

DIGITAL FORENSIC READINESS FOR IOT DEVICES

by

Jaco-Louis Kruger

13025105

Submitted in partial fulfilment of the requirements for the degree

MSc Computer Science

in the

EBIT Faculty

Department of Computer Science

at the

UNIVERSITY OF PRETORIA

Study leader:

Prof. Hein Venter

Date of submission

16 December 2019

TABLE OF CONTENTS

1	CHAPTER 1: INTRODUCTION	2
1.1	PROBLEM STATEMENT	3
1.2	RESEARCH QUESTIONS.....	3
1.2.1	What standards are currently being used for digital forensic readiness that can be applied to the Internet of Things?	3
1.2.2	What are the different categories that Internet of Things devices fall under, and would it be possible to conduct digital forensic investigations on the different types of Internet of Things devices?	3
1.2.3	If it is possible to apply digital forensic readiness on Internet of Things devices, or certain types of Internet of Things devices, would it be possible to proactively apply digital forensic compliance to the devices?	4
1.3	MOTIVATION FOR THE RESEARCH.....	4
1.4	AIMS AND OBJECTIVES	5
1.4.1	The main aim of this dissertation is to investigate processes regarding to digital forensic readiness that can be used in the Internet of Things	5
1.4.2	A secondary aim of this dissertation is to apply trusted digital forensic techniques to the digital forensic readiness model for Internet of Things devices	5
1.4.3	A third aim is to define factors that can be used to ensure that when Internet of Things devices are used for digital forensic readiness, they comply with legal requirements to ensure validity in legal proceedings	6
1.4.4	A fourth aim for the research is to design a proof of concept prototype system....	6
1.5	RESEARCH METHODOLOGY.....	6
1.6	REFERRAL TO RESEARCH QUESTIONS AND OBJECTIVES	8
1.7	LAYOUT	10
2	CHAPTER 2: DIGITAL FORENSICS.....	12
2.1	INTRODUCTION	12
2.2	DEFINITION OF DIGITAL FORENSICS AND DIGITAL FORENSIC INVESTIGATIONS	12
2.3	BACKGROUND ON THE NEED FOR DIGITAL FORENSIC INVESTIGATIONS.....	13
2.4	PROCESSES FOLLOWED IN DIGITAL FORENSIC INVESTIGATIONS	14
2.5	BACKGROUND OF DIGITAL FORENSIC READINESS	15
2.6	DIGITAL FORENSIC READINESS (PROACTIVE) VS DIGITAL FORENSICS (REACTIVE).....	16
2.7	APPLICATIONS OF DIGITAL FORENSIC READINESS.....	16
2.8	CONCLUSION.....	18
3	CHAPTER 3: THE INTERNET OF THINGS	19
3.1	INTRODUCTION	19
3.2	BACKGROUND OF THE IOT	19
3.2.1	Background and history of the Internet of Things.....	20

3.2.2	Different types of the Internet of Things devices and their applications	20
3.3	RELEVANCE OF THE INTERNET OF THINGS DEVICES WITH REGARDS TO DIGITAL FORENSICS	21
3.3.1	Background on the Internet of Things devices in forensic investigations	21
3.3.2	Example of Internet of Things devices forming part of a forensic investigation	23
3.3.3	Possible future uses of Internet of Things devices	24
3.4	CONCLUSION.....	26
4	CHAPTER 4: CURRENT STATE OF DIGITAL FORENSICS FOR THE INTERNET OF THINGS	27
4.1	INTRODUCTION	27
4.2	ISSUES WITH DIGITAL FORENSICS.....	28
4.2.1	Issues with digital forensics in general	28
4.2.2	Issues with mobile forensics.....	29
4.2.3	Issues with digital forensics of the Internet of Things	30
4.3	RESEARCH METHODOLOGY FOR THE STATE OF THE ART OF DIGITAL FORENSICS OF THE INTERNET OF THINGS	30
4.4	STATE OF THE ART IN DIGITAL FORENSICS OF THE INTERNET OF THINGS LITERATURE EVALUATION	30
4.5	SUMMARY OF CONTRIBUTIONS MADE BY CURRENT LITERATURE	32
4.5.1	Topic 1 – Internet of Things.....	33
4.5.2	Topic 2 – Mobile Devices	33
4.5.3	Topic 3 – Digital forensics	34
4.5.4	Topic 4 – Digital forensics of the Internet of Things	34
4.5.5	Topic 5 – Internet of Things legal aspects.....	35
4.5.6	Topic 6 – Digital forensic processes and standards.....	35
4.5.7	Topic 7 – Digital forensic readiness	35
4.6	DISCUSSION	36
4.7	CONCLUSION.....	37
5	CHAPTER 5: LEGAL VALIDITY REQUIREMENTS FOR DIGITAL FORENSIC EVIDENCE	39
5.1	INTRODUCTION	39
5.2	BACKGROUND.....	39
5.2.1	Electronic evidence	39
5.2.2	Admissibility of evidence	40
5.2.3	Data messages	40
5.2.4	Originality and authenticity of evidence	41
5.2.5	Search and seizure	41
5.3	LEGAL ASPECTS REGARDING DIGITAL FORENSICS OF INTERNET OF THINGS.....	42
5.4	CONCLUSION.....	43
6	CHAPTER 6: A MODEL FOR DIGITAL FORENSIC READINESS IN THE INTERNET OF THINGS	44

6.1	INTRODUCTION	44
6.2	RESEARCH METHODOLOGY.....	45
6.3	STATE OF THE ART OF INTERNET OF THINGS FORENSIC TECHNIQUES...45	
6.3.1	Literature review of Internet of Things techniques.....	46
6.3.2	Exposition of Internet of Things techniques.....	46
6.4	ADVANTAGES AND DISADVANTAGES OF THE VARIOUS REQUIREMENTS	48
6.5	THEORETICAL DEVELOPMENT OF AN INTERNET OF THINGS DIGITAL FORENSIC READINESS MODEL BASED ON THE REQUIREMENTS OF INTERNET OF THINGS FORENSICS.....	50
6.5.1	Graphical representation of model solution.....	51
6.5.2	Activity diagrams for the model	53
6.5.3	Mitigating the disadvantages of Internet of Things techniques.....	57
6.6	DISCUSSION	58
6.7	CONCLUSION.....	61
7	CHAPTER 7: A PROTOTYPE FOR DIGITAL FORENSIC READINESS IN THE INTERNET OF THINGS	63
7.1	INTRODUCTION	63
7.2	REITERATION OF THE VARIOUS REQUIREMENTS FOR DIGITAL FORENSICS OF INTERNET OF THINGS.....	63
7.3	POSSIBLE APPROACHES TO IMPLEMENTING THE REQUIREMENTS FOR DIGITAL FORENSICS OF THE INTERNET OF THINGS.....	64
7.4	SOLUTION FOR THE REQUIREMENTS OF DIGITAL FORENSIC FOR THE INTERNET OF THINGS.....	65
7.4.1	Possible open-source digital forensics tools.....	65
7.4.2	Selection of a tool for digital forensics of the Internet of Things	66
7.4.3	Setting up the GRR environment for Internet of Things forensics	68
7.4.4	Install GRR server on a server with a public IP address	70
7.4.5	Installing GRR clients on virtual machines	74
7.4.6	Building custom GRR clients to work on IoT devices	76
7.5	CONCLUSION.....	81
8	CHAPTER 8: TESTING THE PROTOTYPE FOR DIGITAL FORENSIC READINESS IN THE INTERNET OF THINGS.....	83
8.1	INTRODUCTION	83
8.2	FORMULATION OF MEASUREMENTS FOR THE FEASIBILITY OF DFR FOR IOT	83
8.3	OVERVIEW OF TOOLS USED TO MEASURE IMPLEMENTATION FEASIBILITY	85
8.4	STEPS FOR SIMULATING DIGITAL FORENSIC READINESS IN THE INTERNET OF THINGS.....	86
8.4.1	Metrics for measuring device performance	86
8.4.2	Setup of Zabbix and Grafana for the simulation	87
8.4.3	Differences between the various steps of the simulation	90

8.4.4	Detailed information on the smart home simulation program developed for Step 3 and Step 4 of the simulation	92
8.4.5	Detailed information on the DFR setup of the GRR prototype for Step 4	98
8.5	FINDINGS OF SIMULATING DIGITAL FORENSIC READINESS FOR THE INTERNET OF THINGS	111
8.5.1	Processing metrics	112
8.5.2	Storage performance metrics	120
8.5.3	Memory performance metrics	122
8.5.4	Network usage metrics	124
8.6	CONCLUSION	127
9	CHAPTER 9: PROTOTYPE EVALUATION	128
9.1	INTRODUCTION	128
9.2	EVALUATION OF THE FINDINGS OF DFR FOR IOT	129
9.2.1	Processing power	129
9.2.2	Disk usage	129
9.2.3	Memory usage	130
9.2.4	Network usage	130
9.3	EXAMPLE OF FORENSIC VALUE OF DIGITAL FORENSIC READINESS FOR THE INTERNET OF THINGS	131
9.3.1	Physical model of a smart home	131
9.3.2	Fictional scenario for digital forensic readiness in a smart home	132
9.3.3	Forensic investigation into scenario	133
9.4	CRITICAL EVALUATION	135
9.5	CONCLUSION	137
10	CHAPTER 10: CONCLUSION	139
10.1	INTRODUCTION	139
10.2	PROBLEM STATEMENT, RESEARCH QUESTIONS AND RESEARCH OBJECTIVE REVISITED	139
10.3	CONTRIBUTIONS MADE TO BODY OF KNOWLEDGE	140
10.4	FUTURE RESEARCH	142
10.5	CONCLUSION	142
	REFERENCES	144

APPENDICES

APPENDIX A: Systematic literature review evaluation table	152
APPENDIX B: Performance testing tables.....	197
APPENDIX C: Performance testing graphs	209

LIST OF FIGURES

Figure 6-1 Digital forensic readiness for Internet of Things model.....	52
Figure 6-2 Push-based model.....	55
Figure 6-3 Pull-based model.....	56
Figure 7-1 GRR download	71
Figure 7-2 Install script for GRR	71
Figure 7-3 GRR setup parameters.....	72
Figure 7-4 Certificate generation and client repacking.....	73
Figure 7-5 GRR server installation complete	73
Figure 7-6 GRR server running.....	73
Figure 7-7 Logging in to GRR server GUI.....	74
Figure 7-8 GRR admin GUI search clients.....	75
Figure 7-9 New GRR client	75
Figure 7-10 Set up pip virtual environment	76
Figure 7-11 Clone GRR	76
Figure 7-12 Compile protobuf	77
Figure 7-13 Compile protobuf part 2	77
Figure 7-14 Compile protobuf part 3	77
Figure 7-15 Compile protobuf part 4	77
Figure 7-16 Compile protobuf part 5	77
Figure 7-17 GRR installer	77
Figure 7-18 GRR ARM architecture add.....	78
Figure 7-19 GRR ARM architecture add part 2.....	78
Figure 7-20 GRR client configuration.....	79
Figure 7-21 GRR setup fails	80
Figure 7-22 GRR custom client template.....	80
Figure 7-23 GRR custom client template build complete.....	80
Figure 7-24 GRR server build of custom client	80
Figure 7-25 Raspberry Pi client appears in GRR server GUI client list.....	81
Figure 7-26 Raspberry Pi details	81
Figure 8-1 Components of the DFR for IoT simulation	85
Figure 8-2 Zabbix agent install on VM	88
Figure 8-3 Zabbix agent install Raspberry Pi.....	88
Figure 8-4 Zabbix agent configuration	88
Figure 8-5 Automatic detection of new Zabbix agent.....	89
Figure 8-6 Add template to Zabbix host.....	89
Figure 8-7 Successful adding of a new host	89
Figure 8-8 Latest metrics received.....	89

Figure 8-9 Grafana panel creation	90
Figure 8-10 Example of a Grafana panel of metrics	90
Figure 8-11 Physical IoT device smart home simulation.....	93
Figure 8-12 Raspberry Pi as physical IoT device.....	94
Figure 8-13 Smart home program server and client	94
Figure 8-14 Magnetic reed switches used as sensors in physical smart home simulation	95
Figure 8-15 LED lights in physical smart home simulation.....	96
Figure 8-16 Client state change log	96
Figure 8-17 Agent log file.....	98
Figure 8-18 GRR agents.....	100
Figure 8-19 GRR agent details	100
Figure 8-20 Create new GRR cron hunt	101
Figure 8-21 Schedule GRR cron hunt.....	102
Figure 8-22 Cron hunt parameters.....	103
Figure 8-23 GRR agents to execute cron hunt	104
Figure 8-24 Simulation cron hunt parameters.....	104
Figure 8-25 Cron hunt overview.....	105
Figure 8-26 GRR hunt results	106
Figure 8-27 Download evidence	106
Figure 8-28 GRR client details.....	107
Figure 8-29 Virtual file system of evidence file.....	107
Figure 8-30 Timeline of the agent log file.....	108
Figure 8-31 Evidence from Raspberry Pi timeline.....	108
Figure 8-32 Download evidence file.....	109
Figure 8-33 SHA256 hash verify	110
Figure 8-34 Content of agent log file evidence	110
Figure 8-35 Different version of evidence collected	110
Figure 8-36 Step 1's CPU context switches per second for the virtual machines	112
Figure 8-37 Step 4's CPU context switches per second for physical devices	113
Figure 8-38 Step 4's CPU idle time for physical devices	114
Figure 8-39 Step 4's CPU interrupts per second for virtual machines	115
Figure 8-40 Step 4's CPU load 1min average per core for virtual machines.....	116
Figure 8-41 Step 4's CPU system time for virtual machines.....	118
Figure 8-42 Step 4's number of processes for physical devices	119
Figure 8-43 Step 2's number of running processes on virtual machines.....	120
Figure 8-44 Step's 3 free disk space in percentage for physical devices.....	122
Figure 8-45 Step 1's available memory for the virtual machines.....	123
Figure 8-46 Step 2's free swap space in percentage for the virtual machines	124
Figure 8-47 Step 2's outgoing traffic for virtual machines	126

Figure 8-48 Step 4's outgoing traffic for physical devices	127
Figure 9-1 Smart home simulation on a physical IoT device.....	132
Figure 9-2 Deceased moving through the home to finish for work.....	133
Figure 9-3 Deceased left home and unauthorised entry occurs later the day	133
Figure 9-4 Moments before the murder	134
Figure 9-5 Scuffle occurrence.....	134
Figure 10-1 Step 1 CPU context switches per second for VMs	209
Figure 10-2 Step 1 Step 4 CPU context switches per second for physical devices	209
Figure 10-3 Step 2 CPU context switches per second for VMs	210
Figure 10-4 Step 2 CPU context switches per second for physical devices.....	210
Figure 10-5 Step 3 CPU context switches per second for VMs	211
Figure 10-6 Step 3 CPU context switches per second for physical devices.....	211
Figure 10-7 Step 4 CPU context switches per second for VMs	212
Figure 10-8 Step 4 CPU context switches per second for physical devices.....	212
Figure 10-9 Step 1 CPU idle time for VMs	213
Figure 10-10 Step 1 CPU idle time for physical devices	213
Figure 10-11 Step 2 CPU idle time for VMs	214
Figure 10-12 Step 2 CPU idle time for physical devices	214
Figure 10-13 Step 3 CPU idle time for VMs	215
Figure 10-14 Step 3 CPU idle time for physical devices	215
Figure 10-15 Step 4 CPU idle time for VMs	216
Figure 10-16 Step 4 CPU idle time for physical devices	216
Figure 10-17 Step 1 CPU interrupts per second for VMs.....	216
Figure 10-18 Step 1 CPU interrupts per second for physical devices.....	217
Figure 10-19 Step 2 CPU interrupts per second for VMs.....	217
Figure 10-20 Step 2 CPU interrupts per second for physical devices.....	218
Figure 10-21 Step 3 CPU interrupts per second for VMs.....	218
Figure 10-22 Step 3 CPU interrupts per second for physical devices.....	219
Figure 10-23 Step 4 CPU interrupts per second for VMs.....	219
Figure 10-24 Step 4 CPU interrupts per second for physical devices.....	220
Figure 10-25 Step 1 CPU load 1min average per core for VMs.....	220
Figure 10-26 Step 1 CPU load 1min average per core for physical devices	221
Figure 10-27 Step 2 CPU load 1min average per core for VMs.....	221
Figure 10-28 Step 2 CPU load 1min average per core for physical devices	222
Figure 10-29 Step 3 CPU load 1min average per core for VMs.....	222
Figure 10-30 Step 3 CPU load 1min average per core for physical devices	222
Figure 10-31 Step 4 CPU load 1min average per core for VMs.....	223
Figure 10-32 Step 4 CPU load 1min average per core for physical devices	223
Figure 10-33 Step 1 CPU system time for VMs	224

Figure 10-34 Step 1 CPU system time for physical devices	224
Figure 10-35 Step 2 CPU system time for VMs	225
Figure 10-36 Step 2 CPU system time for physical devices	225
Figure 10-37 Step 3 CPU system time for VMs	226
Figure 10-38 Step 3 CPU system time for physical devices	226
Figure 10-39 Step 4 CPU system time for VMs	227
Figure 10-40 Step 4 CPU system time for physical devices	227
Figure 10-41 Step 1 number of processes for VMs	228
Figure 10-42 Step 1 number of processes for physical devices.....	228
Figure 10-43 Step 2 number of processes for VMs	229
Figure 10-44 Step 2 number of processes for physical devices.....	229
Figure 10-45 Step 3 number of processes for VMs	230
Figure 10-46 Step 3 number of processes for physical devices.....	230
Figure 10-47 Step 4 number of processes for VMs	231
Figure 10-48 Step 4 number of processes for physical devices.....	231
Figure 10-49 Step 1 number of running processes on VMs.....	232
Figure 10-50 Step 1 number of running processes on physical devices.....	232
Figure 10-51 Step 2 number of running processes on VMs.....	233
Figure 10-52 Step 2 number of running processes on physical devices.....	233
Figure 10-53 Step 3 number of running processes on VMs.....	234
Figure 10-54 Step 3 number of running processes on physical devices.....	234
Figure 10-55 Step 4 number of running processes on VMs.....	235
Figure 10-56 Step 4 number of running processes on physical devices.....	235
Figure 10-57 Step 1 free disk space in percentage for VMs	236
Figure 10-58 Step 1 free disk space in percentage for physical devices	236
Figure 10-59 Step 2 free disk space in percentage for VMs	237
Figure 10-60 Step 2 free disk space in percentage for physical devices	237
Figure 10-61 Step 3 free disk space in percentage for VMs	238
Figure 10-62 Step 3 free disk space in percentage for physical devices	238
Figure 10-63 Step 4 free disk space in percentage for VMs	239
Figure 10-64 Step 4 free disk space in percentage for physical devices	239
Figure 10-65 Step 1 available memory for the VMs	240
Figure 10-66 Step 1 available memory for the physical devices	240
Figure 10-67 Step 2 available memory for the VMs	241
Figure 10-68 Step 2 available memory for the physical devices	241
Figure 10-69 Step 3 available memory for the VMs	241
Figure 10-70 Step 3 available memory for the physical devices	242
Figure 10-71 Step 4 available memory for the VMs	242
Figure 10-72 Step 4 available memory for the physical devices	243

Figure 10-73 Step 1 free swap space in percentage for the VMs	243
Figure 10-74 Step 1 free swap space in percentage for the physical devices	243
Figure 10-75 Step 2 free swap space in percentage for the VMs	244
Figure 10-76 Step 2 free swap space in percentage for the physical devices	244
Figure 10-77 Step 3 free swap space in percentage for the VMs	245
Figure 10-78 Step 3 free swap space in percentage for the physical devices	245
Figure 10-79 Step 4 free swap space in percentage for the VMs	246
Figure 10-80 Step 4 free swap space in percentage for the physical devices	246
Figure 10-81 Step 1 incoming traffic for VMs.....	247
Figure 10-82 Step 1 outgoing traffic for VMs	247
Figure 10-83 Step 2 incoming traffic for VMs.....	248
Figure 10-84 Step 2 outgoing traffic for VMs	248
Figure 10-85 Step 3 incoming traffic for VMs.....	249
Figure 10-86 Step 3 outgoing traffic for VMs	249
Figure 10-87 Step 4 incoming traffic for VMs.....	250
Figure 10-88 Step 4 outgoing traffic for VMs	250
Figure 10-89 Step 1 incoming traffic for physical devices	251
Figure 10-90 Step 1 outgoing traffic for physical devices	251
Figure 10-91 Step 2 incoming traffic for physical devices	252
Figure 10-92 Step 2 outgoing traffic for physical devices	252
Figure 10-93 Step 3 incoming traffic for physical devices	253
Figure 10-94 Step 3 outgoing traffic for physical devices	253
Figure 10-95 Step 4 incoming traffic for physical devices	254
Figure 10-96 Step 4 outgoing traffic for physical devices	254

LIST OF TABLES

Table 1-1 Summary of research questions and objectives	9
Table 4-1 Paper contributions to the state of the art of DF of IoT	31
Table 6-1 Techniques useful in digital forensics of the Internet of Things	47
Table 6-2 Advantages and disadvantages of digital forensics of Internet of Things techniques.....	49
Table 6-3 Solutions to disadvantages of digital forensics of Internet of Things techniques	57
Table 10-1 Summary of contributions made by different chapters.....	140

ABSTRACT

The Internet of Things (IoT) has evolved to be an important part of modern society. IoT devices can be found in several environments such as smart homes, transportation, the health sector, smart cities and even facilitates automation in organisations. The increasing dependence on IoT devices increases the possibility of security incidents in the physical or cyber environment. Traditional methods of digital forensic (DF) investigations are not always applicable to IoT devices due to their limited data processing resources. A possible solution for conducting forensic investigations on IoT devices is to utilise a proactive approach known as digital forensic readiness (DFR).

This dissertation firstly aims to conduct a thorough review of the available literature in the current body of knowledge to identify a clear process that can be followed to implement DFR tailored for IoT devices. This dissertation then formulates requirements for DFR in IoT based on existing forensic techniques. The requirements for DFR in IoT give rise to the development of a model for DFR in IoT, which is then implemented in a prototype for IoT devices. The prototype is subsequently tested and evaluated on IoT devices that conduct proactive DFR in a simulation of a smart home system. Finally, the dissertation illustrates the feasibility of the DFR processes for IoT and serves as a basis for future research with regards to DFR in IoT. This dissertation will impact future research with regards to developing a standard for DFR in IoT.

Keywords: Digital forensics (DF), Internet of Things (IoT), digital forensic readiness (DFR)

1 CHAPTER 1: INTRODUCTION

The Internet of Things (IoT) is becoming more popular and common in everyday life, but is not always as secure as it should be [1]. The IoT's use of the Internet exposes it to cyber threats, which can result in cybercrimes. Currently there is no silver bullet for digital security, which means that different solutions need to be developed for different scenarios and threats. Security vulnerabilities are exploited when digital security fails, and cybercrimes can be committed. When security vulnerabilities are exploited, harm in the form of denial of service, breach of confidentiality and even compromise of data integrity can occur.

When cybercrimes cause significant harm, investigations can determine the perpetrator(s). This is where DF come into play. DF occur after the conclusion of a cybercrime, which makes it a reactive response by nature. DF can be a time-consuming process due to the amount of time it takes to collect evidence about the cybercrime after its occurrence. However, research has been undertaken to enable proactive DF known as digital forensic readiness (DFR).

DFR is still in its infancy and a lot of research has to be done to increase the acceptance of DFR techniques and standards. This dissertation aims to address the need for DFR in digital forensic investigations, with specific focus on the applicability of DFR on the IoT.

The rest of this chapter is constructed as follows: firstly, the problem statement is defined based on the information provided in this section. The problem statement then leads to the formulation of several research questions. The next section in the chapter provides the motivation for this dissertation. The research questions are then reworked into aims and objectives in the next section of this chapter. This chapter then presents the research methodology followed in this dissertation. This chapter then provides a summarised version of the research questions and objectives to ease indication of contribution towards the research questions and objective throughout the study. The chapter concludes with an overview of the layout of the rest of this dissertation.

DFR is still in its infancy and plagued by a lack of standards for all areas where DF is implemented, not just the IoT. The lack of standards for the implementation of DFR lead to the formulation of the problem statement addressed in this dissertation.

1.1 PROBLEM STATEMENT

As a research area, DFR offers a lot of potential research topics. DFR is still in its infancy and a lot can be done in order to enhance the body of knowledge with regards to the phenomenon as a whole. However, this study is more focused on a specific application of DFR, as it aims to address the application of DFR in IoT – more specifically the DFR processes that can be applied to the IoT. Currently there are no clear processes that can be followed to ensure DFR of IoT devices, which is the specific problem that this dissertation addresses. This problem statement can be divided into smaller research questions, which are discussed in the following section.

1.2 RESEARCH QUESTIONS

There are several research questions that need to be answered in order to address the problem statement of this dissertation. These research questions are discussed in more detail in the next subsections.

1.2.1 What standards are currently being used for digital forensic readiness that can be applied to the Internet of Things?

This question aims to address the apparent lack of DFR processes that can be applied to the IoT by looking at current standards applicable to DFR in the various domains that already make use thereof. It will subsequently be determined whether these standards can be used or adapted to suit the needs of DFR in the IoT. Research with regards to the security of the IoT is also evaluated to determine whether current security standards can be used or adapted in order to incorporate DFR standards.

1.2.2 What are the different categories that Internet of Things devices fall under, and would it be possible to conduct digital forensic investigations on the different types of Internet of Things devices?

This question aims to determine whether a DFR approach can be applied to the IoT as a whole, or perhaps a subset of IoT devices. The IoT can be used to collect, process and

transmit generated data over the internet. It consists of a wide array of devices that can be connected to the internet in order to perform some predefined task. However, the IoT devices that perform these tasks are heterogeneous. They can vary based on size, processing power, hardware resources, software and application (purpose of the device). The different types of IoT devices are investigated, so as to determine whether or not digital forensic investigations can be conducted on all, or a subset of, IoT devices.

1.2.3 If it is possible to apply digital forensic readiness on Internet of Things devices, or certain types of Internet of Things devices, would it be possible to proactively apply digital forensic compliance to the devices?

Assuming that it is possible to apply DFR processes to IoT devices, or a subset of IoT devices, there should be certain compliance requirements that need to be met. Digital forensic compliance in this context would be a combination of certain requirement specifications that need to be met in order for the DFR process to produce evidence that is acceptable in legal proceedings. These required specifications can consist of accepted digital forensic techniques. When all requirement specifications are met, the IoT devices would be considered DFR compliant. DFR compliance would then entail that the evidence produced by the DFR process can be used in legal procedures as sound evidence. Existing DF IoT techniques are evaluated to establish compliance requirements for DFR in IoT.

The research questions drive the research in this dissertation. The research questions therefore aim to address the main problem statement in this dissertation. The motivations that led to the research problem follow.

1.3 MOTIVATION FOR THE RESEARCH

The first motivational factor for this study is that the IoT can be used for malicious purposes, or at least provide insights into malicious activities. DFR can be used to investigate these malicious actions to possibly convict the guilty parties. The next factor that motivates this research is the vast array of different types of IoT devices. The existence of such a variety of IoT devices raises the question as to whether or not these devices can be used to proactively collect data that can be useful in digital forensic investigations.

The next factor motivating this research is the importance of the IoT due to its increasing role in modern society. New advancements in technology can become future influencers of society, for example smart watches which monitor the vitals of wearers to provide insights into health and fitness. They can even be used to notify emergency services of medical emergencies. If IoT devices address a gap in modern society, the adaptation of IoT devices can be rapidly advanced.

The final factor motivating this research is whether or not it would be possible to create standards that need to be adhered to when developing and using IoT devices, so as to ensure that the IoT devices can be used proactively to prevent malicious activities, or to increase the effectiveness of digital forensic investigations when IoT devices formed part of the malicious activity investigated.

The motivation and research questions contribute toward the different aims and objectives of the research, which are stated in the following section.

1.4 AIMS AND OBJECTIVES

This study has several aims. Each of the main aims is presented in numeric form for easy referral throughout the rest of the dissertation. This section provides an overview of the main aims, including the main objectives for each aim.

1.4.1 The main aim of this dissertation is to investigate processes regarding to digital forensic readiness that can be used in the Internet of Things

The following specific requirements are set for this main aim of the dissertation:

- Current DFR standards and processes are evaluated to determine whether these standards and processes can be used in the IoT environment.
- IoT is classified in order to determine whether DFR processes can be used on all types of IoT devices.

1.4.2 A secondary aim of this dissertation is to apply trusted digital forensic techniques to the digital forensic readiness model for Internet of Things devices

The following specific requirements are set for this aim:

- Current trusted digital forensic techniques are investigated to determine techniques that can be used in the IoT.
- Appropriate techniques for the IoT are formalised to establish a model for application of DFR in the IoT.

1.4.3 A third aim is to define factors that can be used to ensure that when Internet of Things devices are used for digital forensic readiness, they comply with legal requirements to ensure validity in legal proceedings

This aim has several objectives that are addressed in this dissertation:

- Current requirements for digital forensic techniques are evaluated to determine the factors that ensure digital forensic evidence is acceptable in legal proceedings.
- Validity requirements for trusted digital forensic evidence are then incorporated in a standard set of requirements for DFR in the IoT to ensure that evidence produced by IoT devices is considered valid in legal proceedings.

1.4.4 A fourth aim for the research is to design a proof of concept prototype system

The final aim for this research has the following set of objectives:

- A proof of concept system is developed in order to demonstrate the working of the model developed from the research in this study.
- The proof of concept prototype must show that the model can be implemented and used to provide trustworthy and forensically sound digital evidence.

1.5 RESEARCH METHODOLOGY

This research study makes use of several established research methodologies. The incorporation of various research methodologies enables a wider variety of investigations in this study, each providing different insights into the research at hand. The combination of methodologies aims to enhance the research conducted in this dissertation.

The first methodology that this research study makes use of is a literature review. The literature review provides the background of the problem addressed in this dissertation, as

well as deeper insight with regards to the IoT, DF, DFR and the legal aspects of forensic investigations. Literature reviews create foundations for the advancement of knowledge, while uncovering areas where further research is needed by understanding what existing research covers about a specific research question [2], [3]. One of the benefits of conducting a literature review is the ability to develop new models that extend existing research based on determining the current state of the body of knowledge [2]. Literature reviews furthermore highlight key issues in the existing body of knowledge [4].

This dissertation furthermore makes use of a systematic literature review (Chapter 4) to formulate the state of the art DF of IoT. A systematic literature review explicitly explains the steps followed in the review [3]. It is also exhaustive in including and reviewing papers relevant to the topic at hand [3], [5]. The main goal of a systematic review is to aggregate all relevant knowledge and be repeatable, which means that other researchers will be able to produce the same results if they follow the steps defined in the systematic literature review [3], [5].

This dissertation also makes use of modelling by introducing a new model for DFR in IoT devices. Modelling is an abstract hypothetical representation of a system with an emphasis on the behaviour and properties of the system [6]. Models are developed based on prior knowledge to satisfy certain design goals [6].

Furthermore, this study implements a prototype for DFR in IoT that is based on the model developed in this dissertation. A prototype is a working model, or partly working model, of a system [7]. Generally, it is a tentative version of a system that highlights important features of that system that will be completely functional at a later stage [7]. Prototyping enhances the ability to visualise problems and aids in the identification of incorrect design assumptions [8].

This dissertation also makes use of simulation to determine the effectiveness and practicality of the prototype developed in this research. A simulation can be seen as a virtual experiment [5] that imitates the operation of a real-world existing or conceptualised system over time to generate an artificial history of the working of the system [9]. The generated artificial history of the system operation is then used to draw conclusions from

the characteristics of the system simulated [9]. Simulation is furthermore used to analyse the working of a system and help with the development of real systems [9].

Finally, critical evaluation was used to evaluate the findings of the implemented model, prototype and simulations conducted in this dissertation. Critical evaluation has several steps that need to be followed [10], including the development of parameters for the evaluation, which entails formulating the need for the evaluation as well as determining what needs to be reported on at the end of the evaluation. The next step is to formulate the methods of the evaluation, which entails choosing the criteria for the evaluation, determining the various ways in which to gather the evidence needed, and developing a plan for collecting this evidence. The next step entails the formulation of standards, collection of evidence and documenting the findings from the evidence collected. The final step in critical evaluation is to report on the findings and make decisions based on them. This step entails checking the evidence against the standards and documenting the findings accordingly [10].

This dissertation has several aims and objectives that it addresses. All of the above-mentioned methodologies work together to address the objectives of this dissertation. Therefore, an easy referral method is required be able to show how this dissertation addresses these various objectives. This leads to the following section, where a method is discussed that contributes toward easy referral to the various aims and objectives of this dissertation.

1.6 REFERRAL TO RESEARCH QUESTIONS AND OBJECTIVES

Due to the complexity and length of the various research questions and objectives, the research questions and objectives are summarised for easy referral throughout this dissertation. When a reference is made to a research question or a research objective, to the number of the research question or objective addressed is included. A few words are also provided for easy reference to the main aspects of the research questions and objectives. Table 1-1 provides the various research questions and objectives, along with abbreviated research questions and objectives. Abbreviated summaries are also provided for the various research questions and objectives, and are used throughout the remainder of this study. The number assigned to each research question and objective correspond to the section number that they appear in in this chapter.

Table 1-1 Summary of research questions and objectives

Number	Abbreviated research question or research objective	Abbreviated summary of research question or objective
1.2.1	What standards are currently being used for digital forensic readiness that can be applied to the Internet of Things?	DFR IoT standards processes
1.2.2	What are the different categories that Internet of Things devices fall under, and would it be possible to conduct digital forensic investigations on the different types of Internet of Things devices?	DFR IoT devices feasibility
1.2.3	If it is possible to apply digital forensic readiness on Internet of Things devices, or certain types of Internet of Things devices, would it be possible to proactively apply digital forensic compliance to the devices?	DFR requirements, legal aspects
1.4.1	The main aim of this dissertation is to investigate processes regarding to digital forensic readiness that can be used in the Internet of Things	DFR processes, IoT device classification
1.4.2	A secondary aim of this dissertation is to apply trusted digital forensic techniques to the digital forensic readiness model for Internet of Things devices	DF techniques, DFR model
1.4.3	A third aim is to define factors that can be used to ensure that when Internet of Things devices are used for digital forensic readiness, they comply with legal requirements to ensure validity in legal proceedings	DF legal requirements, DFR IoT legal compliance
1.4.4	A fourth aim for the research is to design a proof of concept prototype system	Proof of concept, DFR IoT prototype

The various research questions and objectives are addressed throughout the various chapters and sections in this study. Every chapter contains a list of all the contributions, in the introduction and conclusion, that it makes toward the various research questions and objectives. This is done for the convenience of the reader and to enhance easy referral to contributions made in the various chapters of this dissertation.

The next section provides an overview of how the remainder of this dissertation is structured. The research questions and objectives are thus addressed throughout the structure of this dissertation.

1.7 LAYOUT

This section discusses the layout and structure of the chapters and subsections of this dissertation.

Chapter 1 provides the introduction and problem statement to the research. It does this by contextualising the dissertation, stating the problem statement to be addressed, as well as providing the aims and objectives for this research.

Chapter 2 focuses on established digital forensic techniques and standards. The established techniques and standards are analysed in order to determine the level to which these techniques could be applied to digital forensic investigations that include IoT devices. Chapter 2 also reviews current techniques and standards for DFR approaches. Current DFR approaches are evaluated in order to guide the implementation of these standards and approaches for the IoT. This chapter also aims to determine whether DFR techniques can be applied to a subset, or all, of the existing types of IoT devices.

Chapter 3 provides a literature review on the IoT. It aims to formulate the importance of preparing IoT devices for digital forensic investigations.

Chapter 4 aims to provide an overview of all aspects related to DF of the IoT. The chapter starts with an overview of aspects relating to this dissertation and then conducts an in-depth systematic literature review of the current body of knowledge with regards to aspects related to DF of the IoT. Chapter 4 therefore enhances this dissertation by

indicating the current state of affairs with regards to DF of IoT to show that DF of IoT needs to be enhanced.

Chapter 5 surveys the requirements for DF evidence to be considered valid and sound for use in legal proceedings. This will ensure that the formulation of requirements for DFR standards for the IoT will be acceptable in legal proceedings, which contributes to the current body of knowledge.

Chapter 6 focuses on the development of a model for guaranteeing that evidence produced by DFR techniques in IoT devices will be acceptable in legal proceedings.

Chapter 7 is focused on the implementation of a prototype for the model formalised in Chapter 6. The model presented in Chapter 6 is used to determine requirements for the prototype's implementation. The prototype is tested in a simulation of DFR for IoT.

In Chapter 8 a prototype is developed and implemented to test the model developed in this dissertation. Chapter 8 serves to determine the feasibility of DFR for IoT.

Chapter 9 is based on the work done in the preceding chapter. Chapter 9 presents a critical evaluation of the prototype developed for DFR for IoT and makes use of a simulation to determine the validity of the model in future IoT developments.

Chapter 10 summarises the research reflected in this dissertation, and summarises the study's contributions.

2 CHAPTER 2: DIGITAL FORENSICS

2.1 INTRODUCTION

DFR is seen as a specific subset of DF. Since this dissertation focuses on DFR for the IoT, further background information is needed about DF and DFR. This chapter is a literature review of DF and DFR, providing well-rounded definitions for DF and DFR and indicating the various procedures followed in both DF and DFR. In addition, an in-depth comparison between DF and DFR is presented. DFR is a relatively new field of research and academia is still trying to find various areas where DFR can be successfully applied.

This chapter also provides an overview of various applications of DFR in the current literature. It aims to support research question 1.2.1 (DFR IoT standards processes) by providing a background on DFR and the processes that surround DFR. It furthermore partially addresses objective 1.4.1 (DFR processes, IoT device classification) by providing an overview of current DFR processes.

This chapter starts by defining DF as well as digital forensic investigations, followed by an overview on the need for forensic investigations in general and the processes in a forensic investigation. Background on DFR is then provided, which leads to a comparison between DF and DFR. The chapter then provides an overview of the application of DFR.

2.2 DEFINITION OF DIGITAL FORENSICS AND DIGITAL FORENSIC INVESTIGATIONS

DF (a synonym for computer forensics, forensic computing and network forensics) is relatively new as a science [11], [31]. DF is based on the notion of securing and isolating evidence, recording the scene, systematically searching for evidence, collecting and packaging evidence gathered, and finally maintaining the chain of custody over evidence gathered [11]. It can be defined as the use of scientifically derived methods for the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of evidence obtained from electronic and digital sources so as to aid in the reconstruction of criminal events and the actions leading to criminal events [11], [12].

DF is not limited to a particular digital platform, which means that all future digital technologies and platforms must be encompassed in digital forensic standards [11]. DF is

also a multidisciplinary subject that involves engineering, computer science, processing of signals and even criminal justice [12]. DF can be defined as the science of investigating cyber-crimes by identifying, collecting, examining and analysing digital data in a manner that preserves the integrity of information [13], [14]. The digital evidence gathered refers to any digital data that shows that criminal activity has occurred, while also establishing a link between the suspect(s), possible victim(s) and the crime(s) committed [15]. A digital forensic investigation furthermore aims to be able to reconstruct the series of events that resulted in the incident (using the digital evidence gathered), while being valid and trustworthy (forensically sound) in legal proceedings [14].

DF is used as a tool to solve crimes committed using computers and computer systems, or where evidence related to a crime, may be stored on a computer [16]. Crimes that are investigated using DF are not necessarily new crimes [11]. These crimes can be traditional crimes that merely exploit computing power and the availability of information in the digital environment to commit [11]. One of the encouraging factors of using computer systems in crimes is the anonymity that computer systems provide, which leads to a smaller chance of the criminal being caught or prosecuted [11].

As a science, DF must be defined [17]. Science indicates the ability of researchers to conduct repeatable and controllable experiments that produce results that can be reproduced [17]. Science thus entails that anyone can observe phenomena and results can be produced by anyone that have access to all the required training and resources to conduct the experiments [17]. It is, however, difficult to maintain scientific aspects in DF, as different researchers use different data sets to test and evaluate their results [17].

2.3 BACKGROUND ON THE NEED FOR DIGITAL FORENSIC INVESTIGATIONS

DF aims to solve real-world problems of criminal investigators [17]. It must adhere to legal standards, which require that information presented in a court must be an accurate reflection of data acquired [17], [18]. In DF, information is mostly considered accurate when standard procedures are followed, and hash functions indicate that the evidence has not been tampered with [17].

DFR can be defined as the extent to which digital systems record activities and data sufficient enough so that future forensic investigations are able to use the gathered

evidence and be certain of its authenticity [19]. It is based on the ability of an entity governing a system to minimise forensic investigation costs and ensure the maximum applicability of digital evidence in forensic procedures [20]. DFR is a preparation for future incidents, increasing the ease of future digital forensic investigations [19], [20]. Furthermore, it is highly related to the preservation of data, planning for future incidents, policies and monitoring [21].

In order to catch and prosecute criminals that made use of computer systems in their crimes, investigators need to make use of formally accepted and well-defined digital forensic procedures [11]. Digital forensic investigations need to be done in precise, well-formulated steps in order to ensure that the evidence is trustworthy. An overview of such a process is provided in the next section.

2.4 PROCESSES FOLLOWED IN DIGITAL FORENSIC INVESTIGATIONS

This section gives an overview of an abstract model of digital forensic investigations developed by Reith [11], which comprises the following steps:

1. Identify the incident by recognising the different indicators that determine the type of incident.
2. Prepare the tools to be used in the investigation, including the issuing of search warrants and obtaining of relevant authorisations to conduct the investigation.
3. Develop the approach strategy to ensure the maximum collection of potential evidence with the least amount of impact on the victim.
4. Preserve and secure the state of the physical and digital evidence by isolating the evidence. This step also involves preventing other people from using, changing or even damaging the evidence using physical devices such as electromagnetic devices.
5. Collect potential evidence by recording the physical scene and duplicating the potential digital evidence through use of standardised digital forensic procedures.
6. Examine the evidence gathered. This step is where the search for specific pieces of evidence takes place. The search for evidence should be systematic and in-depth. The search should be based on the goal of locating and identifying potential evidence. This step also entails the creation of documentation for further analysis.

7. Analyse the evidence gathered. This step involves the reconstruction of fractions of evidence to draw conclusions with the aim of supporting, proving or disproving a crime theory.
8. Present the evidence. Presentation is about summarising the evidence gathered. The summary should provide an explanation of the evidence and should also be written in layman's terms. The formal explanation of the evidence should also be provided.
9. Return the evidence. In this step the physical and digital evidence is returned to the owner, and it is determined what evidence should be removed from devices.

The steps in the DF process serve as a foundation for producing reliable evidence in investigations after incidents occurred. However, there is another possible approach to forensic investigations, namely proactive forensic investigations or DFR.

2.5 BACKGROUND OF DIGITAL FORENSIC READINESS

DFR is defined as the ability to maximise the potential to use credible digital evidence while minimising the cost of an investigation by gathering evidence before an incident occurs [22]-[25]. DFR also aims to cause minimal disruption to the original system, environment or network in which it is applied [23].

DFR is not only based on procedures but also entails careful pre-incident planning to determine how to maximise the use of the evidence gathered, which means that it entails a great amount of planning for future events [26]. DFR also forms part of the DF investigation lifecycle, which entails the identification, preservation, storage, analysis, and usage of evidence [26]. However, DFR is not just about processes and techniques, but all that is necessary to maximise the ability of the environment to collect credible evidence prior to an incident [27]. DFR is implemented by defining standards that organisations have to conform to when collecting data prior to incidents [28]. It is furthermore seen as a discipline inside the field of DF [29].

Valjarevic [25] provides a great overview of DFR and various aspects relating to it. Several factors that have an impact of DFR include how logging is performed by a system, what a system logs, intrusion detection systems present on a system, the handling of evidence

and the way in which evidence is obtained forensically. These are the main aspects that make up DFR.

2.6 DIGITAL FORENSIC READINESS (PROACTIVE) VS DIGITAL FORENSICS (REACTIVE)

There are several differences between DFR and DF, and this section aims to provide an overview of these differences. The main difference between DF and DFR is that DF is employed as a post-event response whereas DFR refers to pre-event evidence gathering [22], [27], [30].

DF is seen as the science of investigations [43] and as a branch of forensic science that is focused on investigating material found on digital devices, usually related to cyber-crimes [44]. It is based on systematic, formalised examinations that aim to lead to evidence deemed trustworthy and valid in legal proceedings, which in turn leads to sufficient evidence to successfully prosecute guilty parties [22], [28]. DF analytical and investigative techniques involve the preservation, identification, extraction, documentation, analysis and finally interpretation of digital evidence to reconstruct incidents [27]. The application of these techniques proactively leads to DFR [27]. So, where DF is a search and seizure approach, DFR gathers the evidence before an incident occurs, so seizure of devices is not necessarily needed [28].

2.7 APPLICATIONS OF DIGITAL FORENSIC READINESS

DFR is still a relatively new field in academia, which results in researchers still trying to find various areas of society where DFR can be useful and effective. DFR is not only applicable in criminal investigations, such as murders and kidnappings, but it can be part of various other types of investigations [26].

The literature review done in this chapter shows that DFR is currently researched in two broad categories: the business applications of DFR and specific technological aspects where DFR is promising (the IoT). DFR in business environments is one of the main aspects where DFR is being used and researched [29]. DFR in a business environment provides organisations with several benefits when compared to solely depending on normal DF. Digital evidence forms a major part of organisations, as evidence can be found in transactions, emails and calendar events [26]. DFR enables organisations to bolster

their defence when they are faced with a lawsuit, due to them being able to quickly and effectively retrieve the needed evidence gathered through their DFR procedures, which are based on laws and regulations that have been in place prior to an event that requires investigation [22], [27], [30].

Organisations can also use evidence gathering as a deterrent to insider threats by making employees aware of data gathering, as well as using the evidence for internal disciplinary actions [22], [29]. DFR has one key benefit, which is reducing the time and cost of investigations after an incident occurred. This is quite beneficial to organisations as post-incident investigations can be extremely costly [22], [27]. Another benefit of DFR for organisations is that it ensures regulations are met and good governance of the organisation is improved [22]. DFR can be defined as part of an organisational goal to ensure that organisations are well positioned to do investigations and determine root causes of incidents [31].

Rowlingson [22] identifies ten important steps when implementing DFR in an organisation:

1. Define various scenarios in the work environment that require evidence gathering.
2. Identify the sources of evidence for the scenarios provided.
3. Determine the requirements for evidence gathering.
4. Create capacity for securely gathering the evidence needed.
5. Develop policies for secure retention and handling of evidence gathered.
6. Fine-tune monitoring to ensure that all events that could possibly lead to investigations are gathered.
7. Define the circumstances that would warrant a full investigation.
8. Provide training to all those involved in evidence gathering and investigations.
9. Document cases where incidents took place and how it impacted the organisation.
10. Continue with legal actions in response to the incident that took place.

A number of papers that attempt to apply DFR to specific technological areas are addressed in this section of the dissertation. Academia has attempted to apply DFR to wireless sensor networks to ensure that sensor networks are ready for forensic investigations without changing these networks [23]. DFR has also been applied to the cloud in order to prepare it for forensic investigations, as cloud environments are not suitable for the usual search and seizure forensic investigations [24], [28]. DFR has also

been applied to public key infrastructures [25]. However, most of the applications of DFR to technological areas are still in various stages of development.

2.8 CONCLUSION

DFR is still relatively new when compared to DF. While DFR is generally seen as a subset of DF, it is defined differently. DFR is based on the notion of gathering potential evidence before an incident occurs, which increases the time it takes for forensic investigators to obtain all of the devices and evidence they need to investigate the incident.

The major difference between DFR and DF is the fact that DFR is done proactively and DF reactively. As mentioned above, DFR is still relatively new and the lack of available research about the area corroborates this. Academia is still trying to show how DFR can be applied in business environments as well as trying to find technological applications for DFR.

This chapter partly addressed research question 1.2.1 (DFR IoT standards processes) by formulating the various DFR processes and its implementation. The rest of this dissertation aims to apply DFR to IoT based upon the DFR processes defined in this chapter. This chapter also partially satisfied objective 1.4.1 (DFR processes, IoT device classification) of this dissertation, which seeks an evaluation of DFR processes in order to determine whether DFR can be applied to the IoT. This chapter sets the basis for fulfilling this objective.

DFR for IoT devices is even less common in the current body of knowledge, which calls for further research into the field of DFR for IoT devices. This leads to the next chapter, which takes an in-depth look at the IoT and aims to further the link between DF and IoT.

3 CHAPTER 3: THE INTERNET OF THINGS

3.1 INTRODUCTION

The IoT is a dynamic and evolving global infrastructure that is based on network-enabled devices (or things) that have identities, physical attributes and integrate with the information network by gathering information and processing the gathered information by making use of sensors, actuators and processors [32]-[36]. The term IoT was first used in 1999 by Kevin Ashton to describe a system where the physical world can be connected to the internet through the use of sensors [37].

The IoT refers to adding computing power and connectivity to devices and sensors so that objects can be located in the physical world, environments can be monitored and controlled, and data collected and used for analysis and management [38], [39]. According to this concept, just about any physical object can connect to other devices and the internet through technology [40]. There is, however, no formal and universally accepted definition for the IoT, hence several definitions are provided in this study [37].

This chapter aims to establish background information about the IoT and guide the reader towards the application of DF in the IoT environment. It also aims to partly address research question 1.2.2 (DFR IoT devices feasibility) as well as provide background information for aim 1.4.1 (DFR processes, IoT device classification). It starts with background information and some history about the IoT, followed by the different types of IoT devices and how they are applied in modern society. The chapter then establishes the relevance between the IoT and DF by providing a brief background on forensic investigations with a specific emphasis on IoT devices. This is followed by an expansive example that illustrates the benefit of IoT devices in forensic investigations. The chapter is concluded with a brief discussion of possible future uses of IoT, followed by a conclusion of the chapter.

3.2 BACKGROUND OF THE IOT

This section aims to provide some general background information and a brief history about the IoT. It furthermore provides information on different types of IoT devices as well as their applications in modern society

3.2.1 Background and history of the Internet of Things

The IoT aims to connect several aspects of the physical world to the internet, which contributes to the exchange of information and can lead to innovative services and an increase in productivity [41]. The IoT enables physical world entities to become virtual entities [42]. When physical entities become virtual entities, they have a readable, addressable and locatable counterpart on the internet [42]. IoT devices therefore extend computing to anything, anywhere, anytime, anywhere, any service and anyone, making it more pervasive and immersive [20-21]. The IoT is also based on smart sensors interacting with one another, without human interaction, to enable new applications and enhance modern society [43]. IoT devices can work together in order to achieve a shared goal [42]. The IoT devices will also be ubiquitous, self-aware of the context in which it exists and facilitate ambient intelligence [34].

As mentioned above, the IoT is continuously evolving and changing as new technology becomes available and new areas of use become apparent. The IoT develops in different phases. The current phase entails the ability of machines to communicate with one another [43]. The next phase entails more diverse connectivity between different technologies, enabling more intelligent decision making [43]. However, the IoT also faces several challenges such as a lack of trust in the technology and devices, governance, and especially standardisation, which is of concern to the research done in this study [33].

3.2.2 Different types of the Internet of Things devices and their applications

The IoT can be applied to various domains such as automation; medical aids and devices; intelligent management of energy, electricity and traffic; and even smart cities [44], [45]. Another example of the IoT is smart vehicles and smart homes, which eventually contribute towards smart cities [45]. Smart cities are one of the most attractive, large-scale applications of the IoT, as it aims to increase effective resource management [21], [23], [24]. The IoT is also increasing the realisation of Green-IT due to increasing energy efficiency in various domains, such as logistics and smart cities [33]. The IoT is constantly being applied to new domains, creating new forms of business and uses [33].

3.3 RELEVANCE OF THE INTERNET OF THINGS DEVICES WITH REGARDS TO DIGITAL FORENSICS

This section aims to establish a relation between IoT devices and the application of DF to IoT devices. This section is divided into several subsections, of which the first provides a background on IoT devices in forensic investigations. This section elaborates on the concept of IoT devices in forensic investigations with an example. The section is concluded with a discussion of possible future uses of IoT devices and how DF of IoT devices might be useful in the future.

3.3.1 Background on the Internet of Things devices in forensic investigations

The IoT may become an increasingly attractive target for cyberattacks as IoT devices find more applications in modern society [25]. IoT devices can even be used to launch cyberattacks against cyber devices and systems [25]. These attacks can cause great harm to cyber infrastructure, and DF sometimes need to investigate the source of the attacks in order to criminally charge the attackers.

The IoT refers to devices that sense changes in the physical world, interprets what has been sensed and then converts this into digital data or information. This information is then captured and most likely transmitted via the internet. If the information transmitted forms part of possible evidence in some investigation, DF would require access to said information. This begs the question whether it is possible to extract digital forensic evidence from the IoT and how this can be achieved.

Extracting digital forensic evidence from the IoT is no easy task, as the existing digital forensic procedures and tools cannot be applied to the extremely heterogeneous and distributed infrastructure of the IoT [46], [47]. The tremendous amount of IoT devices that generate data and possible digital evidence will create new challenges for data management in forensic investigations [46], [48]. It will also be challenging to collect evidence from IoT devices due to the distributed nature of IoT devices and networks, and the possible uncertainty of the origin of evidence [46], [48]. The heterogenous nature of IoT devices also increases the complexity of digital forensic investigations, as these various devices use different types of operating systems, file formats and protocols [27]. The variety of devices can also lead to problems regarding timelines, as different devices can have different timestamps and time zone references which need to be unified in order

to correlate evidence across multiple devices. When it is possible to collect evidence from IoT devices, digital forensic investigators will need to be able to trust the evidence residing on devices as attackers could have tampered with the devices in order to provide false evidence [46].

IoT digital evidence can be gathered from three different locations: on-device evidence, network evidence and cloud evidence [46]. On-device evidence is evidence that is stored on the IoT devices themselves, network evidence is evidence that is transmitted over networks that the IoT devices form a part of, and cloud evidence is evidence generated by IoT devices which is then sent to cloud services as IoT devices normally have limited storage capacity.

IoT DF is still in its infancy and one of the proposed models for IoT DF makes use of a secure logging scheme [46], more specifically a cloud-based secure digital evidence storing system. IoT devices connect with the system and transmit network logs, registries and sensor readings that is then stored according to different users. The confidentiality of the data is ensured by public-private key encryption. Digital forensic investigators can gain access to the cloud-based evidence using read-only APIs. A problem with public-private key encryption is that it can be too computationally complex for some IoT devices as these devices typically have less powerful processors and resources needed for such computations [48].

The IoT requires unified standards when it comes to digital forensic procedures, as current digital forensic procedures can lead to digital forensic evidence being dismissed in legal proceedings [47]. However, DF in the IoT can be advanced through the implementation of forensically accepted methods and standards specifically created for the IoT. The implementation of these standards in IoT digital forensic procedures may lead to more successful IoT digital forensic investigations [47]. The use of DFR approaches has been proposed as frameworks that include DFR approaches as well as reactive IoT DF [47].

DF that make use of IoT devices can provide new levels of understanding of crimes, as data from various IoT devices can provide more information about the attacker(s). An example of enhanced levels of understanding of crimes is the possibility of examining location data from IoT devices used by suspects at specific times [46].

3.3.2 Example of Internet of Things devices forming part of a forensic investigation

An example of a scenario where IoT forensics can be useful is as follows: a murder took place in a specific room of a house and evidence is needed to prove that they did not commit the murder. The house in this scenario is a smart home. This means that the house is outfitted with several smart devices, such as sensors and actuators that perform certain tasks based on instructions from sensor devices.

The smart home in this scenario is outfitted with ultrasonic sensors, which can detect the presence of humans in a room. These sensors turn on the lights in a room when movement is detected, resulting in the lights turning on when someone enters a room and turning off when no one is present. The smart systems write the different state changes to a secure log file in the smart home system.

The next part in the scenario is the murder itself. The homeowner arrived home at 13:30 on a Sunday afternoon. The homeowner was shot and killed at 13:31 on the same day, 1 minute after their arrival. Suspect X and Suspect Y are suspected of killing the homeowner. The two suspects now need to prove their innocence.

The logs from the smart home systems show that someone entered the house at 13:15 and moved through several rooms of the house, based on the ultrasonic sensors. The person finally entered the room where the murder occurred at 13:25 and remained there until 13:32. The person then left the house at 13:33.

Suspect Y has no clear alibi as to where he was at the time prior, during or after the murder. However, Suspect Y has a smart watch that monitors heart rate, but not GPS-based location. The secure logs from the smart watch are retrieved in a forensically sound manner. The analysis from the heart rate logs of the smart watch shows an incline in the wearer's heart rate from 13:14 to 13:30 on the Sunday afternoon of the murder, after which the wearers heart rate spiked to extremely high rates until 13:35. This is an indication of either extreme exercise or a stressful event that the wearer encountered.

Suspect X also has a smart watch that monitors heart rate as well as GPS-based location. The smart watch also saves location and heart-rate data to a secure log system. The logs

are then retrieved in a forensically sound manner and analysed. The logs from the smart watch shows that Suspect X had a stable heart rate from 09:15 to 15:25, which is an indicator of a peaceful Sunday. The location data from the smart watch also shows that Suspect X was at home from the previous evening until 10:17. The wearer then drove to his/her parents for the rest of the Sunday afternoon. Suspect X's parents confirmed this in court.

The example provided above goes to show the possibility of using IoT devices to gather supporting evidence for investigations into crimes in the physical world. IoT evidence serves as an additional source of evidence in legal proceedings. This dissertation shows how IoT evidence can be used in combination with traditional sources of evidence.

IoT devices can also be used to gather evidence for cyber-based incidents. The example above aims to show how IoT devices can be used to augment information gathered using standard investigation methods. The example shows that Suspect X has an alibi and wasn't at the scene of the murder. However, the example also shows that Suspect Y was in an extremely stressful situation at the time of the murder, due to the spike in their heart rate. Suspect Y also did not provide an alibi for the time of the murder. When combining evidence from the real world and IoT devices, Suspect Y needs to be investigated further, whereas Suspect X is most likely innocent in this scenario.

3.3.3 Possible future uses of Internet of Things devices

The IoT is finding more and more applications in modern society, as the technology is incorporated into an increasing number of scenarios and environments. Several academic papers and publications mention possible future uses for the IoT. Some of these predictions are listed below.

3.3.3.1 Predicting natural disasters

The IoT can be used to predict natural disasters, as these devices can be placed over vast geographical areas [49]. IoT devices make use of sensors and actuators to sense their environment and react upon the information captured by their sensors [49]. IoT devices can therefore sense changes in weather and be used to warn of possible disasters.

3.3.3.2 Monitoring resources

The IoT can be used to monitor resources and detect anomalies such as an abundance or shortage of resources [49], [50]. IoT devices can for example be used to monitor water levels, rainfall and even measure the presence or absence of gas.

3.3.3.3 Smart homes

One of the most significant future applications of the IoT is smart homes. IoT devices can be used in homes to regulate and help reduce unnecessary energy consumption [49], [50]. IoT devices can also be used in homes to increase security and help prevent emergencies, such as automatically turning off gas supplies when detecting gas leaks.

3.3.3.4 Smart medical applications

The IoT is forming part of the modern medical environment and will continue to become a greater part of smart medical applications [49], [50]. IoT devices can be used to monitor patients even though they are not physically at a medical centre. IoT devices make use of their sensors to monitor patients' vital signs and can relay the information to the medical centres for evaluation. If anomalies are detected in the vital signs of patients, medical staff can be alerted, and the patient contacted to take preventative actions.

3.3.3.5 Monitor crops

The agriculture sector can also benefit from the IoT, as the IoT can be used to monitor crops and inform farmers of specific parts of land that might need attention [49]. Farmers can then use limited amounts of resources more effectively for better crop growth.

3.3.3.6 Transportation

Furthermore, the transportation sector can also benefit from the use of the IoT [49], [50]. The IoT can be used to electronically facilitate toll payments, identify traffic congestion automatically, provide self-driving vehicles with alternative routes to reach their destinations on time, regulate pollution generated by vehicles and even detect violations of transportation regulations more efficiently.

3.3.3.7 Smart cities

One of the most documented future applications of the IoT is smart cities. Smart cities can incorporate IoT devices to monitor pollution, regulate energy consumption and even aid in disaster detection and prevention [49], [50].

All the above-mentioned possible future uses of IoT devices can benefit from forensic investigations based on IoT-generated evidence. If the need occurs for investigations into incidents in disasters, resource availability, smart homes, medical environments, agricultural settings, the transportation industry and smart cities, data gathered by IoT devices can prove vital to the reconstruction of the events in the incidents that took place.

3.4 CONCLUSION

This chapter provided background information about the IoT as well as the IoT with regards to DF. This was achieved by conducting a literature review on the IoT, as well as a literature review with a special focus on DF of IoT. The chapter furthermore showed that the IoT is expanding and that more and more uses for the IoT are being developed, thereby indicating that the possible cases where IoT devices form part of forensic investigations are growing exponentially.

This chapter partly addressed research question 1.2.1 (DFR IoT standards processes) by noting some of the various categories of IoT devices that are being utilised in modern society. The literature review conducted in this chapter also showed that new uses for IoT devices are being developed. This chapter furthermore classified various areas where DF of IoT can be useful. This chapter therefore partially addressed objective 1.4.1 (DFR processes, IoT device classification) by classifying various categories of IoT devices where DF can be useful.

IoT devices are starting to take place all around us. They are becoming smarter and more connected to each other and the internet. The IoT enhances modern society in a variety of ways, but with this new technology comes new risks and possibilities for incidents that need to be investigated. This is the fundamental motivation for this dissertation. Forensic investigations are facing a new area of possible investigations and current techniques need to be applied to IoT devices in an effective manner. This leads to the following chapter, which aims to establish the current state-of-the-art DF in IoT.

4 CHAPTER 4: CURRENT STATE OF DIGITAL FORENSICS FOR THE INTERNET OF THINGS

4.1 INTRODUCTION

This chapter aims to evaluate the current state of the art of DF, specifically with regards to IoT, and provide academia with probable future directions for research. A thorough systematic literature review is conducted of topics that relate to the aim of this dissertation. These topics include the IoT, mobile devices (MD), DF, DF of IoT, DF legal implications and proceedings, DF procedures and standards, and DFR. Well-defined search strings are used as part of the methodology to retrieve relevant papers from the body of knowledge. Papers that contribute towards the different topics are marked in a table and then evaluated.

This chapter contributes to all of the research questions raised in this dissertation. It contributes towards research question 1.2.1 (DFR IoT standards processes) and 1.2.2 (DFR IoT devices feasibility) by establishing the state of the art of the IoT, as well as DFR with regards to IoT devices. Research question 1.2.3 (DFR requirements, legal aspects) is partially addressed by incorporating legal aspects into the systematic literature review of the IoT. This is done to determine what the state of the art is with regards to the legal aspects of DF and DFR in the IoT. Objective 1.4.1 (DFR processes, IoT device classification) is also partially addressed by a systematic literature review on the various types of IoT devices that currently exist. Research objective 1.4.2 (DF techniques, DFR model) is also addressed by taking a closer look at existing forensic techniques used with the objective of determining how they can be applied to the IoT.

This chapter furthermore provides a methodology for determining the state of the art of DF for IoT. The methodology is carefully constructed to determine whether DF investigations can be conducted on IoT devices. The state of the art of DF for IoT is formalised in a table (Appendix A), stating which papers in the available body of knowledge contribute to the various topics related to the state of the art of DF for IoT. The various contributions that existing literature makes to topics related to the state of the art of DF for IoT are then indicated in subsections of this chapter. A discussion on the state of the art of DF of IoT based on the systematic literature review conducted in this chapter is provided, concluding the chapter.

In order to conduct the systematic literature review, background about the problems facing IoT, DF and DF of IoT is needed.

4.2 ISSUES WITH DIGITAL FORENSICS

The background provided in Chapter 3 shows what forensic investigations are, why they are necessary and how they are conducted in the real world. DF does, however, face some challenges with regards to general DF issues, mobile DF and DF of IoT. These challenges are discussed in the following subsections.

4.2.1 Issues with digital forensics in general

DF is a growing science and needs to adapt to the changes in the world of IT. The predominant challenges that DF faces is the rapidly increasing number of investigations, size of data stores, and increasing complexity of investigations [16], [48], [51], [52]. Furthermore, DF lacks a universal standard for evidence collection [52]. Existing standards and guidelines for DF evidence gathering are defined in high-level workflow standards rather than precise checklists [52].

DF must often deal with situations where the data is stored on critical systems that cannot be taken offline for investigations, and DF standards often fail to provide procedures to address such challenging data acquisition scenarios [52]. DF also has to deal with devices where flash storage is embedded in the device, making it harder to remove hardware devices for DF imaging [16].

DF also face the so called timelining problem, which entails the challenge of creating timelines of events due to the variety and multitude of devices involved in incidents making use of different time stamps and even different time zones [16], [48]. The timelining problem forms part of normal DF challenges, but becomes a much bigger problem with DF of IoT due to the larger amount of IoT devices that can form part of an investigation.

DF is a relatively new field, which brings with it the lack of trustworthy and established journals to publish DF research. DF is also sometimes limited by legal challenges such as the scope of forensic investigations [16]. DF faces a lack of research and standardisation in a wide variety of operating systems, file systems and user applications [16], [52]. Most of the research in DF is done with a Microsoft Windows background and commonly used

applications such as digital browsers. Another issue with regards to the variety of IoT systems is the implementation of pervasive encryption on some devices that can obstruct forensic investigations [16]. DF also has subsets such as mobile DF, which faces unique challenges. Some of the unique challenges of mobile DF are presented in the following subsection.

4.2.2 Issues with mobile forensics

As a pervasive feature in society, MDs are increasingly becoming part of criminal activities [53]. MDs have seen the biggest increase in the amount of investigations that involve said type of device [51]. MDs have become a prime component in the acquisition of information in modern day digital crime scene investigations [53]. Mobile DF can be defined as the science of retrieving digital evidence from a MD using forensically accepted methods. It takes place over five stages, namely preservation, acquisition, examination, analysis and reporting [53].

There are several challenges with regards to mobile forensics, some of which have been solved and others posing continuous challenges. One of the most prominent challenges of mobile forensics is the large variety of devices [52]-[54]. The process that defines how a mobile phone should be preserved is defined for some devices, but not all. It is therefore not always clear whether or not devices should be turned off when seized, due to uncertainty of what will happen to the data when the device is turned off [16], [53], [54].

MDs furthermore lack standardisation in various areas. Some of the problems with such a vast range of MDs are the different methods of storing data [54]. Some MDs have proprietary operating systems and interfaces, which make the acquisition of data even more challenging for forensic investigators [54]. There exists some uncertainty as to which procedures should be followed to acquire data, as some devices might have PIN code locks that have to be bypassed [53]. Some MDs can even delete all data on the device if the PIN code is entered incorrectly for a certain number of times. The unique codes used for identification of MD users can sometimes be manipulated, leading to improper identification [54]. Examples of such unique codes are the IMEI code of the cell phone and the porting of cell phone numbers [54].

4.2.3 Issues with digital forensics of the Internet of Things

DF is no longer just applicable to personal computers and laptops, but to a much wider variety of devices. A specific subset of devices that DF needs to be applied to, is the IoT [48]. DF with regards to the IoT also faces some unique challenges. The IoT can lead to uncertainty as to where data originated from or where data is stored due to the large quantity of IoT devices and the limits in IoT device storage capacity [48], [55]. Currently there exists a gap in the body of research with regards to an incident response methodology for DF aimed specifically towards IoT devices [55].

4.3 RESEARCH METHODOLOGY FOR THE STATE OF THE ART OF DIGITAL FORENSICS OF THE INTERNET OF THINGS

To determine the current state of research with regards to DF of IoT, the most relevant literature is reviewed. The current body of research is evaluated according to a specific search query in several databases relevant to computer science. The most relevant results are then evaluated according to specified criteria that are defined based on the main topics of current research. The available literature thus guided this chapter to establish the main topics to be evaluated. The search queries of topics are as follows: “Internet of Things” or “internet of things” or “IoT” and “digital forensics” or “digital forensic”.

The various papers are evaluated according to the topics presented in the research methodology. When the paper contributes toward the different topics the paper is evaluated and discussed in order to determine the current state of the art of DF in IoT, as well as identify gaps in the current body of knowledge. The papers that contribute towards the different identified topics are indicated in Table 4-1.

4.4 STATE OF THE ART IN DIGITAL FORENSICS OF THE INTERNET OF THINGS LITERATURE EVALUATION

Table 4-1 presents findings in several ways. All the papers used in the evaluation of current literature are represented in the various rows. The identified topics used for the evaluation of the papers are presented as different columns. If the paper contributes toward one or more of the topics, an X is entered into the appropriate column. The X thus shows that the paper in the current row contributes toward the topic in the column that the X resides in. The final row in the table presents the sum of all the X’s in a specific column to show the number of papers that contributed towards that specific topic.

The first entry in Table 4-1 is used as an example to illustrate the working of the table. The paper “A survey on IoT privacy issues and mitigation techniques”, contributes toward topics 1 and 6. Thus, the paper contributes toward the IoT and processes or standards with regards to the IoT DF processes.

Table 4-1 Paper contributions to the state of the art of DF of IoT

Paper	1 - IoT	2 - MD	3 - DF	4 - DF of IoT	5 - IoT cases	6 - DF processes and standards	7 - DFR
A survey on IoT privacy issues and mitigation techniques [56]	X					X	
Digital forensics on the cheap: Teaching forensics using open-source tools [57]			X		X	X	
Systematically Ensuring the Confidence of Real-Time Home Automation IoT Systems [58]	X						
Current and Future Trends in Mobile Device Forensics: A Survey [59]	X	X	X	X		X	
Embedded Device Forensics and Security [60]	X			X			
Application-Specific Digital Forensics Investigative Model in Internet of Things (IoT) [61]	X		X	X			
Forensic State Acquisition from Internet of Things (FsaIoT): A General Framework and Practical Approach for IoT Forensics Through IoT Device State Acquisition [62]	X	X	X	X			X
Snap Forensics: A Tradeoff Between Ephemeral Intelligence and Persistent Evidence Collection [63]			X	X			X
Standardizing Digital Evidence Storage [64]			X				
Transferring trusted logs across vulnerable paths for digital forensics [65]			X	X			
Harnessing the power of blockchain technology to solve IoT security & privacy issues [66]	X		X	X			X
An Improved Digital Evidence Acquisition Model for the Internet of Things Forensic I: A Theoretical Framework [67]				X		X	
Internet of Things Forensics: Challenges and Approaches [55]	X	X		X	X		X
The Forensics Edge Management System: A Concept and Design [68]	X			X		X	X
Internet of Things (IoT) Digital Forensic Investigation Model: Top-Down Forensic Approach Methodology [69]	X			X			
A Generic Digital Forensic Investigation Framework for Internet of Things (IoT) [47]				X		X	X
Digital Forensic Investigations (DFI) using Internet of Things (IoT) [70]	X						
Challenges of Connecting Edge and Cloud Computing: A Security and Forensic Perspective [71]	X				X		
Cloud-Centric framework for isolating Big Data as Forensic Evidence from IoT Infrastructures [72]	X		X	X		X	
Trust-IoV: A Trustworthy Forensic Investigation Framework for the Internet of Vehicles (IoV) [73]	X			X			X
How an IoT-Enabled “Smart Refrigerator” Can Play a Clandestine Role	X			X		X	X

Paper	1 - IoT	2 - MD	3 - DF	4 - DF of IoT	5 - IoT cases	6 - DF processes and standards	7 - DFR
in Perpetuating Cyber-Crime [74]							
Potential Forensic Analysis of IoT Data [75]	X			X	X	X	X
Enhancing the Security of IOT in Forensics [76]	X					X	
Secure Customer Data over Cloud Forensic Reconstruction [77]	X						
IoT Forensics: Challenges For The Ioa Era [78]	X			X		X	
Internet of Things Forensics: The Need, Process Models, and Open Issues [79]	X			X	X	X	X
Digital forensics: The Missing Piece of the Internet of Things Promise [80]		X		X	X	X	
Digital forensic approaches for Amazon Alexa ecosystem [81]				X			
LEIA: The Live Evidence Information Aggregator [82]			X				
Lightweight Forensics Application: Lightweight Approach to Securing Mobile Devices [83]		X		X			
A Methodology for Privacy-Aware IoT-Forensics [84]				X		X	
The Future of Digital Forensics: Challenges and the Road Ahead [85]	X		X	X		X	
A New Digital Forensics Model of Smart City Automated Vehicles [86]	X		X	X		X	
DROP (Drone Open-source Parser) your drone: Forensic Analysis of the DJI Phantom II [87]	X		X	X	X	X	
IoT Forensic: Bridging the Challenges in Digital Forensic and the Internet of Things [88]				X			X
A review of digital forensic challenges in the Internet of Things (IoT) [89]	X	X		X			
Adding Digital Forensic Readiness as a Security Component to the IoT Domain [90]						X	X
Functional Requirements for Adding Digital Forensic Readiness as a Security Component in IoT Environments [91]						X	X
TOTAL	24	6	13	27	7	19	13

4.5 SUMMARY OF CONTRIBUTIONS MADE BY CURRENT LITERATURE

The different topics are evaluated based on the following specified criteria: the number of papers that address the topic, level of groundbreaking new research and the number of gaps in the current research. The number of papers retrieved, as well as the contribution made by papers towards topics, is evaluated based on throughput and precision. Throughput refers to the amount of papers retrieved when searching a database. Precision refers to the subset of retrieved papers that are relevant to the research done. Usually, high throughput is associated with low precision and a low throughput with high precision.

Table 4-1 presents several interesting insights into the current state of literature and is briefly discussed in the following sections. The different topics are also evaluated based on the above-mentioned criteria in the subsections that follow.

4.5.1 Topic 1 – Internet of Things

The first topic is the IoT. The IoT received the second most contributions (24 papers). Most of the papers note some interesting topics about the IoT, but this is to be expected, as the IoT is still a growing field of research. The searches conducted to gather the papers are focused on the IoT, which also influences the results. The recall on the number of papers returned mostly contained the phrase “Internet of Things” somewhere in the paper. The IoT topic yielded several papers that introduced groundbreaking research, but most of the papers noted the increase in IoT devices, possible future uses for the IoT, as well as possible future research. Likely future uses of IoT devices include smart medicine cabinets, smart cities and home automation.

Not a lot of papers make great advancements towards IoT as a whole, apart from noting that the increase in IoT devices will lead to new challenges. Some of the examples of contributions made towards IoT included new possible encryption algorithms, the indication of the lack of IoT standards and operating systems, and possible future enhancements in IoT security.

4.5.2 Topic 2 – Mobile Devices

The next area of interest is MD. The reason for this is the apparent separation of MD forensics and IoT forensics. Several papers note that MD form part of the IoT, but forensics generally differentiate between MDs and IoT devices due to mobile forensics being a more mature field of research. This is due to the prevalence of established forensic methods, standards and procedures developed specifically for MDs, but not for IoT devices.

The topic of MDs received the lowest amount of contributions from the papers reviewed (6). However, most of the papers that did contribute towards MDs notes that MDs form part of IoT devices, which is a groundbreaking shift in research. However, some papers also mentioned that MD forensics is more mature than IoT forensics. This is a contradiction due to a vast number of papers noting that IoT forensics is still in its infancy, while MD, a subset of IoT devices, are more mature in terms of DF.

The topic of MD forensics also shows an abundance of gaps in available research. In fact, there are very few articles that place MD forensics and IoT forensics in the same category

and introduce a unified solution for mobile and IoT forensics. IoT forensics and mobile forensics generally receive separate treatment in processes and standards. One of the identified gaps in current literature is a DF standard that is applicable to both MDs and IoT devices, as a large amount of papers note that MDs form part of IoT. The development of such standards and procedures will decrease confusion as to which procedures should be used on grey area devices such as smart watches.

4.5.3 Topic 3 – Digital forensics

The topic of DF appeared in almost every paper, but only a limited number of papers (13) contributed towards DF. This could be due to DF being one of the main topics in the search query and a rather established field of research with several accepted standards and procedures. Most of the papers that contribute towards DF, contribute towards DF with regards to computers, which is to be expected. A gap currently exists in the body of knowledge for the enhancement of DF procedures that can cope with the rapid increase in the number of computers, file formats and operating systems.

4.5.4 Topic 4 – Digital forensics of the Internet of Things

The main topic evaluated in the papers is the combination of DF with IoT. The search query yielded several papers, of which some were discarded due to being irrelevant to the topic at hand. In one of the databases the search string resulted in a large throughput with low precision. Other databases used for the retrieval of papers yielded a small throughput with higher precision. This could be due to the large amount of aspects in the search string and the algorithm used by the databases to retrieve papers.

Most of the evaluated papers (27) contribute towards DF of IoT, which is highly beneficial for this dissertation as it determines the state of the art of DF of IoT. This goes to show that the search strings used retrieved accurate results.

To summarise the state of the art of DF of IoT: Due to a lack of standards and procedures in this field a lot of research is needed to ensure the feasibility of DF of IoT. While some papers introduce models, frameworks, standards and procedures for DF of IoT, most of the papers note the gap in current research. The papers introducing innovative solutions for DF of IoT generally note that the frameworks still need testing and evaluation in order to become mature and accepted solutions in DF practice.

There were a limited number of papers that produced groundbreaking research in the field of DF of IoT. The ideas presented usually focused on specific IoT devices or were extremely high-level theoretic designs. The contributions made with regards to DF of IoT is thus lacking in maturity. A lot of research and experiments still need to be done in order to validate and bolster the fragments of research done in the field.

4.5.5 Topic 5 – Internet of Things legal aspects

The retrieved papers contributed extremely little towards this topic. Only 7 papers mentioned IoT devices being used in legal proceedings as sources of digital evidence. There exists a gap in literature as to how to ensure that IoT devices produce forensically sound evidence that can be trusted. A few papers mentioned some challenges or grey areas as to how IoT devices form part of investigations, as well as possible legal challenges such as difficulty in obtaining evidence stored in various locations or at different service providers.

There is a dire need for research with regards to IoT in legal proceedings. While most of the papers mentioned the use of IoT devices in legal proceedings, a formal guideline for such use of IoT digital evidence is still non-existent. The formalisation of such guidelines will increase the likelihood of successful evidence retrieval from IoT devices, which can have several benefits for the legal environment.

4.5.6 Topic 6 – Digital forensic processes and standards

Several papers mentioned established DF procedures, but none made groundbreaking contributions towards IoT DF procedures. However, most of the papers (17) did mention the need for standardised DF procedures specific to IoT devices. There are also some papers that note the validity of existing DF processes and standards for the application of DF of IoT. An example of future research would be the introduction of a tried and tested DF process for IoT.

4.5.7 Topic 7 – Digital forensic readiness

Several papers mention the need for IoT devices to be ready for DF investigations, but only 13 mention the need for proactively preparing IoT devices for DF investigations, indicating another area of possible research. Research in the field of DFR includes the

introduction of the concept in theoretical form and some limited practical examples. There exists an enormous gap in the available literature regarding IoT DFR, as formal guidelines and models still need to be developed, implemented and tested in real-world scenarios.

4.6 DISCUSSION

The evaluation of the current state of literature regarding DF of IoT leads to several questions. Where is research heading? What can be expected in the future of IoT and DF of IoT? What are the actual needs of the industry and academia, and how does it compare with the expected reality?

The current state of DF of IoT research is rather bleak. While many papers note the need for improvement, there is a limited amount of research that aims to enhance the state of DF of IoT. Tried and tested solutions will only be implemented in a few years' time when the body of knowledge becomes saturated with calls for research and solutions. The problem with this is, however, that the adoption of IoT devices is expanding exponentially.

The vastly increasing adaptation of IoT devices will lead to an increase in the amount of data that exist about people monitored by IoT devices, or that make use of IoT devices. This begs the question whether such data can be used and trusted in legal proceedings. Furthermore, if IoT-generated data is seen as useful in investigations, investigators will face an exponential increase in the amount of data to be processed, and investigators will eventually become overwhelmed. Investigators will be in a dire need for faster procedures and solutions that can deal with IoT-generated data.

The problem is that the issues addressed by the industry and academia will be vastly different. The industry will be focused on developing the latest and greatest IoT devices, while academia will be focused on developing innovative technologies to enable IoT devices to be used in more scenarios and environments. Academia will therefore be lacking in the development of standards and procedures with regards to DF of IoT. Academia will only come to light in this regard when investigators can no longer cope with the abundance of data in investigations, leading to unacceptable increases in the time that it takes for investigations into cyber incidents (which already take a lot of time) to be completed.

Academia needs to be informed of the need for the development of concrete, tried and tested solutions for DF of IoT before it becomes unmanageable for forensic investigators to cope with the expanse of IoT-generated data. Academia should direct its attention towards the development of IoT devices that are DF ready. IoT devices should proactively gather data in a manner that will drastically improve forensic procedures, should the need for forensic investigations occur. IoT devices should adhere to globally accepted standards, which require IoT devices to be developed with forensic investigations in mind. This is the ideal picture for the future of DF of IoT. The reality in academia will, however, be different. DFR IoT devices will only be developed as an afterthought as the industry calls for the development of new devices for new use cases, rather than securing devices. This discussion shows that there will be a discrepancy between the need of the industry and the need of academia.

4.7 CONCLUSION

To successfully establish the state of art of DF of IoT a methodology was developed to review and evaluate contributions made by current literature regarding DF of IoT. Several topics related to DF of IoT were identified, and the contributions that the existing literature makes towards the identified topics were addressed and evaluated. The evaluation of these contributions identified several gaps in current literature and showed the need for DF of IoT solutions. DF of IoT solutions should be standardised, trustworthy, efficient and effective to ensure the adoption of said solutions. The methodology processes are repeatable to ensure the validity of the research conducted in this chapter.

The current research on DF of IoT is almost non-existent and this chapter indicated the limited number of papers that aim to address this shortcoming in the DF environment. While a large amount of literature highlights the need for solutions, only a few papers provide them. The provided solutions are mostly high-level solutions that have not yet been thoroughly tried or tested in real-world scenarios.

This chapter contributed towards research question 1.2.1 (DFR IoT standards processes) by taking a closer look at DFR and how DFR standards and approaches are applied to IoT in the current body of knowledge. This chapter also contributed towards research question 1.2.2 (DFR IoT devices feasibility) and research objective 1.4.1 (DFR processes, IoT device classification) by conducting a systematic literature review on various applications

of IoT devices. The systematic literature review also uncovered an interesting finding, which is that even though MD forensics are seen as a subset of IoT forensics, IoT forensics is still in its infancy compared to MD forensics. This chapter contributed towards research question 1.2.3 (DFR requirements, legal aspects) by noting the lack of DFR for IoT and indicating the dire need for further research in this new field. This chapter also took a closer look at research objective 1.4.2 (DF techniques, DFR model) by including DF as a main topic for the systematic literature review. This resulted in a variety of insights with regards to DF of various devices and situations.

A standardised model for DF of IoT that addresses the issues identified in current literature is needed. The model should ensure the trustworthiness of evidence generated by IoT devices, be lightweight and able to handle a large number of devices in heterogeneous environments. It should also incorporate DFR to decrease the amount of time spent retrieving evidence after the occurrence of incidents, as well as the amount of time spent on investigations. The model introduced for IoT should be based on trusted DF and DFR techniques that comply with legal requirements. This leads to the next chapter of this dissertation, which is concerned with the legal aspects of forensic investigations.

5 CHAPTER 5: LEGAL VALIDITY REQUIREMENTS FOR DIGITAL FORENSIC EVIDENCE

5.1 INTRODUCTION

This chapter aims to provide an understanding of the legal environment surrounding cyber incidents and forensic investigations in South Africa (as the author of this dissertation are based in South Africa). The chapter is based on the book *Cyberlaw@SA III: The law of the internet in South Africa* [92] and, based on the Electronic Communications and Transactions Act 25 of 2002 (ECT) [93]. This chapter formalises the requirements for digital evidence from a South African perspective to ensure that the DFR model presented in this dissertation yields valid digital evidence. The evidence generated by the model in this dissertation must adhere to regulations in order to be accepted as evidence in legal proceedings. This chapter therefore presents the current rules and regulations in South Africa with regards to cyber law.

This chapter is based on research question 1.2.3 (DFR requirements, legal aspects) and aims to fully address research objective 1.4.3 (DF legal requirements, DFR IoT legal compliance). It starts with a background on the legal aspects regarding electronic evidence, followed by an overview of the legal relevance of IoT-generated forensic evidence in legal proceedings and a brief discussion of all the above-mentioned aspects.

5.2 BACKGROUND

This section provides background information on the various legal aspects relating to forensic investigations. These aspects are electronic evidence, the admissibility of evidence, data messages, the originality and authenticity of evidence and the details of seizing and searching for digital evidence.

5.2.1 Electronic evidence

The first aspect that needs to be defined is digital (or electronic) evidence. Electronic evidence is when a computer is used to create, manipulate, or store information in digital form, and such evidence is used for forensic purposes [93]. Electronic evidence is located in a vast amount of sources such as computer hard drives, e-mails, e-mail servers, file servers, processing servers, log files, back-ups (even on tape), CDs, DVDs, removable

storage devices such as memory cards, flash disks and floppy disks, and even voicemail [92].

Evidence is categorised into three forms namely: oral, documentary and real. Oral evidence refers to evidence provided by witnesses in the form of testimonies. Documentary evidence is evidence in the form of writing, such as affidavits, notes and sketches. Real evidence is all forms of tangible evidence, including electronic and technological forms.

5.2.2 Admissibility of evidence

The next aspect that needs to be considered is what determines whether evidence is admissible [92]. The first indicator of whether evidence is considered admissible is whether the evidence is relevant. Furthermore, if the evidence is relevant, the evidence can be excluded if it is deemed to be prejudicial.

5.2.3 Data messages

According to the ECT, electronic evidence is based on data and data messages. This definition requires the dissection of the definitions “data” and “data messages”. Data is defined as information represented in any electronic form. Data messages are defined as data that is generated, sent, received or stored by electronic means, which includes voice, voice in an automatic transaction (where one or more of the parties involved in the transaction, is not a natural person), and stored records [94].

Data messages are sometimes not available in their original form, but as a digital copy of the original. However, copies of data messages are still admissible in legal proceedings [92]. The ECT states that data messages are the equivalent of documents, which means that data messages are only admissible based on certain requirements for documents [94]. The requirements for a document to be admissible are as follows: the statements in the document must be relevant, the document must be proven to be authentic, and the original document usually needs to be provided [92], [94]. All three requirements must be met for a document to be admissible in legal proceedings.

5.2.4 Originality and authenticity of evidence

The originality of a document is determined by whether the document has remained complete and unaltered. The integrity of a document is also determined by establishing a chain of custody to determine who had access to the document. The chain of custody can be provided in the form of access control, encrypted access to the documents and entry logs that shows who accessed the documents. A document that is original and unaltered is therefore seen as authentic [92], [94].

The Irish Law Commission provides guidelines that can be used by a court of law to determine the authenticity of electronic evidence [95]. The guidelines are as follows: the computer was functioning as it was supposed to, the program producing the evidence was functioning as it was supposed to, whether or not the storage mediums upon which the evidence was stored has been damaged or tampered with, whether the records were managed in an acceptable manner, whether or not error-checking mechanisms were in place at the time of original evidence generation, and whether the evidence was properly secured from being altered [95]. Electronic evidence is also deemed real evidence when it was generated digitally without any human intervention [92].

After determining that evidence is in fact authentic, the value or weight of the evidence has to be determined. The ECT specifies four factors that are taken into account when determining the weight of electronic evidence [92], [94]. The first factor entails the reliability of the evidence based on how the evidence was generated, stored and communicated. The second factor entails the reliability of the way in which the integrity of the evidence was ensured and enforced. The third factor entails the way in which the originator of the evidence was identified. The final factor is up to the discretion of the legal proceeding, as it entails the possibility of any other relevant factor that can influence the value of the evidence.

5.2.5 Search and seizure

The final important aspect of this chapter is the search for and seizure of electronic evidence. Due to privacy constraints user data cannot just be obtained and investigated; users need to provide consent or lawful authority is needed in order to seize and/or search through user data.

When an applicant in a civil case wants to obtain lawful authority to seize and search for electronic evidence from another respondent, the applicant must establish a need for such lawful authority by providing some basic facts in several forms: the applicant must have a cause for the action against the respondent, the respondent must have some documents that can contain vital evidence in the case at hand, and there must also be a reasonable cause to believe that the evidence might be lost, hidden or destroyed by the time that the case comes to trial [92].

Police officers in a criminal case can only search a person, container, premises or article of a person or occupier of such items in two scenarios. The first scenario is when the police officer has a search warrant to search for potential evidence. The other scenario is when the person or occupier of possible evidence consents to the search [92].

5.3 LEGAL ASPECTS REGARDING DIGITAL FORENSICS OF INTERNET OF THINGS

The legal aspects of forensic investigations are rather well defined in the current body of knowledge and legislation. The ECT is a good example of legislation making provision for forensic investigations. However, legislation is still primarily focused on reactive forensic investigations of electronic evidence and might need to adapt to facilitate the adoption of DFR processes and standards. Even more so, legislation might need to be changed in order to accommodate DFR for IoT devices, as IoT devices are pervasive and can gather a lot of information about users. Users may feel uncomfortable with the level of information gathered about them by their IoT devices in preparation for possible forensic investigations.

When DFR becomes part of the requirements for IoT devices, legislation might need to be changed in order to accommodate such a shift in electronic evidence gathering. Legislation is rather extensive and general enough that it might not be necessary to change legislation with regards to electronic evidence, but rather change legislation with regards to privacy rights. Remote and centralised logging, however, solves the problem of jurisdictional issues due to evidence being stored in a central location [28]. A model for DFR in IoT that makes use of centralised logging therefore solves some of the jurisdictional challenges related to DFR for IoT. The altered privacy right legislation can then be adapted to include proactive evidence gathering. Legislation must define what can

be gathered proactively, what constitutes the release of the gathered evidence for forensic investigators and where such electronic evidence might be proactively stored.

5.4 CONCLUSION

This chapter provided background information on the legal aspects of digital forensic investigations, and showed the relevance of the research done in this dissertation to guide the implementation of a model for DFR in IoT. This chapter ensured that the model developed in this dissertation adheres to the legal requirements for electronic evidence by defining the various legal aspects that determine whether electronic evidence is admissible in legal proceedings. It also partially answered research question 1.2.3 (DFR requirements, legal aspects) and research objective 1.4.3 (DF legal requirements, DFR IoT legal compliance) by taking a closer look at the various aspects that determine the validity of electronic evidence in forensic investigations. These aspects need to be considered in the application of DF and DFR to IoT devices in order to ensure that IoT-generated electronic evidence is acceptable in legal proceedings.

The formulation of requirements for DF of IoT must comply with all the legal aspects mentioned in this chapter. For this reason, the requirements for DF of IoT in the following chapter is based on the underlying knowledge of what makes electronic evidence acceptable in legal proceedings

6 CHAPTER 6: A MODEL FOR DIGITAL FORENSIC READINESS IN THE INTERNET OF THINGS

6.1 INTRODUCTION

This chapter is based on the work done by Kruger and Venter [96] and aims to determine the requirements for IoT forensics. The development of requirements for IoT forensics will make a sound contribution to the current body of knowledge. Its secondary objective is the formulation of a DFR model for the implementation of IoT forensics. The model aims to address the shortfalls of the requirements, as well as incorporate various DF techniques that can be used in IoT environments. The model is presented in theoretical as well as graphical form, to highlight various areas of possible new research. The model is furthermore evaluated to determine the level of contribution it makes to the current body of research.

The model presented in this chapter is based on a systematic literature review of the state of the art of IoT forensics. The systematic literature review provides several techniques that can be used in the DF of IoT devices. The techniques are then formalised into requirements for IoT forensics and addressed through the introduction of a theoretical model for DF of IoT. The model presented in this chapter is also based on legal requirements for evidence to be valid in legal proceedings, which is an attempt to satisfy research question 1.2.3 (DFR requirements, legal aspects). This is therefore a contribution towards objective 1.4.3 (DF legal requirements, DFR IoT legal compliance).

This chapter aims to address research question 1.2.1 (DFR IoT standards processes) by looking at existing DFR techniques and how these techniques are used in various domains. These techniques are then evaluated to determine the possibility of applying said techniques to IoT devices, thus partially addressing objective 1.4.1 (DFR processes, IoT device classification). This chapter also contributes towards objective 1.4.2 (DF techniques, DFR model) by investigating the various DF techniques that can be applied to IoT devices. This chapter also attempts to partially answer research question 1.2.2 (DFR IoT devices feasibility) and objective 1.4.2 (DF techniques, DFR model) by introducing a model for DFR in IoT. This model is developed with a broad scope to attempt to be applicable to IoT devices in general, rather than a specific subset of IoT devices.

The discussion and conclusion of the study evaluates the proposed model and indicates its shortcomings. Furthermore, directions for future research are provided to ensure the advancement of IoT forensics.

6.2 RESEARCH METHODOLOGY

This chapter is based on work done by Kruger and Venter [97], who aimed to determine the current state of research with regards to IoT forensics by evaluating the current body of research according to a specific search query. This search query was used in several databases relevant to computer science.

This chapter in turn aims to address some of the issues raised by Kruger and Venter [97] by developing a theoretical model for IoT forensics. The requirements are formulated based on the results and findings of the systematic literature review done by Kruger and Venter [97]. The methodology used in this chapter is therefore based on a systematic literature review.

The methodology used in this dissertation includes a systematic literature review done by Kruger and Venter [97]. The systematic literature review is an extensive analysis of the current state of the art of IoT. The systematic literature review therefore includes the latest techniques for DF of IoT. The state-of-the-art techniques for IoT forensics are presented in the following section.

6.3 STATE OF THE ART OF INTERNET OF THINGS FORENSIC TECHNIQUES

IoT forensics is still in its infancy. Therefore, it needs to be evaluated and new solutions developed. This section looks at techniques used in IoT forensics, as well as techniques that can possibly be used for IoT forensics. It also expands on the various IoT techniques by analysing the presented techniques to determine why they are valuable for IoT forensics. The presented techniques are then used to formulate several requirements for IoT forensics, which is rather groundbreaking, as requirements for IoT forensics are still non-existent as far as this researcher could determine.

The IoT forensics requirements are then evaluated to present the reader with its various advantages and disadvantages aimed at enhancing the call for further research and

development of IoT forensics requirements. The various requirements and their advantages and disadvantages are then used to propose a theoretical model for IoT forensics.

The development of a theoretical model for IoT forensics is followed by a graphical representation of the proposed model. The model is then discussed to determine how it contributes to the current body of knowledge. It is also evaluated to determine to what extent it addresses the various disadvantages of the requirements used in the model. The feasibility of the implementation of the model and the possibilities of future research are also discussed.

6.3.1 Literature review of Internet of Things techniques

This section is based on the literature review done by Kruger and Venter [97]. The papers in the literature review contribute towards IoT forensics by introducing or evaluating techniques that is, or can be, useful in the IoT forensics. The papers are used to identify possible techniques that can be used for DF of IoT devices. The paper by Kruger and Venter [97] introduces a table with a summary of the contributions made by literature, but due to space constraints the table has not been included in this chapter. The table is added as Appendix A in this dissertation

6.3.2 Exposition of Internet of Things techniques

Table 6-1 summarises the contributions made by the current body of knowledge with regards to IoT forensics techniques and motivates why the chosen technique is useful for IoT forensics. The techniques gathered from the literature review done by Kruger and Venter [97] are formalised into IoT forensics requirements. This section therefore introduces the notion of criteria for a model solution for IoT forensics, and therefore the requirements need to be defined.

The requirements for such a solution aim to guide future research and development in the field of IoT forensics. In addition, it needs to ensure the security, trustworthiness, reliability and verifiability of IoT forensics systems. The implementation of DF systems needs to ensure that evidence generated by IoT devices is suitable for forensic investigations. New sources of evidence are being identified, and the world of DF needs to be proactive in dealing with them, hence the need for the development of formal DF requirements.

The requirements for IoT forensics aim to stipulate aspects that need to be in place in order to ensure the effective implementation of DF procedures with regards to IoT devices. The requirements are then further analysed with regards to the possibility of using these requirements to formulate a model solution for IoT forensics.

Table 6-1 Techniques useful in digital forensics of the Internet of Things

Technique	Discussion of the technique
Open-source tools for IoT forensics	The development of an open-source model solution for IoT forensics is needed to ensure the continuous, rapid expansion of tools for IoT forensics.
Utilisation of lightweight encryption	Data captured and transferred needs to be secured regardless of the limited resources that IoT devices have. Encryption should also be lightweight. It is therefore recommended that a standard be developed that requires all new IoT devices to support lightweight encryption protocols on a standard set of processors.
Unified file format for storing digital evidence and log files	A unified file format ensures interoperability between different devices and processes. Data generated by IoT devices and stored in logs can then be used in a variety of tools for analysis. This ensures the ease of developing new tools and software to work with data generated by IoT devices. To ensure rapid deployment of the new standard, the proposed development of a DFR standard should be limited specifically to IoT.
Automated forensic analysis of data gathered from IoT devices	IoT devices may generate a lot of data, which can prove challenging to forensic investigators if they need to go through all the IoT data after an incident occurred. The use of automated forensics can help ease the burden on forensic investigators and prepare reports about anomalies, which can be more easily digested by users on a regular basis.
Use of logs and state changes as sources of evidence	Logs help provide a deeper insight into system events, with the added benefit of keeping record of the timeline on which events occurred. The incorporation of sensor data ensures that new layers of information are added to forensic evidence. Logs may be compressed and hashed to ensure their integrity, as well as decrease their size.

Technique	Discussion of the technique
Centralised repository for evidence gathered from IoT devices	A centralised repository ensures the aggregation of various sources of evidence, as well as increases the ease of access to the evidence. Data must be encrypted, gathered from devices using an agent on the IoT devices, retained for a reasonable time period and only be accessible through a read-only API.

However, there are additional requirements to ensure that the model is proactive. The model should be developed in a manner that increases the ability to gather trusted evidence for legal proceedings, as well as support reactive forensic investigations by ensuring that forensic investigations can easily gather the data required for the investigation. It should therefore gather and analyse data, as well as report on any anomalies detected.

The problem with a proactive approach is the storage of data prior to a forensic investigation. IoT devices should thus be able to send the data (evidence) to some central repository where it can be accessed in future forensic investigations. This is problematic due to the large volume of data generated by IoT devices. The use of cloud storage can mitigate this problem, as physical environments of IoT devices would not need to maintain hard drives capable of handling such large volumes of data.

A proactive approach is not the only challenge facing the model of DFR for IoT. The techniques identified for possible use in DFR for IoT also has several advantages and disadvantages. The next section of this study analyses the various strengths and weaknesses of the requirements for DFR in IoT.

6.4 ADVANTAGES AND DISADVANTAGES OF THE VARIOUS REQUIREMENTS

The IoT forensics requirements are based on various potential IoT forensic techniques. However, some of the techniques are still new and complex to implement. The requirements therefore hold some advantages for the world of IoT forensics, but can also pose some challenges and drawbacks. In Table 6-2 the various requirements are evaluated based on their respective advantages and disadvantages. They are numbered to save space when referring to them further in the article. An example is that Lightweight encryption is numbered as technique number 1 (shown as #1 in Table 6-2).

Table 6-2 Advantages and disadvantages of digital forensics of Internet of Things techniques

Proposed technique	Advantages	Disadvantages
#1 Lightweight encryption	<ul style="list-style-type: none"> • Resource efficient • Increases data integrity • Reliable data transfer over insecure networks 	<ul style="list-style-type: none"> • Resource intensive on devices with limited processing power • Complex to develop
#2 Unified file format for storing digital evidence and log files	<ul style="list-style-type: none"> • Easy development of new software and tools that use standard file format • Interchangeability of data between tools • Standard procedures and best practises for retrieving evidence 	<ul style="list-style-type: none"> • Complex to develop standard • Time-consuming to develop standard • Time-consuming to get all manufacturers to implement new standard
#3 Automated forensic analysis	<ul style="list-style-type: none"> • Decrease in the duration of forensic investigations • Real-time monitoring and alerts of events in order to highlight anomalies 	<ul style="list-style-type: none"> • Hardware and software capable of conducting near real-time monitoring and analysis of data • Cost implications of required hardware
#4 Use of logs and state changes as sources of evidence	<ul style="list-style-type: none"> • Insight into system and physical events • Enhanced evidence due to timelining of system and physical events 	<ul style="list-style-type: none"> • Need for storage facilities capable of handling large amounts of data
#5 Centralised repository for evidence	<ul style="list-style-type: none"> • Aggregation of all data • Central secure point • Read-only API ensures integrity of evidence • Hashed evidence • Compressed evidence 	<ul style="list-style-type: none"> • Single point of possible failure • Cost implications • Requirement for large storage capabilities • IoT devices must register to central repository

The evaluation of the IoT forensics requirements shows that each requirement has both benefits and challenges. The various challenges need to be addressed to ensure the introduction of models compliant with the IoT forensics requirements. The next section introduces a theoretical model for DFR in IoT. The model presented in the next chapter take the various advantages and disadvantages of the techniques listed in this section into consideration.

6.5 THEORETICAL DEVELOPMENT OF AN INTERNET OF THINGS DIGITAL FORENSIC READINESS MODEL BASED ON THE REQUIREMENTS OF INTERNET OF THINGS FORENSICS

This section introduces a model solution for IoT forensics requirements. The model serves as a base solution to the challenges of the different techniques, as well as a reference solution for the lack of IoT forensics devices. The possibility of solutions for the various challenges of the requirements exacerbates the need for the development of models that address the defined requirements.

The developed model not only takes the disadvantages into consideration, but also aims to mitigate these disadvantages to ensure a highly secure and efficient model that can be used in the industry.

Several articles on IoT forensics highlight proposed techniques for ensuring DF of IoT devices. The proposed model in this chapter is based on the combination of previously proposed solutions, as these solutions have been proposed independently and with different motivations. The model includes techniques based on their merit, impact, benefits and feasibility, and not on how many times they have appeared in papers. Since the field of IoT forensics is rather new, with a limited number of proposed solutions, new ideas are also considered for this model.

Based on the information gathered by Kruger and Venter [97], the model should make use of lightweight encryption to secure the evidence gathered from the IoT devices and ensure the efficient use of limited processing power on IoT devices. The model should furthermore introduce or make use of a unified file format for gathering, storing and presenting forensic results, since no unified standard file format for logs and forensic evidence currently exists. There is an ongoing need for further development of a unified standard for forensic logs.

The model should thus be adaptable to make use of a unified file format, should one be developed in the future. The model should also facilitate automated forensic analysis, meaning it should be able to analyse data automatically and report anomalies in IoT device data. Sources for forensic evidence should include the use of logs and state change logs, as the different states can provide interesting insights into the physical environment that the devices form part of. System logs furthermore provide information into the digital environment, such as packets, requests and data transfer between various IoT devices.

The final aspect of the model is the collection and analysis of evidence. As mentioned earlier, IoT device data should be encrypted and the information sent to a central device for safekeeping and analysis. This central device can form part of the IoT network or be in the cloud. Several papers noted the validity of using the cloud for IoT forensic investigations. The information should be gathered from the IoT devices by software (also referred to as an “agent”) that provides a standardised API with read-only access for the cloud/central evidence device. The centralised device should then analyse the data for anomalies to ensure reactive responses to intrusions and incidents. The centralised device should also ensure safekeeping of evidence using hash functions. The evidence should only be accessible through a read-only API.

The rest of this chapter is structured as follows: firstly, a graphical representation of the model is provided. The model is furthermore discussed based on the graphical representation of the model. This chapter then provides several activity diagrams to show the working of the model. This chapter also provides an overview of how the implemented model mitigates some of the disadvantages of the techniques used in the model. This chapter then presents a conclusive discussion of the model developed.

6.5.1 Graphical representation of model solution

This section provides an overview of the model developed for DFR for IoT. This model is based on all of the requirements in the previous section.

The model is presented in graphical form in Figure 6-1. The model is discussed based on where green letters are placed on key aspects of the model.

The model for DFR for IoT requires one or more IoT devices to connect to a centralised repository to proactively store forensic evidence generated by IoT devices. An IoT device is visualised in Figure 6-1 with the letter “A”. This section discusses the IoT device in singular form, but the same aspects apply to any number of IoT devices that form part of the model for DFR for IoT.

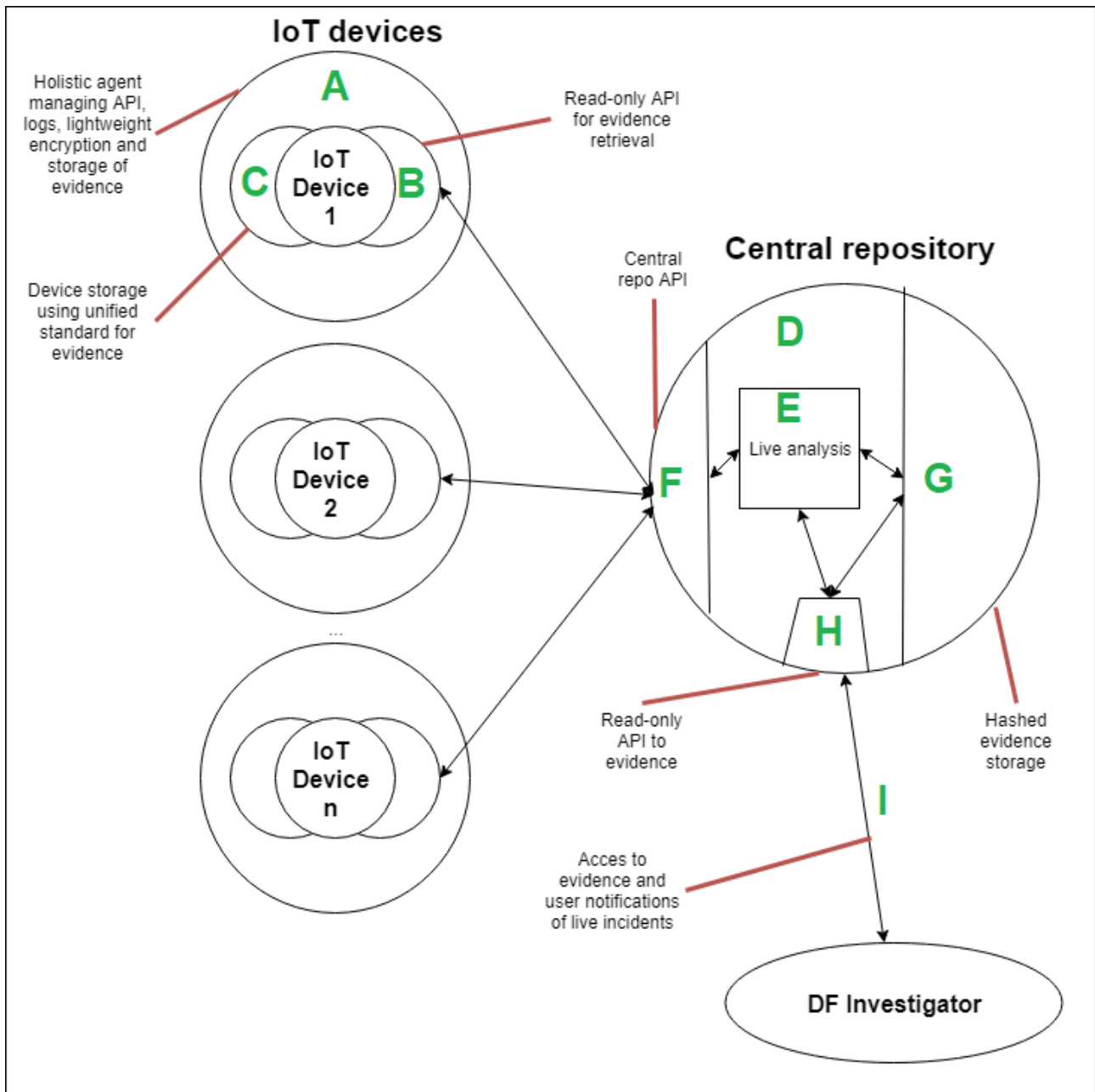


Figure 6-1 Digital forensic readiness for Internet of Things model

The IoT device has agent software installed which proactively gathers forensic evidence. The evidence is stored in a unified file format. The unified file format exists on the IoT device and is represented by the letter “C”. The IoT device furthermore has a read-only

API which enables the secure transfer of evidence to the centralised repository. This ensures that external services can not alter the evidence located on the IoT device. The read-only API is represented by the letter “B” in Figure 6-1.

The model furthermore relies on a centralised repository. All IoT devices with agent software installed connect to the centralised repository. The centralised repository is represented in Figure 6-1 with the letter “D”. The centralised repository gathers the evidence generated by the various IoT devices. The evidence enters the centralised repository through an API. The IoT-agent-facing API is shown in Figure 6-1 with the letter “F”. Once the evidence enters the centralised repository it undergoes live analysis. The live analysis of the evidence aims to identify abnormalities in near real-time. The live analysis of the central repository is shown in Figure 6-1 with the letter “E”.

The evidence gathered in the centralised repository is stored in a secure read-only access database on the centralised repository, along with the hashes of the evidence gathered. The hashed evidence is shown in Figure 6-1 with the letter “G”. The centralised repository offers access to the gathered evidence in a secure manner that ensures the evidence cannot be changed. This is done through a read-only API represented by the letter “H” in Figure 6-1. The read-only API enables investigators the gain access to the evidence gathered from all of the IoT devices. The API also enables notifications of abnormalities detected by the live analysis performed by the centralised repository. The read-only access is represented by the letter “I” in Figure 6-1.

The discussion of the model in graphical representation of DFR for IoT can furthermore be discussed based on the sequence in which the various aspects of the model work together. This is done in the next section when the model is discussed by making use of activity diagrams.

6.5.2 Activity diagrams for the model

The activity diagrams indicate how evidence is gathered from IoT devices. The development of the activity diagrams highlighted the need for further research about whether a push- or pull-based model would be best suited for DFR in IoT. A push-based model (Figure 6-2) is when the IoT devices contact the central repository when new

evidence is available. The centralised repository then retrieves the evidence from the IoT device(s) that indicated the presence of new evidence.

The pull-based model (Figure 6-3) is based on the notion that the centralised repository contacts the various IoT devices at certain time intervals to determine whether the devices have new evidence to be retrieved. The centralised repository then retrieves evidence from the devices that have new evidence. One of the possible disadvantages of this method is that IoT devices might need to store evidence on their local storage for certain periods, which can be challenging due to limited on-board storage space. Another possible disadvantage of this method is the amount of network traffic generated by the centralised repository when there is a large amount of IoT devices in the network and all the IoT devices are contacted at once.

The development of the activity diagrams also introduced an interesting scenario. When users integrate new IoT devices into their network, they tend to be more interested in getting the device to work than ensuring that the devices conform to necessary security requirements, let alone DFR requirements. Users might therefore not register their IoT devices with a centralised repository, potentially leading to IoT device forensic evidence not being trustworthy.

The entire purpose of this model is to ensure that IoT device evidence is trustworthy due to IoT devices being DF-ready. It is worth mentioning that as part of the model, users should be required to register their IoT devices at the centralised repository before the IoT device can serve its purpose in the environment. IoT devices should thus not be able to function if they are not registered with a centralised repository. The activity diagram shows how the device continually checks whether it is registered with a centralised repository for evidence gathering.

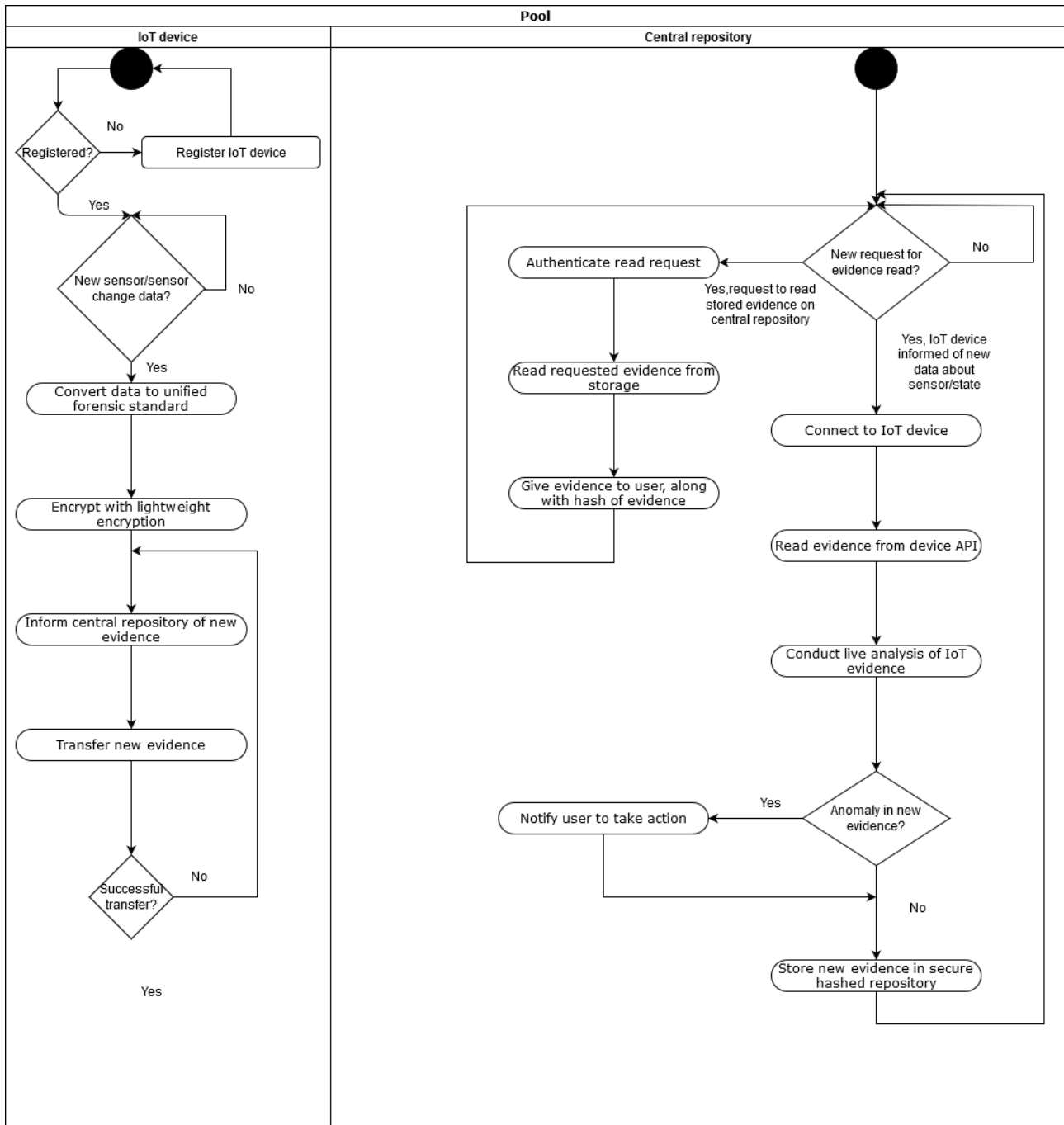


Figure 6-2 Push-based model

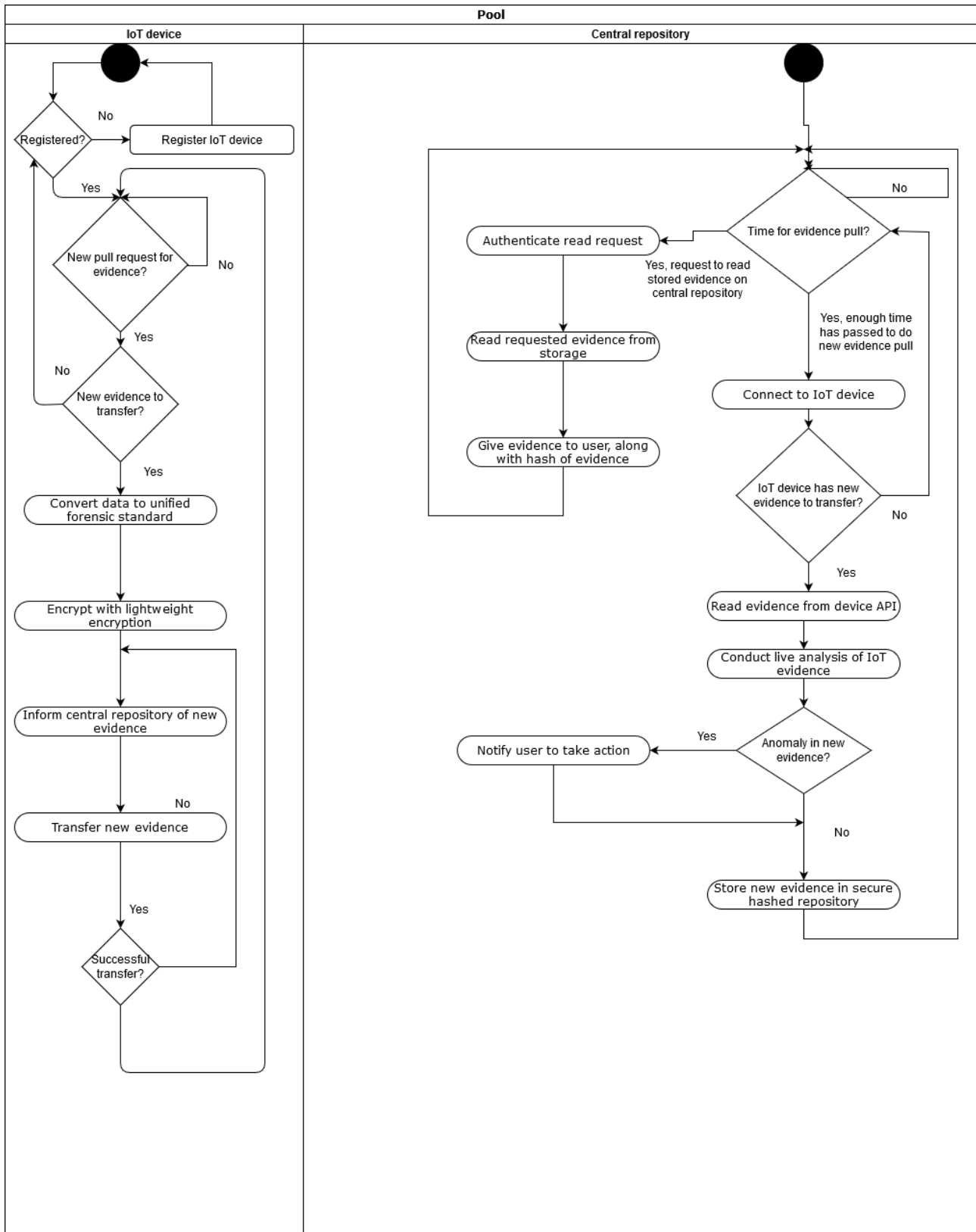


Figure 6-3 Pull-based model

6.5.3 Mitigating the disadvantages of Internet of Things techniques

This section provides a discussion of how the disadvantages (listed in Table 6-2) that are not directly addressed in the model solution can be addressed. However, due to space constraints and the need for further research, not all disadvantages are addressed. Table 6-3 indicates where future research is still needed with regards to the model. The various techniques are not written in full, but are presented numerically. Please refer to Table 6-2 in order to determine which number represents which technique.

Table 6-3 Solutions to disadvantages of digital forensics of Internet of Things techniques

#	Disadvantage	Solution to disadvantage
1	Resource-intensive on devices with limited processing power	Not addressed in the model.
1	Complexity in developing software that uses lightweight encryption	Not addressed in the model.
2	Complexity of developing a new standard file format	The software implemented on the agent is developed in a modular format to ensure that any changes to the underlying storage format can be implemented easily.
2	Time-consuming to develop a new standard	Not addressed in the model. The model only focuses on DF of IoT, which can lead to faster development of a unified standard as only IoT devices should form part of the standard.
2	Time-consuming to get all manufacturers to implement new standard	The modularity of the agent software ensures that different types of storage formats are supported. The idea is that manufacturers should work toward a unified standard and not further disparity.
3	Need for hardware and software capable of conducting near real-time monitoring and analysis of data	Using the cloud as the central repository enables the use of high-performance hardware that can conduct the live processing.
3	Cost implications due to need for extra processing hardware	This disadvantage is not directly addressed in the model.
4	Need for storage facilities	Using the cloud enables long-term, high-volume,

#	Disadvantage	Solution to disadvantage
	capable of handling large amounts of data generated by IoT devices	and secure storage.
5	Single point of possible failure	The possible use of cloud services to enable the central repository.
5	Cost-implications to cloud computing or hosted system	This disadvantage is not directly addressed in the model.
5	Large amount of network traffic to central repository	The use of either push- or pull-methods of transferring data to the central repository. This can be changed in the various implementation environments in which the IoT devices reside based on relative efficiency.
5	Requirement for large hard drives capable of storing evidence for long periods of time	The possibility of using the cloud as the central repository enables the use of high-volume storage hardware that ensures storage of large amounts of digital evidence for long periods of time.
5	Users may not register to a central repository	Users must register a new IoT device to the IoT central repository for the IoT device to be functional.

This section shows how some of the disadvantages of the requirements of DFR for IoT can be addressed. The mitigation of several disadvantages in DFR for IoT calls for a discussion of the potential of the new model for DFR in IoT. The next section provides a discussion of the new model presented in this chapter.

6.6 DISCUSSION

This section not only analyses the model presented in this chapter, but also takes a critical look at how the model uses the IoT forensics requirements to solve some of the challenges facing IoT forensics. Several key IoT forensics techniques, as well as techniques for possible use in IoT forensics, were identified.

There are several possible challenges to the systematic literature review research approach followed in this chapter. One of them is that a systematic literature review can be

very specific in focus due to the use of a specific search string for the retrieval of papers. The search string can be too narrow or too rigid, possibly leaving out some techniques due to not appearing in papers that were retrieved using the specified search string.

Another challenge to this approach is that new papers are constantly added to the body of knowledge. At the time of writing, there might be new techniques presented in newer papers. However, this chapter includes as many techniques as possible by not only listing techniques based on the number of times that they were introduced in various papers, but also their feasibility (regardless of how frequently they are mentioned in papers).

This chapter uses the identified techniques to analyse the possibility of combining the various proposed techniques into one set of requirements. Further research is needed to develop a formal tried-and-tested set of requirements for DF of IoT. This study serves as basis for a new set of standards for DF of IoT.

The development of models based on the requirements presented in this chapter ensures the advancement of security and forensic soundness of evidence generated by IoT devices. The techniques used in the requirements mostly address the integrity and confidentiality of the systems. Availability is the one aspect of computer security that is not directly addressed in the requirements. The addition of these techniques to resource-constrained devices can possibly increase the likelihood of downtime or lack of availability from the IoT devices. The IoT devices can become flooded with computing requests to handle the added overhead of security techniques, or become prone to denial-of-service attacks. Denial-of-service attacks would be easier to achieve due to the added overhead of securing common events. Attackers would merely have to increase the amount of times common events occur to increase the amount of computing that IoT devices need to perform. Further research and development should address this issue and assure that resource-constrained devices have a means of ensuring that the added overhead does not cause the device to be overloaded with processing requests.

One of the mentioned requirements is the need for a unified file format for digital forensic evidence. This requirement is mentioned numerous times in various papers in the systematic literature review and the need for the development of such a file format is expressed by various researchers. The use of such a format in IoT forensics is also

important due to the large variety of devices that can form part of an IoT system. Each manufacturer using their own file formats and techniques increases the complexity of IoT forensics. The vast expanse of IoT devices and the increase of scenarios where IoT devices are useful increase the urgency for the development of such file formats. A unified digital evidence file format needs to be developed sooner rather than later, before it is simply too complex to develop such a standard.

This chapter furthermore identifies several advantages and disadvantages of the various techniques used for the requirements of DF of IoT. The model aims to address several of the disadvantages, however not all disadvantages can be addressed in this model. This study therefore calls for the further development of a tried-and-tested model solution.

This chapter also introduces a model solution for DF of IoT by making use of the developed requirements and incorporating the identified techniques. The model illustrates the working of the various techniques and where they are applied in IoT systems. The proposed model serves as a base for the advancement of DF of IoT models and calls for the addition of new techniques and solutions, as well as the testing and implementation of the model.

One of the limitations of this chapter is that the model is only theoretical and has not yet been implemented and tested in a real-world environment. However, an activity diagram of the model is presented. The activity diagram shows the logic of the model, as well as how the various techniques and components interact with one another and how IoT devices interact with the centralised repository. The activity diagram serves as a high-level overview of the model and illustrates that the model incorporates all the listed requirements for the model solution.

This chapter furthermore gives a brief discussion of how the model mitigates some of the disadvantages of the various techniques presented earlier in the chapter. Most of the addressed problems are solved using cloud service for the central repository, hence the recommendation that the cloud, rather than a local machine, be used for the storage and processing of the evidence. There are, however, several issues that the model does not solve and which require further research. Most of these problems concern the techniques themselves. As mentioned earlier, some of the techniques are quite new and require

further research. The advancement of the various techniques will furthermore increase the effectiveness and efficiency of the proposed model.

This study has a major drawback due to not focusing on privacy issues. One of the main problems of DF is that it aims to collect and analyse as much information about suspects as possible, while privacy aims to reveal as little information about people as possible. This chapter therefore recognises the privacy concerns that might become apparent when analysing the model. Therefore, it calls for further research as to how to ensure that data can always be kept private, especially when using the cloud as a centralised repository. This can be mitigated using privacy agreements and service-level agreements.

6.7 CONCLUSION

This chapter aimed to determine techniques that can be used to ensure the effective and trustworthy gathering of IoT device digital forensic data that can be used in legal proceedings. The systematic literature review on which this chapter is based, is used to extract various possible techniques that can be used to induce DF for IoT. The various identified techniques were evaluated and formulated into IoT forensics requirements. The requirements were furthermore analysed to determine various advantages and disadvantages of the techniques used to formulate IoT forensics requirements. This contributed towards research question 1.2.1 (DFR IoT standards processes) and objective 1.4.1 (DFR processes, IoT device classification).

This chapter introduced a new IoT forensics model to ensure the DFR for IoT in the near future. It will ensure that IoT devices are more secure and trustworthy in providing digital forensic investigators with IoT-generated evidence. This chapter therefore addressed objective 1.4.2 (DF techniques, DFR model) by establishing a model for DFR in IoT. The model presented in this chapter is also based on legal requirements for evidence to be valid in legal proceedings, which contributed towards research question 1.2.3 (DFR requirements, legal aspects). This is also a contribution towards objective 1.4.3 (DF legal requirements, DFR IoT legal compliance).

There is currently a need for a unified file format for the storage of IoT digital evidence, as well as an implemented and tried-and-tested prototype of the model presented in this chapter. Lightweight encryption needs to be enhanced to operate optimally on resource-

constrained IoT devices. Standards need to be developed to require IoT manufacturers to implement the proposed model, which in turn requires IoT devices to be registered to a centralised evidence-gathering repository. This chapter also noted that a lot of research is still needed to realise the DFR for IoT devices.

The model for DFR in IoT introduced in this chapter is a theoretical model and still needs to be tested. This leads to the following chapter in this dissertation, which is the practical implementation of the model for DFR in IoT.

7 CHAPTER 7: A PROTOTYPE FOR DIGITAL FORENSIC READINESS IN THE INTERNET OF THINGS

7.1 INTRODUCTION

This chapter is based on the requirements formulated for DF of IoT in Chapter 6. The aim of this chapter is to show the feasibility of the model for DFR in IoT, to be able to incorporate the requirements into a DFR standard for IoT. In order to show that the model can work in a real-life environment, the requirements need to be implemented in a prototype and then tested, which is exactly what this chapter entails.

This chapter starts with a reiteration of the various requirements for DF of IoT and various possible approaches to implementing these requirements. The chapter then shows the chosen implementation and how it incorporates the various requirements, followed by an in-depth overview of how the solution is implemented and how it works. The chapter concludes with a discussion of whether the requirements seem suitable for use in a real-world environment.

This chapter addresses research question 1.2.1 (DFR IoT standards processes) by using forensic techniques from various DF standards and applying them to the IoT. This chapter also partially addresses research question 1.2.2 (DFR IoT devices feasibility) and 1.2.3 (DFR requirements, legal aspects) by developing a prototype for forensic investigations on IoT devices and implementing a prototype to determine whether or not DF can be applied proactively to IoT devices, resulting in DFR for IoT. Furthermore, this chapter aims to meet objective 1.4.1 (DFR processes, IoT device classification) by determining whether DFR processes can be applied to a subset of IoT devices. However, the main objective that this chapter addresses is objective 1.4.4 (Proof of concept, DFR IoT prototype), which entails the design, development and implementation of a proof of concept prototype for DFR of the IoT.

7.2 REITERATION OF THE VARIOUS REQUIREMENTS FOR DIGITAL FORENSICS OF INTERNET OF THINGS

This section provides a brief overview of the various requirements for DF of IoT. Since the requirements were discussed in detail in the previous chapter of this dissertation, they are merely listed here for quick reference.

1. Open-source tools
2. Lightweight encryption
3. Automated forensic analysis
4. Use of logs and state changes for sources of evidence
5. Centralised repository for evidence

In order to be viable, the solution implemented should address the requirements listed above.

7.3 POSSIBLE APPROACHES TO IMPLEMENTING THE REQUIREMENTS FOR DIGITAL FORENSICS OF THE INTERNET OF THINGS

There are several possible approaches to implementing the requirements for DF of IoT of which two are considered in this dissertation. One is implementing an entire open-source system from scratch, and the other is using already available open-source projects and adopting them to suit the needs of this research.

The first possible solution entails implementing an entirely new open-source solution for DF of IoT. The solution then incorporates all the requirements for DF of IoT from the start of development, which ensures that all the specified requirements are met. The development of an entirely new solution is beneficial to ensuring that all requirements are met and that forensics is a prime aspect of the solution. Forensics is therefore not an afterthought to an IoT system, but forms part of the core of the solution. However, there are some drawbacks to this option. The development of an entirely new solution can be time-consuming and cause incompatibility across devices due to a limited amount of developers when compared to existing open-source projects where an entire community of developers work together on a solution. The first option can therefore result in unnecessarily reinventing the wheel.

The second possible solution entails the customisation of already available open-source DF tools. This option concerns itself with identifying open-source tools that can be used and customised to solve a problem not yet addressed. In order to address the requirements of DF for IoT, the tools identified must address the requirements for DF of

IoT or be able to be adapted accordingly. The advantage of this approach is that the tools have already been tried and tested by a community of developers.

The approach followed in this study is the use of existing open-source tools that can be customised to solve the problems identified in this dissertation. Therefore, the tools must address the requirements for DF of IoT or be customisable in order to address the requirements listed. The next section selects a possible tool that can be used to address the requirements for a model in DFR for IoT.

7.4 SOLUTION FOR THE REQUIREMENTS OF DIGITAL FORENSIC FOR THE INTERNET OF THINGS

This section shows the working of a solution for the requirements of DF for IoT and introduces open-source tools that can be customised for use in this study. This section then selects one tool for customisation and shows how the customised tool addresses the requirements for DF of IoT.

7.4.1 Possible open-source digital forensics tools

This subsection provides a list of possible open-source DF tools that are already available for use in forensic investigations and can possibly be adapted for use in IoT forensics. The tools are listed in no specific order. The list is based on alternatives to Google Rapid Response (GRR) [98], which provides an in-depth overview of various possible tools to be used in this study.

7.4.1.1 MIG (Mozilla InvestiGator)

This is a remote live forensics tool that allows forensic investigations on remote systems. The tool was developed by the Mozilla Foundation and contributors in the open-source community. The tool is command-line based and supports Linux, macOS and Windows [98]-[100].

7.4.1.2 Bitscout

Bitscout, another remote forensics tool, enables an investigator to obtain a disk image clone. The project is based on Linux and is also open-source. However, the tool is maintained by a limited number of developers. The tool also does not enable any remote changes to the system such as querying live memory dumps [98], [101], [102].

7.4.1.3 FIR (Fast Incident Response)

FIR is a tool for incident response and security monitoring. It has a web interface and works on Linux. The tool has a few developers in the open-source community, but not as many as some of the other tools. The tool does not support an API yet [98], [103], [104].

7.4.1.4 TheHive

TheHive is another open-source tool that aims to deal with security incidents. It has an API and web interface. While it has several developers in the open-source community, it is not backed by a major company. The tool supports Linux, but is rather resource-intensive [98], [105], [106].

7.4.1.5 OSSEC

OSSEC is an open-source tool that enables cross-platform monitoring, but seems to be more focused on monitoring than DF. The tool works on Linux, MacOS, Windows, OpenBSD and Solaris [98], [107], [108].

7.4.1.6 GRR

GRR is an open-source tool aimed at supporting DF and investigations. The tool was developed to be fast and scalable. It has a web interface and an API. While it has large support from the open-source community, it is also maintained by Google, as it was started as a Google project. GRR works on Linux, macOS and Windows [98], [109], [110].

This section introduces a variety of possible DF tools that can be customised in this dissertation to facilitate DFR for IoT. After the introduction of the various possible tools, a tool has to be selected for customisation. The next section identifies a tool that is suitable for customisation to satisfy the model presented for DFR in IoT.

7.4.2 Selection of a tool for digital forensics of the Internet of Things

This section aims to provide a brief motivation for the selected tool. The selected tool should address as many of the requirements for DF of IoT as possible. If the tool does not support all the requirements, it must be customisable in order to meet them.

After comparing the various aspects of the possible tools listed in the previous section, the author decided to make use of GRR. The GRR acronym is therefore a recursive acronym. The background information on GRR is based on the official GRR documentation [110], as

sources about the working of GRR are limited. GRR is the tool of choice for this study, due to satisfying most of the requirements for DF of IoT. The first requirement is that the tool must be open-source. The tool must make use of lightweight encryption. The tool must support automated forensic analysis. The tool must make use of logs and state changes as sources for forensic evidence. The final requirement is that the tool must support a centralised evidence repository.

GRR is an open-source project backed by several community developers, as well as a development team within Google itself. The tool is used in Google's data centres for live forensics and monitoring. It has proven to be highly scalable and able to handle an enormous amount of client agents as machines that are monitored [110]. GRR furthermore provides an easy web interface as well as an API. The presence of an API allows for easy extension without having to change core components of the tool.

GRR is an ongoing project, which ensures that the tool is maintained and kept to a high standard. Therefore, the tool is rather robust and reliable. The tool also supports various platforms and architectures for investigations, which is a highly desired quality within the context of the highly heterogeneous IoT environment.

GRR is a modular framework with various components that enables DF investigations and monitoring of devices. GRR consists of two main components, namely the GRR server and the GRR clients. The GRR server consists of several components that work together to form a web-based GUI server with an API that enables investigators to perform actions on and collect data from the clients. The GRR server can conduct various incident response and forensic tasks. It is designed to support artefact collection on a large number of machines and present the results in different ways. The GRR server is also developed with asynchronous tasks in mind, and investigations can therefore be scheduled and performed when needed.

The GRR client refers to an agent deployed to systems or devices to be investigated. The GRR client frequently polls the GRR server for actions to perform. These actions can be a variety of things, such as retrieving memory details or files. The GRR clients run at root level, which enables raw file system access to the various devices and systems. The GRR

client can also be used for monitoring the systems and devices while maintaining imposed resource limits, which ensures that the target systems and devices perform optimally.

The GRR client uses POST requests to communicate with the server and all communication between the GRR server and client is encrypted to ensure that malicious third parties cannot access or change the data or commands passed between the client and server. It is important to note that the GRR clients run with root privilege on all the devices. This means that anyone with access to the GRR server can access the devices on root level. The GRR server should therefore be installed in a secure environment and strict access control be implemented.

GRR is a platform that allows live remote forensics of a wide variety of systems in a forensically sound manner. In addition, it is also adaptable to suit the needs of IoT forensics. All the above-mentioned aspects contribute towards the selection of GRR as tool best suited to this study.

To summarise, GRR meets the requirements For DFR in IoT in various ways:

1. Open-source tools: GRR is open-source.
2. Lightweight encryption: GRR supports lightweight encryption as the traffic between the clients and the GRR server are not too heavily encrypted.
3. Automated forensic analysis: GRR does not natively support automated forensic analysis as hunts can be developed to look for certain artefacts. A hunt is a forensic investigation conducted across multiple host machines. Hunts can also be scheduled to run periodically, which can facilitate automated forensic analysis.
4. Use of logs and state changes: Hunts can be developed in GRR to collect logs of any nature.
5. Centralised repository: GRR stores all collected evidence in a repository along with the hashes of the evidence from the various snapshots of evidence retrieved over time.

7.4.3 Setting up the GRR environment for Internet of Things forensics

The installation of the GRR platform can be done in various ways. These include installing from source, package repositories or Docker images. The GRR server includes prebuilt

GRR client packages for various platforms, but the GRR clients can also be compiled separately.

An installation from source code is used in this study. Installing from source increases the ease of customising the GRR platform to work on IoT devices. The environment is focused on both the simulation of IoT devices and actual IoT devices. In order to facilitate this, the GRR server must be easily accessible by both simulated and actual IoT devices.

Simulated IoT devices enable rapid deployment of multiple devices to show the scalability of the platform. IoT devices are simulated in the form of virtual machines. The virtual machines host small operating systems and are small in storage size to replicate real-world, resource-constrained IoT devices. The virtual machines are also deployed on various physical machines located on different networks in various physical locations to further simulate the heterogeneous environment of actual IoT devices.

Physical IoT devices show the actual implementation of GRR clients and that they run on the IoT devices in an efficient manner. The physical implementation of the GRR client on physical devices furthermore demonstrates the viability and trustworthiness of the GRR platform. The physical IoT device used in this dissertation is a Raspberry Pi Model 3B. The Raspberry Pi is a small IoT device that runs on ARM architecture and enables developers to easily connect hardware to a system using general-purpose input/output (GPIO) pins. The Raspberry Pi also makes use of micro SD cards for storage and has a 1GHz quad-core ARM processor with 1GB of RAM. It supports various operating systems, which makes it ideal for this study, as the GRR client can be built to work on an operating system that is supported by the Raspberry Pi.

In this dissertation the Raspberry Pi and virtual machines are located on various networks in various physical locations. The separation of the devices simulates IoT devices in a real-world scenario. The GRR server should be able to interact with all the devices and vice versa. In order to enable the GRR server and various GRR clients to connect to one another, a server with a public IP is needed. The GRR server is installed on a DigitalOcean droplet located in Europe. A droplet is DigitalOcean's name for a virtual machine (VM). It has a public IP and can be accessed from various physical locations and different networks. The droplet is based on the Ubuntu Server 18.04 MySQL easy

installation, so MySQL comes installed with the droplet. The droplet has two virtual CPU's, 4GB of RAM and 80GB of disk space. The DigitalOcean droplet is based on an Intel(R) Xeon(R) CPU E5-2650L v3 @ 1.80GHz processor.

The installation of the GRR server is fairly easy, as it can be installed from a release DEB package on the DigitalOcean droplet. Linux DEB is a file format for Debian-based Linux software packages. The GRR server is therefore merely installed from a software package. Do note that the GRR server installed from a release DEB on the droplet is not customised. This means that the GRR server is not customised to facilitate the implementation of the model for DFR in IoT. However, the GRR clients installed on the Raspberry Pi are installed from source, as the code is customised for the GRR clients. This means that the GRR clients used on the Raspberry Pi need to be customised in order to implement the model developed for DFR in IoT.

7.4.4 Install GRR server on a server with a public IP address

The method of installing GRR from a release DEB is explained in the following steps. Please note that the figures in this section are based on an installation done on a local virtual machine in order to hide sensitive login details from the actual DigitalOcean droplet with a public IP address. This means that the installation of the actual GRR server used in the implementation of the model for DFR in IoT is not shown. The installation of the actual GRR server is the same as in the figures below, except for the actual GRR server having a different IP address. The figures of the installation done on the local machine is merely provided to hide sensitive information of the actual GRR server.

1. Install Apache2 on the server. Apache2 is used for the web server that hosts the GRR web-based GUI.
2. Install MySQL on the server. MySQL is used as the backend database for the GRR framework.
3. Create specific login details for the GRR server so as to ensure that the database is secured and can only be accessed via the GRR server. MySQL also has to be configured for usage by the GRR server.
4. Retrieve the latest version of GRR from a release DEB and store it on the server as shown in Figure 7-1.

```
ubuntu:~$ wget https://storage.googleapis.com/releases.grr-response.com/grr-server_3.3.0-0_amd64.deb  
--2019-06-23 06:13:28-- https://storage.googleapis.com/releases.grr-response.com/grr-server_3.3.0-0
```

Figure 7-1 GRR download

5. Start the install script to install the GRR server as shown in Figure 7-2.

```
ubuntuserver@ubuntu:~$ sudo apt install -y ./grr-server_3.3.0-0_amd64.deb  
Reading package lists... Done
```

Figure 7-2 Install script for GRR

6. The install script then asks for several configuration settings such as the admin URL for the GUI interface. This is set up based on the public IP of the server. Please note that the example used in this part of the GRR setup uses a local IP address in order to hide the details of the public GRR server on the DigitalOcean droplet. Figure 7-3 shows some of the configuration items that are required during the GRR server setup. Figure 7-4 shows the final configuration items, as well as the generation of certificates for secure communication between the GRR server and GRR clients. Figure 7-5 illustrates the output of a completed GRR server installation. To determine whether the GRR server service is running, the GRR-server service status can be queried as is shown in Figure 7-6.

```
Step 1: Setting Basic Configuration Parameters
We are now going to configure the server using a bunch of questions.

==GRR Datastore==
For GRR to work each GRR server has to be able to communicate with
the datastore. To do this we need to configure a datastore.

Do you want to use REL_DB - the new optimized datastore implementation?
(if not, a legacy datastore will be used) [Yn]: [Y]:
GRR will use MySQL as its database backend. Enter connection details:
MySQL Host [localhost]:
MySQL Port (0 for local socket) [0]:
MySQL Database [grr]:
MySQL Username [root]: grr
Please enter password for database user grr:
Configure SSL connections for MySQL? [yN]: [N]:
Successfully connected to MySQL with the provided details.

==GRR URLs==
For GRR to work each client has to be able to communicate with the
server. To do this we normally need a public dns name or IP address
to communicate with. In the standard configuration this will be used
to host both the client facing server and the admin user interface.

Please enter your hostname e.g. grr.example.com [ubuntu]: 192.168.29.128

==Server URL==
The Server URL specifies the URL that the clients will connect to
communicate with the server. For best results this should be publicly
accessible. By default this will be port 8080 with the URL ending in /control.

Frontend URL [http://192.168.29.128:8080/]:

==AdminUI URL==:
The UI URL specifies where the Administrative Web Interface can be found.

AdminUI URL [http://192.168.29.128:8000]:

==GRR Emails==
GRR needs to be able to send emails for various logging and
alerting functions. The email domain will be appended to GRR
usernames when sending emails to users.

==Monitoring/Email Domain==
Emails concerning alerts or updates must be sent to this domain.

Email Domain e.g example.com [localhost]:

==Alert Email Address==
Address where monitoring events get sent, e.g. crashed clients,
broken server, etc.

Alert Email Address [grr-monitoring@localhost]:
```

Figure 7-3 GRR setup parameters


```

Step 2: Key Generation
All keys will have a bit length of 2048.
Generating executable signing key
Generating CA keys
Generating Server keys
Generating secret key for csrf protection.

Writing configuration to /etc/grr//server.local.yaml.
I0626 02:31:05.973725 140512034703104 config_lib.py:490] Writing back configu
ml
Initializing the datastore.
I0626 02:31:06.524421 140512034703104 server_logging.py:186] Writing log file
site-packages/grr_response_core/var/log//GRRlog.txt

Step 3: Adding GRR Admin User
Please enter password for user 'admin':

Step 4: Repackaging clients with new configuration.
Server debs include client templates. Re-download templates? [yN]: [N]: y
Repack client templates? [Yn]: [Y]: y

```

Figure 7-4 Certificate generation and client repacking

```

#####
Setting up dh-strip-nondeterminism (0.015-1) ...
Processing triggers for libc-bin (2.23-0ubuntu1) ...
ubuntuuser@ubuntu:~$ █

```

Figure 7-5 GRR server installation complete

```

ubuntuuser@ubuntu:~$ sudo service grr-server status
● grr-server.service - GRR Service
   Loaded: loaded (/lib/systemd/system/grr-server.service; enabled; vendor preset: enabled)
   Active: active (exited) since Wed 2019-06-26 02:25:14 PDT; 9min ago

```

Figure 7-6 GRR server running

7. The DigitalOcean server did pose some challenges in this regard, as the droplets have custom firewalls that block all ports by default, except for port 22 (SSH), port 80 (HTTP), port 443 (HTTPS) and port 3306 (MySQL). The GRR server requires port 8000 for the web interface, so the firewall is customised to allow port 8000. This step therefore entails the opening of the required ports in the firewall on the host machine, or the firewall guarding the host machine, to ensure that the GRR server is accessible and in working condition.
8. Log in to the GRR admin GUI web interface with the admin login details provided during the GRR server installation. In order to hide the actual installation location, the screenshot in Figure 7-7 is based on the local installation and not the DigitalOcean installation. However, further screenshots are based on the actual DigitalOcean installation.

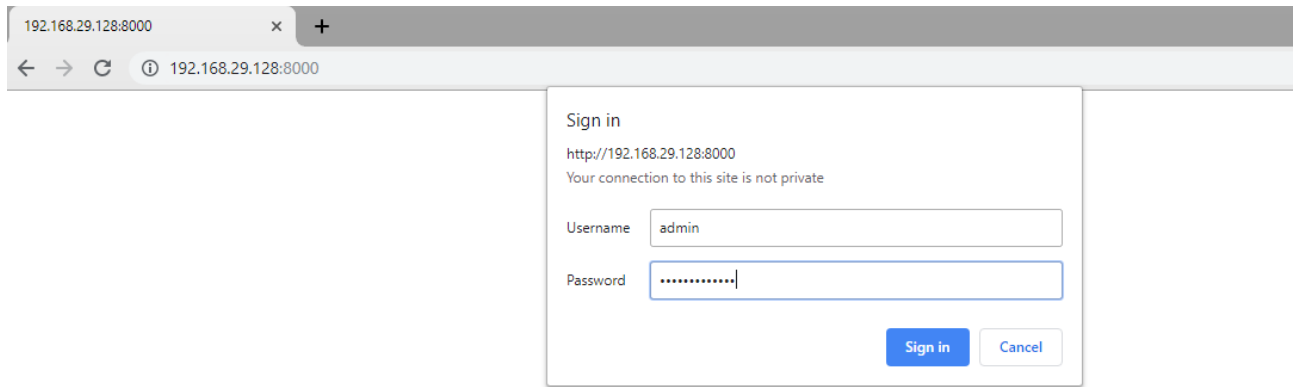


Figure 7-7 Logging in to GRR server GUI

7.4.5 Installing GRR clients on virtual machines

Installing the GRR client on a virtual machine is simple, but highly complex on an IoT device. GRR is built for investigating servers in a server farm environment or investigating desktop computers. This means that GRR only supports the amd64 and i386 architectures.

The installation of GRR clients on virtual machines is rather simple, as virtual machines on physical computers use amd64 or i386 architectures. The first step to installing a GRR client on a virtual machine, based on a GRR supported architecture (amd64 or i386), is to install a minimal Linux distribution on a virtual machine, which simulates the working of a physical IoT device. The Linux distribution used for this simulation is the 32bit Linux Lite version 3.8 distribution.

The Linux Lite distribution is used in this simulation due to its hardware requirements. According to the official Linux Lite website [111], the operating system requires only a 1GHz processor, 768MB of RAM and 8GB of storage space to function. These are relatively modest hardware requirements when compared to that of full-fledged operating systems, which require much more RAM, storage space and powerful processors. The lightweight hardware requirements for the Linux Lite operating system simulate IoT devices with very little processing power, storage space and RAM. The Linux Lite operating system requirements are therefore very close to the physical hardware of the Raspberry Pi, which is why the Linux Lite distribution is used in the simulation of IoT devices.

The steps followed to install the GRR client on a Debian virtual machine are as follows:

1. Download a hypervisor, such as VMware Workstation Player, and install it.
2. Download the Linux Lite operating system.
3. Create a new virtual machine in VMware.
4. Select the ISO file for the Linux Lite operating system.
5. Configure VMware for a Linux installation.
6. Allocate resources to the VM. The resources allocated to the Linux Lite VMs are as close as possible to the actual hardware of the Raspberry Pi as an actual IoT device.
7. Start the VM and complete the installation of the operating system.
8. Download the GRR client from the GRR server GUI by entering the server login details and then navigate to “Manage Binaries”. Click on the client that needs to be installed on the VM, in this case “linux/installers/grr_3.2.4.6_i386.deb”.
9. Save the installer file so that it can be opened for installation.
10. Navigate to the installer package and open the installer by clicking on “Install Package”.

The GRR client is then successfully installed and contacts the GRR server in order to check for new action requests. The installation of a GRR client on a VM without logging in to the GRR admin GUI is as follows:

1. Obtain a copy of the GRR client installer.
2. Copy the GRR installation file onto the virtual machine.
3. Repeat steps 9 and 10 from the steps on how to install a GRR client on a VM.

Upon successful installation of a GRR client, it will appear in the list of GRR clients connected to the GRR server in the GRR admin GUI. The list can be viewed by logging in to the GRR server admin GUI and pressing the “Enter” key in the search box as shown in Figure 7-8. The VM GRR client that was added in the steps above is shown in Figure 7-9.



Figure 7-8 GRR admin GUI search clients

<input type="checkbox"/>	Online	Subject	Host	OS Version	MAC	Usernames	First Seen	Client version	Labels	Last Checkin	OS Install Date
<input type="checkbox"/>	●	C.0e016696966c1c98	linuxlitedesktop-virtual-machine	16.4	00:00:00:00:00:00 00:0c:29:d7:d4:56	linuxlitedesktop	2019-06-26 13:12:12 UTC	3246	desktop-vm	2019-06-26 13:18:44 UTC	2019-06-26 12:52:17 UTC

Figure 7-9 New GRR client

7.4.6 Building custom GRR clients to work on IoT devices

The installation of GRR clients on actual IoT devices, such as the Raspberry Pi, is extremely complex. To this researcher's knowledge, it has never been done before. A core developer of the GRR platform was contacted via email, who was also unaware of this being done before [112].

The GRR client needs to be rebuilt in order to run on the ARM architecture, as the ARM architecture is not officially supported by the GRR platform. Rebuilding the GRR client for the ARM architecture requires several changes to the GRR code and will be discussed in the steps that follow. The steps assume that an external GRR server has been set up and is in a working state. This means that the GRR server can be contacted by various devices with GRR clients installed, in various physical locations and various network configurations.

1. Ensure the Raspberry Pi has all of the needed packages for the GRR platform installed.
2. Set up a virtual environment for the GRR platform to be installed on. This is shown in Figure 7-10.

```
pi@raspberrypi:~/Desktop $ sudo pip install --upgrade pip virtualenv
Collecting pip
  Downloading https://files.pythonhosted.org/packages/5c/e0/be401c0031
420/pip-19.1.1-py2.py3-none-any.whl (1.4MB)
  99% |#####| 1.4MB 1.1MB/s eta 0:00:01
```

Figure 7-10 Set up pip virtual environment

3. Download the entire GRR platform on the IoT device by cloning the latest version of the GRR platform. The entire GRR platform needs to be downloaded on the Raspberry Pi, as the GRR configuration needs to be run in order to configure the GRR clients to communicate with the actual GRR server (shown in future steps). Downloading the GRR platform from source onto the Raspberry Pi is shown in Figure 7-11.

```
pi@raspberrypi:~/Desktop $ git clone https://github.com/google/grr.git
Cloning into 'grr'...
remote: Enumerating objects: 95, done.
```

Figure 7-11 Clone GRR

4. Next Protoc/Protobuf is installed. Protoc/Protobuf is a platform and language neutral tool used for serialising structured data [113]. Custom download Protoc/Protobuf on

the IoT device as the GRR platform requires Google's protocol buffers to read and write data from the GRR clients.

5. Compile PROTOC/PROTOBUF on the Raspberry Pi, as the code needs to be compiled for the ARM architecture. This part of the process takes quite a long time and is shown in from Figure 7-12 to Figure 7-16.

```
pi@raspberrypi:~/Desktop/grr/protobuf $ ./autogen.sh
+ mkdir -p third_party/googletest/m4
```

Figure 7-12 Compile protobuf

```
pi@raspberrypi:~/Desktop/grr/protobuf $ ./configure
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... armv7l-unknown-linux-gnueabi
checking host system type... armv7l-unknown-linux-gnueabi
```

Figure 7-13 Compile protobuf part 2

```
pi@raspberrypi:~/Desktop/grr/protobuf $ make
make all-recursive
make[1]: Entering directory '/home/pi/Desktop/grr/protobuf'
```

Figure 7-14 Compile protobuf part 3

```
pi@raspberrypi:~/Desktop/grr/protobuf $ make check
Making check in .
make[1]: Entering directory '/home/pi/Desktop/grr/protobuf'
make check-local
```

Figure 7-15 Compile protobuf part 4

```
pi@raspberrypi:~/Desktop/grr/protobuf $ sudo make install
Making install in .
make[1]: Entering directory '/home/pi/Desktop/grr/protobuf'
```

Figure 7-16 Compile protobuf part 5

6. Add Protobuf to the PATH variable so that it can be accessed whenever it is needed by the GRR client.
7. Create a virtual environment where GRR is installed.
8. Run the Linux installer for the GRR platform as shown in Figure 7-17.

```
pi@raspberrypi:~/Desktop/grr $ travis/install.sh
+ source /home/pi/INSTALL/bin/activate
++ deactivate nondestructive
++ unset -f pydoc
```

Figure 7-17 GRR installer

9. Customise the GRR platform by adding the arm7l architecture, which the Raspberry Pi is based on, to several of the GRR configuration files. The original GRR platform does not recognise the ARM architecture in the list of supported architectures for the platform, which means that the installer will ultimately fail if the platform is not customised. Figure 7-18 shows the first file that is modified to add the arm7l context to the GRR platform. Figure 7-19 shows how the arm7l architecture is added to the YAML configuration of the GRR platform.

```

pi@raspberrypi:~/Desktop/grr/grr/core/grr_response_core/config $ sudo nano contexts.py
GNU nano 2.7.4 File: contexts.py

#!/usr/bin/env python
"""This file defines valid configuration contexts."""

from __future__ import absolute_import
from __future__ import division
from __future__ import unicode_literals

from grr_response_core.lib import config_lib

# Different client platforms.
PLATFORM_DARWIN = config_lib.DEFINE_context("Platform:Darwin")
PLATFORM_LINUX = config_lib.DEFINE_context("Platform:Linux")
PLATFORM_WINDOWS = config_lib.DEFINE_context("Platform:Windows")

# Client architectures.
ARCH_ARMV7L = config_lib.DEFINE_context("Arch:armv7l")
ARCH_AMD64 = config_lib.DEFINE_context("Arch:amd64")
ARCH_I386 = config_lib.DEFINE_context("Arch:i386")
ARCH_PPC64LE = config_lib.DEFINE_context("Arch:ppc64le")
ARCH_AARCH64 = config_lib.DEFINE_context("Arch:aarch64")

```

Figure 7-18 GRR ARM architecture add

```

pi@raspberrypi:~/Desktop/grr/grr/core/install_data/etc $ sudo nano grr-server.yaml
GNU nano 2.7.4 File: grr-server.yaml

AdminUI.bind: "://"
AdminUI.webauth_manager: BasicWebAuthManager
AdminUI.use_precompiled_js: True
API.DefaultRouter: ApiCallRouterWithoutChecks
Client.company_name: GRR Project
Client.description: "%(name) %(platform) %(arch)"
Client.name: GRR
Client.proxy_servers:
Logging.engines: "stderr,file"
Logging.verbose: False

# These must be in the domain specified above as Logging.domain.
# Monitoring.emergency_access_email: grr-emergency-access@example.com
# Monitoring.alert_email: grr-monitoring@example.com

AdminUI Context:
  Logging.filename: "%(Logging.path)/grr-ui.log"

# The client's reported architecture string depends on the Arch context
Arch:armv7l:
  Client.arch: armv7l
Arch:amd64:
  Client.arch: amd64
Arch:i386:
  Client.arch: i386

```

Figure 7-19 GRR ARM architecture add part 2

10. Add the details of the GRR server that the clients should connect to. This is done in order to obtain the packages for rebuilding client templates. The installation step configures the Raspberry Pi as if it will be a GRR server, but it is not used as a GRR server. It is only used to provide the details for custom GRR clients. The

installation therefore configures the clients to be built with the necessary information to contact the real GRR server that is hosted on the DigitalOcean droplet. The configuration for the GRR server is done based on the details of the actual DigitalOcean droplet GRR server, as is shown in Figure 7-20.

```
(INSTALL) pi@raspberrypi:~/Desktop/grr $ grr_config_updater initialize
Checking write access on config /home/pi/Desktop/grr/grr/core/install_data/etc/server.local.yaml

Step 0: Importing Configuration from previous installation.
No old config file found.

Step 1: Setting Basic Configuration Parameters
We are now going to configure the server using a bunch of questions.

--GRR Datastore--
For GRR to work each GRR server has to be able to communicate with
the datastore. To do this we need to configure a datastore.

Do you want to use REL_DB - the new optimized datastore implementation?
(if not, a legacy datastore will be used) [Yn]: [Y]:
GRR will use MySQL as its database backend. Enter connection details:
MySQL Host [localhost]:
MySQL Port (0 for local socket) [0]:
MySQL Database [grr]:
MySQL Username [root]: grr
Please enter password for database user grr:
Configure SSL connections for MySQL? [yN]: [N]:
Successfully connected to MySQL with the provided details.

--GRR URLs--
For GRR to work each client has to be able to communicate with the
server. To do this we normally need a public dns name or IP address
to communicate with. In the standard configuration this will be used
to host both the client facing server and the admin user interface.

Please enter your hostname e.g. grr.example.com [raspberrypi]: [REDACTED]

--Server URL--
The Server URL specifies the URL that the clients will connect to
communicate with the server. For best results this should be publicly
accessible. By default this will be port 8080 with the URL ending in /control.
Frontend URL [REDACTED]8080/]:

--AdminUI URL--:
The UI URL specifies where the Administrative Web Interface can be found.
AdminUI URL [REDACTED]8000]:

--GRR Emails--
GRR needs to be able to send emails for various logging and
alerting functions. The email domain will be appended to GRR
usernames when sending emails to users.

--Monitoring/Email Domain--
Emails concerning alerts or updates must be sent to this domain.
Email Domain e.g example.com [localhost]: █
```

Figure 7-20 GRR client configuration

The standard installer will eventually fail, as the GRR server component is not supported on the ARM architecture. This is not a problem, however, as the configuration files are changed and the GRR clients are repacked in the following steps. The GRR server installation failure is shown in Figure 7-21.


```

Initializing the datastore.
I0627 17:13:57.539889 1996206080 server_logging.py:186] Writing log file to /home/pi/
e_core/var/log/GRRlog.txt
Traceback (most recent call last):
  File "/home/pi/INSTALL/bin/grr_config_updater", line 11, in <module>
    load_entry_point('grr-response-server', 'console_scripts', 'grr_config_updater')(

```

Figure 7-21 GRR setup fails

- Build custom client templates. This entails building a new template on the IoT device and repacking the template on the GRR server. Use the following command on the IoT devices to build the new client template. This is shown in Figure 7-22 and Figure 7-23:

```

grr_client_build build --output mytemplates
(INSTALL) pi@raspberrypi:~/Desktop/grr $ grr_client_build build --output mytemplates
Clearing directory /tmp/grr-build/grr-build
Clearing directory /tmp/grr-build/workpath
Clearing directory /tmp/grr-build/dist

```

Figure 7-22 GRR custom client template

```

Building EXE from out00-EXE.toc
Appending archive to ELF section in EXE /tmp/grr-build/workpath/grr/grr-client
Building EXE from out00-EXE.toc completed successfully.
checking COLLECT
Building COLLECT because out00-COLLECT.toc is non existent
Building COLLECT out00-COLLECT.toc
Building COLLECT out00-COLLECT.toc completed successfully.
Unable to remove directory: /tmp/grr-build/dist/debian/grr-client/pytz
Generating zip template file at mytemplates/grr_3.3.0.1_i386.deb.zip
(INSTALL) pi@raspberrypi:~/Desktop/grr $ █

```

Figure 7-23 GRR custom client template build complete

- Repack the GRR template created in the previous step. This is done on the GRR server, due to the GRR server knowing all the configurations, such as cryptographic details for the connections with GRR clients. Details are redacted in Figure 7-24 in order to hide details about the actual GRR server.

```

grr_client_build repack --template mytemplates/*.zip --output_dir mytemplates
(INSTALL) grr_client_build repack --template mytemplates/*.zip --output_dir mytemplates
Repacking template: mytemplates/grr_3.3.0.1_i386.deb.zip
Using context: [u'ClientBuilder Context', u'ClientBuilder Context', u'Arch:i386', u'Platform:Linux', u'Target:Linux', u'rget:LinuxDeb', u'Target:Linux', u'ClientBuilder Context'] and labels: []
Writing back configuration to file /tmp/tmpkdQVXb/grrd.yaml
Created package mytemplates/grr_3.3.0.1_i386.deb
Repacked into mytemplates/grr_3.3.0.1_i386.deb
(INSTALL) █

```

Figure 7-24 GRR server build of custom client

The result of repacking the template is an installer that can be installed on IoT devices based on the arm7l (ARM) architecture. Do note that the name of the custom client still ends with i386.deb, which is reminiscent of the i386 architecture, but it is in fact built for the ARM architecture, it has just not been changed in the compiler files.

- Install the new repacked template GRR client on the client by simply copying the installer to the client machine and then running the installer. The GRR client then runs as a background process on the IoT device. This is the same as the steps for installing a GRR client on a VM machine described earlier in this chapter.

14. Verify that the GRR server and GRR client can connect to each other. This is done by opening the GRR server GUI and listing all the GRR clients. If the IoT device is added successfully, a new GRR client will appear in the list of clients (Figure 7-25) and while the architecture will be arm7l (Figure 7-26).

<input type="checkbox"/>	<input checked="" type="checkbox"/>	C.6e2f6e356217e288	raspberrypi	9.8	00:00:00:00:00:00 b8:27:eb:0f:b9:65 b8:27:eb:5a:ec:30	pi	2019-05-02 13:15:23 UTC	3249	raspberrypi 1	2019-06-27 16:32:11 UTC	2018-11-13 14:21:15 UTC
--------------------------	-------------------------------------	--------------------	-------------	-----	---	----	-------------------------	------	---	-------------------------	-------------------------

Figure 7-25 Raspberry Pi client appears in GRR server GUI client list

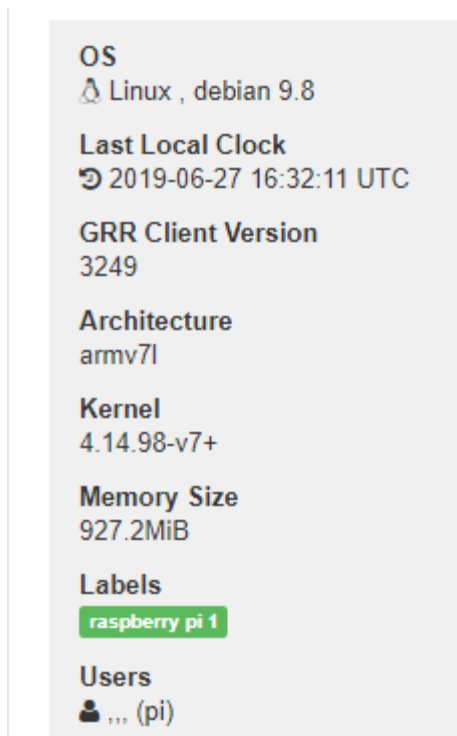


Figure 7-26 Raspberry Pi details

7.5 CONCLUSION

This chapter aimed at finding a solution for the requirements of DF for IoT by implementing a prototype for forensic investigations on IoT devices. The prototype is based on the requirements for DF of IoT defined in Chapter 6, and indicated that it is possible to satisfy these requirements.

This chapter stated various arguments for and against a complete custom developed prototype or a customised implementation of available software, with the latter option being chosen. Available software solutions for the problem at hand were then investigated. The various options were evaluated and the GRR platform selected based on the possibility of satisfying most requirements for DF of IoT. GRR is, however, developed for desktop and server hardware, not IoT devices. The customisation of the GRR platform for

IoT devices is therefore an enormous challenge. This chapter then made a major contribution to the current body of knowledge by customising an already established remote live forensics platform to accommodate IoT forensics.

The customisation of the GRR platform for IoT devices was presented in this chapter in detail. The steps taken to customise the platform were presented in various screenshots with detailed explanations.

This chapter formed an integral part of this study, as it addressed several research questions and achieved a variety of the research objectives. It addressed research question 1.2.1 (DFR IoT standards processes) by determining what standards of DFR can be applied to the IoT. This was achieved by enabling the application of DFR techniques employed to server farms and desktop computers to IoT devices by altering the GRR platform to also support IoT DFR. Research question 1.2.2 (DFR IoT devices feasibility) was addressed by altering a forensic investigation platform to work on a subset of IoT devices. In this dissertation the Raspberry Pi was used as a subset of IoT devices to show the working of GRR on IoT devices. This chapter furthermore addressed research question 1.2.3 (DFR requirements, legal aspects) by showing that it is indeed possible to apply DFR on certain types of IoT devices.

With regard to research objectives, this chapter partially addressed objective 1.4.1 (DFR processes, IoT device classification) by investigating current software solutions that facilitate the DFR of IoT devices. In addition, this chapter also showed that some of the current DFR processes can be used for the IoT. However, this chapter made the biggest contribution towards objective 1.4.4 (Proof of concept, DFR IoT prototype) by implementing a working prototype for DFR in IoT devices. The prototype still needs to be tested and evaluated in order to determine the level to which it enables DFR for IoT devices, as presented in the next chapter.

8 CHAPTER 8: TESTING THE PROTOTYPE FOR DIGITAL FORENSIC READINESS IN THE INTERNET OF THINGS

8.1 INTRODUCTION

This chapter aims to show the working of the DFR for IoT devices model prototype developed in this study. Several measurements are formulated for the successful implementation of a DFR solution for IoT devices, after which the prototype developed is tested to determine whether the model is feasible. All the testing steps are shown in order to provide an accurate test environment for DFR in IoT.

This chapter aims to answer research question 1.2.1 (DFR IoT standards processes) by determining which forensic readiness approaches can be applied to the IoT (1.2.2, DFR IoT devices feasibility) by conducting DF investigations on a subset of IoT devices and testing a DFR compliant prototype on IoT devices (1.2.3, DFR requirements, legal aspects). The model for DFR in IoT is practically tested to satisfy objective 1.4.4 (Proof of concept, DFR IoT prototype) of this dissertation and to show that the model produces trustworthy forensic evidence.

This chapter starts with a basic formulation of measurements used to determine the feasibility of DFR for IoT devices. An overview of the various tools used to measure the feasibility of DFR for IoT devices is then provided, followed by an indication of the steps followed to simulate the working of DFR for IoT, as well as how data is captured to measure the feasibility of the model and prototype. This chapter is concluded with a presentation of the testing procedures' findings.

8.2 FORMULATION OF MEASUREMENTS FOR THE FEASIBILITY OF DFR FOR IOT

This section aims to introduce various measurements to determine whether DFR for IoT is possible, which entails that IoT devices send evidence prior to a forensic investigation in a proactive manner and not after an incident occurred. The ability of IoT devices to send evidence proactively is therefore a measurement of the feasibility of DFR for IoT.

IoT devices must furthermore remain in the physical position where they normally function. Contrasting to the normal forensic investigations where devices are seized and then searched for evidence, IoT devices should still function when investigations occur. IoT devices can sometimes be placed in hard to reach places, which adds to the motivation of

applying DFR to IoT devices. IoT devices can also be part of critical systems, which means that IoT devices cannot be removed for investigations. This is where DFR plays a big role, as the evidence is gathered prior to the investigation and normal IoT functioning can continue. IoT devices therefore do not need to be physically relocated in order to facilitate DFR for IoT. This is another measurement of the feasibility of DFR for IoT.

The final and main measurement of feasibility for DFR in IoT is the measurement of continuous reliable performance of IoT devices when the model for DFR in IoT is implemented. This means that IoT devices should still function without any loss of service due to the model being implemented on IoT devices. Normal operation means that the storage of the IoT devices should not be hindered from storing or gathering data. IoT device storage should not be impacted significantly due to the introduction of a prototype for DFR in IoT.

IoT devices are generally also constrained with regards to processing power. In fact, IoT devices do not have high performance CPU's and often make use of reduced instruction set processors. The introduction of a prototype for DFR in IoT should therefore not be CPU intensive, but allow the IoT devices to perform normally. CPU usage of the DFR prototype should therefore be kept to a minimum.

Also, IoT devices do not generally have much RAM installed and RAM usage should therefore be kept to a minimum in order to ensure accommodation of other processes running on IoT devices.

The measurement of a successful implementation of a prototype for DFR in IoT devices is therefore rather simple. IoT devices should be kept operational when investigations occur, and the prototype must not use too much of the limited resources on IoT devices upon which it is installed. The testing of the prototype should show how the IoT devices perform under normal conditions, as well as how the IoT devices perform when the simulation takes place. The testing should thus be transparent and show how the various tools impact the performance of the IoT devices.

8.3 OVERVIEW OF TOOLS USED TO MEASURE IMPLEMENTATION FEASIBILITY

This section provides an overview of the various tools and programs used to measure the feasibility of DFR for IoT and ensure that a realistic simulation is conducted. The more realistic the simulation of the prototype, the more accurate the measurement of the validity of DFR for IoT.

There are several components needed to conduct the simulation of DFR for IoT. The components are shown in Figure 8-1 and referenced with numbers which correlate to the numbers of the components mentioned in this section. Components in this case refer to devices, programs and tools. The first component is IoT devices, which can be physical or virtual IoT devices. The IoT devices in this simulation are two Raspberry Pi's and three VMs. The setup of the physical IoT devices and virtual machines was presented in detail in chapter 7.

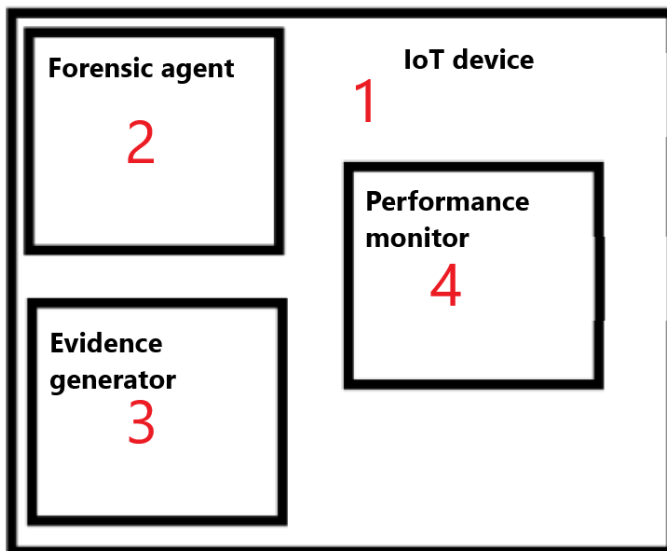


Figure 8-1 Components of the DFR for IoT simulation

The next three components of the simulation of DFR for IoT are software-based. As is shown in Figure 8-1 all of the software components are installed on the IoT devices used in the simulation. The second component needed for the simulation is the actual forensic agent installed on the IoT devices to collect the evidence generated by the IoT devices. In this dissertation the GRR platform is successfully adapted to cater for IoT devices. The process of preparing IoT devices for the GRR platform is also presented in detail in Chapter 7.

The third component of this simulation is a program that generates evidence to be gathered by the GRR platform. The program creates evidence (state-based logs) based on a smart home environment. These state-based logs contain sensor-based logs, which means that whenever IoT device sensors detect changes in the environment, the changes are logged. The actuator actions of the IoT devices are also logged. In other words, when an IoT sensor detects that a door is opened, the sensing information is logged. When the IoT device turn lights on or off based on the sensed changed in the environment, the actions are also logged.

The fourth and final component of this simulation is a platform that monitors all the devices' performance metrics. The platform must be able to monitor all the main resources of the IoT devices. This includes the devices' CPU, RAM and storage usage during the various stages of the simulations and testing of the prototype. All the components mentioned in this section play a role in the steps followed in the simulation of DFR for IoT.

8.4 STEPS FOR SIMULATING DIGITAL FORENSIC READINESS IN THE INTERNET OF THINGS

This section provides detailed steps followed to test the prototype developed and determine whether DFR for IoT is possible. All the various steps in the testing need to be transparent to ensure that reliable data about the simulation and testing is presented.

This section starts with a formulation of the various metrics analysed in this simulation, followed by an overview of the differences of the simulation to show how each step adds value towards the main simulation of DFR for IoT.

8.4.1 Metrics for measuring device performance

All the various steps in measuring IoT device performance during the simulation are conducted over a continuous period of 8 hours. Each of the steps in the simulation is conducted over the same timeframe while performing the various aspects of the simulation. The platform used for measuring the performance of the IoT devices is Zabbix, software designed to monitor a wide variety of metrics from various devices in real time [114]. The platform consists of a server where all the metrics are collected, and agents that are installed on the various devices. The agents gather the data from the various metrics

on the devices and send it to the server. The server then enables aggregation of the various metrics across the various devices and graphically presents the data.

The reporting function of Zabbix is somewhat limited. Therefore, Grafana is integrated with Zabbix to enable better reporting capabilities. Grafana is an open-source platform that is used to visualise and query metrics easily [115]. Grafana also enables the exporting of data captured in a comma-separated (CSV) format, which enables further analysis of the data gathered from IoT devices.

Zabbix supports several types of metrics that can be monitored by devices with the agents installed. In this study, the Linux machine template is used to monitor the metrics of the various IoT devices. This enables the collection of several metrics, including the available memory space, CPU context switches per second, CPU idle time, CPU interrupts per second, CPU load average over 1 minute, CPU system time, free disk space in percentage, free swap space in percentage, network traffic on the various network ports, number of processes, number of running processes, and the ping of the Zabbix agent on the IoT devices.

Data from the various metrics is collected from the Zabbix agents every few seconds. The data is captured and presented in graph form in Grafana. The data is also exported to CSV files for further analysis. The following subsection provides an overview of the monitoring setup used in the simulation.

8.4.2 Setup of Zabbix and Grafana for the simulation

The Zabbix server is installed on an Ubuntu server on the local network where the IoT devices and VMs are located. This is done due to the limited availability of servers with public IP's and separation of the various physical machines used in this study. The initial setup of Zabbix is rather simple and not explained in this dissertation. The complicated part of the setup is adding all the monitored devices to the Zabbix server, for which the various steps are shown in screenshots in the paragraphs that follow.

The first step is to add an agent to the Zabbix server by downloading the Zabbix Agent on the device that is to be monitored. This is shown in Figure 8-2 for a VM and in Figure 8-3 for a Raspberry Pi.

```
linuxlitevm3@linuxlitevm3-virtual-machine:~/Desktop/zabbix$ dpkg -i zabbix-release_3.4-1+bionic_all.deb
dpkg: error: requested operation requires superuser privilege
linuxlitevm3@linuxlitevm3-virtual-machine:~/Desktop/zabbix$ sudo dpkg -i zabbix-release_3.4-1+bionic_all.deb
```

Figure 8-2 Zabbix agent install on VM

```
pi@raspberrypi:~$ sudo apt-get install zabbix-agent
Reading package lists... Done
```

Figure 8-3 Zabbix agent install Raspberry Pi

The Zabbix agent is then configured to connect to the Zabbix server. The Zabbix server is configured to automatically add new hosts, as Zabbix agents are configured to actively poll the Zabbix server. This is done by configuring the `ServerActive` parameter in the Zabbix agent configuration file. The `Hostname` parameter is unique for the various agents, as it is used to identify the various hosts connected to the Zabbix server. Figure 8-4 shows the `ServerActive` and `Hostname` parameter being configured for a VM. The configurations for the Raspberry Pi are the same as the configuration for VMs, except for Hostnames being unique across the hosts.

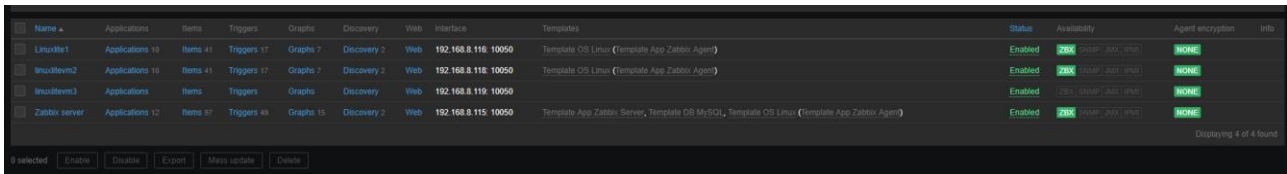
```
ServerActive=192.168.8.115

### Option: Hostname
#   Unique, case sensitive hostname.
#   Required for active checks and must match hostname as configured on the server.
#   Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostname=

Hostname=linuxlitevm3
```

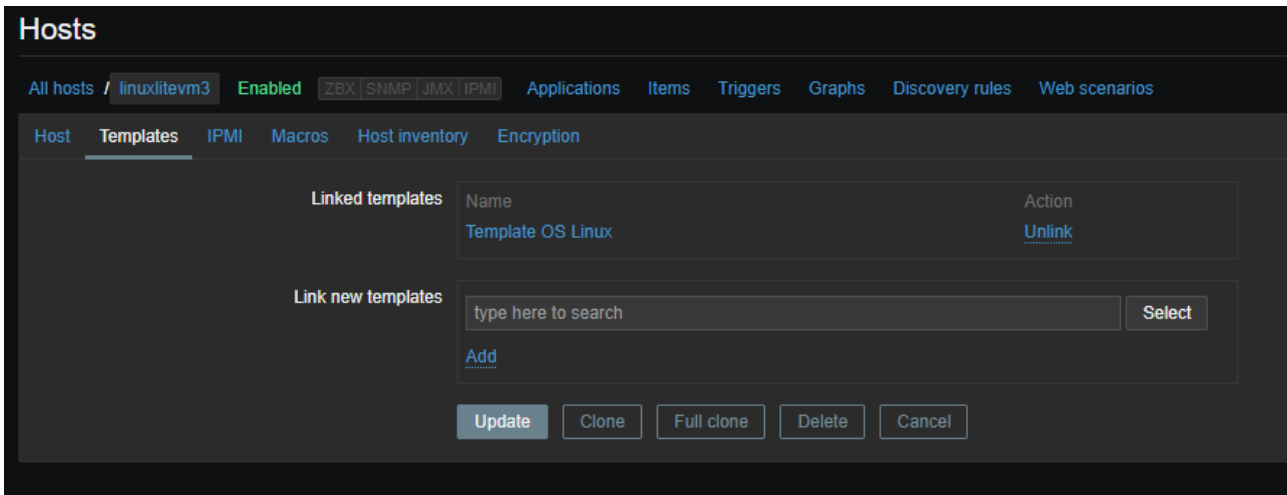
Figure 8-4 Zabbix agent configuration

Due to the automatic adding of hosts, the newly configured host appears in the Zabbix server interface. A newly added host is shown in Figure 8-5, but note that the “ZBX” label is not yet green next to the newly added host. This is due to the absence of a monitoring template for the newly added host. A monitoring template is then applied to the newly added host so that the Zabbix agent knows what metrics to collect about the host. This is shown in Figure 8-6. The “ZBX” label turns green when metrics are being received from a host. Figure 8-7 indicates the success of adding a new host with a template that is providing the Zabbix server with metrics.



Name	Applications	Items	Triggers	Graphs	Discovery	Web	Interface	Templates	Status	Availability	Agent encryption	Info
linuxite1	Applications 10	Items 41	Triggers 17	Graphs 7	Discovery 2	Web	192.168.8.116:10050	Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX (SNMP, JMX, IPMI)	NONE	
linuxitev2	Applications 10	Items 41	Triggers 17	Graphs 7	Discovery 2	Web	192.168.8.116:10050	Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX (SNMP, JMX, IPMI)	NONE	
linuxitev3	Applications	Items	Triggers	Graphs	Discovery	Web	192.168.8.119:10050		Enabled	ZBX (SNMP, JMX, IPMI)	NONE	
Zabbix server	Applications 12	Items 97	Triggers 49	Graphs 15	Discovery 2	Web	192.168.8.115:10050	Template App Zabbix Server, Template DB MySQL, Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX (SNMP, JMX, IPMI)	NONE	

Figure 8-5 Automatic detection of new Zabbix agent



Hosts

All hosts / linuxitevm3 Enabled ZBX | SNMP | JMX | IPMI Applications Items Triggers Graphs Discovery rules Web scenarios

Host Templates IPMI Macros Host inventory Encryption

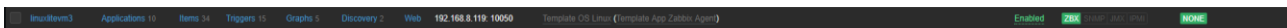
Linked templates

Name	Action
Template OS Linux	Unlink

Link new templates

type here to search

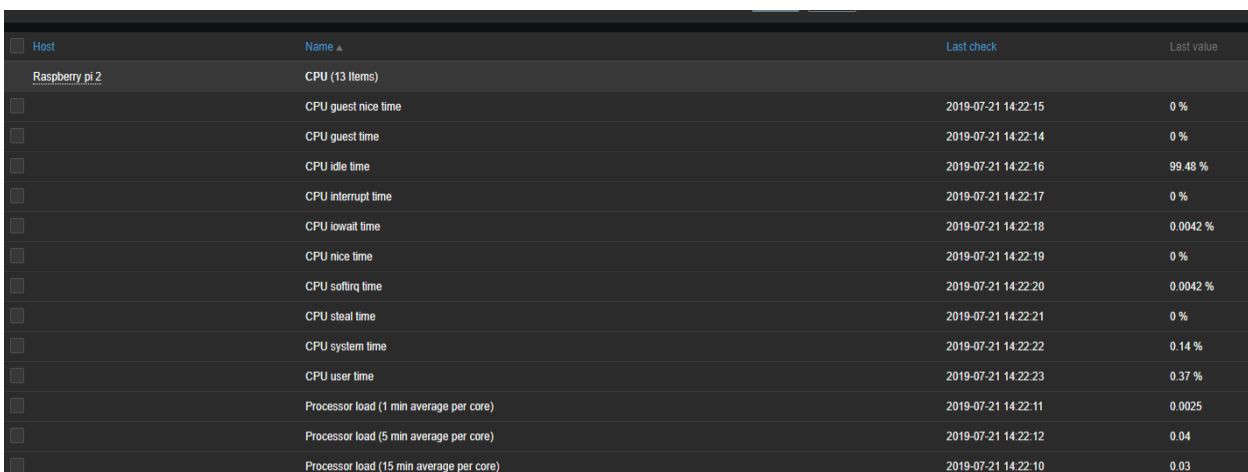
Figure 8-6 Add template to Zabbix host



Host	Templates	IPMI	Macros	Host inventory	Encryption						
linuxitevm3	Applications 10	Items 54	Triggers 15	Graphs 5	Discovery 2	Web	192.168.8.119:10050	Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX (SNMP, JMX, IPMI)	NONE

Figure 8-7 Successful adding of a new host

After the successful installation of a Zabbix agent on a host, the metrics need to be evaluated. The latest metrics can be viewed in Zabbix (Figure 8-8). Readability of metrics is improved using a Grafana panel (Figure 8-9). The final Grafana panel in Figure 8-10 shows the metrics from all of the monitored IoT hosts and allows easy exporting of the metrics from all devices.



Host	Name	Last check	Last value
Raspberry pi 2	CPU (13 Items)		
	CPU guest nice time	2019-07-21 14:22:15	0 %
	CPU guest time	2019-07-21 14:22:14	0 %
	CPU idle time	2019-07-21 14:22:16	99.48 %
	CPU interrupt time	2019-07-21 14:22:17	0 %
	CPU iowait time	2019-07-21 14:22:18	0.0042 %
	CPU nice time	2019-07-21 14:22:19	0 %
	CPU softirq time	2019-07-21 14:22:20	0.0042 %
	CPU steal time	2019-07-21 14:22:21	0 %
	CPU system time	2019-07-21 14:22:22	0.14 %
	CPU user time	2019-07-21 14:22:23	0.37 %
	Processor load (1 min average per core)	2019-07-21 14:22:11	0.0025
	Processor load (5 min average per core)	2019-07-21 14:22:12	0.04
Processor load (15 min average per core)	2019-07-21 14:22:10	0.03	

Figure 8-8 Latest metrics received

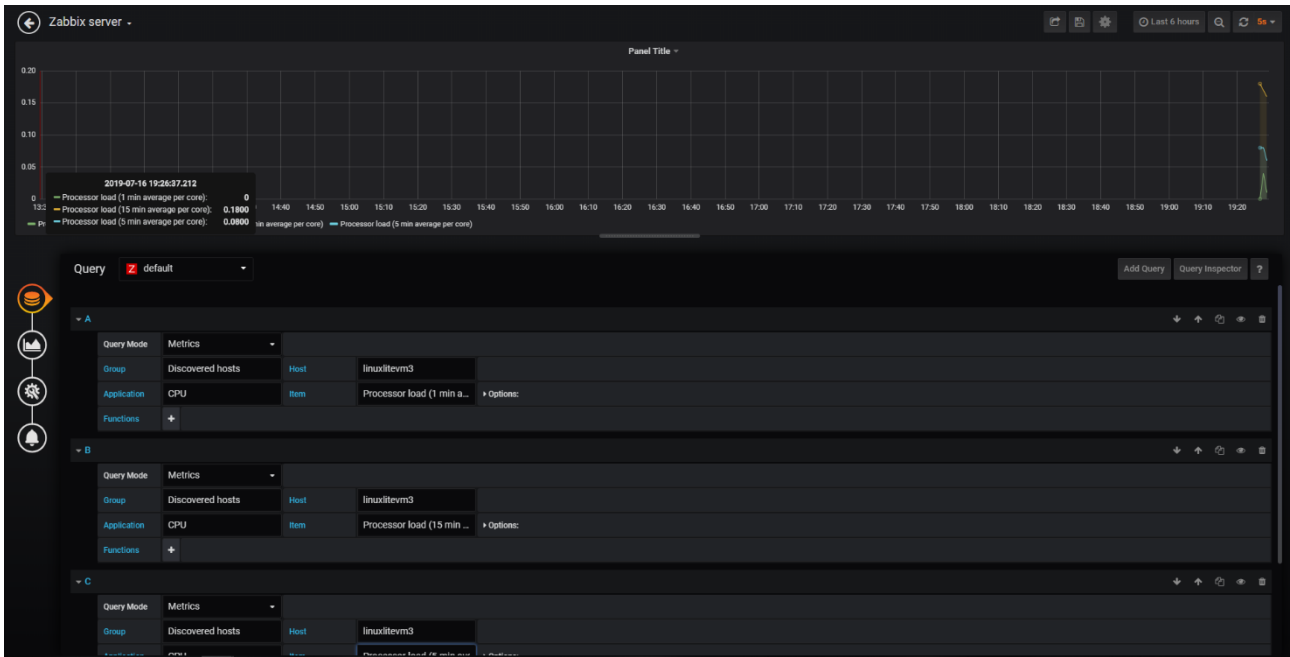


Figure 8-9 Grafana panel creation



Figure 8-10 Example of a Grafana panel of metrics

After the successful installation of Zabbix on all the VMs and Raspberry Pi’s used for the simulation, the various steps in the simulation of DFR for IoT can commence. The following subsection presents an overview of the various aspects of the simulation and how the various steps are conducted.

8.4.3 Differences between the various steps of the simulation

The simulation consists of four steps. Each step requires more resources from the IoT devices, as the simulation of a smart home and DFR for IoT are added to the IoT devices

in the various steps. The 4 steps last eight hours each, conducted over a long period of time to ensure that a good baseline for performance is established. The 8-hour timeframe allows for several iterations of proactive evidence gathering and a large number of simulated actions in a simulated smart home. Both the simulated and physical IoT devices were left completely undisturbed during the various steps to ensure that no extra processing occurred, which could hinder the accuracy of the metrics collected. No interaction with the IoT devices took place during the various steps of the simulation. The IoT devices did therefore not receive any additional instructions once the simulation started.

Step 1 took place from 22 July 2019 21:00 to 23 July 2019 05:00. This timeframe is referred to as Step 1 in the rest of this dissertation. Step 1 entails monitoring the IoT devices with zero programs running on the devices. This serves as a baseline to compare the various steps to and ensures that the impact of the various tools on the IoT devices is transparent. The physical IoT devices are therefore merely turned on for this step in the simulation.

Step 2 took place from 23 July 2019 09:00 to 23 July 2019 17:00. This timeframe is referred to as Step 2 of the simulation in the rest of this dissertation. Step 2 refers to keeping the prototype program on the IoT devices continuously active without any investigations and measuring how it impacts IoT device performance. The prototype program in this dissertation is the modified GRR client. The prototype program is therefore active during the eight hours in Step 2, but the prototype program is not gathering any forensic evidence. Step 2 merely measures the impact of an idle prototype program.

Step 3 took place from 23 July 2019 22:00 to 24 July 06:00. This timeframe is referred to as Step 3 of the simulation in the rest of this dissertation. Step 3 measured the impact of the smart home simulation continuously on the IoT device performance during this eight hours. This ensures that the main function of the IoT device is monitored. IoT devices need to gather and react upon sensor data. The data gathered from the sensors and actuators also need to be transferred to the internet. In Step 3, the smart home program is deactivated, as this step only analyses the impact of the smart home simulation program.

Step 4 took place from 24 July 2019 21:00 to 25 July 05:00. This timeframe is referred to as Step 4 of the simulation in the rest of this dissertation. In this step the smart home simulation program is run on all the IoT devices while proactively gathering forensic evidence using the custom GRR prototype. This is the most resource intensive step of the simulation. The goal is for the IoT devices to perform their usual tasks while proactively gathering evidence without sacrificing performance in their normal functioning. In Step 4 the smart home simulation and the GRR client is active throughout the 8-hour timeframe. The prototype program (modified GRR clients) furthermore conduct evidence gathering, which is more resource intensive when compared to Step 2 with only an idle prototype program. This means that in Step 4 evidence is generated by the smart home simulation program and the evidence is then gathered by the prototype program and transferred to the centralised repository.

The smart home simulation program used in the various steps of the simulation has several interesting characteristics to ensure that the simulation resembles a real-world scenario, and is therefore discussed in more detail in the next section. This ensures that the working of the smart home simulation program is better defined in the overhead context of the simulation of DFR for IoT.

8.4.4 Detailed information on the smart home simulation program developed for Step 3 and Step 4 of the simulation

In order to simulate the working of IoT devices, a Python smart home simulation program was developed. The Python program aims to simulate the working of IoT devices that gather data from sensors and react upon changes in the environment, for example sensors that detect the opening/closing of doors or the switching on/off of lights.

The Python program is installed on the VMs as well as the physical IoT devices. Due to the Python program being installed on the simulated IoT devices (VMs), not all the IoT devices have actual sensors that can be used to sense environmental changes. The VMs do not have the physical hardware pins to support the sensors used by the physical IoT devices in this simulation. This calls for simulated changes in the environment that result in simulated actions as a result of state changes. Simulated changes in this simulation are doors being opened and closed, which result in lights being turned on and off in the various rooms of the simulated smart home. The sensor data and actions taken by IoT

device actuators are therefore simulated. This is achieved by using mock objects for the sensor data and actuator responses. The physical IoT devices are connected to actual sensors as shown in Figure 8-11. The physical IoT devices connect to sensors in a physical model of a smart home with three rooms. Each of the three rooms has sensors which detect the presence of people inside the room. When the sensors detect the presence of someone in a room, the light in that room is turned on. The physical IoT device smart home model is discussed in detail further in this section. The physical IoT device is a Raspberry Pi which is shown in Figure 8-12.

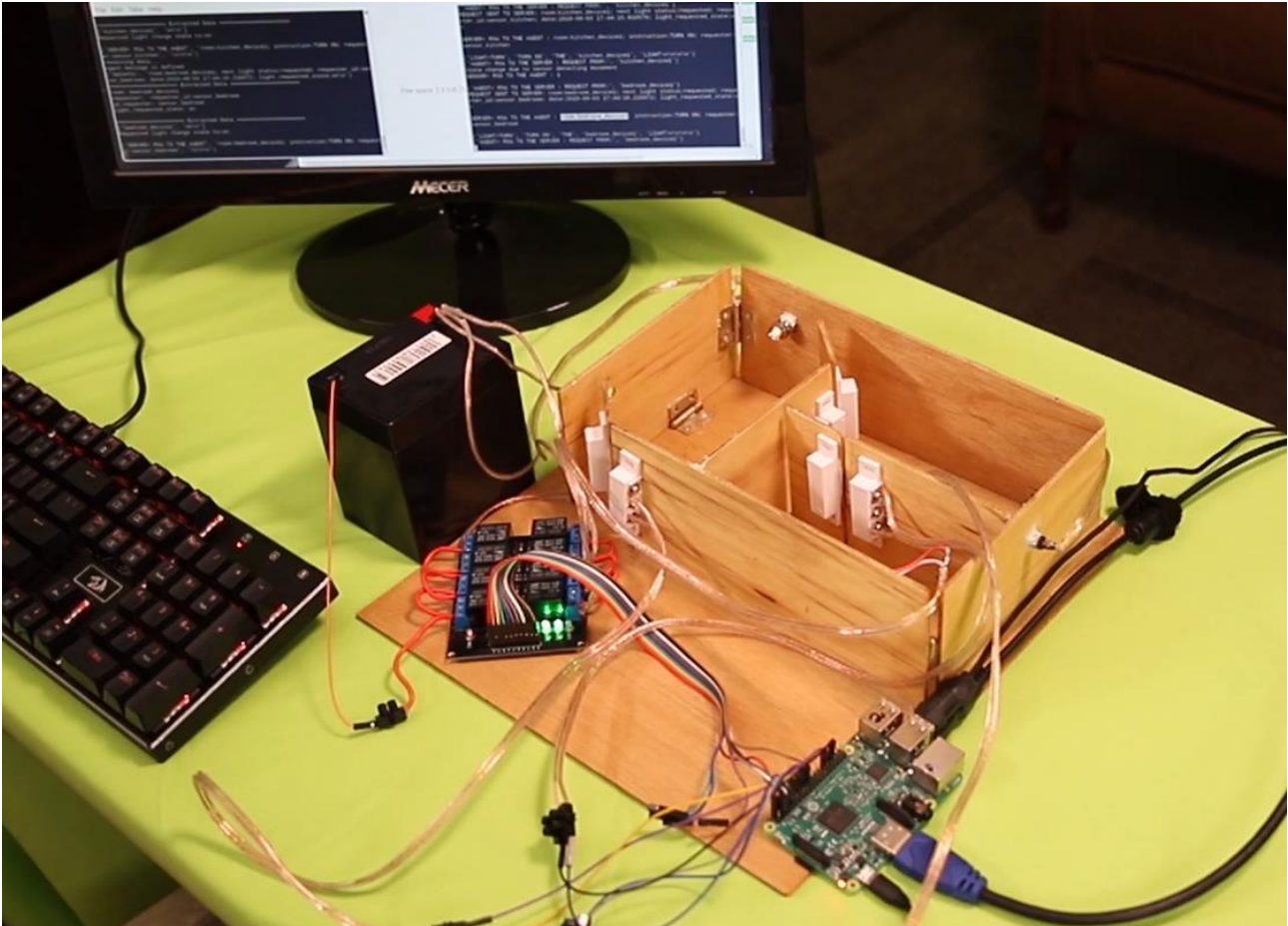


Figure 8-11 Physical IoT device smart home simulation



Figure 8-12 Raspberry Pi as physical IoT device

The Python program consists of two separate programs. One is the smart home server program and the other the smart home client. The smart home client is installed on the IoT devices that have sensors connected to them. The Python client interfaces with the sensors, logs the changes and notifies the Python server of these state changes. The Python server then receives the state changes, logs the changes and instructs the Python clients to react accordingly. In the physical smart home simulation, the smart home server and client is seen in Figure 8-11, and is indicated in Figure 8-13. Figure 8-11 and Figure 8-13 shows the two separate Python programs running on the Raspberry Pi. The program on the left is the smart home simulation server and the program on the right is the smart home simulation client.



Figure 8-13 Smart home program server and client

The simulated smart home program simulates IoT devices that detect movement inside various rooms of a house. The movement is registered through sensors on doors that detect when a door is opened or closed. For simplicity, an open door indicates the

presence of a person in a room, while a closed door indicates the absence of a person according to the assumption that when a person is in a room, the door will not be closed. Whenever a person is present in a room, the IoT device turns on the light in that specific room. In a real-world scenario the sensors would be movement sensors combined with door sensors to determine the presence or absence of people in the various rooms, but this is a limited testing environment with the assumptions stated above. The physical IoT device is connected to reed switches, which are magnetic switches that act as the sensors in the smart home simulation that indicate when doors in the model smart home are opened or closed. The sensors (magnetic reed switches) are indicated by the blue squares in Figure 8-14.

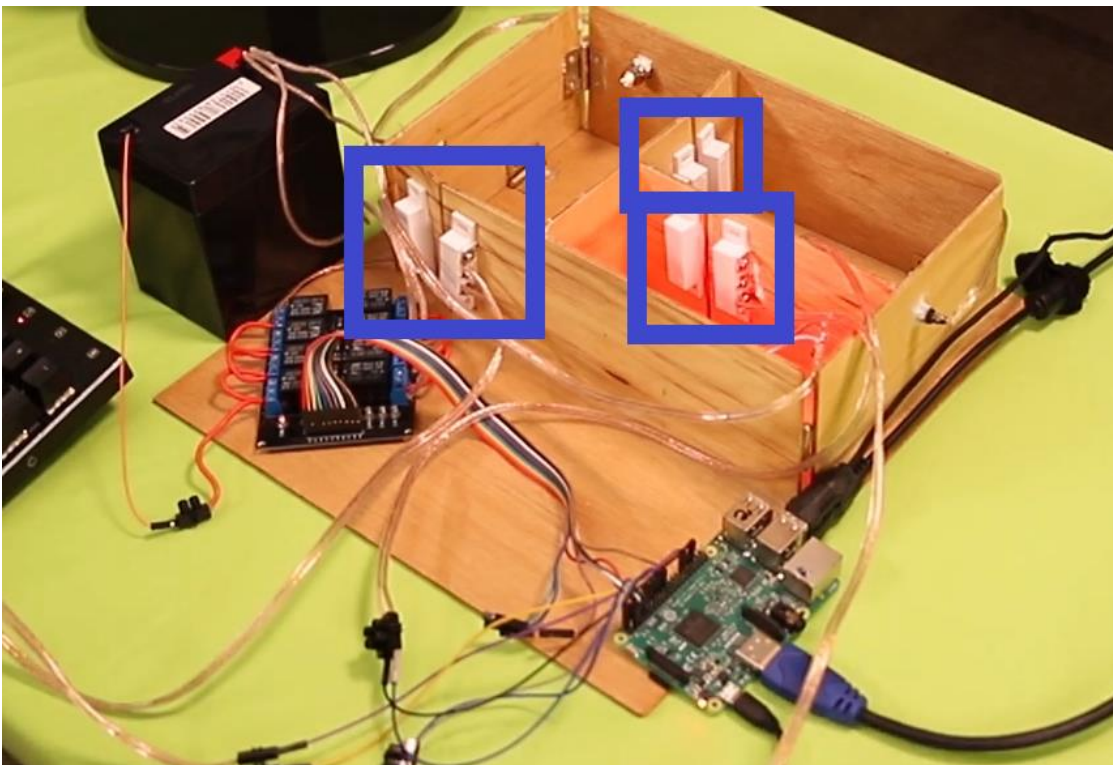


Figure 8-14 Magnetic reed switches used as sensors in physical smart home simulation

As soon as a door is opened, the Python smart home client registers the presence of someone in the room, logs the changes and notifies the Python smart home server of the state changes. The Python smart home server logs the changes in log files for each of the separate rooms. The server then responds by authorising an actuator to respond to the state change. In the case of the smart home simulation, it would be to turn the light in the room on or off. The lights that are turned on and off in the physical smart home simulation are shown inside blue squares in Figure 8-15.

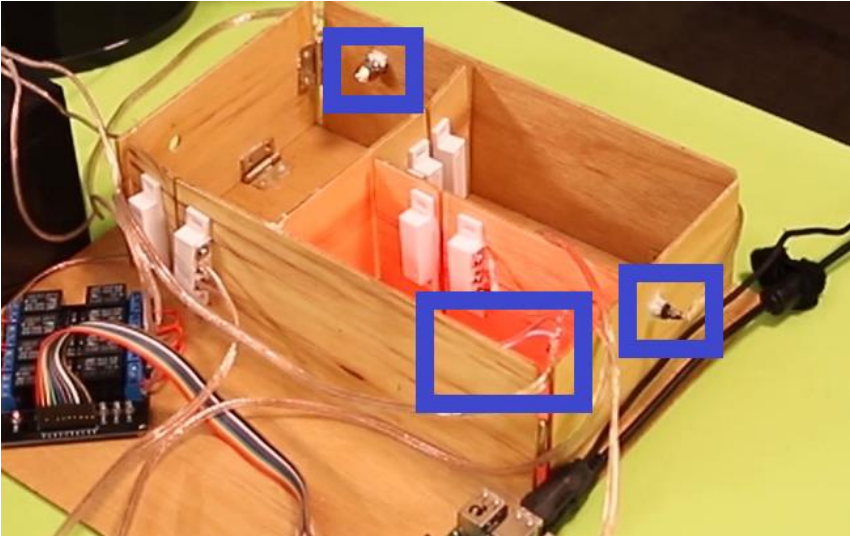


Figure 8-15 LED lights in physical smart home simulation

The simulation of the smart home program is done in a specific manner to ensure that the same sequence of events takes place across all the various IoT devices. The simulation is based on intervals of a few seconds of no activity in the smart home (doors are not opened or closed). The intervals of no activity is then followed by a state change (opening and closing of doors). The agent then transmits the state change and receives authorisation from the server to change the state of the light in the room. The process then repeats indefinitely.

The Python smart home simulation makes use of sockets to transfer data between the client and server. This aims to simulate the connection between real-world IoT sensors and IoT servers. The log files for the client and server are quite similar. In the client, all of the events are logged to a single log file. The log file contains a variety of information that can be used for the timelining of events. Figure 8-16 shows the detected state change in the physical environment being logged to the agent log file. The client then receives a response from the server with regards to the actuator (light) being turned on or off as a result of the sensed state change.

```
turn_on_file = open("agent_log.txt", "a")
turn_on_file.write(
    "room:{0}; next light status:requested; requester_id:{1}; date:{2}; light_requested_state:{3}\n".format(
        self.room_name, self.id_requester, date, next_state))
turn_on_file.close()
```

Figure 8-16 Client state change log

After some simulation of smart home operation, the agent log file contains quite a number of state change requests and responses from the smart home server. An example of the output of the agent log file is shown in Figure 8-17. The example from the log file shows the process of a light state change being requested by the agent due to a sensor being triggered. The agent then logs the request with the time of the request. An example of such a request is shown in line 1 of Figure 8-17.

The contents of line 1 is as follows: `room:livingroom_device1; next light status: requested; requester_id:sensor_livingRoom; date:2019-07-23 20:41:59.970178; light_requested_state:off`. The first part of line 1 (`room:livingroom_device1;`) indicates the room in which the state change request occurred. In this case it is the living room. The next part of line 1 (`next light status: requested;`) indicates that a new state change for the light is requested. The next part of line 1 (`requester_id:sensor_livingRoom;`) shows the id of the sensor that requested the state change. In this case it is the sensor with the id “`sensor_livingRoom`”. The reason for specifying the id of the sensor requesting the state change is that there might be more than one sensor in a specific room of a real smart home which can request state changes. The next part of line 1 (`date:2019-07-23 20:41:59.970178;`) indicates the date and time at which the state change request took place. This is extremely useful in the timelining of the sequence of events that occur during an incident. The final part of line 1 (`light_requested_state:off`) indicates the specific state that the sensor is requesting the light to change to. Therefore, line 1 indicates that the sensor with id `sensor_livingRoom`, requested a state change of the light in the room with id `livingroom_device1` to off at 2019-07-23 20:41:59.970178.

The server then responds with the authorisation of a state change of the light in the room where the sensor was triggered. The agent logs the response of the server and shows the state of the light in the room. An example of such an event occurs on line 2 of Figure 8-17. The process is repeated for various rooms in the smart home. The log file of the server is similar to that of the agent, with the only exception being that the state changes for the various rooms are logged in different files for the various rooms.

The contents of line 2 is as follows: `room:livingroom_device1; current light status:TURN OFF; requester_id:sensor_livingRoom; date:2019-07-23 20:41:59.982223`. The first part of line 2 (`room:livingroom_device1;`), indicates that the server has acknowledged a state

change in the room with id livingroom_device1. The next part of line 2 (current light status:TURN OFF;) indicates that the light has change state based on the request received and that the new state of the light is an off state. The next part of line 2 (requester_id:sensor_livingRoom;) indicates the id of the sensor that requested the state change. The final part of line 2 (date:2019-07-23 20:41:59.982223) indicates the time at which the state of the light in the room changed state.

```

1 room:livingroom_device1; next light status:requested; requester_id:sensor_livingRoom; date:2019-07-23 20:41:59.970178; light_requested_state:off
2 room:livingroom_device1; current light status:TURN OFF; requester_id:sensor_livingRoom; date:2019-07-23 20:41:59.982223
3 room:livingroom_device1; next light status:requested; requester_id:sensor_livingRoom; date:2019-07-23 20:42:09.993817; light_requested_state:on
4 room:livingroom_device1; current light status:TURN ON; requester_id:sensor_livingRoom; date:2019-07-23 20:42:09.997902
5 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-07-23 20:42:20.008995; light_requested_state:off
6 room:kitchen_device1; current light status:TURN OFF; requester_id:sensor_kitchen; date:2019-07-23 20:42:20.011116
7 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-07-23 20:42:30.021628; light_requested_state:on
8 room:kitchen_device1; current light status:TURN ON; requester_id:sensor_kitchen; date:2019-07-23 20:42:30.023837
9 room:bedroom_device1; next light status:requested; requester_id:sensor_bedroom; date:2019-07-23 20:42:40.027547; light_requested_state:off
10 room:bedroom_device1; current light status:TURN OFF; requester_id:sensor_bedroom; date:2019-07-23 20:42:40.029632
11 room:bedroom_device1; next light status:requested; requester_id:sensor_bedroom; date:2019-07-23 20:42:50.030281; light_requested_state:on
12 room:bedroom_device1; current light status:TURN ON; requester_id:sensor_bedroom; date:2019-07-23 20:42:50.032375
13 room:livingroom_device1; next light status:requested; requester_id:sensor_livingRoom; date:2019-07-23 20:43:00.042931; light_requested_state:off
14 room:livingroom_device1; current light status:TURN OFF; requester_id:sensor_livingRoom; date:2019-07-23 20:43:00.045026
15 room:livingroom_device1; next light status:requested; requester_id:sensor_livingRoom; date:2019-07-23 20:43:10.055523; light_requested_state:on
16 room:livingroom_device1; current light status:TURN ON; requester_id:sensor_livingRoom; date:2019-07-23 20:43:10.057767
17 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-07-23 20:43:20.068273; light_requested_state:off

```

Figure 8-17 Agent log file

8.4.5 Detailed information on the DFR setup of the GRR prototype for Step 4

The simulation of DFR for IoT requires a simulation of DF evidence being gathered in a simulated environment. In this simulation the Python smart home is used as the evidence generator, and the GRR prototype as the forensic tool that gathers evidence in a proactive manner. In order for this evidence to be considered trustworthy, mathematical hashes have to be created for the evidence. Hashes are the unique digital values that indicate that files are identical [116]. A hash is a one-way mathematical function that is created using the contents of a file. If any of the content of a file is changed, the hash value generated is completely different to the original. Two identical files, or digital evidence, generate the same hash values when the same hashing algorithm is used, indicating the trustworthiness of digital evidence [116].

The smart home program is separate to the GRR prototype. The GRR prototype serves as an add-on to existing IoT systems that do not support forensic investigations or DFR. The GRR prototype simulates how DFR can be facilitated for existing programs on IoT devices. programs usually result in changes to a system, which can be detected by forensic investigations. In this simulation the changes made by the smart home simulation are stored in log files. The GRR prototype therefore has to facilitate the proactive forensic acquisition of the smart home log files.

GRR is based on the idea of flows and hunts for forensic investigations. A flow is a forensic investigation launched on one specific host machine (machine with the GRR client installed). A hunt is a forensic investigation conducted across multiple host machines. In order to simulate DFR evidence gathering on IoT devices, the various agent log files need to be retrieved from the various IoT devices in the simulation (from VMs as well as from physical devices).

In order to establish the trustworthiness of the evidence gathered by the GRR hunts, the hashes of the various log files need to be created upon collection of the evidence. The GRR hunts also need to create timelines for the various versions of the evidence gathered from the IoT devices due to the log files continually changing with new state change data. Timelines show the various versions of the same log files collected over time. The smart home program updates the log files when doors are opened and closed, with the evidence collected therefore resulting in various versions of log files over time. The hashes for the different versions of the log are retained in the timeline. This ensures that older versions of a log file can still be used as evidence due to the availability of the hashes from the different versions of the log files.

Figure 8-18 shows a list of the GRR clients registered to the GRR server. Figure 8-18 shows the details of the three VMs as well as the two physical devices (Raspberry Pi's) used in the simulation. A label called "grr simulation" has been applied to all of the devices used in the simulation to limit the GRR hunts to only a subset of the IoT devices connected to this GRR server. The GRR hunts are customisable to only apply to devices that match certain criteria. This facilitates hunting for evidence across a subset of all of the GRR agents listed in the GRR server.

The simulation of DFR evidence gathering of IoT devices calls for a custom hunt to be developed in GRR. The hunt needs to run continually to gather the latest evidence from the various IoT devices. As mentioned earlier, the hunt needs to gather evidence from multiple IoT devices to show the ability of the platform to facilitate DFR for IoT devices on scale.

Online	Subject	OS Version	MAC	Unames	First Seen	Client version	Labels	Last Checkin	OS Install Date
<input type="checkbox"/>	C.2c2525f4f7fee134		00:00:00:00:00:00 00:0c:29:cc:f5:f7		2019-07-23 05:41:07 UTC	3246	grr-simulation	2019-07-26 19:42:02 UTC	2019-07-15 19:39:12 UTC
<input type="checkbox"/>	C.4554627910a37fae		00:00:00:00:00:00 b8:27:eb:dd:1d:16 b8:27:eb:88:48:43		2019-07-09 17:32:10 UTC	3249	raspberrypi2 physical-sensors grr-simulation	2019-08-05 18:48:57 UTC	2018-11-13 14:21:15 UTC
<input type="checkbox"/>	C.6e2f6e356217e288		00:00:00:00:00:00 b8:27:eb:0f:b9:65 b8:27:eb:5a:ec:30		2019-05-02 13:15:23 UTC	3249	raspberrypi1 grr-simulation	2019-08-05 18:51:52 UTC	2018-11-13 14:21:15 UTC
<input type="checkbox"/>	C.952bbc83e8815972		00:00:00:00:00:00 00:0c:29:9d:64:66		2019-07-23 05:40:02 UTC	3246	grr-simulation	2019-07-26 17:40:02 UTC	2019-07-11 13:00:09 UTC
<input type="checkbox"/>	C.aebc9f5f69c23171		00:00:00:00:00:00 00:0c:29:71:b4:7f		2019-07-23 05:41:23 UTC	3246	grr-simulation	2019-07-26 19:41:25 UTC	2019-07-16 17:04:49 UTC

Figure 8-18 GRR agents

In order to understand the working of the GRR server GUI better, Figure 8-19 shows a single GRR agent. Each of the columns represents some information about the specific GRR agent. In Figure 8-19 the green dot shows that the agent is powered on and connected to the GRR Server. The “Subject” column shows the id that GRR creates for the agent. The “OS Version” column is empty in this example, as the operating system used in the simulation is unfamiliar to the GRR platform. Normally the operating system used by GRR agents are some popular version of Linux or Windows. The “MAC” column shows the MAC address of the GRR agent. The “Unames” column is empty in this example, but when the GRR agent detects the usernames on the host machine, the usernames of all the users are listed in this column. The “First Seen” column indicates the date and time that the GRR agent initially contacted the GRR Server. The “Client Version” column indicates the version of the GRR agent installed on the IoT device. The “Labels” column shows custom labels that are applied to the agent, which are used for grouping GRR agents together. The “Last Checkin” column shows the last timestamp that the GRR agent contacted the GRR server to check for new hunt instructions. The final column, “OS Install Date” indicates the timestamp when the operating system was installed on the IoT device.

Online	Subject	OS Version	MAC	Unames	First Seen	Client version	Labels	Last Checkin	OS Install Date
<input checked="" type="checkbox"/>	C.2c2525f4f7fee134		00:00:00:00:00:00 00:0c:29:cc:f5:f7		2019-07-23 05:41:07 UTC	3246	grr-simulation	2019-07-26 19:42:02 UTC	2019-07-15 19:39:12 UTC

Figure 8-19 GRR agent details

A recurring hunt is known as a “cron hunt” in GRR. The DFR gathering of evidence can occur regularly, and as such the DFR for IoT GRR simulation is based on a cron hunt. Regular hunts only run when started by the investigator. Proactively gathering evidence from the various IoT devices means that evidence need to be collected regularly. Therefore, a GRR cron hunt is used. The cron hunt regularly collects the same log files, hashes them, and then creates entries in the timeline of the log file to show how the log

files change over time. The previous versions of the log files are hereby preserved in a trustworthy manner.

A new cron hunt called “GRR DFR for IoT” is created to show the working of the DFR for IoT simulation as shown in Figure 8-20. The name of the cron hunt used during the simulation is called “hunt download” but consist of the same parameters as the cron hunt created in this subsection.

The new cron hunt called “GRR DFR for IoT” is created over a few steps. The first step is to click on the plus icon in the “Cron Job Viewer” page of the GRR server web GUI shown in Figure 8-20.

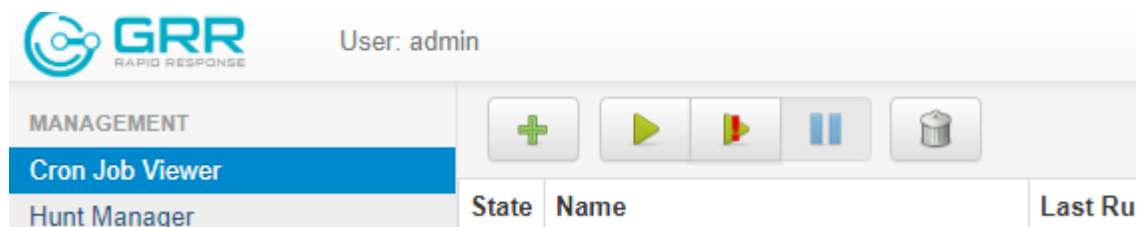


Figure 8-20 Create new GRR cron hunt

The occurrence and lifespan of the cron hunt is then defined as shown in Figure 8-18. The description is the name of the cron hunt and the periodicity the interval at which a new instance of the hunt is launched. In Figure 8-21 the periodicity of the cron hunt is 25 minutes. Therefore, new DFR evidence gathering takes place every 25 minutes. The evidence gathering takes place every 25 minutes in order to be able to easily distinguish time periods when the IoT devices gather evidence. The time frame of 25 minutes is therefore chosen to ensure significant differences in the log files, as the smart home simulation program generates several entries over this time period.

The lifetime of the cron hunt is set to an hour. This is done to ensure that all the IoT devices are able to conduct the evidence gathering of an individual cron hunt. Sometimes GRR agents poll the GRR server infrequently, which results in not all GRR agents conducting a new hunt precisely when it is created. Some GRR agents take a bit longer to receive the new hunt instructions. The 1-hour timeframe is sufficient in ensuring that all GRR agents receive the new hunt instructions and execute the evidence gathering. This also means that if a new GRR agent is added to the GRR server within the lifetime of

existing cron hunt, the new agent will also execute the hunt if it matches the parameters of the cron hunt. For example, if there are five GRR agents that satisfy the parameters for a hunt, all these agents will execute the hunt. If another GRR agent is added after a hunt is started, but still within the lifetime of the hunt, and the new GRR agent meets the parameters for the hunt, the new GRR agent will also execute the hunt. In the above-mentioned scenario, a total of six agents will then have executed the hunt at the end of the lifetime of the hunt. This shows that the GRR platform is able to handle expanding systems, such as IoT systems where new devices can be added at any given moment.

The “Allow overruns” option is enabled in Figure 8-21, which means that if a version of a cron hunt is still in execution and a new occurrence of the cron hunt is started, the old occurrence of the cron hunt is allowed to finish normally. This means that if, for example, a cron hunt is started and five GRR agents match the parameters for the hunt, five agents need to execute the hunt. However, if for some reason some agents are not able to execute the hunt before another instance of the same cron hunt is started, the agent(s) still need(s) to complete the previous instance of the cron hunt. A new instance of the same cron hunt can then be started regardless of whether previous instances of the cron hunt completed successfully. Should the GRR agent(s) then become able to complete the cron hunts, all the instances of the cron hunt are executed by the agent. Multiple cron hunts can therefore exist parallel to each other.

Schedule Cron Hunt - Cron Job properties

Step 1 out of 6

Description	GRR DFR of IoT
Periodicity	25m
Lifetime	1h
Advanced ▾	
Allow overruns	<input checked="" type="checkbox"/>

Figure 8-21 Schedule GRR cron hunt

The next step in setting up the cron hunt is the details of the hunt itself as shown in Figure 8-19. In this DFR for IoT simulation the “File Finder” GRR artefact is used. The “File Finder” artefact locates files based on certain paths. Actions can then be defined for the file when the file is located on the GRR client machines. In Figure 8-22 the path for the “File Finder” artefact is “/grr-home-automation/agent_log.txt”. This is the location of the Python smart home agent log file on the various IoT devices. The action defined in Figure 8-22 is “download”, which creates a hash of the file(s) found on the GRR agent(s) and retrieves them from the agent(s). This represents the application of DFR for IoT devices, as the evidence is gathered before the potential occurrence of an incident. The hashes of the log files are available, as well as the log files itself. This is shown later in this section.

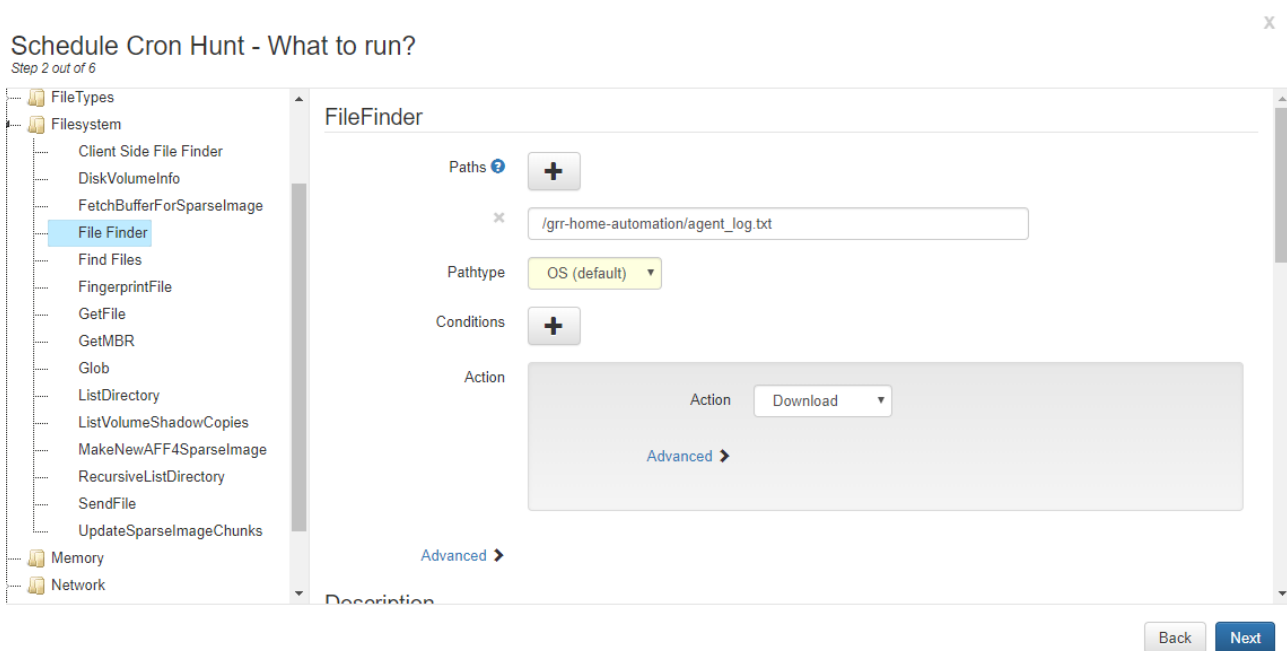


Figure 8-22 Cron hunt parameters

The next part of the cron hunt configuration is determining which GRR hosts need to execute the cron hunt. In Figure 8-23 the cron hunt is configured to only run on GRR agents with the “grr simulation” label, in this case the VMs and physical devices.

Schedule Cron Hunt - Where to run?

Step 4 out of 6

Match mode Match all (default) ▾

Rules +

×

Rule type Label ▾

Match mode Match all (default) ▾

Add label +

Label name grr simulation ▾ ×

Back Next

Figure 8-23 GRR agents to execute cron hunt

The final step in the cron hunt configuration shows all the configured parameters of the cron hunt. Figure 8-24 shows the configured parameters of the “hunt download” cron hunt, which is the actual cron hunt used in the simulation of DFR for IoT. The “hunt download” cron hunt is therefore the cron hunt that was executed during the monitoring of the performance of the various IoT devices in Step 4 of the simulation.

Details		Runs	
ID	FileFinder_48913		
Cron Job ID	FileFinder_48913		
Description	hunt download		
Enabled	false		
Frequency	25m		
Last Run Time	2019-08-01 18:38:24 UTC		
Max Iteration Lifetime	1h		
Allow Overruns	false		
Cron Arguments	Action type HUNT_CRON_ACTION		
Hunt cron action	Hunt runner args	Description	hunt download (cron)
		Client rule set	Rules
		Crash Limit	100
		Average Results-per-Client Limit	1000
		Average CPU-Usage-per-Client Limit (seconds)	60
		Average Network-Usage-per-Client Limit (bytes)	10485760
		Flow name	FileFinder
		Paths	/grr-home-automation/agent_log.txt
		Action	DOWNLOAD
		Download	

Figure 8-24 Simulation cron hunt parameters

To illustrate the working of the “hunt download” GRR cron hunt, evidence gathered by the hunt is shown in the following paragraphs. Step 4 of the simulation collected the agent log files from the various IoT devices. Due to the Python smart home simulation executing continuously on the various IoT devices, the same log file for a specific IoT device changes over time. The GRR cron hunt must show this difference in the evidence collected.

The overview of a GRR hunt is shown in the “*Hunt Manager*” page of the GRR web GUI. Figure 8-25 shows an example of one of the instances of the “*hunt download*” cron hunt executed during Step 4 of the simulation. The overview shows that the hunt was completed on a total of five clients, which is the total of all the VMs and physical devices used in the simulation. The hunt therefore completed the DFR evidence gathering on all the IoT devices in the simulation. Figure 8-25 also shows that the cron hunt resulted in 711,6KiB of network traffic and used a second of CPU time in total, which is extremely low.

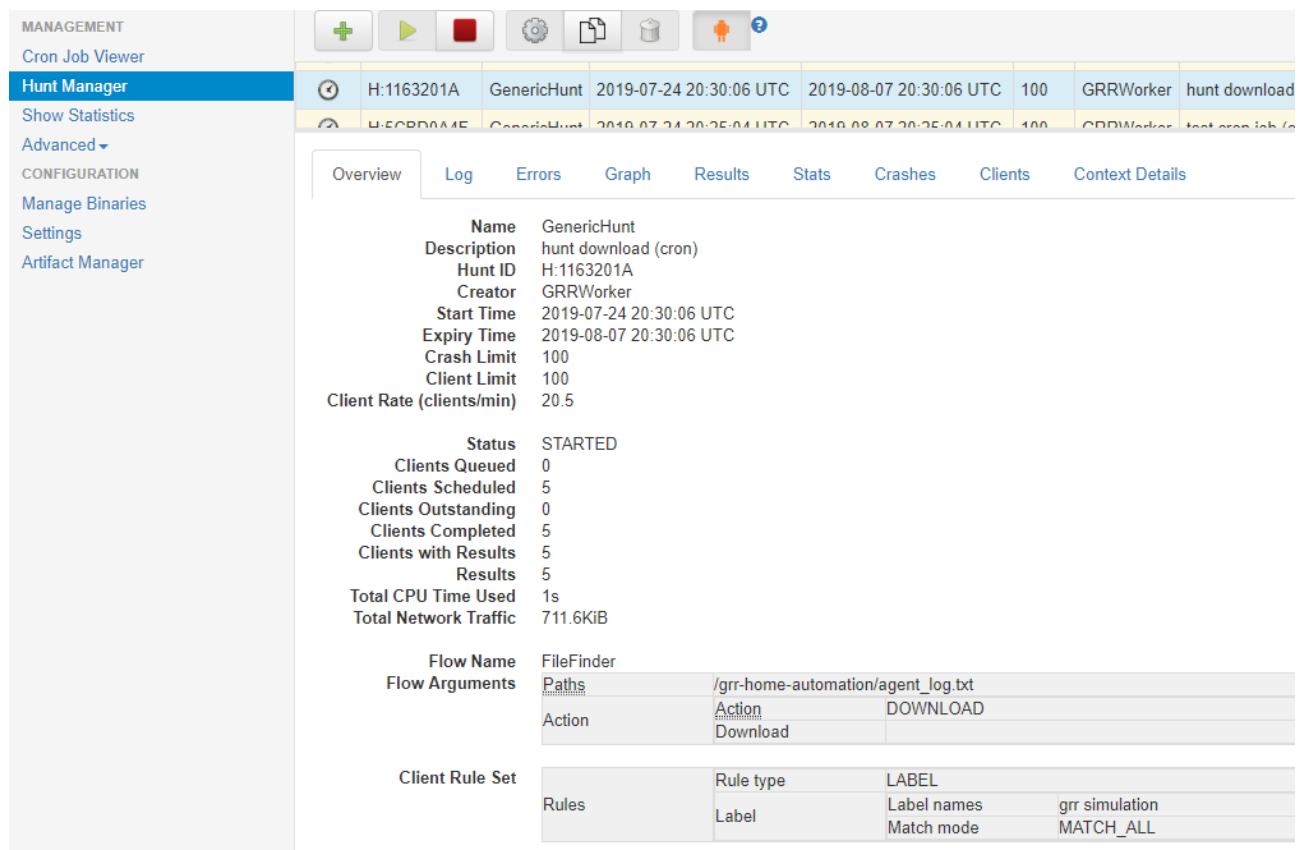
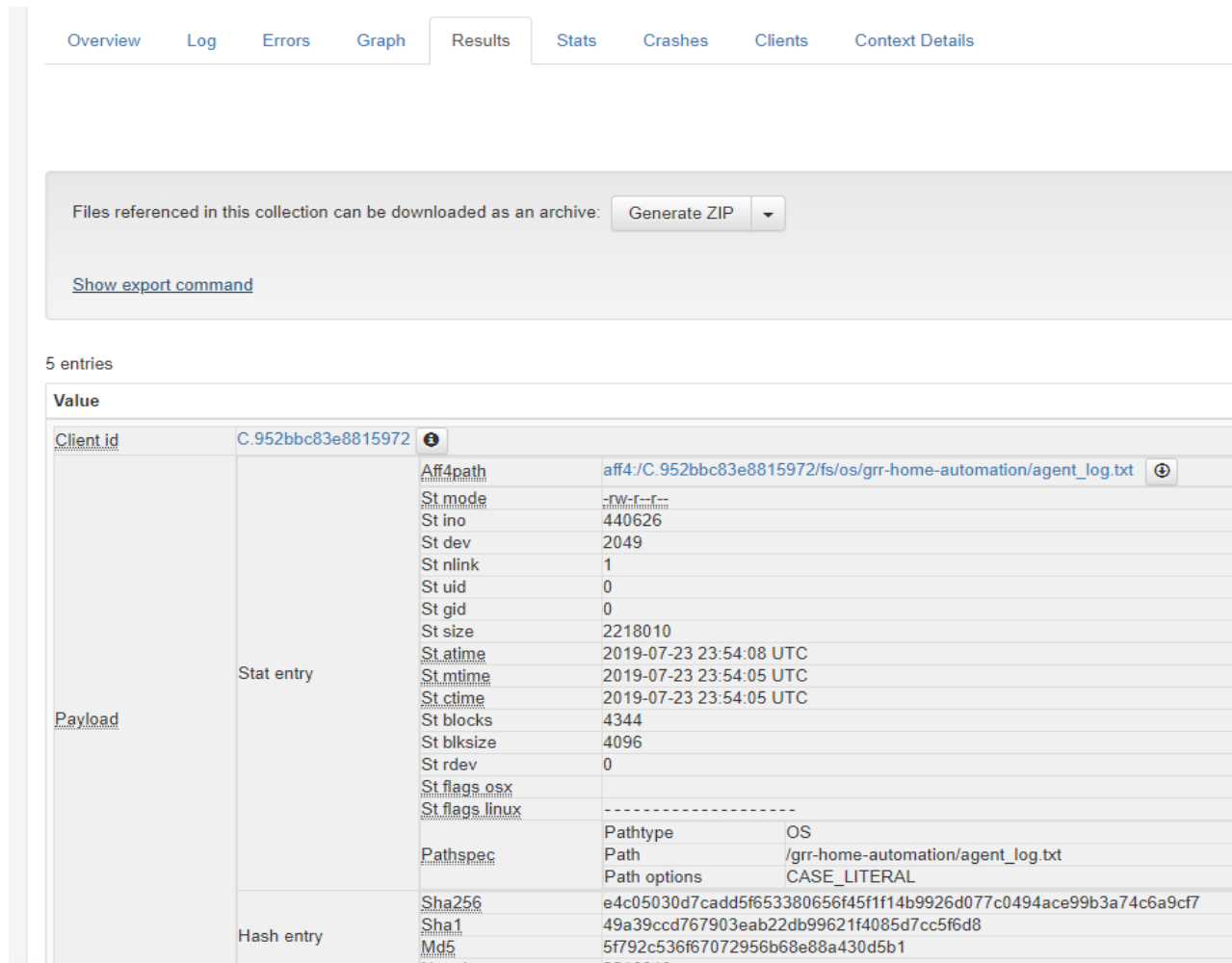


Figure 8-25 Cron hunt overview

The results tab shows the details of the evidence gathered by the GRR hunt from all the GRR clients that participated in the hunt. Figure 8-26 shows the client id of the host from which the evidence is gathered, as well as the various attributes of the log file retrieved from the IoT device. Hashes for the evidence collected are also shown. The evidence can be downloaded by clicking on the download icon shown in Figure 8-27.



Files referenced in this collection can be downloaded as an archive: [Generate ZIP](#)

[Show export command](#)

5 entries

Value																													
<u>Client id</u>	C.952bbc83e8815972 ⓘ																												
<u>Payload</u>	<table border="1"> <tr> <td><u>Aff4path</u></td> <td>aff4:/C.952bbc83e8815972/fs/os/grr-home-automation/agent_log.txt ⓘ</td> </tr> <tr> <td><u>St mode</u></td> <td>-rw-r--r--</td> </tr> <tr> <td><u>St ino</u></td> <td>440626</td> </tr> <tr> <td><u>St dev</u></td> <td>2049</td> </tr> <tr> <td><u>St nlink</u></td> <td>1</td> </tr> <tr> <td><u>St uid</u></td> <td>0</td> </tr> <tr> <td><u>St gid</u></td> <td>0</td> </tr> <tr> <td><u>St size</u></td> <td>2218010</td> </tr> <tr> <td><u>St atime</u></td> <td>2019-07-23 23:54:08 UTC</td> </tr> <tr> <td><u>St mtime</u></td> <td>2019-07-23 23:54:05 UTC</td> </tr> <tr> <td><u>St ctime</u></td> <td>2019-07-23 23:54:05 UTC</td> </tr> <tr> <td><u>St blocks</u></td> <td>4344</td> </tr> <tr> <td><u>St blksize</u></td> <td>4096</td> </tr> <tr> <td><u>St rdev</u></td> <td>0</td> </tr> </table>	<u>Aff4path</u>	aff4:/C.952bbc83e8815972/fs/os/grr-home-automation/agent_log.txt ⓘ	<u>St mode</u>	-rw-r--r--	<u>St ino</u>	440626	<u>St dev</u>	2049	<u>St nlink</u>	1	<u>St uid</u>	0	<u>St gid</u>	0	<u>St size</u>	2218010	<u>St atime</u>	2019-07-23 23:54:08 UTC	<u>St mtime</u>	2019-07-23 23:54:05 UTC	<u>St ctime</u>	2019-07-23 23:54:05 UTC	<u>St blocks</u>	4344	<u>St blksize</u>	4096	<u>St rdev</u>	0
	<u>Aff4path</u>	aff4:/C.952bbc83e8815972/fs/os/grr-home-automation/agent_log.txt ⓘ																											
	<u>St mode</u>	-rw-r--r--																											
	<u>St ino</u>	440626																											
	<u>St dev</u>	2049																											
	<u>St nlink</u>	1																											
	<u>St uid</u>	0																											
	<u>St gid</u>	0																											
	<u>St size</u>	2218010																											
	<u>St atime</u>	2019-07-23 23:54:08 UTC																											
	<u>St mtime</u>	2019-07-23 23:54:05 UTC																											
	<u>St ctime</u>	2019-07-23 23:54:05 UTC																											
	<u>St blocks</u>	4344																											
	<u>St blksize</u>	4096																											
<u>St rdev</u>	0																												
<u>Stat entry</u>																													
<u>Hash entry</u>	<table border="1"> <tr> <td><u>Sha256</u></td> <td>e4c05030d7cadd5f653380656f45f1f14b9926d077c0494ace99b3a74c6a9cf7</td> </tr> <tr> <td><u>Sha1</u></td> <td>49a39ccd767903eab22db99621f4085d7cc5f6d8</td> </tr> <tr> <td><u>Md5</u></td> <td>5f792c536f67072956b68e88a430d5b1</td> </tr> </table>	<u>Sha256</u>	e4c05030d7cadd5f653380656f45f1f14b9926d077c0494ace99b3a74c6a9cf7	<u>Sha1</u>	49a39ccd767903eab22db99621f4085d7cc5f6d8	<u>Md5</u>	5f792c536f67072956b68e88a430d5b1																						
<u>Sha256</u>	e4c05030d7cadd5f653380656f45f1f14b9926d077c0494ace99b3a74c6a9cf7																												
<u>Sha1</u>	49a39ccd767903eab22db99621f4085d7cc5f6d8																												
<u>Md5</u>	5f792c536f67072956b68e88a430d5b1																												
<u>Pathspec</u>	<table border="1"> <tr> <td><u>Pathtype</u></td> <td>OS</td> </tr> <tr> <td><u>Path</u></td> <td>/grr-home-automation/agent_log.txt</td> </tr> <tr> <td><u>Path options</u></td> <td>CASE_LITERAL</td> </tr> </table>	<u>Pathtype</u>	OS	<u>Path</u>	/grr-home-automation/agent_log.txt	<u>Path options</u>	CASE_LITERAL																						
<u>Pathtype</u>	OS																												
<u>Path</u>	/grr-home-automation/agent_log.txt																												
<u>Path options</u>	CASE_LITERAL																												

Figure 8-26 GRR hunt results

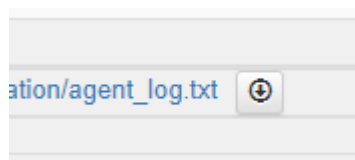
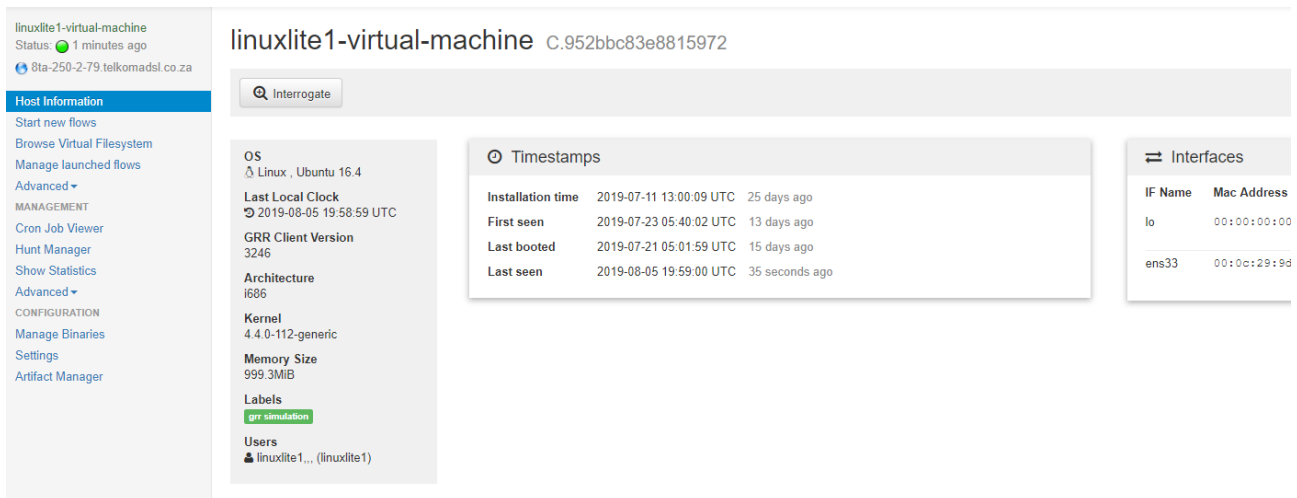


Figure 8-27 Download evidence

Timelining of the specific evidence file is possible by navigating to the file on the virtual file system of the specific client. This is done by clicking on the client id (Figure 8-26). The details about the specific GRR client is then shown, as in Figure 8-28. Next, one can click

on “Browse Virtual Filesystem” for the specific IoT device to locate the evidence file. This enables the acquisition of the agent log file that is in the directory where the Python smart home client logs all state changes. This is shown in Figure 8-29.



linuxlite1-virtual-machine C.952bbc83e8815972

Interrogate

OS
Linux, Ubuntu 16.4
Last Local Clock
2019-08-05 19:58:59 UTC
GRR Client Version
3246
Architecture
i686
Kernel
4.4.0-112-generic
Memory Size
999.3MiB
Labels
grr simulation
Users
linuxlite1, (linuxlite1)

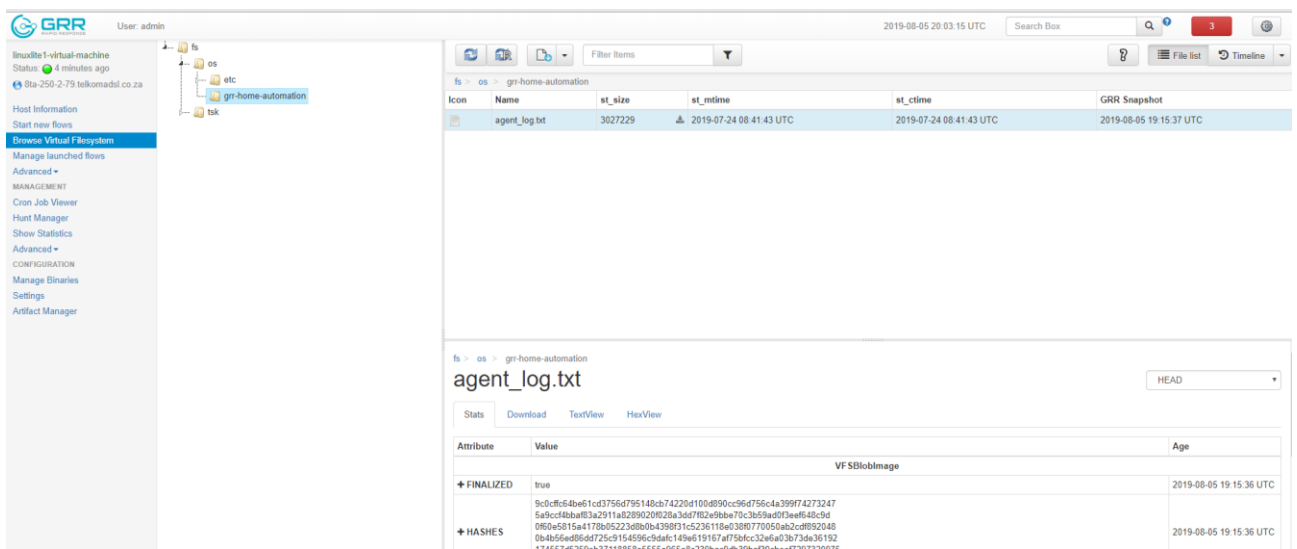
Timestamps

Installation time	2019-07-11 13:00:09 UTC	25 days ago
First seen	2019-07-23 05:40:02 UTC	13 days ago
Last booted	2019-07-21 05:01:59 UTC	15 days ago
Last seen	2019-08-05 19:59:00 UTC	35 seconds ago

Interfaces

IF Name	Mac Address
lo	00:00:00:00:00:00
ens33	00:0c:29:9d:...

Figure 8-28 GRR client details



fs > os > grr-home-automation

Icon	Name	st_size	st_mtime	st_ctime	GRR Snapshot
📄	agent_log.txt	3027229	2019-07-24 08:41:43 UTC	2019-07-24 08:41:43 UTC	2019-08-05 19:15:37 UTC

fs > os > grr-home-automation

agent_log.txt

Stats Download TextView HexView

Attribute	Value	Age
VF SBlobImage		
+ FINALIZED	true	2019-08-05 19:15:36 UTC
+ HASHES	5c0c4c54be61cd3756d79f148cb74220d1004890cc96d756c4a399f74273247 5a9ccf4bbaf83a2911a8289020f028a3dd7802e9bbe70c3b59ad0f3ee46c9d 080e5815a4178b05223d8b0b4398f31c5236118e0380770050a02cdf892048 0b4856e288d0725c9154596c9d0a149a019167a77590cc32e6a03673da36192 1746274c765a43111880c4e2e2c0d5c1f710a4c46b10b4a4075075967e	2019-08-05 19:15:36 UTC

Figure 8-29 Virtual file system of evidence file

The timeline of all the versions of the file gathered by the cron hunts can be viewed by clicking on the file’s “Timeline” button (Figure 8-30). This will show the different versions of the file collected by the various cron hunts. Figure 8-31 shows the timeline for the agent log of one of the physical devices. The cron hunt creates various instances at the same timestamp, which can be seen inside the red square in Figure 8-31. The rest of the timeline is presented, but unfortunately does not all fit into one screenshot. The evidence can be downloaded by clicking on the “Download x bytes” button shown in Figure 8-32.

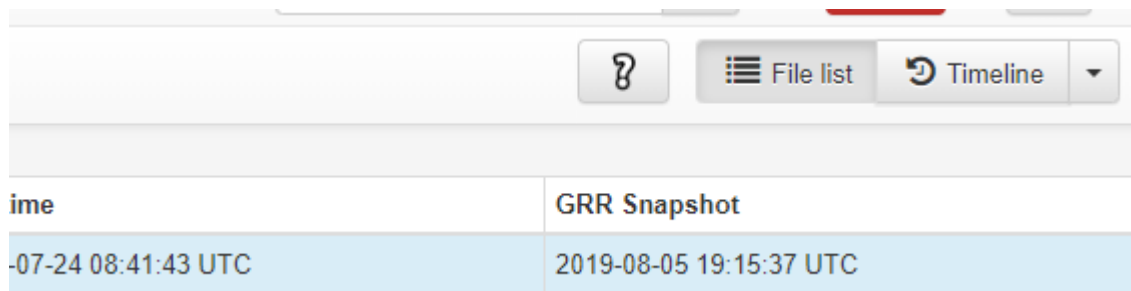


Figure 8-30 Timeline of the agent log file

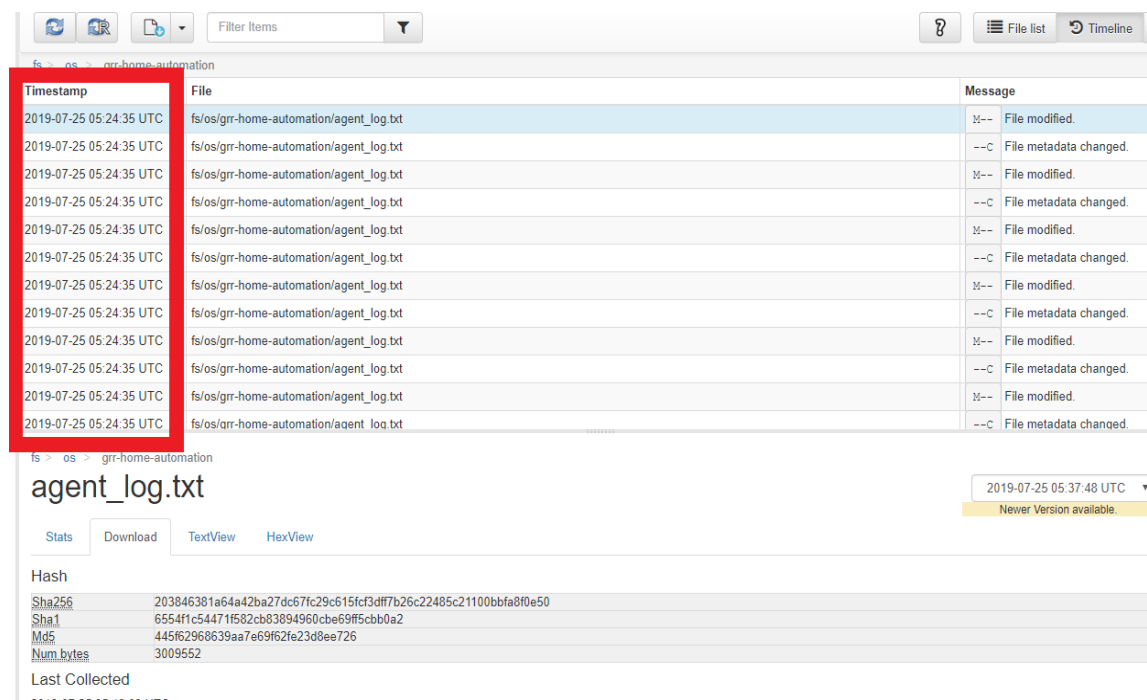


Figure 8-31 Evidence from Raspberry Pi timeline

Md5	445f62968639aa7e69
Num bytes	3009552

Last Collected

2019-07-25 05:13:03 UTC

Download

Download (3019800 bytes)

Figure 8-32 Download evidence file

The validity of the evidence file can be determined by comparing the hash from the GRR server with a hash checksum. The GRR server indicates the SHA256 hash to be 203846381a64a42ba27dc67fc29c615fcf3dff7b26c22485c21100bbfa8f0e50, as shown in Figure 8-31. An online SHA256 file checksum application is used to determine the SHA256 hash of the file [117]. Any other SHA256 application (whether it is an online or installable app) could have been used. The online SHA256 file checksum also calculates the SHA256 hash as

203846381a64a42ba27dc67fc29c615fcf3dff7b26c22485c21100bbfa8f0e50, as is shown in Figure 8-33. Due to the two hashes that are clearly the same, the content of the agent log file is indeed that of the Python smart home agent as is shown in Figure 8-34.

As more data is added to the log file, the hash values will obviously change. For example, as shown in Figure 8-35, the SHA256 hash for the file changed to 11ff1103bbcd4a674b764e89d8b587d28056c180721e2f81f225949b8bddba79, as the Python smart home simulation changed the content of the agent log file at time stamp 05:24:35 UTC.

Online Tools

SHA256 File Checksum

SHA256 online hash file checksum function



Hash	File Hash
CRC-16	CRC-16
CRC-32	CRC-32
MD2	MD2
MD4	MD4
MD5	MD5
SHA1	SHA1
SHA224	SHA224
SHA256	SHA256
SHA384	SHA384
SHA512	SHA512
SHA512/224	SHA512/224
SHA512/256	SHA512/256
SHA3-224	SHA3-224
SHA3-256	SHA3-256
SHA3-384	SHA3-384
SHA3-512	SHA3-512
Keccak-224	Keccak-224
Keccak-256	Keccak-256
Keccak-384	Keccak-384

Figure 8-33 SHA256 hash verify

```

18676 room:kitchen_devicel; current light status:TURN ON; requester_id:sensor_kitchen; date:2019-07-25 00:22:45.053570
18677 room:bedroom_devicel; next light status:requested; requester_id:sensor_bedroom; date:2019-07-25 00:22:55.064117; light_requested_state:off
18678 room:bedroom_devicel; current light status:TURN OFF; requester_id:sensor_bedroom; date:2019-07-25 00:22:55.066227
18679 room:bedroom_devicel; next light status:requested; requester_id:sensor_bedroom; date:2019-07-25 00:23:05.076739; light_requested_state:on
18680 room:bedroom_devicel; current light status:TURN ON; requester_id:sensor_bedroom; date:2019-07-25 00:23:05.078604
18681 room:livingroom_devicel; next light status:requested; requester_id:sensor_livingRoom; date:2019-07-25 00:23:15.089141; light_requested_state:off
18682 room:livingroom_devicel; current light status:TURN OFF; requester_id:sensor_livingRoom; date:2019-07-25 00:23:15.091105
18683 room:livingroom_devicel; next light status:requested; requester_id:sensor_livingRoom; date:2019-07-25 00:23:25.101584; light_requested_state:on
18684 room:livingroom_devicel; current light status:TURN ON; requester_id:sensor_livingRoom; date:2019-07-25 00:23:25.103644
18685 room:kitchen_devicel; next light status:requested; requester_id:sensor_kitchen; date:2019-07-25 00:23:35.114145; light_requested_state:off
18686 room:kitchen_devicel; current light status:TURN OFF; requester_id:sensor_kitchen; date:2019-07-25 00:23:35.116162
18687 room:kitchen_devicel; next light status:requested; requester_id:sensor_kitchen; date:2019-07-25 00:23:45.126688; light_requested_state:on
18688 room:kitchen_devicel; current light status:TURN ON; requester_id:sensor_kitchen; date:2019-07-25 00:23:45.128448
18689 room:bedroom_devicel; next light status:requested; requester_id:sensor_bedroom; date:2019-07-25 00:23:55.139045; light_requested_state:off
18690 room:bedroom_devicel; current light status:TURN OFF; requester_id:sensor_bedroom; date:2019-07-25 00:23:55.141235
  
```

Figure 8-34 Content of agent log file evidence

File Name	Size	Downloaded	Collected	Uploaded	Size
agent.pyc	7117		2019-07-23 18:08:00 UTC	2019-07-23 18:08:00 UTC	20
agent_log.txt	3037591		2019-07-25 05:24:35 UTC	2019-07-25 05:24:35 UTC	20
main_agent.py	2844		2019-07-23 18:30:24 UTC	2019-07-23 18:30:24 UTC	20
main_server.py	403		2019-07-23 17:31:15 UTC	2019-07-23 17:31:15 UTC	20
server.py	7058		2019-07-23 18:27:05 UTC	2019-07-23 18:27:05 UTC	20
server.pyc	3854		2019-07-23 18:27:08 UTC	2019-07-23 18:27:08 UTC	20

fs > os > grr-home-automation

agent_log.txt

[Stats](#)
[Download](#)
[TextView](#)
[HexView](#)

Hash

Sha256	11ff1103bbcd4a674b764e89d8b587d28056c180721e2f81f225949b8bddba79
Sha1	b8fca3643f45f667456103a94a61b38c80fd324f
Md5	261a354c23939de45a05e2d9d572d33d
Num bytes	3037591

Last Collected

Figure 8-35 Different version of evidence collected

The cron hunt collects evidence from the various IoT devices at a predefined time interval. It enables the collection of evidence from the various IoT devices in the form of their respective log files. The IoT evidence collected from the various IoT devices is also hashed to ensure its trustworthiness.

This process shows the working of the GRR prototype. It also shows that the evidence gathered reflects the working of a simulated smart home system, due to the logs collected from the various IoT devices showing the events from the smart home system. The GRR prototype thus facilitates proactive forensic collection of evidence – that is, DFR – on IoT. This section showed the setup of the GRR cron hunts in order to demonstrate the evidence collection conducted during Step 4 of the simulation.

8.5 FINDINGS OF SIMULATING DIGITAL FORENSIC READINESS FOR THE INTERNET OF THINGS

This section provides an overview of the findings from simulating the DFR for IoT devices and is divided in subsections for the various metrics collected during the steps of the simulation. The metrics are grouped into various overhead performance categories, namely processing related metrics (CPU), storage metrics (disk usage), memory metrics (RAM), and network metrics (data transferred to and from the IoT devices).

There are some important differences between the VMs and physical IoT devices that need to be highlighted before the presentation of the findings. Firstly, the VMs and physical IoT devices use two different operating systems. Secondly, while the VMs only have a single CPU core per VM, the physical IoT devices have four CPU cores per device. Finally, the VMs are based on the x86 architecture, where the physical IoT devices are based on the ARM7 architecture.

The VMs and physical IoT devices are identical in one aspect: they both have 1GB RAM each. As mentioned earlier in this section, the various differences between the VMs and physical devices call for the separation of graphs for the various metrics. The graphs with the performance metrics for this chapter can be found in Appendix C. Example of some graphs are provided in this chapter, but all the graphs can be found in the respective appendices of this study. This chapter also presents a discussion of the findings from the various metric graphs.

Each of the various analysed metrics results in two graphs per step. One graph is used to represent the performance of the metric for the IoT VMs, and the other graph is used to represent the performance of the physical IoT devices. This ensures easy comparison between the performance of the VMs and physical IoT devices. The X-axes of the graphs represent the amount of data points over time. Due to the simulation only incorporating two physical IoT devices and three VMs, there are more data points collected about the VMs. Thus, for every second of data collected, two points of data are collected for the physical IoT devices and three points of data are collected for the VMs. This means that there are more scatter points in the graphs showing performance metrics from the VMs. The graphs with VM metrics therefore have more values on the X-axis. The metrics from the various devices are also shown as one series in the graphs. This facilitates a better overview of the averages of the various metrics. An example of a graph based on one of the various metrics is presented in Figure 8-36

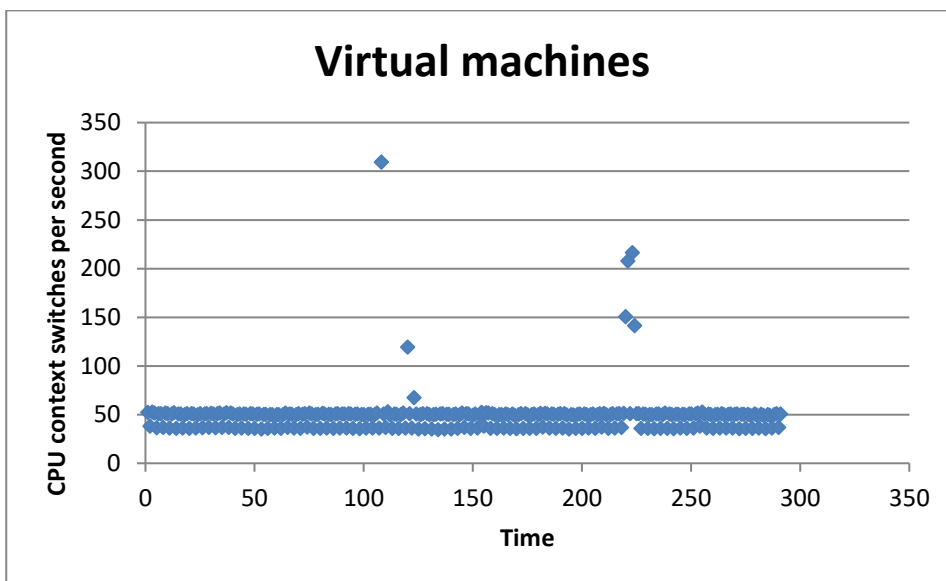


Figure 8-36 Step 1's CPU context switches per second for the virtual machines

8.5.1 Processing metrics

This subsection focuses on metrics related to the processing power of the IoT devices. The metrics analysed by Zabbix that relate to processing power are CPU context switches per second, CPU idle time, CPU interrupts per second, CPU load 1min average per core, CPU system time, number of processes and number of running processes.

CPU context switches

The VMs' CPU context switches over the four steps show that the number of context switches did not increase significantly over the course of the steps. However, it is interesting to note that the context switches increased slightly in Step 2 and 4, which indicates that the GRR prototype requires some context switches to operate. While the CPU context switches spiked periodically in Step 4 (timeframes where evidence was gathered) they were still not extreme.

The CPU context switches for the physical devices were generally more than those of the VMs over the four steps, most likely due to the different operating systems. The Raspberry Pi's also have GPIO pins that are regularly polled for inputs, which could increase CPU context changes. Similar to the CPU context switches of the VMs, the GRR prototype increases CPU context switches slightly over the four steps. When the GRR clients perform evidence gathering in Step 4, the amount of CPU context switches spike periodically. This is shown in Figure 8-37.

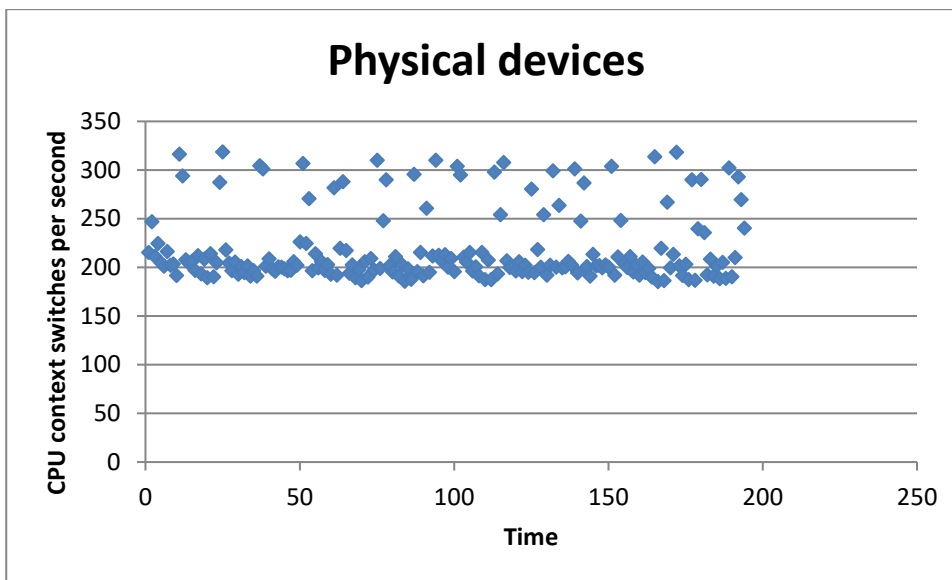


Figure 8-37 Step 4's CPU context switches per second for physical devices

CPU idle time

The CPU idle time for the VMs remained mostly constant throughout the various steps in the simulation. This is of importance to the simulation, as it indicates that the various tools and the GRR prototype did not negatively impact the processing performance of the IoT devices. The CPU idle time did have minor instances where it dropped, but these events were not frequent.

The CPU idle time for the physical devices were more spread out, but still between 99-100%. The physical devices indicated more frequent interrupts in CPU idle time, but the CPU idle time never fell below 99%. This goes to show that the physical IoT devices were more interrupted, but still not disruptive to the operation of the physical devices. The most dramatic drops in CPU idle time were in Step 4, which indicates that the GRR prototype did impact the overall CPU idle time, but it was minimal due to still resulting in above 99% CPU idle time. This is shown in Figure 8-38.

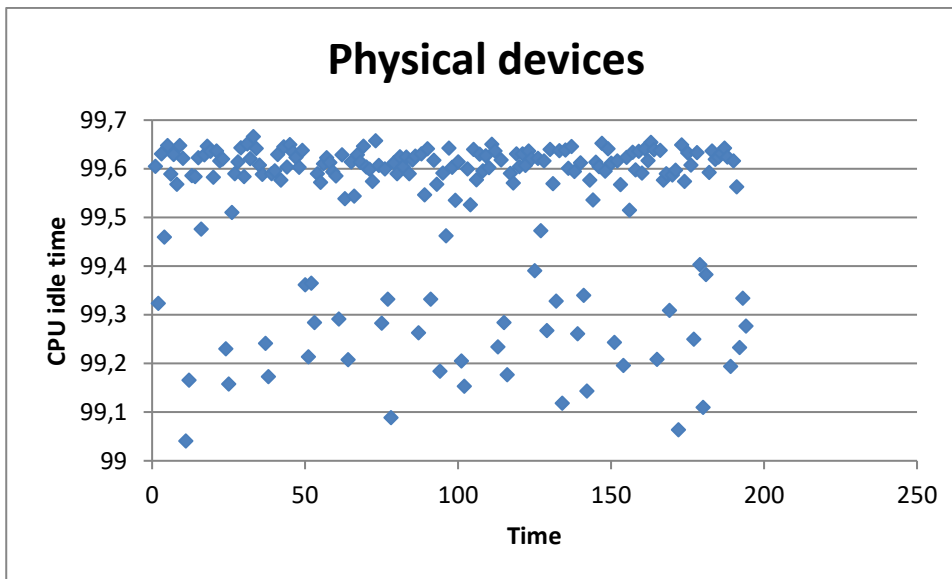


Figure 8-38 Step 4's CPU idle time for physical devices

CPU interrupts per second

The amount of CPU interrupts per second were low for the VMs throughout the entire simulation. The amount of CPU interrupts per second remained around 35. The exception is in Step 4, where the amount of CPU interrupts intermittently spiked to around 45 interrupts per second (Figure 8-39). This is most likely due to the fact that GRR DFR evidence gathering requires processing power. The number of interrupts remained low throughout all of the steps, which indicates that the GRR prototype did not increase interrupts drastically.

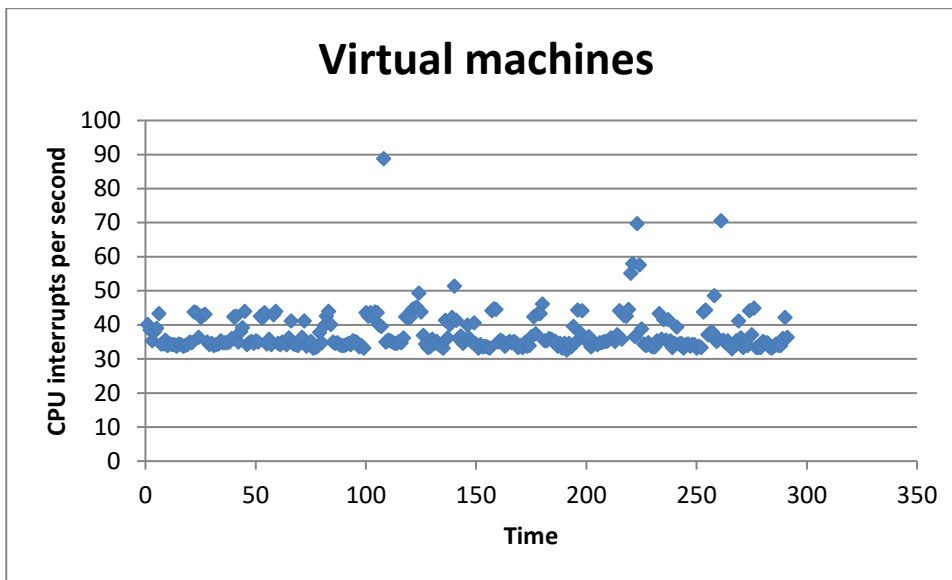


Figure 8-39 Step 4's CPU interrupts per second for virtual machines

The amount of CPU interrupts per second for the physical devices were significantly higher than that of the VMs. The physical devices registered around 390 CPU interrupts per second for most of the steps in the simulation. The higher number of interrupts per second is most likely attributed to the difference in operating systems. The physical devices have GPIO pins that can be polled to check for inputs, which can result in higher CPU interrupts. Spikes in the amount of CPU interrupts were registered in Step 4 of the simulation, which corroborates the findings of spikes in CPU interrupts due to DFR evidence gathering.

CPU load 1 minute average per core

The CPU load average per core for the VMs averaged around 0,02-0,04 per core for the various steps of the simulation. This is extremely low usage of the CPU for simulation. This is not a measurement of CPU usage per minute, but an average over a minute. This means that the CPU core load might have been significantly higher for a few seconds out of a minute timeframe, but the overall CPU core load per minute was extremely low for all of the steps in the simulation. An interesting finding is that the CPU load average per core was higher in Step 2 than in Step 3, indicating that an idling GRR prototype used more CPU processing than the Python smart home program that was used in Step 3 to simulate the working of a smart home. The CPU load 1 minute average per core was the highest in Step 4, which entails the GRR prototype conducting DFR processes on a running simulation of a smart home (Figure 8-40). This combination of the smart home simulation

and DFR processes resulted in an average CPU load per 1 min of 0,044 which is still rather small.

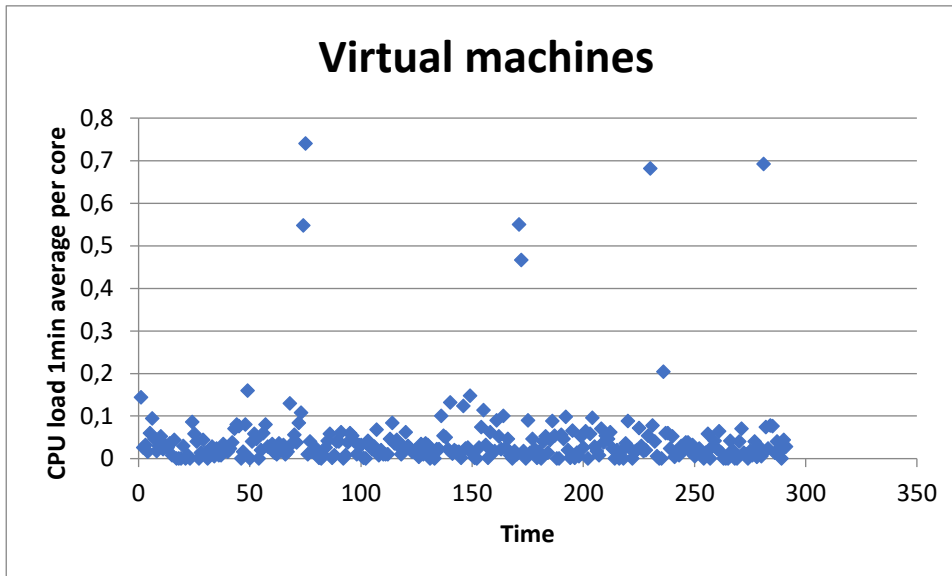


Figure 8-40 Step 4's CPU load 1min average per core for virtual machines

The CPU load 1 minute average for the VMs was quite similar to the CPU load 1 minute average of the physical devices in Step 1 of the simulation, extremely fragmented in Step 2, more closely related in Step 3, and a bit more scattered in Step 4.

The CPU load 1 minute average per core for the physical devices was much lower than that of the VMs. This is attributed to the fact that the physical devices have a total of four processing cores each, where the VMs only had one processing core on a hypervisor. Nevertheless, the physical devices yielded similar results for CPU load average per core when compared to the VMs for the simulation. The CPU load 1 minute average per core for the physical devices averaged around 0,002-0,005 for the various steps of the simulation. Similar to the findings of the CPU load average for the simulation, Step 2 used more processing power than Step 3. This confirms that the GRR prototype uses more processing power than the smart home simulation program, even when the GRR prototype is merely running without performing any investigations. The highest CPU load 1 min average per core for the physical devices is 0,005686 in Step 4. This is still extremely low CPU usage for IoT devices with constrained resources.

The CPU load averages for the physical devices were mostly fragmented across the various steps. However, they were more closely grouped in Step 1 and 3, which is similar to the findings of the load averages for the VMs. This suggests that the Python smart home simulation program results in fragmented CPU usage loads, rather than a constant increase.

CPU system time

The CPU system time for the VMs ranged from 0,2-0,8 for the various steps of the simulation. What is interesting is that the CPU systems averaged higher in Step 1 when compared to Step 2, where the GRR prototype was active. A possible reason for this is that the VMs were conducting system maintenance tasks during the first step, which could have increased the CPU system time. The CPU system time peaked during Step 1 with an outlier CPU system time of 15. The CPU system time only had one occurrence of an outlier in Step 2, which was around 1,7 and significantly lower than the outliers in Step 1 of the simulation. The final step in the simulation did have a few outliers with a maximum CPU system time of 40, but was quite low in frequency.

The CPU system time for the VMs averaged 0,834 for Step 4, which was the most resource-intensive of the steps. This is shown in Figure 8-41. This is likely due to the GRR prototype acquiring the log files from the smart home program and then sending the forensic evidence to the GRR server. The overall CPU system time remained relatively low throughout all of the steps in the simulation, which indicates the relatively low impact of DFR for IoT devices on CPU system time.

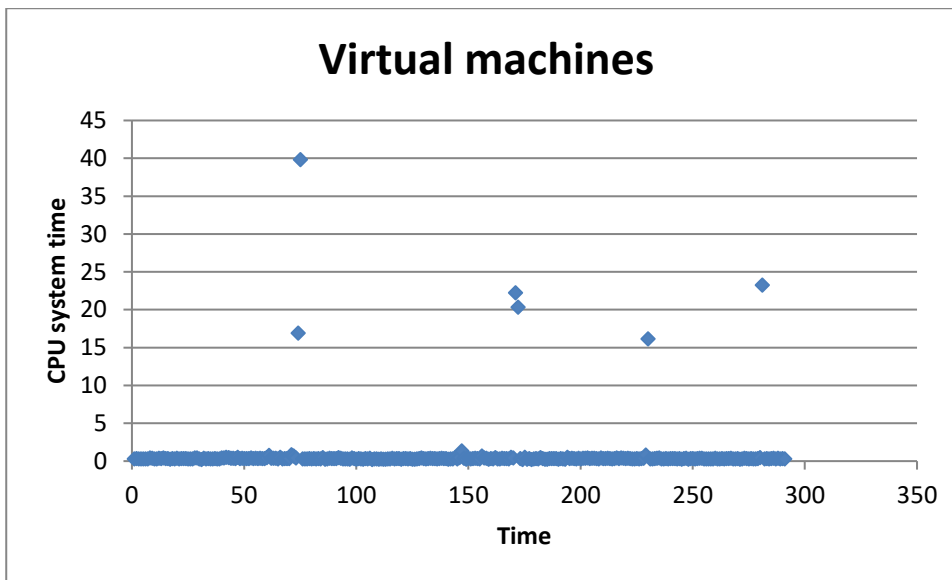


Figure 8-41 Step 4's CPU system time for virtual machines

The CPU system time for the physical devices ranged from 0,08-0,1 during the various steps of the simulation. The lowest being Step 1 and the highest Step 4. The CPU system time was closely grouped in Step 1, but more spread out for the rest of the steps. The maximum measured CPU system time was 0,15 during Step 4, which is still rather small. The DFR for IoT investigations therefore did not have a significant impact on the CPU system time of the physical devices.

Number of processes

The average number of processes for the VMs ranged from 215-228 for the various steps of the simulation. The lowest being the first step of the simulation and the highest being the final step of the simulation. The largest increase in the amount of processes was between the first two steps. Step 1 had an average of 215 processes and Step 2 had an average of 225 processes. This shows that the GRR prototype results in a higher number of processes, even when idle and not conducting forensic investigations. The GRR prototype resulted in a higher number of processes due to the complexity of the GRR prototype and the multitude of operations that the GRR prototype can perform.

The average number of processes for the physical devices ranged from 126-136 for the various steps of the simulation. The first step had an average of 126 processes and the final step had an average of 136 processes (Figure 8-42). Similar to the observation made

for the VMs, the largest increase was between the first two steps, which confirms that the GRR prototype results in a higher number of processes even when idle.

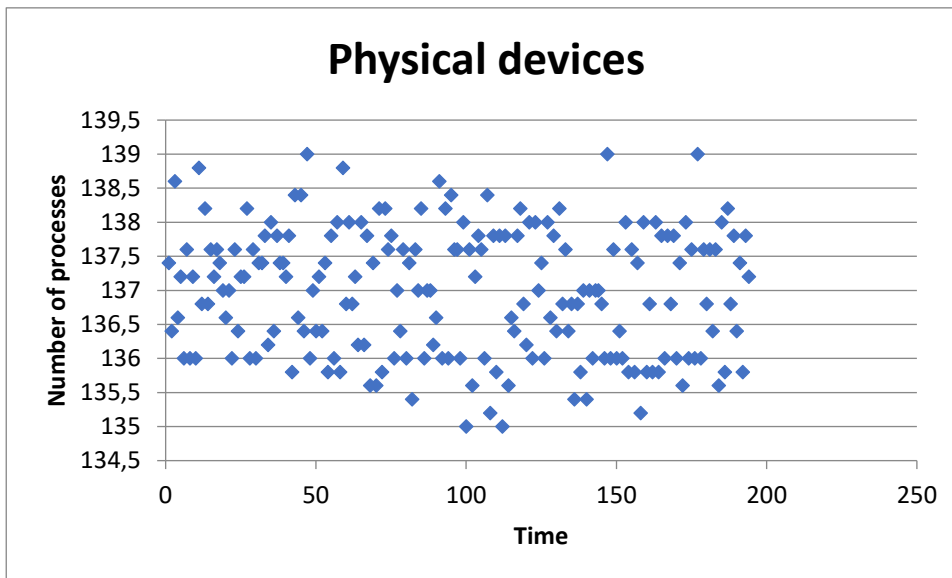


Figure 8-42 Step 4's number of processes for physical devices

Number of running processes

The average number of running processes ranged from 2,226-2,252 for the various steps of the simulation. This is shown in Figure 8-43. The final step was the highest and the first step was the lowest. An interesting observation is that the average number of running processes was higher in Step 2 than in Step 4. This indicates that an idle GRR prototype uses more running processes than the smart home simulation. This is to be expected, as the GRR prototype is multi-faceted. The GRR prototype requires a variety of software tools and programs to operate. This is backed up with the example of having to compile protobuf for the physical devices in order for the GRR prototype to function on physical IoT devices. The number of running processes for the VMs was a bit spread out in all of the steps.

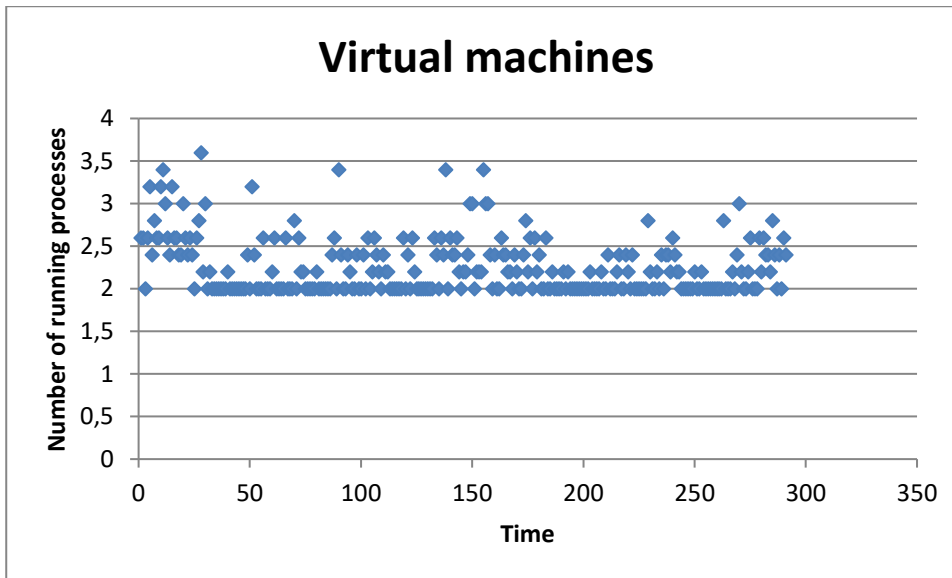


Figure 8-43 Step 2's number of running processes on virtual machines

The average number of processes for the physical devices ranged from 2,031-2,058 for the various steps. The highest average being Step 4 and the lowest being Step 1. Similar to the findings of the VMs, Step 2 resulted in the highest increase in the number of running processes between the GRR prototype and the smart home simulation. The number of running processes was relatively closely grouped over all the various steps of the simulation. Step 2 and Step 4 showed a bigger increase in the number of running processes, but the outliers were closely grouped, which shows that the processes were related. This also shows that external unrelated processes were not activated during the various steps of the simulation.

8.5.2 Storage performance metrics

The next category for measuring IoT device performance is that of storage mediums. There is one metric in the simulation monitored for storage performance, namely free disk space in percentage.

Free disk space in percentage

The free disk space for the VMs ranged from 52-50,85% over the course of the simulation. The highest amount of free disk space was recorded during Step 1 and the lowest amount of free disk space during Step 4. This is to be expected, as the Python smart home simulation generated log files for the IoT device sensor events, as well as the smart home server log events. This is due to the Python smart home simulation programs being

completely installed on the various IoT devices. Thus, each VM had a server program running for the smart home program, as well as the client for the smart home program. There was a drop in the available amount of disk space for the various VMs. The drop is not massive, bearing in mind that the VMs have small disks allocated to them. A better implementation of the Python smart home program would be to delete old logs on IoT devices after they are collected as evidence by the DFR GRR prototype. The logs would, however, still be stored in the GRR server database.

One aspect that needs to be defined is that Git was installed on the various VMs after Step 2. This resulted in a drop in the amount of free disk space between the two steps. Git was used to install the Python smart home simulation program. The smart home program was also installed after Step 2 in the simulation, which furthermore resulted in a decrease in the amount of available disk space for the VMs.

The free disk space for the physical devices ranged from 21,1313-21,10928 for the various steps of the simulation (Figure 8-44). This enforces the findings of the free disk space in percentage for the VMs. The small decrease in the amount of free disk space in percentage shows that the GRR prototype does not result in excessive disk space usage over time. The main cause for a change in the amount of available disk space is the various log files generated by the Python smart home simulation program. The physical devices did have Git installed before the start of the various steps in the simulation, but this was not the case with the VMs. The decrease in the amount of available disk space for the physical devices is therefore smaller over the course of the various steps in the simulation and only impacted by the installation of the smart home simulation program and the various log files generated by the smart home simulation program.

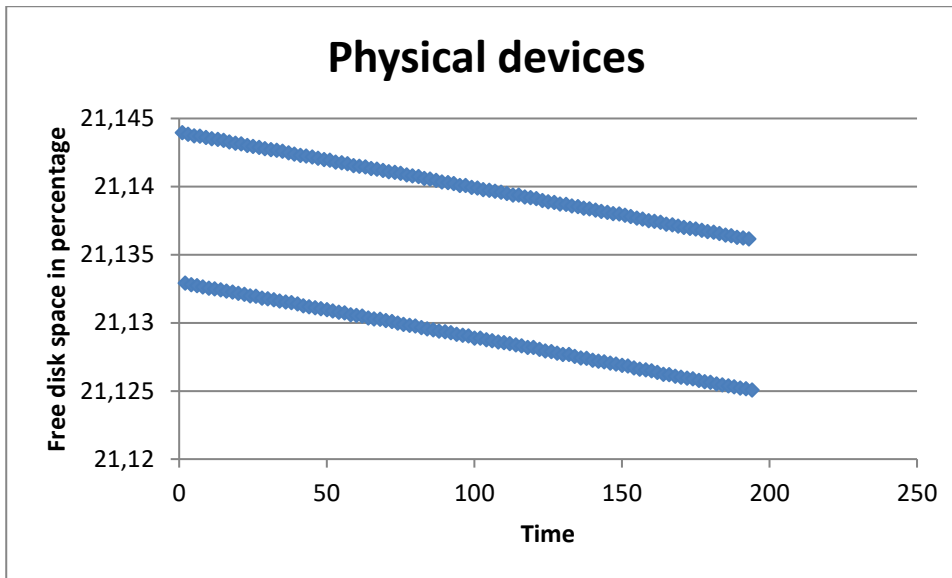


Figure 8-44 Step's 3 free disk space in percentage for physical devices

8.5.3 Memory performance metrics

The memory performance metrics are divided into two metrics, namely available memory and the free swap space in percentage. The free swap space is also related to storage space, as swap space resides on storage rather than the RAM of devices. Swap space is used as an alternative for memory, which is why it is included under the memory category.

Available memory

The available memory for the VMs remained rather constant over the course of Step 1 to Step 3 (Figure 8-45). In Step 4 the available memory for the VMs decreased slowly at a constant rate. The memory usage in the first three steps indicates that the operating system made consistent use of the available memory, and the Python simulated smart home program and idle GRR prototype did not use up too much memory. The memory usage did not increase too much when the GRR prototype or Python program was started. The DFR evidence gathering did result in more memory usage over time, but it did not take up too much of the available memory. The memory steadily decreased, which is not ideal. This is most likely due to the GRR hunts not ending in time. This might result in the GRR hunts still keeping the evidence in memory until the hunts finish. The hunts used in Step 4 were not configured to end shortly. This meant that if a GRR agent detected the hunt parameters, it would conduct the hunt as soon as possible, but it kept listening for parameters from the same hunt. This is not ideal and future tests can create hunts of

shorter lifespan to potentially mitigate this problem. Further tests are needed to determine that this is indeed the cause for the steady loss of available memory for the VMs.

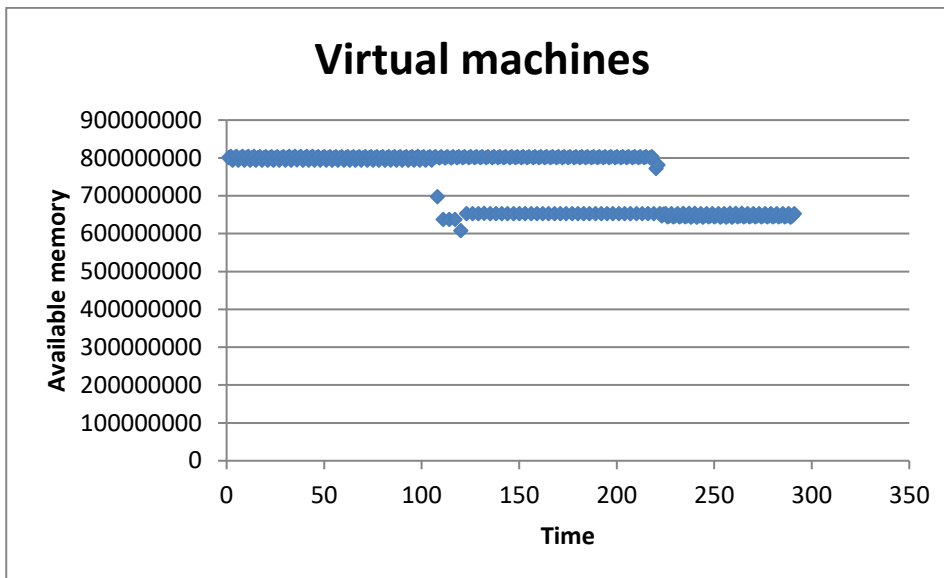


Figure 8-45 Step 1's available memory for the virtual machines

The available memory for the physical devices yielded some interesting results. It steadily decreased over the various stages of the simulation. This suggests that the operating system used for the physical devices resulted in some continuous loss of available memory. An interesting future investigation into this loss of memory over time would be beneficial. Step 2 resulted in a slight decrease in the amount of available memory. In Step 3 the Python simulation used less memory when compared to the GRR prototype in Step 2. Step 4 resulted in less available memory, with the continuation of a decrease in the amount of available memory. This shows that the GRR prototype uses more memory than the simulated Python smart home program. It would be interesting to determine if the memory is freed after long periods of time or whether the physical devices would eventually run out of available memory, which is not desirable. Further tests are needed to determine whether the GRR prototype contributes to a continual loss of available memory, or if the continual loss of memory is solely attributed to the operating system used by the physical devices.

Free swap space in percentage

The amount of available free swap space for the VMs ranged from 100%-94,61% (Figure 8-46). The highest amount of available memory was recorded during Step 1, and the

lowest available memory was recorded during Step 4. The various steps indicated that some swap space was used during Step 1, which is quite peculiar as no programs apart from the operating system were active then. This suggests that the operating system used by the VMs require usage of swap space after a certain amount of time of operation. The amount of swap space used by the operating system is relatively low, but still noteworthy. The available swap space remained rather constant during Step 2, which indicates that the GRR prototype operation did not require additional swap space. Step 3 did result in some additional usage of swap space, which suggests that the Python smart home simulation program results in additional usage of swap space. This continued throughout Step 4 of the simulation.

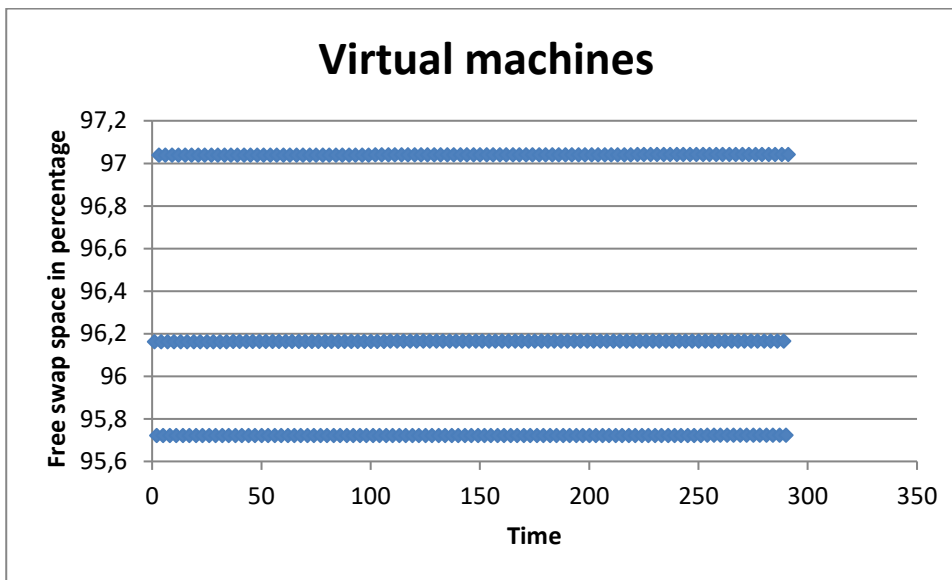


Figure 8-46 Step 2's free swap space in percentage for the virtual machines

The available swap space for the physical devices remained constant at 100% throughout all of the steps of the simulation. This suggests that the particular operating system used on the simulated devices contributes towards the usage of swap space due to the VMs using swap space but not the physical devices.

8.5.4 Network usage metrics

The final category of performance measurement for the various devices in the simulation is network usage. Network usage is measured in both ingoing and outgoing traffic.

VM network usage

The average incoming network traffic usage for the VMs ranged from 1979bps-2742bps. The lowest incoming average network usage was recorded during Step 1 of the simulation and the highest incoming traffic average during Step 4. Step 2 also had a slightly higher incoming traffic average than Step 3, which indicates that the GRR agent receives data from the GRR server. This is to be expected, as the GRR client polls the server for hunt requests. Overall, the incoming traffic was very low over the course of the simulation, which is good for IoT devices, which typically use a lot of data.

The outgoing traffic for the VMs ranged from 1368bps-1931bps over the course of the four steps of the simulation (Figure 8-47). The lowest average usage was recorded during Step 1 of the simulation and the highest average was recorded during Step 4. Step 2 also had a slightly higher average for outgoing traffic, which is to be expected, as the GRR client polls the GRR server for hunts. Do note that both the Python home server and Python smart home clients operated locally on the IoT devices. This is a bit restrictive for the gathering of network traffic in the smart home simulation, as the network traffic was local to the various devices and not transferred across the network. However, the purpose of the simulation is to determine how the GRR prototype impacts IoT devices for DFR in IoT, so the tests were conducted in this manner to ensure maximum insight into the GRR prototypes. The final aspect worth noting with regards to outgoing traffic of the VMs is spikes in outgoing network traffic at certain intervals of Step 4. These spikes were higher in bandwidth usage than the spikes in the other steps. This is most likely caused by the transfer of the DFR evidence gathered by the GRR clients, which is to be expected as the DFR evidence results are usually larger than merely polling the server for hunt requests.

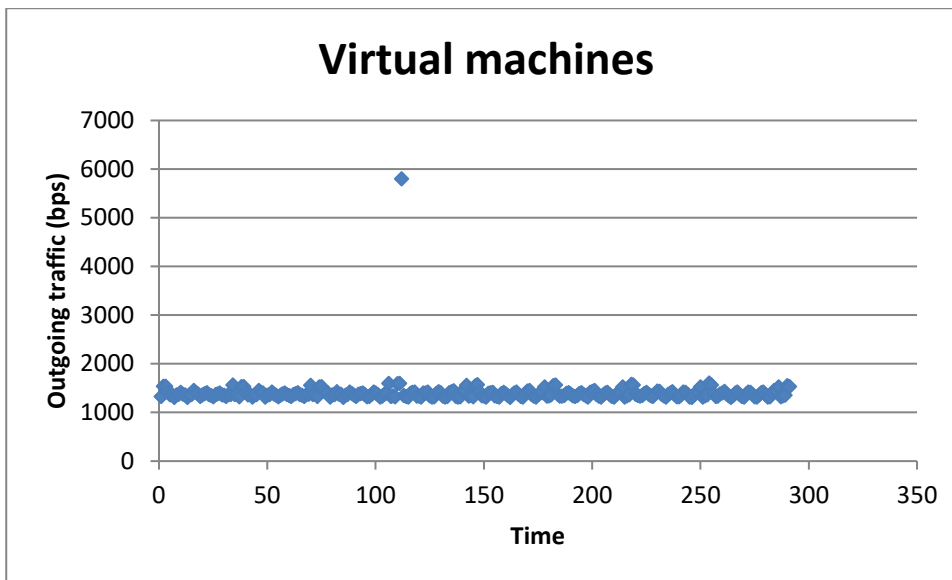


Figure 8-47 Step 2's outgoing traffic for virtual machines

Physical device network usage

The average incoming traffic for the physical devices ranged from 1817bps-2242bps for the various steps of the simulation, which is slightly higher than that of the VMs. This indicates a possible system update or higher overhead for network usage. However, further investigation is needed to determine the exact cause of higher network traffic for the physical devices, but it is most likely related to the operating system used by the physical devices.

The average outgoing network usage for the physical devices ranged from 2052bps-4332bps over the course of the various steps of the simulation. The lowest average of outgoing traffic was observed during Step 1 of the simulation and the highest during Step 4. This is a result of the GRR prototype transmitting the results of the DFR evidence gathering. The network usage of the physical devices also showed intermittent spikes in the outgoing traffic. This is attributed to the transfer of DFR evidence to the GRR server. The spikes in outgoing traffic for Step 4 of the simulation were generally about 1800bps more than the average outgoing network traffic (Figure 8-48). This indicates that the GRR prototype requires more network traffic for the evidence gathering, but it is not a continual increase in network traffic. This is great for DFR in IoT devices, as a lot of IoT devices can generate a significant amount of network traffic. The intermittent nature of the network traffic for DFR is good for overall network traffic, as the slight increase in network traffic

would not negatively impact the general network usage of the IoT devices. IoT devices are thus able to still operate normally with minimal to no service disruptions.

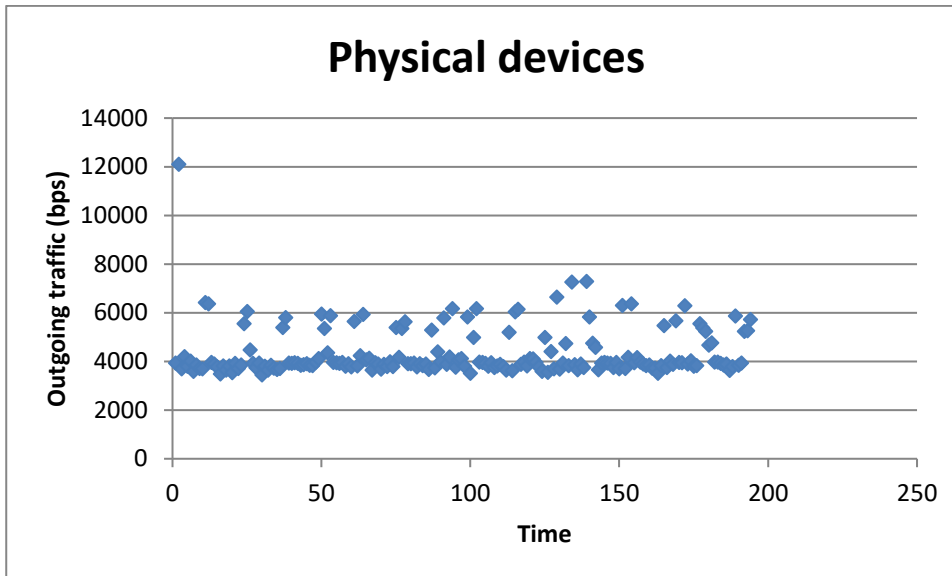


Figure 8-48 Step 4's outgoing traffic for physical devices

8.6 CONCLUSION

This chapter showed the testing of a prototype for DFR in IoT. A predefined set of steps were followed in order to simulate the working of a prototype based on a model for DFR in IoT. A variety of metrics was used to monitor the performance of various IoT devices during a simulation of DFR for IoT.

This chapter provided an in-depth overview of the configuration and setup of various tools and programs used in the simulation. It also showed how the prototype is used to proactively collect evidence from various IoT devices in a forensically sound manner. All the research questions in this dissertation were partially answered through the use of forensic techniques in simulated forensic investigations on IoT devices. The main objective of this study is to determine if it is possible to apply forensic techniques to IoT devices and facilitate DFR for IoT devices. This chapter showed that it is indeed possible to prepare IoT devices for DFR. The successful implementation of DFR for IoT calls for an evaluation of this dissertation's findings, which is provided in the following chapter.

9 CHAPTER 9: PROTOTYPE EVALUATION

9.1 INTRODUCTION

The previous chapter provides an in-depth overview of the findings of the various metrics monitored during the steps of the simulation. This chapter evaluates the findings on a higher level and with regards to the greater context of this study.

This chapter contributes towards all the various research questions of this dissertation by evaluating the findings of the simulation relevant to the problem statement. This chapter furthermore contributes towards objective 1.4.4 (Proof of concept, DFR IoT prototype) due to evaluating the proof of concept of the model for DFR in IoT based on the simulations of a prototype.

The simulation conducted in the previous chapter consists of four steps. Each step is designed to test a different aspect of the DFR for IoT. The steps are briefly repeated here for convenience as they are used in this chapter again. The first step establishes a baseline for the performance of the VMs and physical IoT devices. The second step determines the impact that the GRR prototype has on the overall performance of the various IoT devices (both VMs and physical devices). The third step determines the impact that the simulation of a smart home program has on the various IoT devices. The final step determines the impact that a full-fledged smart home program and DFR prototype has on the IoT devices when the DFR prototype conducts DFR evidence gathering. The various steps and performance impacts are evaluated in the next section.

The rest of this chapter is structured as follows: firstly, the findings of the simulation of DFR for IoT are evaluated and an example of the forensic value of DFR for IoT is presented. This chapter then critically evaluates the findings of simulating the model for DFR in IoT.

The various steps in the simulation for DFR in IoT is briefly presented in this introduction. The next section starts the in-depth evaluation of DFR for IoT, based on the simulation of DFR for IoT using a prototype.

9.2 EVALUATION OF THE FINDINGS OF DFR FOR IOT

The metrics gathered during the various steps of the simulation focused on four main categories of the various IoT devices, which are processing power, disk usage, memory usage and network usage. The various findings for the overhead categories are evaluated in the following subsections.

9.2.1 Processing power

All the various metrics used for measuring the impact of DFR for IoT on IoT device processing performance indicate the feasibility of DFR for IoT. The various metrics show that the GRR DFR prototype increased processing usage during the various steps. This was expected, as the GRR prototype is a feature-rich program that has access to a variety of IoT device resources. The GRR prototype used the most processing power when conducting DFR evidence gathering. However, the prototype used an acceptable amount of the IoT devices' processing power. This indicates that the GRR prototype can be used on IoT devices for DFR. The prototype might, however, be a bit too resource-intensive for IoT devices with some heavily reduced instruction set micro controllers such as an 8bit micro controller. IoT devices therefore require small controller devices with processors capable of running the GRR prototypes. This study has proved that the small controllers need not be powerful computers, but that ARM processors would be sufficient.

9.2.2 Disk usage

The disk usage of the DFR prototype indicated that the GRR prototype uses some disk space when installed on the host system. This was expected, as the GRR prototype is feature-rich, resulting in a program of mentionable size (+/- 14MB). However, the disk usage of the GRR prototype is moderate, and it can easily reside on IoT devices with limited storage space.

The main issue with disk usage during the simulation is the amount of disk space (up to 2% of the total amount of available disk space) used by the Python smart home program logs. As previously noted, the Python server program and Python client were hosted on the same devices. This resulted in more disk usage than when only the clients are installed on the IoT devices. In a real-world scenario, the clients are installed on the IoT devices. In the simulation both the client and server were installed on the same devices to increase the amount of disk space used, as both the server and client program write log

files of the state changes. Nevertheless, the clients generate log files that increase in size over time. A solution to this problem would be to clear the logs after they are transferred by the GRR prototype. The logs are preserved in a forensically sound manner by the GRR platform using hashing techniques, which means that the space on the IoT devices can be freed up for future log content. This is an improvement that can be made in future iterations of DFR for IoT.

9.2.3 Memory usage

The memory usage of the IoT devices was considerably low over the various steps of the simulation. The simulation did however show a steady decrease in the amount of available memory when DFR evidence was being gathered. At the end of Step 4 almost 25% of the total amount of available RAM in the VMs were consumed. This is most likely due to the GRR hunts not ending before Step 4 of the simulation concluded. This is another aspect that needs further investigation by tweaking the configuration of the GRR hunts to have a shorter lifespan before they are concluded.

The simulation did indicate that the physical devices suffered a continuous loss of available memory from Step 1 onwards. This is most likely due to the operating system and not the GRR platform. Further investigation is needed to verify this matter. The loss of memory persisted in Step 4, which is most likely a combination of the operating system memory usage and the GRR hunts not ending in a timely manner.

9.2.4 Network usage

The network usage of the IoT devices remained rather low during the various simulation steps (+/- 1600bps to +/- 4000bps). Even though the outgoing network traffic did increase during the final step of the simulation due to the DFR evidence being transferred to the GRR server, the network traffic was still relatively low and intermittent. The GRR prototype did not result in an overall increase in network traffic, merely sporadic traffic. One of the fundamental properties of IoT devices is that they are networked, and a slight increase in network traffic from IoT devices is not a general concern.

The above-mentioned sections show how well the prototype for DFR in IoT performed in a simulation. The next section shows the potential value of a functioning DFR compliant IoT system by making use of an example.

9.3 EXAMPLE OF FORENSIC VALUE OF DIGITAL FORENSIC READINESS FOR THE INTERNET OF THINGS

This section provides a scenario to illustrate the value of DFR for IoT. The example is done using the model proposed in this study, the smart home simulation program and the GRR prototype developed here. This section expands on the previous chapter by introducing a physical model of a home to enforce the scenario for DFR in IoT.

9.3.1 Physical model of a smart home

In order to enforce the value of DFR for IoT a scenario of a murder at a personal residence is provided in this section. The scenario is simulated using the physical model of a smart home. The smart home in this scenario has sensors to detect the presence of a human in a room. Whenever a human is present in a room, the lights in the room turn on. When the human leaves the room the lights turn off. The physical model home in this scenario uses magnetic switches on the doors of the various rooms for simplicity of simulation. In reality a combination of sensors would be necessary to accurately detect the presence of a human in a room. However, the assumption in this example is that when the door to a room is open, a human is present therein. The smart home built for this simulated example scenario is seen in Figure 9-1.

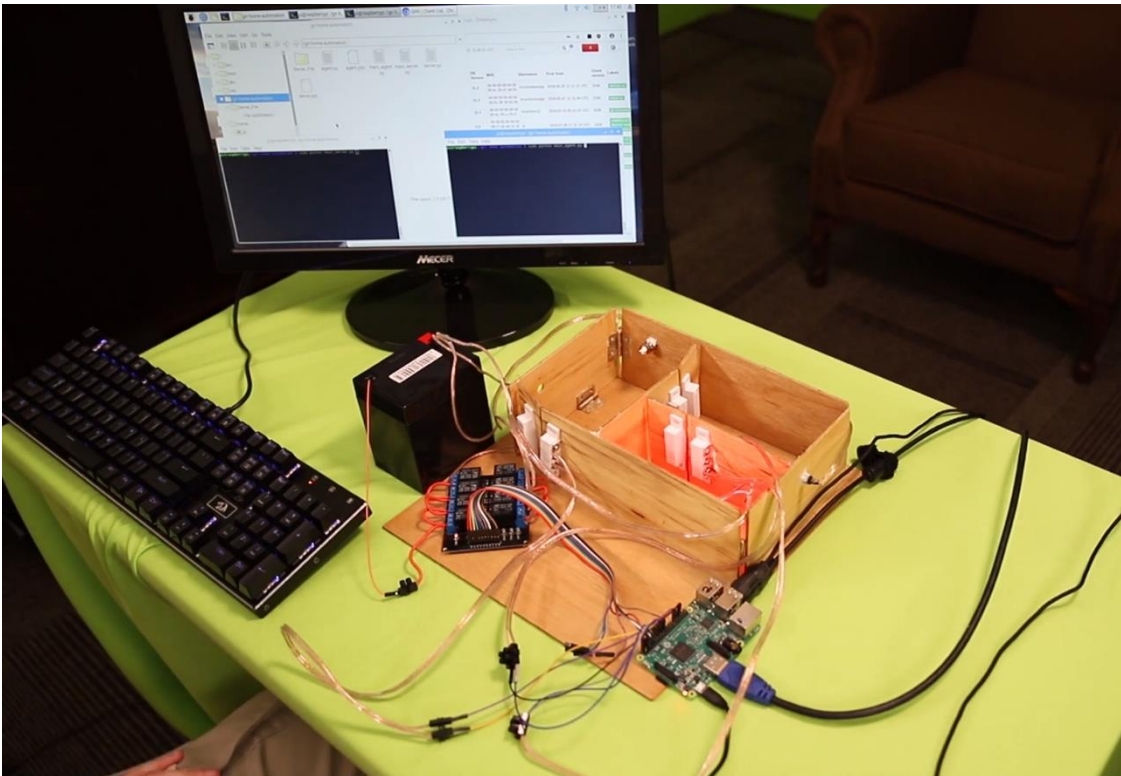


Figure 9-1 Smart home simulation on a physical IoT device

The magnetic switches on the various doors are connected to a Raspberry Pi that registers the opening and closing of doors in the smart home. The Python smart home simulation program developed in this study then logs the various changes in the physical environment of the smart home in the form of log entries indicating the various doors opening and closing. The smart home program also sends the various log entries to the smart home server, which in turn logs the entries into different log files for the different rooms of the smart home. The smart home program also turns the lights on or off based on the registered state changes from the magnetic switches on the doors.

9.3.2 Fictional scenario for digital forensic readiness in a smart home

The owner of a smart home is murdered when he returns home from work one day. The smart home can be used as evidence to support the alibi of a potential suspect.

The owner of the smart home woke up late on the morning of the murder and quickly rushed through his home to finish up for work. He went into several of the rooms in his home to get ready for work before he eventually left the property. The owner of the smart home returned to his home late the same afternoon where he was murdered within a few

seconds upon his arrival at his home. A passer-by, Suspect X, is suspected of the murder as they were witnessed passing the home of the deceased at roughly the same time of the murder, but Suspect X pleads their innocence and that they have never visited the deceased's home.

9.3.3 Forensic investigation into scenario

The logs collected from the smart home system are deleted every few minutes in order to save space on the smart home system. Luckily, the DFR prototype sends the updated logs to the GRR server in a trustworthy and forensically sound manner. The forensic investigator assigned to the case looks at the evidence located in the remote GRR server repository with a hashed version of the log files of the smart home system.

The log files reveal that the deceased rushed through the various rooms in the smart home before leaving for work, as is shown in Figure 9-2. The logs show that the deceased left home at 07:57, as is shown in Figure 9-3.

```

159 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-08-26 22:35:07.401005; light_requested_state:on
160 room:kitchen_device1; current light status:TURN ON; requester_id:sensor_kitchen; date:2019-08-26 22:35:07.402138
161 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-08-26 22:35:07.402387; light_requested_state:off
162 room:kitchen_device1; current light status:TURN OFF; requester_id:sensor_kitchen; date:2019-08-26 22:35:07.403293
163 room:livingroom_device1; next light status:requested; requester_id:sensor_livingRoom; date:2019-08-27 07:56:48.691118; light_requested_state:off
164 room:livingroom_device1; current light status:TURN OFF; requester_id:sensor_livingRoom; date:2019-08-27 07:56:48.698889
165 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 07:56:48.699617; light_requested_state:off
166 room:kitchen_device1; current light status:TURN OFF; requester_id:sensor_kitchen; date:2019-08-27 07:56:48.702152
167 room:bedroom_device1; next light status:requested; requester_id:sensor_bedroom; date:2019-08-27 07:56:48.702748; light_requested_state:off
168 room:bedroom_device1; current light status:TURN OFF; requester_id:sensor_bedroom; date:2019-08-27 07:56:48.705206
169 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 07:56:54.392298; light_requested_state:on
170 room:kitchen_device1; current light status:TURN ON; requester_id:sensor_kitchen; date:2019-08-27 07:56:54.394087
171 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 07:56:54.397674; light_requested_state:off
172 room:kitchen_device1; current light status:TURN OFF; requester_id:sensor_kitchen; date:2019-08-27 07:56:54.399952
173 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 07:56:54.401194; light_requested_state:on
174 room:kitchen_device1; current light status:TURN ON; requester_id:sensor_kitchen; date:2019-08-27 07:56:54.403454
175 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 07:56:58.350646; light_requested_state:off
176 room:kitchen_device1; current light status:TURN OFF; requester_id:sensor_kitchen; date:2019-08-27 07:56:58.352334
177 room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 07:56:58.352884; light_requested_state:on
178 room:kitchen_device1; current light status:TURN ON; requester_id:sensor_kitchen; date:2019-08-27 07:56:58.355255

```

Figure 9-2 Deceased moving through the home to finish for work

```

room:kitchen_device1; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 07:57:21.559816; light_requested_state:off
room:kitchen_device1; current light status:TURN OFF; requester_id:sensor_kitchen; date:2019-08-27 07:57:21.560853

```

Figure 9-3 Deceased left home and unauthorised entry occurs later the day

The forensic investigator discovers an interesting sequence of events before the deceased arrived home: the logs show the presence of a human in several rooms from 11:20 (also shown in Figure 9-3). The logs show that movement in the home continued for a while, with the intruder finally settling in the bedroom (Figure 9-4). This is inferred due to the last state change being the bedroom lights being turned on, which indicates a presence in the room until the light is eventually turned off. Figure 9-4 then shows another entry to the smart home on line 327 of the logs where the living room's lights are turned on without the

bedroom light being turned off. This indicates the entry of the deceased, as the other light was not turned off prior to his arrival.

```
room:kitchen_devicel; current light status:TURN OFF; requester_id:sensor_kitchen; date:2019-08-27 14:32:58.705688  
room:kitchen_devicel; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 14:32:58.709714; light requested state:on  
room:kitchen_devicel; current light status:TURN ON; requester_id:sensor_kitchen; date:2019-08-27 14:32:58.710729  
room:kitchen_devicel; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 19:55:04.046801; light requested state:off  
room:kitchen_devicel; current light status:TURN OFF; requester_id:sensor_kitchen; date:2019-08-27 19:55:04.048007  
room:bedroom_devicel; next light status:requested; requester_id:sensor_bedroom; date:2019-08-27 19:55:12.151004; light requested state:on  
room:bedroom_devicel; current light status:TURN ON; requester_id:sensor_bedroom; date:2019-08-27 19:55:12.152193
```

Figure 9-4 Moments before the murder

The forensic investigation into the logs of the smart home also shows the rapid movement of entities between the various rooms of the smart home at the time when the deceased arrived at home. This indicates that a scuffle took place before the murder. This is supported by an autopsy of the deceased indicating several bruises. This is shown from line 371 onwards in the log files of Figure 9-5.

```
room:kitchen_devicel; current light status:TURN ON; requester_id:sensor_kitchen; date:2019-08-27 19:55:52.914297  
room:kitchen_devicel; next light status:requested; requester_id:sensor_kitchen; date:2019-08-27 19:55:52.915248  
room:livingroom_devicel; next light status:requested; requester_id:sensor_livingRoom; date:2019-08-27 19:55:59.115847; light requested state:off  
room:livingroom_devicel; current light status:TURN OFF; requester_id:sensor_livingRoom; date:2019-08-27 19:55:59.117002  
room:bedroom_devicel; next light status:requested; requester_id:sensor_bedroom; date:2019-08-27 19:56:01.678820; light requested state:off  
room:bedroom_devicel; current light status:TURN OFF; requester_id:sensor_bedroom; date:2019-08-27 19:56:01.679976  
room:bedroom_devicel; next light status:requested; requester_id:sensor_bedroom; date:2019-08-27 19:56:04.677095; light requested state:on  
room:bedroom_devicel; current light status:TURN ON; requester_id:sensor_bedroom; date:2019-08-27 19:56:04.682164  
room:livingroom_devicel; next light status:requested; requester_id:sensor_livingRoom; date:2019-08-27 19:56:05.813505; light requested state:on  
room:livingroom_devicel; current light status:TURN ON; requester_id:sensor_livingRoom; date:2019-08-27 19:56:05.814606
```

Figure 9-5 Scuffle occurrence

Suspect X owns a smart watch as IoT device, which keeps track of the heart rate of the wearer, but does not track the GPS location of the wearer as some smart watches do. In this scenario we assume that the smart watch is also DFR compliant. The forensic investigator is therefore able to gather logs of the heart rate from the wearer of the day of the murder.

The heart rate logs show a dramatic increase in the wearer's heart rate from 11:19-12:14, as well as 19:55-20:13 during the day of the murder. The latter spike in heart rate coincides with the time of death determined by the autopsy of the deceased. The spike in heart rate between 11:19-12:14 coincides with the logs of the smart home that indicates unauthorised entry to the home. Furthermore, a co-worker of Suspect X testifies in court that they saw Suspect X leave work at around 11:00. The morning's increase in heart rate supports the case that the suspect entered the smart home and then settled in as the heart rate dropped at around 12:14. The spike in the suspect's heart rate at 19:55 indicates the

sudden shock when the deceased entered the smart home and the suspect realised that the murder was about to take place. The decrease in heart rate at around 20:13 supports the notion of an increase in adrenaline levels of the suspect that also had to get away from the scene of the murder. The decrease in heart rate is possible due to the reduction in danger of being caught red-handed at the scene of the murder.

The combination of evidence from traditional sources, as well as from IoT devices, provides a compelling amount of evidence to indicate that Suspect X might be the murderer in this example. The example makes use of logs proactively gathered from the smart home in a trustworthy manner, as well as fictional logs from the suspect's smart watch. This example aims to show the extent to which DFR for IoT adds value to forensic investigations. DFR IoT evidence can be used as an extension of traditional sources of evidence. With careful cross examination of DFR IoT evidence, new insights can be added to investigations. A demonstration of an investigation using the custom GRR prototype developed in this study is available at <https://youtu.be/fxX0POwu6SQ>. The video shows the working of the physical model of a smart home and goes through the various steps required to perform a forensic investigation of proactively gathered IoT evidence using the GRR platform.

9.4 CRITICAL EVALUATION

This chapter shows that with some implementation and custom development, DFR for IoT is possible. The implementation of DFR for IoT brings new opportunities to the world of DF. DFR for IoT devices opens new sources of potential evidence that can form part of forensic investigations. The combination of evidence from IoT devices and conventional forensic evidence provides new levels of insight into incidents, as IoT devices include a wide variety of sensors. The next step of DFR for IoT is for researchers to determine additional scenarios where forensic IoT evidence can play a vital role in legal proceedings. Wherever IoT devices form a part of an environment, they can provide possible insights into incidents.

Smart home systems with embedded DFR for IoT enable new insight into home automation and security. In medical environments, IoT devices with DFR evidence gathering enabled can be used to combat medical malpractice or even prevent some medical emergencies. DFR of smart cities can be used to combat crime and potentially

prevent crime by detecting indicators of imminent crimes. DFR of personal IoT devices can be used as alibies or as convicting evidence by using location data from these smart devices. DFR for IoT devices can provide cyber security professionals with more data, which can be used to combat cybercrimes. DFR offers a lot of potential waiting to be unlocked by future research.

The DFR for IoT is possible without adding a lot of overhead to IoT devices. The model for DFR in IoT is successfully implemented and tested in this study. A next possible step of research would be to develop a standard for DFR in IoT based on the DFR for IoT requirements and model presented in this study. The model for DFR in IoT enhances insights into IoT sensor events and even provides additional security to IoT communication when the data gathered is stored in a trustworthy and secure manner. All of this is possible with some initial effort when designing and implementing IoT systems. Registering IoT devices to a centralised secure forensic evidence repository enables forensic investigators to gain insight into a whole new world of forensic evidence due to the correlation of evidence from a variety of sources. The development of a standard for DFR in IoT can also incorporate a new file standard for forensic evidence storage, which is much needed in the forensic environment.

DFR for IoT does have some challenges. It is not possible to implement DFR for IoT on all IoT devices, due to some devices being too resource constrained. The advancement in microprocessors may result in them becoming more affordable. Future IoT devices can then be based on the model for DFR in IoT and employ more powerful microprocessors that can support DFR for IoT. Another possible solution to this problem is the development of a lightweight DFR agent. Lightweight DFR agents for IoT devices can be designed to focus on specific types of evidence to be collected. As shown in this study, the DFR agents can be developed to collect only logs from IoT state changes and IoT system events. The smaller DFR agents can then be developed for even more resource-constrained IoT devices than those employed in this study. The DFR agents can also be designed to merely transfer the evidence to the centralised cloud forensic repository, whereafter the codification of the data collected occurs entirely in the cloud repository, ensuring less resource usage by the IoT devices.

Another possible solution for adding DFR to existing IoT devices that make use of microprocessors is to add another more powerful IoT device to the system. All the existing microprocessor-based IoT devices then connect to the more powerful IoT device. The more powerful IoT device then gathers data from the microprocessor-based IoT devices and conduct proactive evidence gathering. This enables DFR for existing IoT systems that make use of IoT devices that are extremely resource constrained and unable to handle DFR processes on their own.

DFR for IoT must also be applied carefully as it may result in privacy concerns. The information gathering of IoT devices can result in privacy concerns. This matter needs to be addressed, as this study does not take privacy concerns into consideration. Users can be made aware of the data collection from IoT devices and data can be anonymised where the focus of the evidence collection is not on individuals. An example of where DFR for IoT might not result in privacy concerns is work environments where IoT devices gather data to help individuals, like hospitals. IoT devices can be used to monitor patients and make sure that patients receive the correct treatments.

Users can also opt-in to the forensic data collection in some cases. Smart homes are an example of opting in to DFR evidence gathering of IoT devices. Evidence gathered in smart homes can be encrypted in a private cloud so that only the owners of the smart home can access the data. Users therefore need to be made aware of the possible benefits of the DFR for IoT devices in their specific applications of IoT devices.

9.5 CONCLUSION

The simulation of DFR for IoT devices yielded some interesting results based on the metrics collected from both simulated and physical IoT devices. Metrics were collected in the categories of processing power, disk usage, memory usage and network traffic usage.

The metrics from the various categories indicated an increase in resources used by the DFR prototype implemented in this study. This increase in resource usage was limited and did not restrict the normal operation of the IoT devices. The DFR prototype conducted successful DFR evidence gathering on IoT devices using a tried-and-tested remote live forensic platform. The simulations therefore showed that DFR for IoT devices is feasible and scalable to a multitude of devices, which is ideal for the IoT environment.

The simulation also showed that DFR for IoT provides deeper insight into incidents due to providing information on environmental state changes and system events of IoT devices. The combination of IoT-generated evidence enhances existing timelines for forensic investigations. Based on all the above-mentioned aspects, it is clear that DFR for IoT is feasible and advantageous to the world of DF.

This chapter constituted the final answer to the problem statement of this dissertation. It showed that there are new possible processes to ensure DFR for IoT, and finally contributed towards objective 1.4.4 (Proof of concept, DFR IoT prototype) by evaluating the simulation of the model and prototype developed in this research.

DFR for IoT shows a multitude of possible beneficial applications in the modern society. It is up to academia to build upon the possibility of DFR for IoT to enhance industries and further the effectiveness of DF. New possibilities bring new challenges, as is the case with DFR for IoT, but challenges call for future research and the actualisation of new opportunities. IoT devices can enhance our insight into the processes of tomorrow and DF offers new sources of evidence. DFR for IoT is feasible, and exciting opportunities exist for this application on IoT devices. DFR for IoT is there to secure IoT forensic evidence for the legal proceedings of tomorrow.

10 CHAPTER 10: CONCLUSION

10.1 INTRODUCTION

This chapter aims to summarise the work done in this dissertation. It starts with an overview of the problem statement, revisits the research questions and research objectives, and then provides an overview of this study's contributions to the current body of knowledge. This chapter ends with a final conclusion and possible future work.

10.2 PROBLEM STATEMENT, RESEARCH QUESTIONS AND RESEARCH OBJECTIVE REVISITED

This section briefly revisits the problem statement this dissertation set out to address. Furthermore, it looks at the various formulated research questions and objectives defined for this dissertation. This dissertation's problem statement is that there is currently no clear process that can be followed to ensure DFR for IoT devices. The problem statement indicates that DFR for IoT devices is still non-existent and this study developed a research methodology to solve this problem.

The research methodology entails a thorough literature review of various relevant areas of research, including the Internet of Things, digital forensic techniques and standards and DFR. A systematic literature review is then used to determine the current state of DF as applied to the IoT. This is followed by a literature review of the legal aspects of DF that can be relevant to DFR for IoT devices. Next a model for DFR in IoT is developed based on requirements for DFR in IoT. The requirements for DFR in IoT are formulated based on the findings of the systematic literature review. Thereafter a working prototype for DFR in IoT is developed. This prototype is then tested through simulation. Finally, the findings from the simulation of DFR for IoT are evaluated.

The various methodologies used in this dissertation was formulated to answer the various research questions as based on the problem statement. The various research questions are as follows:

1.2.1 What standards are currently being used for digital forensic readiness that can be applied to the Internet of Things?

1.2.2 What are the different categories that Internet of Things devices fall under, and would it be possible to conduct digital forensic investigations on the different types of Internet of Things devices?

1.2.3 If it is possible to apply digital forensic readiness on Internet of Things devices, or certain types of Internet of Things devices, would it be possible to proactively apply digital forensic compliance to the devices?

The various research questions are then reworked into aims and objectives for this dissertation. The various research aims, and objectives are as follows:

1.4.1 The main aim of this dissertation is to investigate processes regarding to digital forensic readiness that can be used in the Internet of Things.

1.4.2 A secondary aim of this dissertation is to apply trusted digital forensic techniques to the digital forensic readiness model for Internet of Things devices.

1.4.3 A fourth aim for the research is to design a proof of concept prototype system.

1.4.4 A fourth aim for the research is to design a proof of concept prototype system.

This study answered all research questions, and all aims and objectives were addressed. Each chapter's contributions towards research aims and objectives are indicated in the conclusions of the various chapters. The next section of this dissertation reiterates the contributions made by this study.

10.3 CONTRIBUTIONS MADE TO BODY OF KNOWLEDGE

This section aims to show how this dissertation contributed to the existing body of knowledge. The various contributions of the different chapters are presented in tabular form, followed by a brief discussion. Table 10-1 shows an overview of which research questions and objectives are addressed by which chapters. The colours of the cells indicate the level of contribution a chapter made to that specific research question or objective. The final row in the table is used as an example of how the table should be read. The final row in Table 10-1 indicates that research objective 1.4.4 is addressed in-depth in Chapter 8 and somewhat addressed in Chapter 9.

Table 10-1 Summary of contributions made by different chapters

	Chapters								Partially
	2	3	4	5	6	7	8	9	Extensively
Research questions									
1.2.1 Standards for DFR applicable to IoT	X		X		X	X	X	X	
1.2.2 Different categories of IoT devices		X	X		X	X	X	X	
1.2.3 DFR compliance of IoT devices			X	X	X	X	X	X	

	Chapters									Partially
	2	3	4	5	6	7	8	9	Extensively	
Aims and objectives										
1.4.1 Current DFR processes	x				x	x				
1.4.1 IoT classification		x	x			x				
1.4.2 Trusted forensic techniques			x		x					
1.4.2 DF techniques lead to IoT DFR model					x					
1.4.3 Factors influencing legal validity of evidence				x						
1.4.3 DFR for IoT conforms to legal requirements				x	x					
1.4.4 Proof of concept of the model						x	x	x		
1.4.4 Show model can be implemented						x	x	x		
1.4.4 Model produces trustworthy evidence							x	x		

Research question 1.2.1 aims to determine which DFR processes and standards are relevant to IoT devices – if they exist at all. The systematic literature review shows that the state of DFR for IoT is underdeveloped. DFR is still a relatively new field of research, and while DFR is starting to be applied to a larger variety of research domains, IoT is not one of the main focus areas of DFR. The research that introduces techniques with relevance to DFR that can be applied to IoT devices is limited. DFR for IoT is therefore a rather new concept and papers that address DFR for IoT are mostly theoretical and do not provide tried-and-tested models or solutions. This dissertation therefore makes a major contribution in this regard. It also contributes towards research objective 1.4.1 (DFR processes, IoT device classification) with the implementation of a new model specifically for DFR in IoT, which in turn satisfies research objective 1.4.2 by incorporating various existing DF techniques into a working model for DFR in IoT.

Research question 1.2.2 is based on the concern that IoT devices are quite heterogeneous and that it might not be possible to apply DFR to all IoT devices. This study developed a working prototype for DFR in IoT and applied it to IoT devices. This dissertation identified a subset of IoT devices that can be used for DFR in IoT, but whether it is possible to apply DFR to all IoT devices is still to be determined. This dissertation does however indicate that DFR agents with reduced capabilities can possibly be developed and applied to IoT devices with even less processing power than the devices used in this dissertation. The successful implementation of a proof of concept prototype for DFR in IoT indicates the successful completion of research objective 1.4.4.

Research question 1.2.3 is concerned with the digital forensic compliance of IoT devices that incorporate DFR. An entire chapter was dedicated to the legal aspects of forensic investigations to determine the various established methods of ensuring that electronic evidence is trustworthy. These methods were incorporated into the model developed for DFR in IoT. This dissertation therefore shows that DFR can be implemented for IoT devices and comply with requirements for evidence to be valid in legal proceedings. This indicates the completion of research objective 1.4.3.

10.4 FUTURE RESEARCH

This dissertation shows that existing tools can be customised to enable DFR for IoT devices, and serves as the basis for future research with regards to the feasibility of DFR for IoT. Further research is needed to determine the extent of existing IoT devices that are able to support DFR for IoT.

A lot of research is needed to establish a standard file format for forensic evidence. This is needed not only in IoT forensics, but forensics in general. This will greatly enhance forensic tools, as well as simplify the process of storing and preserving forensic evidence. This dissertation can serve as the starting point for developing a new standard for DFR in IoT. This dissertation developed a model for DFR in IoT that can be expanded and enhanced.

Future research is needed to determine the social impact of DFR for IoT. This dissertation did not consider the privacy concerns that may arise when IoT devices are proactively gathering forensic evidence about the physical world in which the IoT devices are located. Future research can furthermore analyse the difference in various sectors where IoT devices proactively collect evidence. Users might be more open to DFR in IoT devices when the IoT devices do not collect personal information about them. An example of a sector where DFR for IoT might be welcomed is the industrial sector where IoT devices monitor production plants and hardware. DFR in IoT can then be used to make the industrial workplace safer.

10.5 CONCLUSION

This dissertation conducted a systematic literature review to determine the state of the current body of knowledge with regards to DFR for IoT. This literature review determined

that research with regards to DFR for IoT is still lacking. Various techniques for DFR were then analysed, which led to the development of a new set of requirements for DFR in IoT. The requirements for DFR in IoT were then incorporated in the development of a new model for DFR in IoT. The model was then implemented by developing a prototype based on existing tools and frameworks. This is a major contribution towards the current body of knowledge, with the author being unable to identify any similar prototypes in existing literature.

A simulation was then developed to test the prototype. The simulation contained several steps to determine the performance of the prototype and if it is feasible to conduct DFR on IoT devices in a manner that produces trustworthy electronic evidence. During the various steps of the simulation, a variety of performance metrics were collected from simulated and physical IoT devices. The evaluation of the simulation metrics showed that DFR for IoT is indeed feasible.

This dissertation showed that DFR for IoT is possible and that there are several benefits to proactively gathering evidence from IoT devices. This research served as the starting point for future research with regards to incorporating DFR in the growing world of the IoT.

REFERENCES

- [1] Q. Jing *et al*, "Security of the Internet of Things: perspectives and challenges," *Wireless Networks*, vol. 20, (8), pp. 2481-2501, Nov 2014, 2014.
- [2] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quarterly*, pp. xiii-xxiii, 2002.
- [3] C. Okoli and K. Schabram, "A guide to conducting a systematic literature review of information systems research," 2010.
- [4] C. Hart, *Doing a Literature Review: Releasing the Research Imagination*. Sage, 2018.
- [5] B. Kitchenham *et al*, "Systematic literature reviews in software engineering—a systematic literature review," *Information and Software Technology*, vol. 51, (1), pp. 7-15, 2009.
- [6] H. Vangheluwe, J. De Lara and P. J. Mosterman, "An introduction to multi-paradigm modelling and simulation," in *Proceedings of the AIS'2002 Conference (AI, Simulation and Planning in High Autonomy Systems)*, Lisboa, Portugal, 2002, pp. 9-20.
- [7] P. Beynon-Davies, D. Tudhope and H. Mackay, "Information systems prototyping in practice," *J. Inf. Technol.*, vol. 14, (1), pp. 107-120, 1999.
- [8] J. Menold, T. W. Simpson and K. Jablokow, "The prototype for X framework: exploring the effects of a structured prototyping framework on functional prototypes," *Research in Engineering Design*, vol. 30, (2), pp. 187-201, 2019.
- [9] J. Banks, *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. John Wiley & Sons, 1998.
- [10] M. Baehr, "Evaluation methodology," 2004.
- [11] M. Reith, C. Carr and G. Gunsch, "An examination of digital forensic models," *International Journal of Digital Evidence*, vol. 1, (3), pp. 1-12, 2002.
- [12] E. Delp, N. Memon and M. Wu, "Digital forensics," *IEEE Signal Process. Mag.*, vol. 26, (2), pp. 14-15, 2009.
- [13] S. Almulla, Y. Iraqi and A. Jones, "A STATE-OF-THE-ART REVIEW OF CLOUD FORENSICS," *The Journal of Digital Forensics, Security and Law : JDFSL*, vol. 9, (4), pp. 7-28, 2014.
- [14] Y. Prayudi, A. Ashari and T. K. Priyambodo, "A Proposed Digital Forensics Business Model to Support Cybercrime Investigation in Indonesia," *International Journal of Computer Network and Information Security*, vol. 7, (11), pp. 1-8, Oct 2015, 2015.
- [15] M. O. Hewling, "Digital Forensics: An Integrated Approach for the Investigation of Cyber/Computer Related Crimes." , University of Bedfordshire (United Kingdom), England, 2013.

- [16] S. L. Garfinkel, "Digital forensics research: The next 10 years," *Digital Investigation*, vol. 7, pp. S64-S73, 2010.
- [17] S. Garfinkel *et al*, "Bringing science to digital forensics with standardized forensic corpora," *Digital Investigation*, vol. 6, pp. S2-S11, 2009.
- [18] R. Leigland and A. W. Krings, "A formalization of digital forensics," *International Journal of Digital Evidence*, vol. 3, (2), pp. 1-32, 2004.
- [19] N. M. Karie and S. M. Karume, "DIGITAL FORENSIC READINESS IN ORGANIZATIONS: ISSUES AND CHALLENGES," *The Journal of Digital Forensics, Security and Law : JDFSL*, vol. 12, (4), pp. 43-53, 2017.
- [20] S. Park *et al*, "Research on Digital Forensic Readiness Design in a Cloud Computing-Based Smart Work Environment," *Sustainability*, vol. 10, (4), pp. 1203, 2018.
- [21] M. Elyas *et al*, "Towards a systemic framework for digital forensic readiness," *Journal of Computer Information Systems*, vol. 54, (3), pp. 97-105, 2014.
- [22] R. Rowlingson, "A ten step process for forensic readiness," *International Journal of Digital Evidence*, vol. 2, (3), pp. 1-28, 2004.
- [23] F. Mouton and H. Venter, "A prototype for achieving digital forensic readiness on wireless sensor networks," in *IEEE Africon'11*, 2011, pp. 1-6.
- [24] V. R. KEBANDE and H. Venter, "Obfuscating a cloud-based botnet towards digital forensic readiness," in *Iccws 2015-the Proceedings of the 10th International Conference on Cyber Warfare and Security*, 2015, pp. 434.
- [25] A. Valjarevic and H. S. Venter, "Towards a digital forensic readiness framework for public key infrastructure systems," in *2011 Information Security for South Africa*, 2011, pp. 1-10.
- [26] A. Mouhtaropoulos, C. Li and M. Grobler, "Digital forensic readiness: are we there yet," *J.Int'T Com.L.& Tech.*, vol. 9, pp. 173, 2014.
- [27] C. P. Grobler and C. Louwrens, "Digital forensic readiness as a component of information security best practice," in *IFIP International Information Security Conference*, 2007, pp. 13-24.
- [28] P. M. Trenwith and H. S. Venter, "Digital forensic readiness in the cloud," in *2013 Information Security for South Africa*, 2013, pp. 1-5.
- [29] J. Danielsson and I. Tjostheim, "The need for a structured approach to digital forensic readiness," in *IADIS International Conference E-Commerce*, 2004, pp. 417-421.
- [30] M. Elyas *et al*, "TOWARDS A SYSTEMIC FRAMEWORK FOR DIGITAL FORENSIC READINESS," *The Journal of Computer Information Systems*, vol. 54, (3), pp. 97-105, Spring 2014, 2014.

- [31] K. Reddy, H. Venter and M. S. Olivier, "Using time-driven activity-based costing to manage digital forensic readiness in large organisations," *Inf. Syst. Front.*, vol. 14, (5), pp. 1061-1077, 2012.
- [32] L. Da Xu, W. He and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, (4), pp. 2233-2243, 2014.
- [33] L. Coetzee and J. Eksteen, "The internet of things-promise for the future? an introduction," in *IST-Africa Conference Proceedings, 2011*, 2011, pp. 1-9.
- [34] A. Whitmore, A. Agarwal and L. Da Xu, "The Internet of Things—A survey of topics and trends," *Inf. Syst. Front.*, vol. 17, (2), pp. 261-274, 2015.
- [35] D. Miorandi *et al*, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, (7), pp. 1497-1516, 2012.
- [36] O. Vermesan *et al*, "Internet of things strategic research roadmap," *Internet of Things-Global Technological and Societal Trends*, vol. 1, (2011), pp. 9-52, 2011.
- [37] T. T. Mulani and S. V. Pingle, "Internet of things," *International Research Journal of Multidisciplinary Studies*, vol. 2, (3), 2016.
- [38] P. B. Pathak, "Internet of Things: A Look at Paradigm Shifting," *International Journal of Advanced Research in Computer Science*, vol. 7, (2), pp. n/a, Mar 2016, 2016.
- [39] J. Kaur and K. Kaur, "Internet of Things: A Review on Technologies, Architecture, Challenges, Applications, Future Trends," *International Journal of Computer Network and Information Security*, vol. 9, (4), pp. 57, Apr 2017, 2017.
- [40] K. Tucker *et al*, "Internet Industry: A Perspective Review Through Internet of Things and Internet of Everything," *International Management Review*, vol. 14, (2), pp. 26-35, 56-57, 2018.
- [41] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," *Wireless Personal Communications*, vol. 58, (1), pp. 49-69, 2011.
- [42] R. Roman, P. Najera and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, (9), pp. 51-58, 2011.
- [43] A. Al-Fuqaha *et al*, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, (4), pp. 2347-2376, 2015.
- [44] A. Zanella *et al*, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, (1), pp. 22-32, 2014.
- [45] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, (1), pp. 3-9, 2014.

- [46] S. Zawoad and R. Hasan, "Faiot: Towards building a forensics aware eco system for the internet of things," in *Services Computing (SCC), 2015 IEEE International Conference On*, 2015, pp. 279-284.
- [47] V. R. KEBANDE and I. Ray, "A generic digital forensic investigation framework for internet of things (iot)," in *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference On*, 2016, pp. 356-362.
- [48] D. Lillis *et al*, "Current challenges and future research areas for digital forensic investigation," *arXiv Preprint arXiv:1604.03850*, 2016.
- [49] R. Khan *et al*, "Future internet: The internet of things architecture, possible applications and key challenges," in *Frontiers of Information Technology (FIT), 2012 10th International Conference On*, 2012, pp. 257-260.
- [50] F. Wortmann and K. Flüchter, "Internet of things," *Business & Information Systems Engineering*, vol. 57, (3), pp. 221-224, 2015.
- [51] M. Al Fahdi, N. L. Clarke and S. M. Furnell, "Challenges to digital forensics: A survey of researchers & practitioners attitudes and opinions," in *Information Security for South Africa, 2013*, 2013, pp. 1-8.
- [52] N. Beebe, "Digital forensic research: The good, the bad and the unaddressed," in *IFIP International Conference on Digital Forensics*, 2009, pp. 17-36.
- [53] S. Raghav and A. K. Saxena, "Mobile forensics: Guidelines and challenges in data preservation and acquisition," in *Research and Development (SCORed), 2009 IEEE Student Conference On*, 2009, pp. 5-8.
- [54] A. Zareen and S. Baig, "Notice of violation of IEEE publication principles mobile phone forensics: Challenges, analysis and tools classification," in *Systematic Approaches to Digital Forensic Engineering (SADFE), 2010 Fifth IEEE International Workshop On*, 2010, pp. 47-55.
- [55] E. Oriwoh *et al*, "Internet of things forensics: Challenges and approaches," in *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference On*, 2013, pp. 608-615.
- [56] B. Tank, H. Upadhyay and H. Patel, "A survey on IoT privacy issues and mitigation techniques," in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, Udaipur, India, 2016, pp. 2:1-2:4.
- [57] R. D. Austin, "Digital forensics on the cheap: Teaching forensics using open source tools," in *Proceedings of the 4th Annual Conference on Information Security Curriculum Development*, Kennesaw, Georgia, 2007, pp. 6:1-6:5.
- [58] L. Bu *et al*, "Systematically Ensuring the Confidence of Real-Time Home Automation IoT Systems," *ACM Trans.Cyber-Phys.Syst.*, vol. 2, (3), pp. 22:1-22:23, jun, 2018.
- [59] K. Bampatsalou *et al*, "Current and Future Trends in Mobile Device Forensics: A Survey," *ACM Computing Surveys*, vol. 51, (3), Jul 2018, 2018.

- [60] K. R. Choo *et al*, "Embedded Device Forensics and Security," *ACM Trans.Embed.Comput.Syst.*, vol. 16, (2), pp. 50:1-50:5, dec, 2016.
- [61] T. Zia, P. Liu and W. Han, "Application-specific digital forensics investigative model in internet of things (IoT)," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, Reggio Calabria, Italy, 2017, pp. 55:1-55:7.
- [62] C. Meffert *et al*, "Forensic state acquisition from internet of things (FSAIoT): A general framework and practical approach for IoT forensics through IoT device state acquisition," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, Reggio Calabria, Italy, 2017, pp. 56:1-56:11.
- [63] Y. Yu and T. T. Tun, "Snap forensics: A tradeoff between ephemeral intelligence and persistent evidence collection," in *Proceedings of the 1st ACM SIGSOFT International Workshop on Software Engineering and Digital Forensics*, Paderborn, Germany, 2017, pp. 10-11.
- [64] Group, The Common Digital Evidence Storage Format Working, "Standardizing Digital Evidence Storage," *Commun ACM*, vol. 49, (2), pp. 67-68, feb, 2006.
- [65] A. Tomono, M. Uehara and Y. Shimada, "Transferring trusted logs across vulnerable paths for digital forensics," in *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*, Kuala Lumpur, Malaysia, 2009, pp. 487-492.
- [66] A. Ouaddah, A. A. Elkalam and A. Ouahman, "Harnessing the power of blockchain technology to solve IoT security & privacy issues," in *Second Int. Conf. Internet Things, Data Cloud Comput.(ICC 2017)*, 2017, .
- [67] M. Harbawi and A. Varol, "An improved digital evidence acquisition model for the internet of things forensic I: A theoretical framework," in *2017 5th International Symposium on Digital Forensic and Security, ISDFS 2017*, 2017, .
- [68] E. Oriwoh and P. Sant, "The forensics edge management system: A concept and design," in *Proceedings - IEEE 10th International Conference on Ubiquitous Intelligence and Computing, UIC 2013 and IEEE 10th International Conference on Autonomic and Trusted Computing, ATC 2013*, 2013, pp. 544-550.
- [69] S. Perumal, N. M. Norwawi and V. Raman, "Internet of things(IoT) digital forensic investigation model: Top-down forensic approach methodology," in 2015, pp. 19-23.
- [70] K. Dhar and Y. Pingle, "Digital forensic investigations (DFI) using internet of things (IoT)," in *Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom 2016*, 2016, pp. 1443-1447.
- [71] C. Esposito *et al*, "Challenges of Connecting Edge and Cloud Computing: A Security and Forensic Perspective," *IEEE Cloud Computing*, vol. 4, (2), pp. 13-17, 2017.
- [72] V. R. Kebande, N. M. Karie and H. S. Venter, "Cloud-centric framework for isolating big data as forensic evidence from IoT infrastructures," in *2017 1st International*

Conference on Next Generation Computing Applications, NextComp 2017, 2017, pp. 54-60.

[73] M. Hossain, R. Hasan and S. Zawoad, "Trust-IoV: A trustworthy forensic investigation framework for the internet of vehicles (IoV)," in *Proceedings - 2017 IEEE 2nd International Congress on Internet of Things, ICIOT 2017, 2017*, pp. 25-32.

[74] V. R. Kebande *et al*, "How an IoT-enabled "smart refrigerator" can play a clandestine role in perpetuating cyber-crime," in 2017, pp. 1-10.

[75] C. Shin *et al*, "Potential forensic analysis of IoT data: An overview of the state-of-the-art and future possibilities," in 2017, pp. 705-710.

[76] M. Banday, "Enhancing the security of IOT in forensics," in 2017, pp. 193-198.

[77] J. Surbiryala and C. Rong, "Secure customer data over cloud forensic reconstruction," in 2018, pp. 1-4.

[78] A. MacDermott, T. Baker and Q. Shi, "Iot forensics: Challenges for the ioa era," in 2018, pp. 1-5.

[79] M. Chernyshev *et al*, "Internet of Things Forensics: The Need, Process Models, and Open Issues," *IT Professional*, vol. 20, (3), pp. 40-49, 2018.

[80] S. Watson and A. Dehghantanha, "Digital forensics: the missing piece of the Internet of Things promise," *Comput. Fraud Secur.*, vol. 2016, (6), pp. 5-8, 2016.

[81] H. Chung, J. Park and S. Lee, "Digital forensic approaches for Amazon Alexa ecosystem," *Digit. Invest.*, vol. 22, pp. S15-S25, 2017.

[82] I. Homem, S. Dosis and O. Popov, "LEIA: The live evidence information aggregator: Towards efficient cyber-law enforcement," in *2013 World Congress on Internet Security, WorldCIS 2013, 2013*, pp. 156-161.

[83] H. A. Brumfitt, B. Askwith and B. Zhou, "Lightweight forensics application: Lightweight approach to securing mobile devices," in *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Secure Computing, DASC 2015 and 13th IEEE International Conference on Pervasive Intelligence and Computing, PICom 2015, 2015*, pp. 795-800.

[84] A. Nieto, R. Rios and J. Lopez, "A methodology for privacy-aware iot-forensics," in *Proceedings - 16th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 11th IEEE International Conference on Big Data Science and Engineering and 14th IEEE International Conference on Embedded Software and Systems, Trustcom/BigDataSE/ICCESS 2017, 2017*, pp. 626-633.

[85] L. Caviglione, S. Wendzel and W. Mazurczyk, "The Future of Digital Forensics: Challenges and the Road Ahead," *IEEE Secur. Privacy*, vol. 15, (6), pp. 12-17, 2017.

- [86] X. Feng, E. S. Dawam and S. Amin, "A new digital forensics model of smart city automated vehicles," in *Proceedings - 2017 IEEE International Conference on Internet of Things, IEEE Green Computing and Communications, IEEE Cyber, Physical and Social Computing, IEEE Smart Data, iThings-GreenCom-CPSCoM-SmartData 2017*, 2018, pp. 274-279.
- [87] D. R. Clark *et al*, "DROP (DRone Open source Parser) your drone: Forensic analysis of the DJI Phantom III," *Digit. Invest.*, vol. 22, pp. S3-S14, 2017.
- [88] N. H. N. Zulkipli, A. Alenezi and G. B. Wills, "IoT forensic: Bridging the challenges in digital forensic and the internet of things," in *IoT BDS 2017 - Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*, 2017, pp. 315-324.
- [89] O. Yakubu, N. C. Babu and O. Adjei, "A review of digital forensic challenges in the internet of things (IOT)," *Int. J. Mech. Eng. Technol.*, vol. 9, (1), pp. 915-923, 2018.
- [90] V. R. Kebande, N. M. Karie and H. S. Venter, "Adding Digital Forensic Readiness as a security component to the IoT domain," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, (1), pp. 1-11, 2018.
- [91] V. R. Kebande, N. M. Karie and H. S. Venter, "Functional requirements for adding digital forensic readiness as a security component in IoT environments," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, (2), pp. 342-349, 2018.
- [92] S. Papadopoulos and S. Snail, *Cyberlaw@ SA III: The Law of the Internet in South Africa*. Van Schaik Pretoria, 2012.
- [93] A. Nieman, "Cyberforensics: bridging the law/technology divide," *Journal of Information, Law and Technology*, vol. 2009, (1), 2009.
- [94] Anonymous "Electronic Communications and Transactions Act," vol. 25, 2002.
- [95] Anonymous "Consultation Paper on Documentary and Electronic Evidence (LRC CP 57 - 2009)," 2009.
- [96] J. Kruger and H. Venter, "Requirements for IoT forensics," in *NextComp 2019*, Mauritius, 2019.
- [97] J. Kruger and H. Venter, "State of the art in digital forensics for the internet of things," in *14th International Conference on Cyber Warfare and Security*, Stellenbosch University, South Africa, 2019.
- [98] (2019). *GRR Rapid Response alternatives*. Available: <https://linuxsecurity.expert/tools/grr-rapid-response/alternatives/>.
- [99] (2019). *MIG: Mozilla InvestiGator*. Available: <https://mig.mozilla.org/>.
- [100] (2018). *MIG (Mozilla InvestiGator)*. Available: <https://linuxsecurity.expert/tools/mig/>.
- [101] (2018). *Bitscout*. Available: <https://linuxsecurity.expert/tools/bitscout/>.

- [102] (14 May 2019). *Project Bitscout*. Available: <https://github.com/vitaly-kamluk/bitscout>.
- [103] (2019). *FIR*. Available: <https://github.com/certsocietegenerale/FIR>.
- [104] (2018). *FIR (Fast Incident Response)*. Available: <https://linuxsecurity.expert/tools/fir/>.
- [105] (2019). *SECURITY INCIDENT RESPONSE FOR THE MASSES*. Available: <https://thehive-project.org/>.
- [106] (5 June 2019). *TheHive*. Available: <https://linuxsecurity.expert/tools/thehive/>.
- [107] (2018). *OSSEC*. Available: <https://linuxsecurity.expert/tools/ossec/>.
- [108] (2019). *The World's Most Widely Used Host-based Intrusion Detection System*. Available: <https://www.ossec.net/>.
- [109] (4 July 2019). *GRR Rapid Response*. Available: <https://linuxsecurity.expert/tools/grr-rapid-response/>.
- [110] (2019). *GRR Rapid Response*. Available: <https://grr-doc.readthedocs.io/en/latest/>. DOI: Revision b16af3f1.
- [111] Anonymous (2018, February 1). *Linux Lite*. Available: <https://www.linuxliteos.com/>. DOI: 2019, June 26.
- [112] M. Bushkov, vol. Email, April, 2019.
- [113] (2019). *Protocol Buffers*. Available: <https://developers.google.com/protocol-buffers/>.
- [114] (2019). *Zabbix*. Available: <https://www.zabbix.com/>.
- [115] (2019). *Grafana*. Available: <https://grafana.com/grafana>.
- [116] V. Schmitt and J. Jordaan, "Establishing the Validity of Md5 and Sha-1 Hashing in Digital Forensic Practice in Light of Recent Research Demonstrating Cryptographic Weaknesses in These Algorithms," *International Journal of Computer Applications*, vol. 68, (23), pp. n/a, 2013.
- [117] (2019). *SHA256 Checksum*. Available: http://emn178.github.io/online-tools/sha256_checksum.html.
- [118] M. Harbawi and A. Varol, "An improved digital evidence acquisition model for the internet of things forensic I: A theoretical framework," in 2017, pp. 1-6.

APPENDIX A – Systematic literature review evaluation table

This appendix contains the table used for evaluation of the current body of knowledge with regards to the state of the art of DF of IoT.

This table serves as the research for chapter 4 of this dissertation.

Paper name	IOT	Mobile devices	DF	IOT DF	Cases in law	Process or standard followed	DFR
A Survey on IoT Privacy Issues and Mitigation Techniques	The first paper contributes toward the body of knowledge by enhancing privacy in the IoT, while also increasing security in the IoT.					The paper shows that the Constrained Application Protocol was standardized by the Internet Engineering Task force that could be used with IoT devices. The paper however shows that security is still lacking and therefore they introduced the paper to address privacy challenges.	
Digital Forensics on the Cheap: Teaching Forensics Using Open-source Tools	No direct contribution.		The paper discusses the possible use of open-source digital forensic software, but is limited to standard desktop computers, not IoT devices. The availability of open-source	No contribution	The paper mentioned a legal case where MD5 as a hashing algorithm to prove the authenticity of digital forensic evidence. The contribution towards this paper would	The paper briefly discusses the forensic process developed by Casey in. The paper only provided a basic introduction to a standard digital forensic investigation; thus, it is not of extreme value towards this	

			digital forensic software aimed at IoT devices would however be highly beneficial towards the advancement of research with regards to digital forensics in IoT.		thus be a recommendation to use a stronger form of encryption in IoT devices, while also making sure that the encryption protocol is lightweight enough so that devices with limited resources would still be able to encrypt the data if they need to generate hashes of data.	paper.	
Systematically Ensuring the Confidence of Real-Time Home Automation IoT Systems	The paper introduces techniques that can be used to assist people in confidence checking of home automation systems. The paper does however not make a						

	significant contribution towards this paper [58].							
Current and Future Trends in Mobile Device Forensics: A Survey	The paper focuses on mobile device forensics and see the MDs as possible IoT devices. The paper thus contributes towards IoT, as it expands the inclusion of MDs into IoT devices.	The paper focuses primarily on MD digital forensics. The paper sees MDs as an integral part of modern society. The paper describes MDs as digital witnesses for their owners, which shows how important mobile digital forensics is.	The paper contributes towards digital forensics as it mentions that mobile forensics is a sub-domain of digital forensics. Mobile forensics is defined as the process of acquiring evidence about some incident or criminal activity that involved MDs. As mentioned earlier, the paper also introduces the concept of MDs forming part of IoT devices, therefore, IoT	The paper furthermore highlights the need for forensic tools to make use of unified formats and data structures to facilitate interoperability between digital forensic software. The contribution towards this paper is that IoT forensics will add to the interoperability issue if standards for IoT forensics and IoT forensic tools are not developed.			The paper shows the need for automated procedures that can help facilitate investigations.	

			forensics can also be a subdomain of digital forensics.				
Embedded Device Forensics and Security	The paper contributes towards the IoT by introducing the idea that rather than developing lightweight cryptographic algorithms for embedded devices, resource intensive computations should be sent to nearby devices with more computing power.			The paper notes the need for embedded device digital forensic techniques. Due to the close relationship between embedded devices and IoT devices, this is a contribution towards IoT digital forensics.			
Application-Specific Digital Forensics Investigative Model in Internet of Things (IoT)	The paper contributes towards IoT due to noting the lack of unified standards for the IoT. The		The paper notes that it is vital that digital forensic measures are developed alongside	The paper contributes enormously towards IoT digital forensics. The paper notes that currently there is			

	<p>lack of standards noted in this paper is mostly aimed towards digital forensic standards. The paper notes that 84% of IoT adopters have been involved in security breaches and that a lot of work still needs to be done in order to secure IoT technologies.</p>		<p>security measures so as to ensure that attacks can be traced, and sources of attacks can be identified. The paper also notes that standard formats for digital forensic evidence will increase the process of finding evidence.</p>	<p>little focus on IOT digital forensics in academic literature and research. The paper notes that due to the various complexities of storage mediums in IoT devices, IoT digital forensics requires specialized forensic methods. The paper introduces an Application-Specific Digital Forensics Investigative Model. The model is based on the notion that the flow of information for the forensic process is based on the type of application that is investigated. The model uses the example of smart homes, wearable technology and</p>			
--	--	--	--	--	--	--	--

				<p>smart cities. The model notes that say for instance smart home devices form part of the investigation, then the information collected from the devices will be different from the information collection when wearable technology formed part of the investigation. The application of smart home devices can for example give information about Wi-Fi connections at the time of the incident, and wearable technology would be able to give information about heart rates and body postures at the time of the incident. The</p>			
--	--	--	--	---	--	--	--

				information collected from the IoT devices will follow a similar digital forensic investigation, but the evidence will be contextual based on the different applications.			
Forensic State Acquisition from Internet of Things (FSAIoT): A General Framework and Practical Approach for IoT Forensics Through IoT Device State Acquisition	The paper notes that IoT devices are not developed based on standards and they are extremely volatile due to always being connected. The paper notes that there is currently no standardized operating system for IoT devices and there exist a lack of standardization	The paper contributes greatly towards MDs, as the paper classifies MDs as IoT devices.	The paper notes that logs are the most common data source used for digital forensic investigations.	The paper makes an enormous contribution towards IoT digital forensics by introducing a Forensic State Acquisition from Internet of Things (FSAIoT). Their model is based on the notion of using the states of IoT devices to gather evidence surrounding an event, rather than using system data. They used controllers to connect IoT			The paper contributes toward digital forensic readiness by noting that most digital forensic practitioners chase historical data, rather than being able to have forensic data beforehand, which increases the speed of forensic Investigations.

with regards to IoT devices.

devices to, which then logged the state changes of devices. They based their model on state changes as IoT historical records are limited or non-existent. Their model also requires controllers to be forensically sound and not alter the states of the devices. The controllers should also provide reliable logs of times and dates of events. The controller should also be able to securely store collected IoT data. The paper furthermore states that most IoT digital forensic research is theoretical and that practical tests are limited. The paper did however

				<p>conduct a practical test using a scenario which proved the model to be feasible.</p> <p>The paper does however note that the connectivity between different IoT devices is challenging due to a lack of standardized interfaces and that the model does not give access to historical data, as the data is collected and then stored.</p>			
<p>Snap Forensics: A Tradeoff Between Ephemeral Intelligence and Persistent Evidence Collection</p>			<p>The paper notes that live forensics can be challenging to implement but can be highly beneficial for forensic investigations. The paper notes the need for live</p>	<p>The paper notes that live forensics may be inevitable and vital in digital forensics where IoT devices is involved. The paper thus contributes greatly towards this paper, due to noting the importance of live forensics with</p>			<p>The paper contributes greatly towards DFR due to noting that some evidence, such as network packets, are not always stored and accessible when forensic</p>

			forensics by using the example of network packets that only live temporarily and may not even be stored and accessible later in time when forensic investigations take place.	regards to IoT devices as IoT devices change states continuously.			Investigations take place. The paper thus shows the importance of preparing systems for capturing potential evidence.
Standardizing Digital Evidence Storage			The paper does not contribute excessively towards this paper, as the paper is only concerned with the lack of standardization in digital forensic formats.				
Transferring Trusted Logs Across Vulnerable Paths for Digital			The paper contributes toward digital forensics by developing a process that	The process introduced by the paper can be useful in IoT forensics as evidence can be			

Forensics			can be used to guarantee the integrity of logs transferred over insecure networks. The process requires each server that logs pass through to be sent to the final server. The final server then compares the hash files of each server along the route to determine whether or not the evidence has been tampered with.	transferred across unsecure routes.			
Harnessing the Power of Blockchain Technology to Solve IoT Security & Privacy Issues	The paper introduces the concept of using blockchain technology to grant access to IoT devices. The concept relies on the security of the		Blockchain technology can be used to provide integrity to requests and keep a ledger of all requests, or in blockchain	The blockchain can be used to secure IoT transactions, but it is computationally intensive and can take time to compute the necessary blocks to approve			The use of the Blockchain can be useful for DFR as a blockchain keeps a record of all transactions that took place. Each block

	blockchain technology to validate requests and grant access to IoT devices.		terminology, transactions.	transactions. The concept developed in the paper can possible be useful in digital forensics using IoT devices.			builds on the previous block, thus increasing the integrity of older blocks.
An Improved Digital Evidence Acquisition Model for the Internet of Things Forensic I: A Theoretical Framework				The first paper contributes greatly towards IoT digital forensics. The paper introduces a revised evidence acquisition model for IoT devices. The paper does this due to noting that most of the proposed IoT digital forensics models are premature and need verification. The model proposed in this paper is based on the notion of starting the investigation from the IoT devices that produced the last evidence and then navigate through devices		The paper introduces a revised process that can be followed when forensics investigations are faced with IoT devices.	

				that interacted with the IoT device. This limits the number of devices and nodes in the network that needs to undergo digital forensic investigations.[118]			
Internet of Things Forensics: Challenges and Approaches	The paper contributes towards the IoT by noting that data generated by IoT Devices will increase significantly.	The paper also notes that MDs form part of the IoT.		The paper notes that in IoT investigations, the diverse types of devices that need to be investigated will increase significantly. The paper also notes that there will be an enormous increase in the amount of file formats that will be used by the variety of IoT devices. The paper notes that IoT devices cannot always be seized for investigation (such as pacifiers), but the next best thing	The paper contributes toward the legal implication of IoT devices in forensic investigations due to noting that cases where IoT devices are involved, the complexity of the cases will increase due to IoT devices sometimes travelling between networks and even various barriers such as different countries. Cases		The paper Contributes toward DFR by noting that DFR should not encourage situations where all information is stored, such as personal information. There should be limits to the data captured beforehand.

				<p>can be investigated, such as logs etc. The digital forensic investigation remains the same as standard digital forensic investigations with the exception of extracting data from the next best thing.</p>	<p>involving IoT devices will also face challenges due to IoT devices being used between private, personal and public networks by individuals. The paper notes that legal frameworks must be developed in order to allow investigators to do digital forensic investigations on IoT devices, as such devices may not be traditional computing devices. Legal frameworks should also allow digital forensic investigators to seize control of IoT systems that</p>	
--	--	--	--	---	---	--

					<p>can cause harm to external parties, such as botnets running on IoT devices. Owners may refuse to turn off IoT systems as owners can see the systems as critical. Investigators may need the authority to require malicious systems to be turned off and legal frameworks need to be developed to enable them to do so.</p>		
<p>The Forensics Edge Management System: A Concept and Design</p>	<p>The paper introduces the Forensics Edge Management System (FEMS) that aims to autonomously provide security and forensic</p>			<p>The paper makes a great contribution towards IoT digital forensics through the introduction of the FEMS. The system provides users and digital forensic</p>		<p>The paper follows standardized digital forensic investigation techniques but applies the techniques toward the IoT.</p>	<p>The FEMS is reactive not proactive.</p>

	services within an IoT context.			<p>investigators with reports of events that occurred as well as digital forensic information as to what caused events. The FEMS follows standardized forensic techniques but applies it to the IoT.</p> <p>The paper notes that automated forensics will save digital forensic investigators time due to IoT being complex.</p> <p>The paper further notes that forensic solutions that do not take in to account the nature of the IoT, will eventually become too slow to be used for the IoT.</p> <p>The FEMS will ensure that data collected will be</p>			
--	---------------------------------	--	--	---	--	--	--

				compressed and stored for a period. The data collected will also be mapped onto a timeline of events and users will be alerted of the events that occurred. The FEMS is a reactive system.			
Internet of Things(IoT) Digital Forensic Investigation Model: Top-Down Forensic Approach Methodology	The paper contributes towards the IoT by noting that the main challenges facing the IoT is as follows: there is no clear approach for unique identifiers of IoT devices, new IoT reference architectures are not being developed, the exchange of sensor information is difficult in the			The paper identifies the gap in current research in IoT digital forensics by noting that very little research has been done in IoT digital forensics. The paper provides a good overview of the existing IoT digital forensic models. The Digital Forensic Investigation Model (DFIM) is a four-tier model that aims to expose evidence that is hidden. The model			

heterogeneous nature of the IoT, trust establishment is an on-going challenge in the IoT environment and there is a lack of large scale testing and learning environments for the IoT.

is however not fit for the IoT as the IoT relies heavily on physical evidence. The next model is the Hybrid model. The Hybrid model aims to look at both crime scene investigation and laboratory investigation, thus physical evidence and digital evidence. The model is however very time consuming and not fit for the IoT due to its narrow application. The next model is the 1-2-3 Zones of Digital Forensics. The 1-2-3 Zones model is the most applicable to the IoT as divides the internal network into Zone zero, zone two as the border network and zone three as

				<p>the external network. The model does however lack clear information on how to conduct digital forensic investigations.</p> <p>The paper introduces the IoT Based Digital Forensic Model. The model starts with the planning of the investigation and the obtainment of a warrant. Investigators must then identify the medium used to communicate with the device involved in the incident. Once the device has been located all data related to the device must be obtained. Once the data has been obtained, investigators should continue with common</p>			
--	--	--	--	---	--	--	--

				digital forensic investigation procedures such as chain of custody, lab analysis, storage and reporting. The paper does however not describe the model in detail and limited information is given as to how to conduct the steps provided.			
A Generic Digital Forensic Investigation Framework for Internet of Things (IoT)				The paper contributes towards IoT digital forensics by noting that there is no accepted digital forensic framework for IoT environments. The paper introduces the Digital Forensic Investigation Framework for IoT (DFIF-IoT). The model consists of a proactive process, IoT forensics and a		The paper is based on ISO 274043 which is of great advantage to the model developed, due to the knowledge that the framework complies with set standards. The use of ISO 274043 increases the likelihood that the evidence gather by the IoT devices, will be acceptable in legal proceedings.	The paper notes that the proactive process is a science. The DFR involves planning and preparing for potential IoT incidents.

				reactive process which represents the forensic investigation after the occurrence of an incident. The model is of high value to IoT digital forensics due to the introduction of forensic readiness which can help ensure that investigations come to conclusions much faster.			
Digital Forensic Investigations (DFI) using Internet of Things (IoT)	The paper made no contribution as the paper used IoT devices to generate data for standard forensic investigations.						
Challenges of Connecting Edge and Cloud Computing: A Security	The paper contributes towards the IoT by noting the paradigm of fog computing. Fog				The paper mentions working parties established by the European Union to		

and Forensic Perspective	computing entails the introduction of intermediate computing nodes between IoT devices and the cloud. The fog devices can greatly enhance the computing power of the IoT by aggregating data before sending the data to the cloud.				evaluate data protection issues of the IoT. The legal requirements mentioned in the paper only concerns user consent with regards to personal data collected by the IoT. The paper merely contributes by noting that fog nodes can increase the security of users by giving users more control over their data.		
Cloud-Centric framework for isolating Big Data as Forensic Evidence from IoT Infrastructures	The paper contributes toward the IoT by introducing a cloud-centric framework that isolates IoT data from cloud environments as forensic evidence.		The paper notes that digital forensics does not just involve the process of obtaining evidence but includes the forensic analysis of the	The paper contributes towards IoT digital forensics with the introduction of a reactive digital forensic process for obtaining forensic evidence from the cloud environment that		The process developed in the paper relies heavily on ISO/IEC 27043 which is a great contribution towards standards, as ISO27403 defines standardized processes for the preservation of	

			evidence as well as the presentation of the evidence in legal environments such as courtrooms.	was produced by IoT devices. The solution developed by the paper is an Agent-Based Solution (ABS). This entails applications running on the IoT devices to capture information about events such as IoT device state changes or connections to services. The agents then transfer the data to the cloud environment in a forensically sound manner which allows for the data to be analysed in digital forensic investigations.		digital evidence. The paper furthermore notes that there existed no standard procedures for IoT devices and that needs to be addressed. The paper provides a standard as mentioned earlier to address the lack of IoT standards. The paper also notes that frameworks cannot be easily accepted due to the lack of a standard for the development of IoT devices.	
Trust-IoV: A Trustworthy Forensic Investigation Framework for the Internet of Vehicles (IoV)	The paper contributes towards the IoT by focusing on a specific subset of IoT devices, namely the			The paper contributes towards the IoT by introducing the first framework specifically for digital forensics			The paper contributes Towards DFR due to storing all interactions in a trusted and secure

	Internet of Vehicles (IoV).			<p>with regards to the IoV. The model proposed in the paper aims to be tamper proof and store evidence in a trustworthy manner.</p> <p>The model is based on the notion of a Forensic Header Handler that requires all interactions to be sent through the Forensic Header Handler. All interactions is signed and stored in a database with a read only API. Evidence is published in a secure blockchain so as to ensure the integrity of interactions.</p>			<p>manner.</p> <p>The model also makes use of a blockchain to store interactions in a secure manner so as to increase the trustworthiness of information that can be used in later digital forensic Investigations.</p>
How an IoT-Enabled “Smart Refrigerator” Can Play a Clandestine	The paper contributes towards the IoT by providing an example of how	The paper also sees MDs as part of the IoT.		The paper contributes to IoT digital forensics by introducing a proactive digital		The paper contributes towards processes and standards by noting the gap in the	The paper contributes to DFR by proactively preventing

<p>Role in Perpetuating Cyber-Crime</p>	<p>IoT devices can be used to launch further attacks on other systems, once said IoT devices have been hacked and used as intermediaries in cyberattacks. The paper contributes towards the IoT by highlighting the fact that some known vulnerabilities exist in the IoT and that more efforts should be made to secure IoT devices.</p>			<p>forensics model that filters communication which aims to detect vulnerable points in IoT smart home devices. The model introduced in this paper is the Smart Refrigerator Crime Propagation Model. The model mentions possible countermeasures that can help to secure IoT devices. The countermeasures include: DFR, intrusion detection systems, Multi-factor authentication, the use of various passwords, data encryption and best practices such as only sharing a minimal amount of information. The paper furthermore</p>		<p>amount of available processes that aims to secure IoT devices.</p>	<p>IoT-based digital crimes.</p>
---	---	--	--	---	--	---	----------------------------------

				<p>contributes towards IoT digital forensics by indicating the need for IoT digital forensic systems that makes use of the countermeasures mentioned. The paper is thus a high-level approach to prevent cybercrimes taking place due to the exploitation of IoT devices.</p>			
<p>Potential Forensic Analysis of IoT Data</p>	<p>The paper contributes towards the IoT by noting several examples of IoT devices and their underlying technologies, such as the Amazon Echo.</p>			<p>The paper contributes to the IoT digital forensics by noting that home routers are placed in zone 2 (mentioned earlier) and therefore routers have access to all network traffic between all IoT devices in the home network, and the internet. The</p>	<p>The paper makes a great contribution towards the use of IoT devices in legal proceedings. The paper provides an example of a murder case in 2017 where the data from a smart heater provided</p>	<p>The paper notes that there is currently a lack of techniques for extracting digital evidence from IoT devices.</p>	<p>The paper contributes towards DFR by noting that certain devices such as routers can be used to log information in order to aid with forensic investigations.</p>

				<p>router in zone two is therefore a great candidate for IoT evidence gathering. The paper notes that there exists an abundance of research possibilities in IoT digital forensics.</p>	<p>evidence into excessive hot water being used to clean the crime scene. The case provides an example as to how IoT devices can provide deeper insight into criminal investigations and not just cybercrimes.</p>		
Enhancing the Security of IOT in Forensics	<p>The paper notes that the IoT needs to be secured more effectively in order to enhance the working of IoT digital forensics.</p>					<p>The paper notes that portable criminology needs to be enhanced in order to augment IoT digital forensics.</p>	
Secure Customer Data over Cloud Forensic Reconstruction	<p>No contribution apart from noting that IoT devices can generate a significant amount of data that can lead to big data in the</p>						

	cloud environment.						
IoT Forensics: Challenges For The IoA Era	The paper makes an interesting contribution towards the IoT by naming the IoT, the Internet of Anything (IoA). The paper goes to show how many possibilities of IoT devices exist.			The paper highlights the existence of procedures for digital forensic investigations in computers, hard drives and mobile phones, but not for IoT devices.		The paper highlights the ever-increasing number of devices that can be of forensic interest, due to the dawn of the IoT. The paper notes that standards for IoT digital forensics should have the same objectives as physical crime scenes: identifying the crime that took place, analysing the evidence available and then to identify the parties involved in the crime. The paper notes that objects of interest may not always be available or accessible which can be challenging for IoT digital forensic investigations. The paper notes that cloud investigations	

						<p>will become important as IoT devices most likely stores data in the cloud.</p> <p>The paper identifies three sources of IoT digital forensic evidence: evidence on the smart devices, evidence collected from hardware and software that provide a connection with IoT devices such as firewalls and then finally hardware and software outside the network of investigation such as cloud services and even Internet Service Providers.</p> <p>The paper makes an interesting contribution by stating that the traditional search and seizure procedures are impractical due to</p>	
--	--	--	--	--	--	---	--

						the data being stored in the cloud.	
Internet of Things Forensics	The paper contributes towards the IoT by noting the complexity of the underlying infrastructure of IoT devices.			The paper mentions the already discussed Next Best Thing model, the Last on Scene Model, Forensic Aware IoT Model and the Digital Forensic Investigation Framework for IoT model that will be evaluated further in this paper. The paper contributes to the processes by noting that the FAIoT and the LoS model is yet to be tested in practical environments.	The paper mentions the value of IoT devices in legal cases and mentions that IoT wearable devices have already aided in cases where personal injury or murder took place. The paper furthermore contributes towards the legal implications of IoT digital forensic cases by noting that IoT devices introduces a new level of complexity with regards to cross-border jurisdictional issues as IoT devices can	The paper contributes towards the IoT digital forensics by noting several challenges with regards to IoT digital forensics: uncertainty about the location of stored data, how data is stored and what attributes about data are stored. The next challenge is the difficulty in maintaining chain of custody over IoT generated evidence. Another identified challenge is that of the lack of applicableness of current digital forensic techniques on IoT devices. The final challenge identified is that of diverse types of storage and file formats used in IoT	The paper Contributes Towards DFR by noting That there exists a need for secure real time evidence storage repositories. The paper however notes that how this can be achieved is still an on-going issue. The paper also notes that there exist teams that aim to enhance the security of the IoT, such as the US IoT Cybersecurity Improvement Act (2017). Improved security does

					store and transfer data over a variety of logical and physical borders.	devices. The paper notes that automation of the IoT digital forensic process can be greatly beneficial to the digital forensic environment by ensuring effective gathering and analysis of forensic evidence as well as ensuring the repeatability of the process.	however not necessarily improve DFR. The paper also notes that trusted repositories may lack the ability to intelligently analyse the data stored as the collected data will be made up of different formats and granularity levels, due to the lack of standards.
Digital forensics: the missing piece of the Internet of Things promise		The paper contributes towards MDs in relation to this paper by noting that even investigations into MDs are challenging due to investigators		The paper contributes towards IoT digital forensics by noting that IoT digital forensics can be challenging due to the data related to incidents residing on a multitude of possible locations such as on the	The paper notes that device manufacturers should be aware of the possibility that their devices might be at the centre of digital forensic investigations. Device	The paper notes that there currently does not exist any standard that can be followed in order to retrieve data from IoT devices due to the occurrence of cyber-events.	

		struggling to gain access to tools that supports the latest device types, encryption capabilities and advancements in mobile operating systems, such as embedded devices and wearable tech.		device or even in the cloud.	manufacturers should thus be able to provide statements as to how data can be extracted from the device; otherwise they should be able to provide reasons as to why that data cannot be extracted from the devices.		
Digital forensic approaches for Amazon Alexa ecosystem				The paper contributes towards IoT digital forensics by conducting an experiment specifically on the Amazon Alexa smart home system. The paper discovered that the Amazon system does not yet support native API access to the public, but they made use of			

				<p>unofficial API's to gain access to the data generated and received by the Amazon Echo Smart home device. The paper concludes that API's that integrate with IoT devices can be highly beneficial for digital forensic investigations. The also concluded that the provision of timestamps with the data generated by the API's is of extreme value to creating timelines of events.</p>			
LEIA: The Live Evidence Information Aggregator			<p>The paper contributes towards digital forensics by developing the LEIA modal to deal with the large scale of data that digital forensic investigators</p>				

			<p>will be facing. The paper argues that one of the sources of such copious amounts of data, is the increasing adoption of the IoT. The modal introduced in the paper is a Host-based Hypervisor (HbH), Peer-to-peer Distribution Architecture (P2P-da), Cloud-based Backend (CBB) with a law enforcement controller (LEC). The HbH is a secure virtualization operating system</p>				
--	--	--	---	--	--	--	--

			<p>monitoring the operating system for intrusions. HbH communicate over P2P-Da to inform other HbH of issues in order to counter vulnerabilities. The CBB checks uploaded hashes of system files with the trusted hashes of system files.</p>				
<p>Lightweight Forensics Application: Lightweight Approach to Securing Mobile Devices</p>		<p>The paper introduces the LFA to enhance the security of MDs, but it can also be used in the IoT environment.</p>		<p>The LFA contributes towards the IoT digital forensics as it is designed to be able to function in the IoT environment as well. The model consists of the LFA the runs on the devices, a Central</p>			

				<p>Security Manager (CSM) that is connected to the network and a collaborative component (Col) that operates between the devices in the network.</p> <p>The LFA is lightweight and collects forensic-valuable data and sends it to the CSM. The CSM requires processing power to analyse data collected from the devices and provide reports. The CSM also operated automatically. The Col enables the same device to operate differently in different networks.</p> <p>Example: more freedom in home networks but</p>			
--	--	--	--	--	--	--	--

				restricted in work environments.			
A Methodology for Privacy-Aware IoT-Forensics				The paper contributes towards IoT digital forensics by introducing PProFIT which is an IoT-forensics model that takes privacy into consideration based on ISO29100. The paper ensures that the user is aware of data collection and that the user can consent to it. The intent for the data collection must be specified, only the minimum required information may be collected, and the data must remain safe until deletion after the investigation. The model is based on software installed on the IoT devices		The model is based on ISO29100 which is of significant importance for user privacy.	

				<p>which facilitates the requirements specified in the ISO standard.</p> <p>The paper notes the lack of IoT-specific digital forensic models.</p>		
<p>The Future of Digital Forensics: Challenges and the Road Ahead</p>	<p>The paper contributes towards the IoT by noting that IoT devices can be targeted by cyberattackers and turned into “zombies” that can be used to launch attacks on other systems. The sheer amount of IoT devices can cause great havoc if they are turned into “zombies”.</p>		<p>The paper contributes towards digital forensics by noting several challenges facing digital forensics. Some of the challenges relevant to this paper include the increase in the heterogeneity of devices that need to be imaged, as well as the increase in use of cryptography in devices.</p>	<p>The paper contributes towards IoT digital forensics by noting that IoT devices do not only provide information about the digital world, but also the physical world as IoT sensors detect and react upon changes in the physical environment. Furthermore, the paper notes several challenges facing IoT digital forensics, such as restricted resources in IoT devices causing lack of persistent</p>		<p>The paper notes the need for standard file formats in order to assist digital forensic investigations to take place faster.</p>

				recording, different proprietary interfaces to IoT devices and reduced energy in devices causing incomplete data.			
A New Digital Forensics Model of Smart City Automated Vehicles	The paper extends the IoT by noting the importance of Vehicles as part of the IoT device list.		The paper introduces a DF model for the investigation of Autonomous Automated Vehicles (AAV)	The paper notes the need for forensically sound evidence from IoT vehicles, as the data from these vehicles can lead to court cases where accidents occurred.		The paper notes the lack of a standard and procedures when dealing with digital evidence from vehicles.	
DROP (Drone Open-source Parser) your drone: Forensic analysis of the DJI Phantom III	The paper contributes toward the IoT by noting that drones form part of IoT devices		The paper introduces a process for investigating a case which involved a drone. This is a rather unique case for DF, but drones are becoming increasingly part of society and the paper noted several examples	The paper introduces a tool that can be used to parse DAT files from the DJI drones into a forensically sound manner. The paper notes the need for further research with regards to drones and evidence extraction from the drones.	The paper notes the possible increase in the amount of legal cases where drones can be involved and notes the need for processes dealing with evidence in such cases.	The paper introduces procedures that can be used by examiners to investigate cases where DJI drones are involved.	

			where drones have already been used for malicious activities.				
IoT Forensic: Bridging the Challenges in Digital Forensic and the Internet of Things				<p>The paper notes the existence of DF models, but notes the lack of models applicable to the IoT.</p> <p>The paper notes the importance of real-time forensics where suspicious traffic or events trigger the forensic phase.</p>			<p>The paper notes the need for readiness for DF investigations. The paper presents DFR as preparing management aspects, such as investigation strategies, preparing the tools and infrastructure to support the investigations as well as obtaining authorization for investigations. The paper also notes technical readiness to ensure that only relevant</p>

							devices form part of the investigation. The paper makes interesting contributions towards DFR, but not in a manner of systems to retrieve the data from the devices.
A review of digital forensic challenges in the Internet of Things (IoT)	The paper contributes towards the IoT by including MDs as well as the cloud in the scope of IoT devices	The paper contributes towards MDs by seeing MDs as part of the IoT.		The paper contributes towards IoT DF by noting that several digital forensic models exist, but that they are unable to extract evidence reliably and timely from IoT devices. The paper thus notes the need for models that can ensure the detection and analysis of attacks in the IoT as soon as the attacks take place.			

				<p>The paper furthermore notes the relevance between the cloud and IoT devices and also views the cloud as a valid source of evidence for IoT devices.</p> <p>The paper also evaluates several of the other papers evaluated in this paper but notes that they are not yet sufficient.</p>		
Adding Digital Forensic Readiness as a Security Component to the IoT Domain						<p>The architecture presented in the paper is based on ISO 27043 which is an accepted standard, thus the paper is based on accepted standards.</p> <p>The paper makes a significant contribution towards DFR by introducing architecture for IoT DFR. The architecture also complies with ISO 27043. The architecture helps maximize the potential use of evidence.</p>

							<p>The architecture makes use of IoT sensor monitoring where sensor data can be transmitted to a centre for incident analysis. Packets sent to and from IoT devices can also be monitored. Log analysis also forms part of the architecture</p>
<p>Functional Requirements for Adding Digital Forensic Readiness as a Security Component in IoT Environments</p>						<p>The paper introduces several processes that are based on ISO 27043, which is an accepted international standard.</p>	<p>The paper extends on the previous paper by the same authors (discussed in the previous row in the table), by adding functional requirements. The paper is a</p>

							huge contribution towards DFR in the IoT, due to providing the first paper that addresses DFR, rather than just introducing reactive forensic processes.
--	--	--	--	--	--	--	--

APPENDIX B – Performance testing tables

Performance metrics

Several performance metrics were collected during the various steps of the simulation of DFR for IoT. The metrics were collected by Zabbix before they were exported to Excel documents. The Excel documents were then used to generate graphs for the performance metrics. The graphs can be found in the next appendix of this dissertation. Due to space constraints, only one of the metrics is listed in tables in this chapter. The full analysis of performance metrics exists in the form of graphs in the next appendix of this dissertation. The tables present in this appendix is the amount of context switches per second for the VMs and the Raspberry Pi's during Step 1 of the simulation For DFR in IoT.

linuxlite1: Context switches per second	7/22/201 9 21:00	52.2	Raspberry pi 1: Context switches per second	7/22/201 9 21:00	189. 2
linuxlitevm2: Context switches per second	7/22/201 9 21:00	38	Raspberry pi 2: Context switches per second	7/22/201 9 21:00	193
linuxlitevm3: Context switches per second	7/22/201 9 21:00	52.4	Raspberry pi 1: Context switches per second	7/22/201 9 21:05	174. 6
Linuxlite1: Context switches per second	7/22/201 9 21:05	51.6	Raspberry pi 2: Context switches per second	7/22/201 9 21:05	178. 6
linuxlitevm2: Context switches per second	7/22/201 9 21:05	36.4	Raspberry pi 1: Context switches per second	7/22/201 9 21:10	175
linuxlitevm3: Context switches per second	7/22/201 9 21:05	50.8	Raspberry pi 2: Context switches per second	7/22/201 9 21:10	178. 8
Linuxlite1: Context switches per second	7/22/201 9 21:10	51	Raspberry pi 1: Context switches per second	7/22/201 9 21:15	173. 2
linuxlitevm2: Context switches per second	7/22/201 9 21:10	36.6	Raspberry pi 2: Context switches per second	7/22/201 9 21:15	178. 4
linuxlitevm3: Context switches per second	7/22/201 9 21:10	51.6	Raspberry pi 1: Context switches per second	7/22/201 9 21:20	171. 4
Linuxlite1: Context switches per second	7/22/201 9 21:15	50.8	Raspberry pi 2: Context switches per second	7/22/201 9 21:20	175. 8
linuxlitevm2: Context switches per second	7/22/201 9 21:15	36	Raspberry pi 1: Context switches per second	7/22/201 9 21:25	170
linuxlitevm3: Context switches per second	7/22/201 9 21:15	50.6	Raspberry pi 2: Context switches per second	7/22/201 9 21:25	174. 8
Linuxlite1: Context switches per second	7/22/201 9 21:20	51.8	Raspberry pi 1: Context switches per second	7/22/201 9 21:30	171. 2
linuxlitevm2: Context switches per second	7/22/201 9 21:20	35.8	Raspberry pi 2: Context switches per second	7/22/201 9 21:30	173. 2
linuxlitevm3: Context switches per second	7/22/201 9 21:20	50.6	Raspberry pi 1: Context switches per second	7/22/201 9 21:35	174. 8
Linuxlite1: Context switches per second	7/22/201 9 21:25	50.6	Raspberry pi 2: Context switches per second	7/22/201 9 21:35	172. 8
linuxlitevm2: Context switches per second	7/22/201 9 21:25	36	Raspberry pi 1: Context switches per second	7/22/201 9 21:40	173. 8

linuxlitevm3: Context switches per second	7/22/201	9 21:25	50	Raspberry pi 2: Context switches per second	7/22/201	9 21:40	177.2
Linuxlite1: Context switches per second	7/22/201	9 21:30	51	Raspberry pi 1: Context switches per second	7/22/201	9 21:45	170.6
linuxlitevm2: Context switches per second	7/22/201	9 21:30	35.8	Raspberry pi 2: Context switches per second	7/22/201	9 21:45	174.4
linuxlitevm3: Context switches per second	7/22/201	9 21:30	50.8	Raspberry pi 1: Context switches per second	7/22/201	9 21:50	169.2
Linuxlite1: Context switches per second	7/22/201	9 21:35	50.4	Raspberry pi 2: Context switches per second	7/22/201	9 21:50	180.4
linuxlitevm2: Context switches per second	7/22/201	9 21:35	36	Raspberry pi 1: Context switches per second	7/22/201	9 21:55	168.2
linuxlitevm3: Context switches per second	7/22/201	9 21:35	50	Raspberry pi 2: Context switches per second	7/22/201	9 21:55	180.8
Linuxlite1: Context switches per second	7/22/201	9 21:40	51	Raspberry pi 1: Context switches per second	7/22/201	9 22:00	171.8
linuxlitevm2: Context switches per second	7/22/201	9 21:40	36.4	Raspberry pi 2: Context switches per second	7/22/201	9 22:00	180.8
linuxlitevm3: Context switches per second	7/22/201	9 21:40	50.6	Raspberry pi 1: Context switches per second	7/22/201	9 22:05	167.4
Linuxlite1: Context switches per second	7/22/201	9 21:45	51	Raspberry pi 2: Context switches per second	7/22/201	9 22:05	171.4
linuxlitevm2: Context switches per second	7/22/201	9 21:45	36.6	Raspberry pi 1: Context switches per second	7/22/201	9 22:10	169.8
linuxlitevm3: Context switches per second	7/22/201	9 21:45	51.2	Raspberry pi 2: Context switches per second	7/22/201	9 22:10	180
Linuxlite1: Context switches per second	7/22/201	9 21:50	50.4	Raspberry pi 1: Context switches per second	7/22/201	9 22:15	175.2
linuxlitevm2: Context switches per second	7/22/201	9 21:50	36.4	Raspberry pi 2: Context switches per second	7/22/201	9 22:15	181.4
linuxlitevm3: Context switches per second	7/22/201	9 21:50	50.8	Raspberry pi 1: Context switches per second	7/22/201	9 22:20	177.4
Linuxlite1: Context switches per second	7/22/201	9 21:55	51.6	Raspberry pi 2: Context switches per second	7/22/201	9 22:20	178.6
linuxlitevm2: Context switches per second	7/22/201	9 21:55	36.8	Raspberry pi 1: Context switches per second	7/22/201	9 22:25	172.8
linuxlitevm3: Context switches per second	7/22/201	9 21:55	50.4	Raspberry pi 2: Context switches per second	7/22/201	9 22:25	174.8
Linuxlite1: Context switches per second	7/22/201	9 22:00	51.8	Raspberry pi 1: Context switches per second	7/22/201	9 22:30	173
linuxlitevm2: Context switches per second	7/22/201	9 22:00	36.8	Raspberry pi 2: Context switches per second	7/22/201	9 22:30	176.4
linuxlitevm3: Context switches per second	7/22/201	9 22:00	51.4	Raspberry pi 1: Context switches per second	7/22/201	9 22:35	172.2
Linuxlite1: Context switches per second	7/22/201	9 22:05	50.4	Raspberry pi 2: Context switches per second	7/22/201	9 22:35	172.8
linuxlitevm2: Context switches per second	7/22/201	9 22:05	35.8	Raspberry pi 1: Context switches per second	7/22/201	9 22:40	172.6
linuxlitevm3: Context switches per second	7/22/201	9 22:05	49.8	Raspberry pi 2: Context switches per second	7/22/201	9 22:40	168.2
Linuxlite1: Context switches per second	7/22/201		50.6	Raspberry pi 1: Context switches per second	7/22/201		182.

switches per second	9 22:10		switches per second	9 22:45	6
linuxlitevm2: Context	7/22/201		Raspberry pi 2: Context	7/22/201	178.
switches per second	9 22:10	36.2	switches per second	9 22:45	6
linuxlitevm3: Context	7/22/201		Raspberry pi 1: Context	7/22/201	174.
switches per second	9 22:10	50.6	switches per second	9 22:50	2
Linuxlite1: Context	7/22/201		Raspberry pi 2: Context	7/22/201	170.
switches per second	9 22:15	50.2	switches per second	9 22:50	8
linuxlitevm2: Context	7/22/201		Raspberry pi 1: Context	7/22/201	172.
switches per second	9 22:15	35.8	switches per second	9 22:55	2
linuxlitevm3: Context	7/22/201		Raspberry pi 2: Context	7/22/201	
switches per second	9 22:15	50.8	switches per second	9 22:55	169
Linuxlite1: Context	7/22/201		Raspberry pi 1: Context	7/22/201	180.
switches per second	9 22:20	50.4	switches per second	9 23:00	8
linuxlitevm2: Context	7/22/201		Raspberry pi 2: Context	7/22/201	178.
switches per second	9 22:20	35.8	switches per second	9 23:00	2
linuxlitevm3: Context	7/22/201		Raspberry pi 1: Context	7/22/201	174.
switches per second	9 22:20	50.2	switches per second	9 23:05	2
Linuxlite1: Context	7/22/201		Raspberry pi 2: Context	7/22/201	168.
switches per second	9 22:25	50.4	switches per second	9 23:05	6
linuxlitevm2: Context	7/22/201		Raspberry pi 1: Context	7/22/201	
switches per second	9 22:25	35	switches per second	9 23:10	177
linuxlitevm3: Context	7/22/201		Raspberry pi 2: Context	7/22/201	171.
switches per second	9 22:25	49.8	switches per second	9 23:10	4
Linuxlite1: Context	7/22/201		Raspberry pi 1: Context	7/22/201	179.
switches per second	9 22:30	50.2	switches per second	9 23:15	2
linuxlitevm2: Context	7/22/201		Raspberry pi 2: Context	7/22/201	
switches per second	9 22:30	35.6	switches per second	9 23:15	174
linuxlitevm3: Context	7/22/201		Raspberry pi 1: Context	7/22/201	
switches per second	9 22:30	50	switches per second	9 23:20	174
Linuxlite1: Context	7/22/201		Raspberry pi 2: Context	7/22/201	172.
switches per second	9 22:35	50	switches per second	9 23:20	4
linuxlitevm2: Context	7/22/201		Raspberry pi 1: Context	7/22/201	175.
switches per second	9 22:35	36.2	switches per second	9 23:25	6
linuxlitevm3: Context	7/22/201		Raspberry pi 2: Context	7/22/201	
switches per second	9 22:35	50	switches per second	9 23:25	171
Linuxlite1: Context	7/22/201		Raspberry pi 1: Context	7/22/201	176.
switches per second	9 22:40	50	switches per second	9 23:30	4
linuxlitevm2: Context	7/22/201		Raspberry pi 2: Context	7/22/201	179.
switches per second	9 22:40	35.8	switches per second	9 23:30	2
linuxlitevm3: Context	7/22/201		Raspberry pi 1: Context	7/22/201	169.
switches per second	9 22:40	49.8	switches per second	9 23:35	8
Linuxlite1: Context	7/22/201		Raspberry pi 2: Context	7/22/201	174.
switches per second	9 22:45	51.4	switches per second	9 23:35	8
linuxlitevm2: Context	7/22/201		Raspberry pi 1: Context	7/22/201	
switches per second	9 22:45	36.8	switches per second	9 23:40	171
linuxlitevm3: Context	7/22/201		Raspberry pi 2: Context	7/22/201	176.
switches per second	9 22:45	50.4	switches per second	9 23:40	6
Linuxlite1: Context	7/22/201		Raspberry pi 1: Context	7/22/201	
switches per second	9 22:50	50.2	switches per second	9 23:45	170
linuxlitevm2: Context	7/22/201		Raspberry pi 2: Context	7/22/201	175.
switches per second	9 22:50	36.2	switches per second	9 23:45	8

linuxlitevm3: Context switches per second	7/22/201	9 22:50	49.8	Raspberry pi 1: Context switches per second	7/22/201	9 23:50	175.6
Linuxlite1: Context switches per second	7/22/201	9 22:55	51	Raspberry pi 2: Context switches per second	7/22/201	9 23:50	174.2
linuxlitevm2: Context switches per second	7/22/201	9 22:55	35.8	Raspberry pi 1: Context switches per second	7/22/201	9 23:55	174.8
linuxlitevm3: Context switches per second	7/22/201	9 22:55	50.2	Raspberry pi 2: Context switches per second	7/22/201	9 23:55	173
Linuxlite1: Context switches per second	7/22/201	9 23:00	50.8	Raspberry pi 1: Context switches per second	7/23/201	9 0:00	176.4
linuxlitevm2: Context switches per second	7/22/201	9 23:00	37.2	Raspberry pi 2: Context switches per second	7/23/201	9 0:00	179.8
linuxlitevm3: Context switches per second	7/22/201	9 23:00	51.4	Raspberry pi 1: Context switches per second	7/23/201	9 0:05	168.8
Linuxlite1: Context switches per second	7/22/201	9 23:05	50.6	Raspberry pi 2: Context switches per second	7/23/201	9 0:05	168.8
linuxlitevm2: Context switches per second	7/22/201	9 23:05	35.6	Raspberry pi 1: Context switches per second	7/23/201	9 0:10	174.8
linuxlitevm3: Context switches per second	7/22/201	9 23:05	50	Raspberry pi 2: Context switches per second	7/23/201	9 0:10	167.8
Linuxlite1: Context switches per second	7/22/201	9 23:10	50.2	Raspberry pi 1: Context switches per second	7/23/201	9 0:15	171.2
linuxlitevm2: Context switches per second	7/22/201	9 23:10	35.8	Raspberry pi 2: Context switches per second	7/23/201	9 0:15	174
linuxlitevm3: Context switches per second	7/22/201	9 23:10	51.2	Raspberry pi 1: Context switches per second	7/23/201	9 0:20	171.4
Linuxlite1: Context switches per second	7/22/201	9 23:15	50.4	Raspberry pi 2: Context switches per second	7/23/201	9 0:20	168.4
linuxlitevm2: Context switches per second	7/22/201	9 23:15	36.2	Raspberry pi 1: Context switches per second	7/23/201	9 0:25	173
linuxlitevm3: Context switches per second	7/22/201	9 23:15	50.2	Raspberry pi 2: Context switches per second	7/23/201	9 0:25	168.6
Linuxlite1: Context switches per second	7/22/201	9 23:20	49.8	Raspberry pi 1: Context switches per second	7/23/201	9 0:30	170.2
linuxlitevm2: Context switches per second	7/22/201	9 23:20	35.6	Raspberry pi 2: Context switches per second	7/23/201	9 0:30	170
linuxlitevm3: Context switches per second	7/22/201	9 23:20	50.4	Raspberry pi 1: Context switches per second	7/23/201	9 0:35	168
Linuxlite1: Context switches per second	7/22/201	9 23:25	50.8	Raspberry pi 2: Context switches per second	7/23/201	9 0:35	170
linuxlitevm2: Context switches per second	7/22/201	9 23:25	36	Raspberry pi 1: Context switches per second	7/23/201	9 0:40	170.6
linuxlitevm3: Context switches per second	7/22/201	9 23:25	50.6	Raspberry pi 2: Context switches per second	7/23/201	9 0:40	169.6
Linuxlite1: Context switches per second	7/22/201	9 23:30	50.6	Raspberry pi 1: Context switches per second	7/23/201	9 0:45	170.6
linuxlitevm2: Context switches per second	7/22/201	9 23:30	36	Raspberry pi 2: Context switches per second	7/23/201	9 0:45	171.4
linuxlitevm3: Context switches per second	7/22/201	9 23:30	51	Raspberry pi 1: Context switches per second	7/23/201	9 0:50	170.8
Linuxlite1: Context switches per second	7/22/201		50.6	Raspberry pi 2: Context switches per second	7/23/201		166.

switches per second	9 23:35		switches per second	9 0:50	4
linuxlitevm2: Context	7/22/201		Raspberry pi 1: Context	7/23/201	
switches per second	9 23:35	35.6	switches per second	9 0:55	169
linuxlitevm3: Context	7/22/201		Raspberry pi 2: Context	7/23/201	166.
switches per second	9 23:35	50.2	switches per second	9 0:55	2
Linuxlite1: Context	7/22/201		Raspberry pi 1: Context	7/23/201	175.
switches per second	9 23:40	50.4	switches per second	9 1:00	8
linuxlitevm2: Context	7/22/201		Raspberry pi 2: Context	7/23/201	175.
switches per second	9 23:40	35.4	switches per second	9 1:00	6
linuxlitevm3: Context	7/22/201		Raspberry pi 1: Context	7/23/201	170.
switches per second	9 23:40	50	switches per second	9 1:05	4
Linuxlite1: Context	7/22/201		Raspberry pi 2: Context	7/23/201	173.
switches per second	9 23:45	49.8	switches per second	9 1:05	4
linuxlitevm2: Context	7/22/201		Raspberry pi 1: Context	7/23/201	166.
switches per second	9 23:45	36	switches per second	9 1:10	4
linuxlitevm3: Context	7/22/201		Raspberry pi 2: Context	7/23/201	167.
switches per second	9 23:45	50.2	switches per second	9 1:10	4
Linuxlite1: Context	7/22/201		Raspberry pi 1: Context	7/23/201	182.
switches per second	9 23:50	50.2	switches per second	9 1:15	8
linuxlitevm2: Context	7/22/201		Raspberry pi 2: Context	7/23/201	183.
switches per second	9 23:50	36	switches per second	9 1:15	6
linuxlitevm3: Context	7/22/201		Raspberry pi 1: Context	7/23/201	170.
switches per second	9 23:50	50	switches per second	9 1:20	4
Linuxlite1: Context	7/22/201		Raspberry pi 2: Context	7/23/201	171.
switches per second	9 23:55	51.4	switches per second	9 1:20	2
linuxlitevm2: Context	7/22/201		Raspberry pi 1: Context	7/23/201	173.
switches per second	9 23:55	36.2	switches per second	9 1:25	8
linuxlitevm3: Context	7/22/201	309.	Raspberry pi 2: Context	7/23/201	169.
switches per second	9 23:55	2	switches per second	9 1:25	2
Linuxlite1: Context	7/23/201		Raspberry pi 1: Context	7/23/201	174.
switches per second	9 0:00	50.6	switches per second	9 1:30	2
linuxlitevm2: Context	7/23/201		Raspberry pi 2: Context	7/23/201	171.
switches per second	9 0:00	36.6	switches per second	9 1:30	8
linuxlitevm3: Context	7/23/201		Raspberry pi 1: Context	7/23/201	
switches per second	9 0:00	52.8	switches per second	9 1:35	171
Linuxlite1: Context	7/23/201		Raspberry pi 2: Context	7/23/201	165.
switches per second	9 0:05	49.6	switches per second	9 1:35	6
linuxlitevm2: Context	7/23/201		Raspberry pi 1: Context	7/23/201	
switches per second	9 0:05	35.6	switches per second	9 1:40	170
linuxlitevm3: Context	7/23/201		Raspberry pi 2: Context	7/23/201	166.
switches per second	9 0:05	50.4	switches per second	9 1:40	6
Linuxlite1: Context	7/23/201		Raspberry pi 1: Context	7/23/201	168.
switches per second	9 0:10	50.2	switches per second	9 1:45	4
linuxlitevm2: Context	7/23/201		Raspberry pi 2: Context	7/23/201	
switches per second	9 0:10	35.8	switches per second	9 1:45	168
linuxlitevm3: Context	7/23/201		Raspberry pi 1: Context	7/23/201	172.
switches per second	9 0:10	49.8	switches per second	9 1:50	8
Linuxlite1: Context	7/23/201		Raspberry pi 2: Context	7/23/201	167.
switches per second	9 0:15	51.4	switches per second	9 1:50	6
linuxlitevm2: Context	7/23/201		Raspberry pi 1: Context	7/23/201	167.
switches per second	9 0:15	36	switches per second	9 1:55	2

linuxlitevm3: Context switches per second	7/23/201	119.9	0:15	4	Raspberry pi 2: Context switches per second	7/23/201	166.8	1:55	8
Linuxlite1: Context switches per second	7/23/201	51	0:20	51	Raspberry pi 1: Context switches per second	7/23/201	174.2	2:00	2
linuxlitevm2: Context switches per second	7/23/201	36.2	0:20	36.2	Raspberry pi 2: Context switches per second	7/23/201	179.8	2:00	8
linuxlitevm3: Context switches per second	7/23/201	67.4	0:20	67.4	Raspberry pi 1: Context switches per second	7/23/201	169.6	2:05	6
Linuxlite1: Context switches per second	7/23/201	50.6	0:25	50.6	Raspberry pi 2: Context switches per second	7/23/201	171.4	2:05	4
linuxlitevm2: Context switches per second	7/23/201	35.2	0:25	35.2	Raspberry pi 1: Context switches per second	7/23/201	170.4	2:10	4
linuxlitevm3: Context switches per second	7/23/201	49.8	0:25	49.8	Raspberry pi 2: Context switches per second	7/23/201	170	2:10	170
Linuxlite1: Context switches per second	7/23/201	50.8	0:30	50.8	Raspberry pi 1: Context switches per second	7/23/201	172.4	2:15	4
linuxlitevm2: Context switches per second	7/23/201	35.4	0:30	35.4	Raspberry pi 2: Context switches per second	7/23/201	171.2	2:15	2
linuxlitevm3: Context switches per second	7/23/201	51	0:30	51	Raspberry pi 1: Context switches per second	7/23/201	168.2	2:20	2
Linuxlite1: Context switches per second	7/23/201	49.8	0:35	49.8	Raspberry pi 2: Context switches per second	7/23/201	167.6	2:20	6
linuxlitevm2: Context switches per second	7/23/201	35.2	0:35	35.2	Raspberry pi 1: Context switches per second	7/23/201	171	2:25	171
linuxlitevm3: Context switches per second	7/23/201	49.8	0:35	49.8	Raspberry pi 2: Context switches per second	7/23/201	170.6	2:25	6
Linuxlite1: Context switches per second	7/23/201	50.2	0:40	50.2	Raspberry pi 1: Context switches per second	7/23/201	170	2:30	170
linuxlitevm2: Context switches per second	7/23/201	34.6	0:40	34.6	Raspberry pi 2: Context switches per second	7/23/201	172.6	2:30	6
linuxlitevm3: Context switches per second	7/23/201	50.4	0:40	50.4	Raspberry pi 1: Context switches per second	7/23/201	169.2	2:35	2
Linuxlite1: Context switches per second	7/23/201	51	0:45	51	Raspberry pi 2: Context switches per second	7/23/201	170.2	2:35	2
linuxlitevm2: Context switches per second	7/23/201	35	0:45	35	Raspberry pi 1: Context switches per second	7/23/201	171.8	2:40	8
linuxlitevm3: Context switches per second	7/23/201	50.2	0:45	50.2	Raspberry pi 2: Context switches per second	7/23/201	167.4	2:40	4
Linuxlite1: Context switches per second	7/23/201	50.2	0:50	50.2	Raspberry pi 1: Context switches per second	7/23/201	171.6	2:45	6
linuxlitevm2: Context switches per second	7/23/201	35	0:50	35	Raspberry pi 2: Context switches per second	7/23/201	175.4	2:45	4
linuxlitevm3: Context switches per second	7/23/201	50	0:50	50	Raspberry pi 1: Context switches per second	7/23/201	167.6	2:50	6
Linuxlite1: Context switches per second	7/23/201	50.4	0:55	50.4	Raspberry pi 2: Context switches per second	7/23/201	173	2:50	173
linuxlitevm2: Context switches per second	7/23/201	35.6	0:55	35.6	Raspberry pi 1: Context switches per second	7/23/201	170	2:55	170
linuxlitevm3: Context switches per second	7/23/201	49.8	0:55	49.8	Raspberry pi 2: Context switches per second	7/23/201	168.4	2:55	4
Linuxlite1: Context switches per second	7/23/201	51.4		51.4	Raspberry pi 1: Context switches per second	7/23/201	176.		176.

switches per second	9 1:00		switches per second	9 3:00	2
linuxlitevm2: Context	7/23/201		Raspberry pi 2: Context	7/23/201	
switches per second	9 1:00	36.6	switches per second	9 3:00	167
linuxlitevm3: Context	7/23/201		Raspberry pi 1: Context	7/23/201	170.
switches per second	9 1:00	50.8	switches per second	9 3:05	4
Linuxlite1: Context	7/23/201		Raspberry pi 2: Context	7/23/201	164.
switches per second	9 1:05	50.4	switches per second	9 3:05	2
linuxlitevm2: Context	7/23/201		Raspberry pi 1: Context	7/23/201	169.
switches per second	9 1:05	35.8	switches per second	9 3:10	4
linuxlitevm3: Context	7/23/201		Raspberry pi 2: Context	7/23/201	
switches per second	9 1:05	49.2	switches per second	9 3:10	165
Linuxlite1: Context	7/23/201		Raspberry pi 1: Context	7/23/201	174.
switches per second	9 1:10	50.4	switches per second	9 3:15	8
linuxlitevm2: Context	7/23/201		Raspberry pi 2: Context	7/23/201	169.
switches per second	9 1:10	36	switches per second	9 3:15	6
linuxlitevm3: Context	7/23/201		Raspberry pi 1: Context	7/23/201	173.
switches per second	9 1:10	49.8	switches per second	9 3:20	6
Linuxlite1: Context	7/23/201		Raspberry pi 2: Context	7/23/201	164.
switches per second	9 1:15	52.2	switches per second	9 3:20	8
linuxlitevm2: Context	7/23/201		Raspberry pi 1: Context	7/23/201	173.
switches per second	9 1:15	38	switches per second	9 3:25	8
linuxlitevm3: Context	7/23/201		Raspberry pi 2: Context	7/23/201	167.
switches per second	9 1:15	52	switches per second	9 3:25	2
Linuxlite1: Context	7/23/201		Raspberry pi 1: Context	7/23/201	172.
switches per second	9 1:20	51.4	switches per second	9 3:30	2
linuxlitevm2: Context	7/23/201		Raspberry pi 2: Context	7/23/201	
switches per second	9 1:20	35.8	switches per second	9 3:30	165
linuxlitevm3: Context	7/23/201		Raspberry pi 1: Context	7/23/201	
switches per second	9 1:20	50.8	switches per second	9 3:35	168
Linuxlite1: Context	7/23/201		Raspberry pi 2: Context	7/23/201	
switches per second	9 1:25	49.8	switches per second	9 3:35	163
linuxlitevm2: Context	7/23/201		Raspberry pi 1: Context	7/23/201	171.
switches per second	9 1:25	35.8	switches per second	9 3:40	6
linuxlitevm3: Context	7/23/201		Raspberry pi 2: Context	7/23/201	161.
switches per second	9 1:25	49.8	switches per second	9 3:40	2
Linuxlite1: Context	7/23/201		Raspberry pi 1: Context	7/23/201	173.
switches per second	9 1:30	50	switches per second	9 3:45	8
linuxlitevm2: Context	7/23/201		Raspberry pi 2: Context	7/23/201	165.
switches per second	9 1:30	36	switches per second	9 3:45	6
linuxlitevm3: Context	7/23/201		Raspberry pi 1: Context	7/23/201	171.
switches per second	9 1:30	50.6	switches per second	9 3:50	6
Linuxlite1: Context	7/23/201		Raspberry pi 2: Context	7/23/201	168.
switches per second	9 1:35	49.8	switches per second	9 3:50	6
linuxlitevm2: Context	7/23/201		Raspberry pi 1: Context	7/23/201	170.
switches per second	9 1:35	35.8	switches per second	9 3:55	4
linuxlitevm3: Context	7/23/201		Raspberry pi 2: Context	7/23/201	169.
switches per second	9 1:35	50.4	switches per second	9 3:55	4
Linuxlite1: Context	7/23/201		Raspberry pi 1: Context	7/23/201	174.
switches per second	9 1:40	49.4	switches per second	9 4:00	8
linuxlitevm2: Context	7/23/201		Raspberry pi 2: Context	7/23/201	180.
switches per second	9 1:40	35.4	switches per second	9 4:00	4

linuxlitevm3: Context switches per second	7/23/201	9 1:40	50	Raspberry pi 1: Context switches per second	7/23/201	9 4:05	171.8
Linuxlite1: Context switches per second	7/23/201	9 1:45	50.8	Raspberry pi 2: Context switches per second	7/23/201	9 4:05	172.4
linuxlitevm2: Context switches per second	7/23/201	9 1:45	35.8	Raspberry pi 1: Context switches per second	7/23/201	9 4:10	165.6
linuxlitevm3: Context switches per second	7/23/201	9 1:45	51	Raspberry pi 2: Context switches per second	7/23/201	9 4:10	168.4
Linuxlite1: Context switches per second	7/23/201	9 1:50	49.6	Raspberry pi 1: Context switches per second	7/23/201	9 4:15	168.8
linuxlitevm2: Context switches per second	7/23/201	9 1:50	35.6	Raspberry pi 2: Context switches per second	7/23/201	9 4:15	174.8
linuxlitevm3: Context switches per second	7/23/201	9 1:50	50	Raspberry pi 1: Context switches per second	7/23/201	9 4:20	167.2
Linuxlite1: Context switches per second	7/23/201	9 1:55	50.6	Raspberry pi 2: Context switches per second	7/23/201	9 4:20	171.8
linuxlitevm2: Context switches per second	7/23/201	9 1:55	35.8	Raspberry pi 1: Context switches per second	7/23/201	9 4:25	171.4
linuxlitevm3: Context switches per second	7/23/201	9 1:55	50	Raspberry pi 2: Context switches per second	7/23/201	9 4:25	173
Linuxlite1: Context switches per second	7/23/201	9 2:00	51.2	Raspberry pi 1: Context switches per second	7/23/201	9 4:30	170.8
linuxlitevm2: Context switches per second	7/23/201	9 2:00	37	Raspberry pi 2: Context switches per second	7/23/201	9 4:30	169.2
linuxlitevm3: Context switches per second	7/23/201	9 2:00	51.2	Raspberry pi 1: Context switches per second	7/23/201	9 4:35	171
Linuxlite1: Context switches per second	7/23/201	9 2:05	50.6	Raspberry pi 2: Context switches per second	7/23/201	9 4:35	169.4
linuxlitevm2: Context switches per second	7/23/201	9 2:05	36	Raspberry pi 1: Context switches per second	7/23/201	9 4:40	172.2
linuxlitevm3: Context switches per second	7/23/201	9 2:05	50.6	Raspberry pi 2: Context switches per second	7/23/201	9 4:40	173
Linuxlite1: Context switches per second	7/23/201	9 2:10	50.2	Raspberry pi 1: Context switches per second	7/23/201	9 4:45	170.2
linuxlitevm2: Context switches per second	7/23/201	9 2:10	35.8	Raspberry pi 2: Context switches per second	7/23/201	9 4:45	170.8
linuxlitevm3: Context switches per second	7/23/201	9 2:10	50.2	Raspberry pi 1: Context switches per second	7/23/201	9 4:50	171.6
Linuxlite1: Context switches per second	7/23/201	9 2:15	50.8	Raspberry pi 2: Context switches per second	7/23/201	9 4:50	172.4
linuxlitevm2: Context switches per second	7/23/201	9 2:15	36.2	Raspberry pi 1: Context switches per second	7/23/201	9 4:55	169.8
linuxlitevm3: Context switches per second	7/23/201	9 2:15	51	Raspberry pi 2: Context switches per second	7/23/201	9 4:55	170.4
Linuxlite1: Context switches per second	7/23/201	9 2:20	49.6	Raspberry pi 1: Context switches per second	7/23/201	9 5:00	174.6
linuxlitevm2: Context switches per second	7/23/201	9 2:20	35	Raspberry pi 2: Context switches per second	7/23/201	9 5:00	174.2
linuxlitevm3: Context switches per second	7/23/201	9 2:20	50				
Linuxlite1: Context switches per second	7/23/201		49.6				

switches per second	9 2:25	
linuxlitevm2: Context	7/23/201	
switches per second	9 2:25	35.6
linuxlitevm3: Context	7/23/201	
switches per second	9 2:25	50.2
Linuxlite1: Context	7/23/201	
switches per second	9 2:30	50.6
linuxlitevm2: Context	7/23/201	
switches per second	9 2:30	35.6
linuxlitevm3: Context	7/23/201	
switches per second	9 2:30	50.6
Linuxlite1: Context	7/23/201	
switches per second	9 2:35	49.8
linuxlitevm2: Context	7/23/201	
switches per second	9 2:35	36
linuxlitevm3: Context	7/23/201	
switches per second	9 2:35	50
Linuxlite1: Context	7/23/201	
switches per second	9 2:40	50.6
linuxlitevm2: Context	7/23/201	
switches per second	9 2:40	35.8
linuxlitevm3: Context	7/23/201	
switches per second	9 2:40	49.8
Linuxlite1: Context	7/23/201	
switches per second	9 2:45	50.8
linuxlitevm2: Context	7/23/201	
switches per second	9 2:45	37
linuxlitevm3: Context	7/23/201	
switches per second	9 2:45	51.2
Linuxlite1: Context	7/23/201	
switches per second	9 2:50	49.8
linuxlitevm2: Context	7/23/201	
switches per second	9 2:50	35.6
linuxlitevm3: Context	7/23/201	
switches per second	9 2:50	49.8
Linuxlite1: Context	7/23/201	
switches per second	9 2:55	51
linuxlitevm2: Context	7/23/201	
switches per second	9 2:55	36.2
linuxlitevm3: Context	7/23/201	
switches per second	9 2:55	50.4
Linuxlite1: Context	7/23/201	
switches per second	9 3:00	51
linuxlitevm2: Context	7/23/201	
switches per second	9 3:00	36.6
linuxlitevm3: Context	7/23/201	
switches per second	9 3:00	51.2
Linuxlite1: Context	7/23/201	
switches per second	9 3:05	150.
linuxlitevm2: Context	7/23/201	
switches per second	9 3:05	6
linuxlitevm2: Context	7/23/201	
switches per second	9 3:05	207.
switches per second	9 3:05	6

linuxlitevm3: Context switches per second	7/23/201 9 3:05	51
Linuxlite1: Context switches per second	7/23/201 9 3:10	216.2
linuxlitevm2: Context switches per second	7/23/201 9 3:10	141.4
linuxlitevm3: Context switches per second	7/23/201 9 3:10	51
Linuxlite1: Context switches per second	7/23/201 9 3:15	50.8
linuxlitevm2: Context switches per second	7/23/201 9 3:15	35.8
linuxlitevm3: Context switches per second	7/23/201 9 3:15	50.4
Linuxlite1: Context switches per second	7/23/201 9 3:20	50.2
linuxlitevm2: Context switches per second	7/23/201 9 3:20	35.6
linuxlitevm3: Context switches per second	7/23/201 9 3:20	50
Linuxlite1: Context switches per second	7/23/201 9 3:25	49.8
linuxlitevm2: Context switches per second	7/23/201 9 3:25	35.4
linuxlitevm3: Context switches per second	7/23/201 9 3:25	50.4
Linuxlite1: Context switches per second	7/23/201 9 3:30	50.6
linuxlitevm2: Context switches per second	7/23/201 9 3:30	35.8
linuxlitevm3: Context switches per second	7/23/201 9 3:30	49.8
Linuxlite1: Context switches per second	7/23/201 9 3:35	51.6
linuxlitevm2: Context switches per second	7/23/201 9 3:35	35.6
linuxlitevm3: Context switches per second	7/23/201 9 3:35	50.6
Linuxlite1: Context switches per second	7/23/201 9 3:40	50
linuxlitevm2: Context switches per second	7/23/201 9 3:40	35.4
linuxlitevm3: Context switches per second	7/23/201 9 3:40	50.2
Linuxlite1: Context switches per second	7/23/201 9 3:45	50.2
linuxlitevm2: Context switches per second	7/23/201 9 3:45	36
linuxlitevm3: Context switches per second	7/23/201 9 3:45	50.2
Linuxlite1: Context switches per second	7/23/201 9 3:45	50

switches per second	9 3:50	
linuxlitevm2: Context	7/23/201	
switches per second	9 3:50	35.6
linuxlitevm3: Context	7/23/201	
switches per second	9 3:50	50.4
Linuxlite1: Context	7/23/201	
switches per second	9 3:55	50.4
linuxlitevm2: Context	7/23/201	
switches per second	9 3:55	36
linuxlitevm3: Context	7/23/201	
switches per second	9 3:55	50.6
Linuxlite1: Context	7/23/201	
switches per second	9 4:00	51.6
linuxlitevm2: Context	7/23/201	
switches per second	9 4:00	38.2
linuxlitevm3: Context	7/23/201	
switches per second	9 4:00	52.6
Linuxlite1: Context	7/23/201	
switches per second	9 4:05	50.2
linuxlitevm2: Context	7/23/201	
switches per second	9 4:05	36
linuxlitevm3: Context	7/23/201	
switches per second	9 4:05	50.4
Linuxlite1: Context	7/23/201	
switches per second	9 4:10	50
linuxlitevm2: Context	7/23/201	
switches per second	9 4:10	35.6
linuxlitevm3: Context	7/23/201	
switches per second	9 4:10	50
Linuxlite1: Context	7/23/201	
switches per second	9 4:15	50.2
linuxlitevm2: Context	7/23/201	
switches per second	9 4:15	36.2
linuxlitevm3: Context	7/23/201	
switches per second	9 4:15	50.8
Linuxlite1: Context	7/23/201	
switches per second	9 4:20	50
linuxlitevm2: Context	7/23/201	
switches per second	9 4:20	36
linuxlitevm3: Context	7/23/201	
switches per second	9 4:20	50.2
Linuxlite1: Context	7/23/201	
switches per second	9 4:25	50.4
linuxlitevm2: Context	7/23/201	
switches per second	9 4:25	36
linuxlitevm3: Context	7/23/201	
switches per second	9 4:25	50.2
Linuxlite1: Context	7/23/201	
switches per second	9 4:30	50.4
linuxlitevm2: Context	7/23/201	
switches per second	9 4:30	35.4

linuxlitevm3: Context switches per second	7/23/201 9 4:30	50.2
Linuxlite1: Context switches per second	7/23/201 9 4:35	50.2
linuxlitevm2: Context switches per second	7/23/201 9 4:35	35.8
linuxlitevm3: Context switches per second	7/23/201 9 4:35	50.2
Linuxlite1: Context switches per second	7/23/201 9 4:40	49.4
linuxlitevm2: Context switches per second	7/23/201 9 4:40	35.6
linuxlitevm3: Context switches per second	7/23/201 9 4:40	50.4
Linuxlite1: Context switches per second	7/23/201 9 4:45	49.6
linuxlitevm2: Context switches per second	7/23/201 9 4:45	36
linuxlitevm3: Context switches per second	7/23/201 9 4:45	50.2
Linuxlite1: Context switches per second	7/23/201 9 4:50	49.2
linuxlitevm2: Context switches per second	7/23/201 9 4:50	35.4
linuxlitevm3: Context switches per second	7/23/201 9 4:50	50
Linuxlite1: Context switches per second	7/23/201 9 4:55	49.4
linuxlitevm2: Context switches per second	7/23/201 9 4:55	36.2
linuxlitevm3: Context switches per second	7/23/201 9 4:55	49.8
Linuxlite1: Context switches per second	7/23/201 9 5:00	50.8
linuxlitevm2: Context switches per second	7/23/201 9 5:00	36.6
linuxlitevm3: Context switches per second	7/23/201 9 5:00	50.6

APPENDIX C – Performance testing graphs

Performance metrics

The first processing metric analysed in this subsection is CPU context switches per second. Figure 10-1, Figure 10-2, Figure 10-3, Figure 10-4, Figure 10-5, Figure 10-6, Figure 10-7 and Figure 10-8 shows the metrics for the various steps and devices.

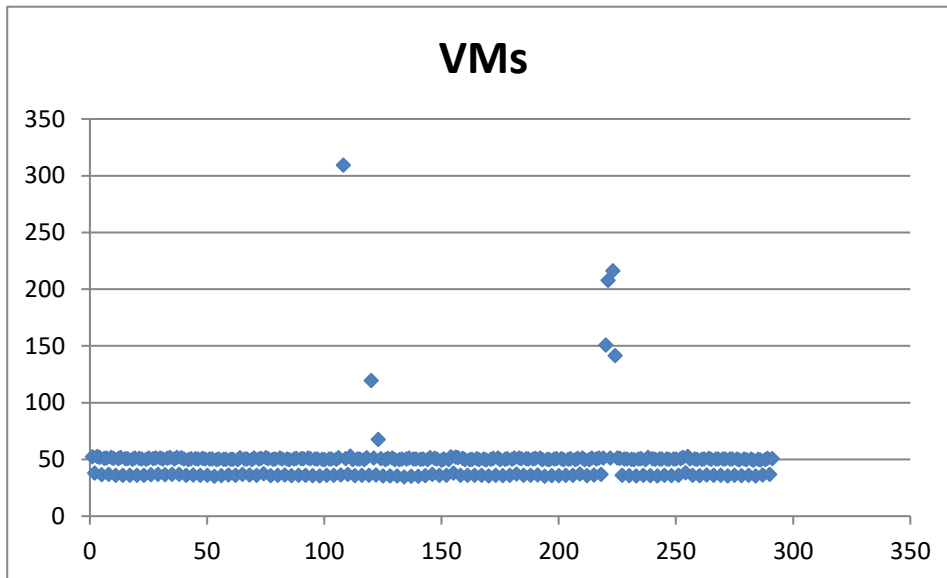


Figure 10-1 Step 1 CPU context switches per second for VMs

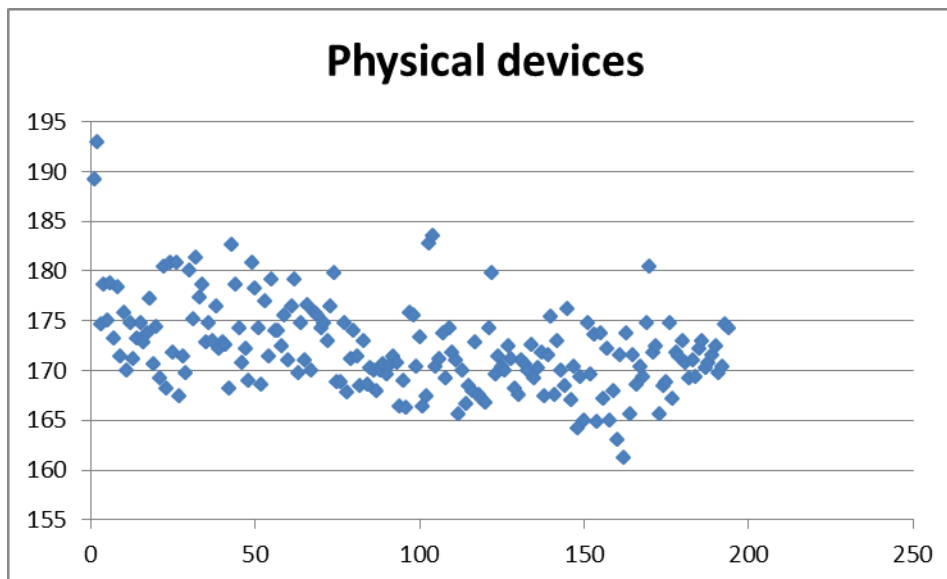


Figure 10-2 Step 1 Step 4 CPU context switches per second for physical devices

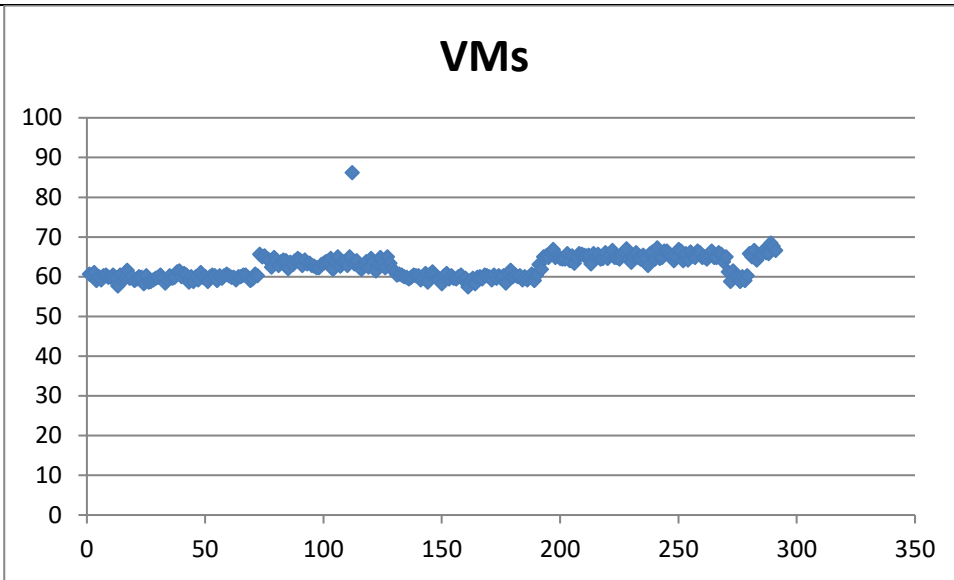


Figure 10-3 Step 2 CPU context switches per second for VMs

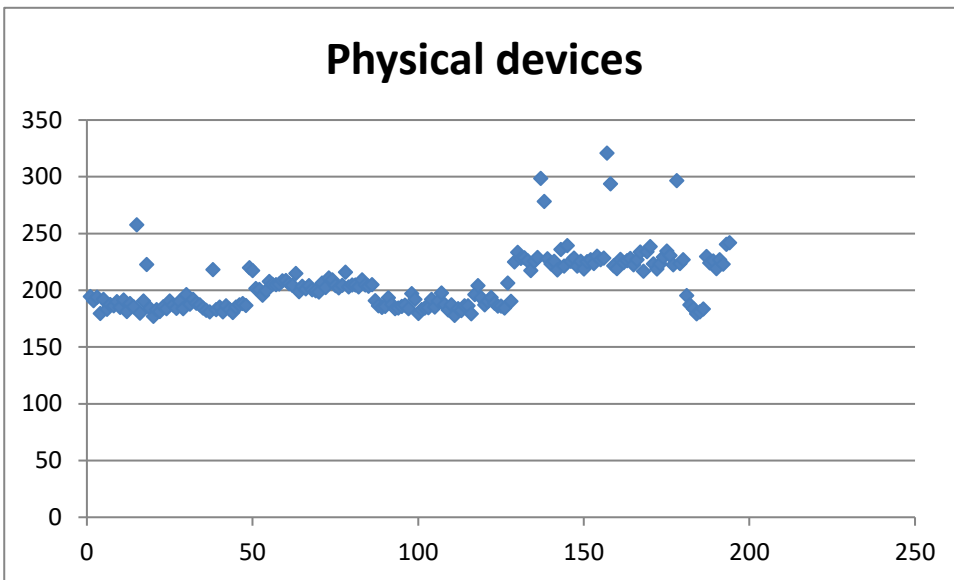


Figure 10-4 Step 2 CPU context switches per second for physical devices

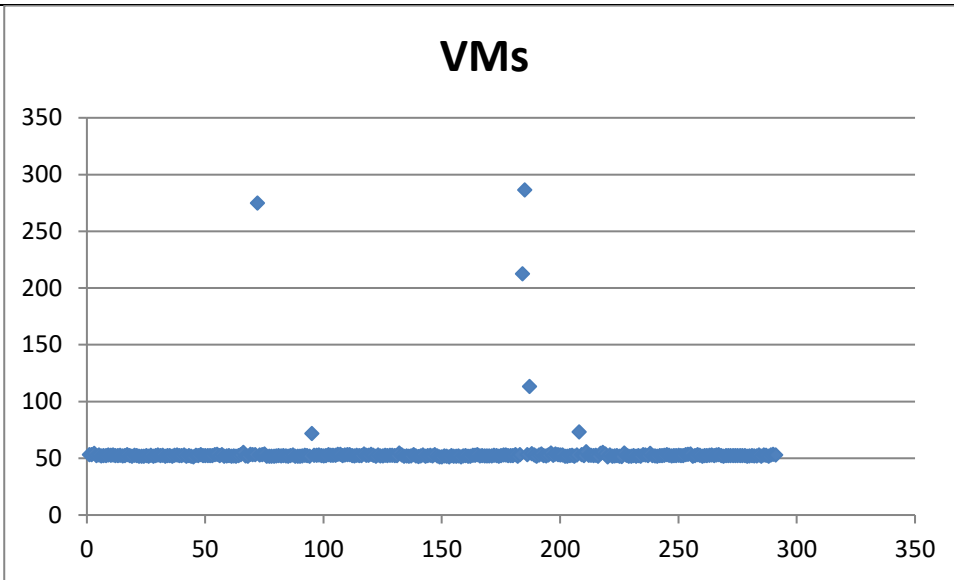


Figure 10-5 Step 3 CPU context switches per second for VMs

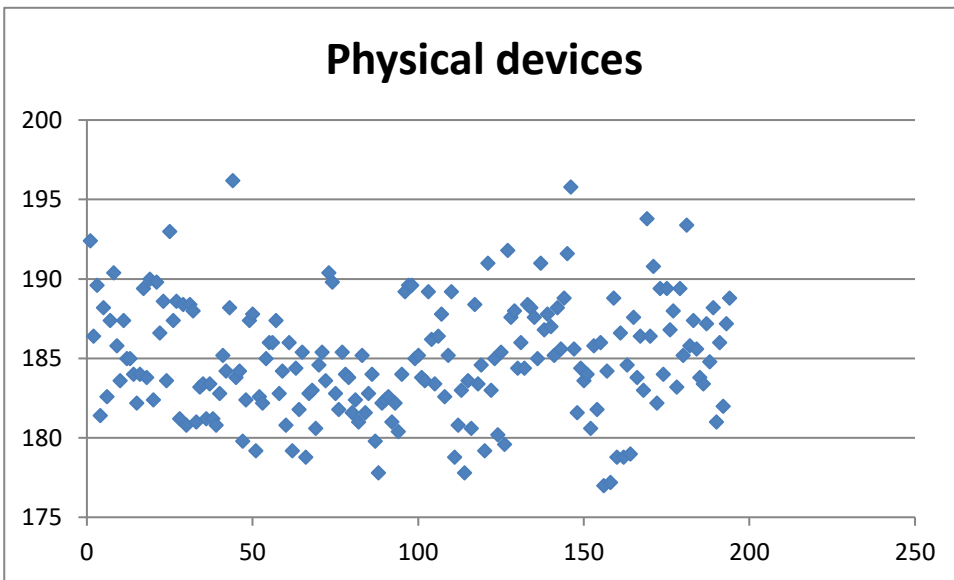


Figure 10-6 Step 3 CPU context switches per second for physical devices

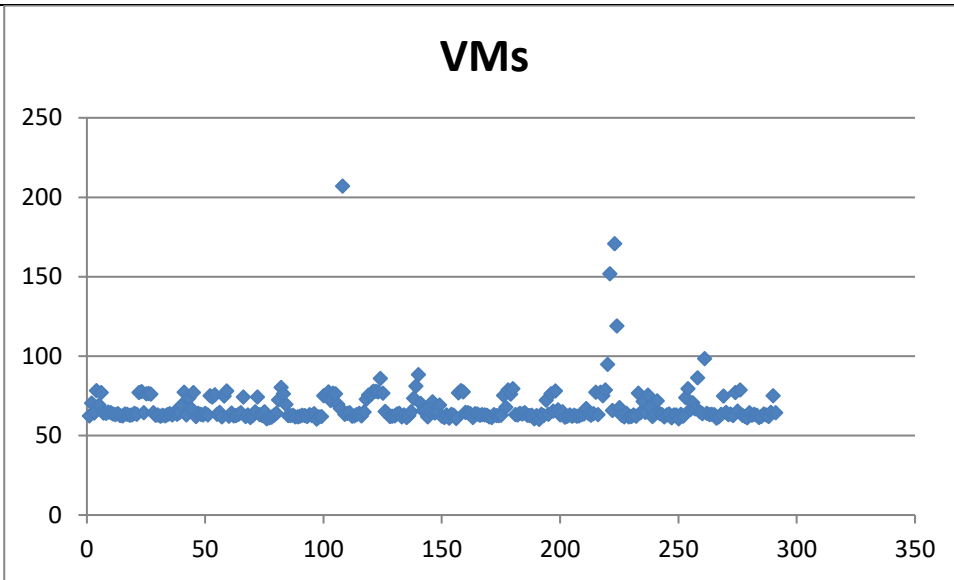


Figure 10-7 Step 4 CPU context switches per second for VMs

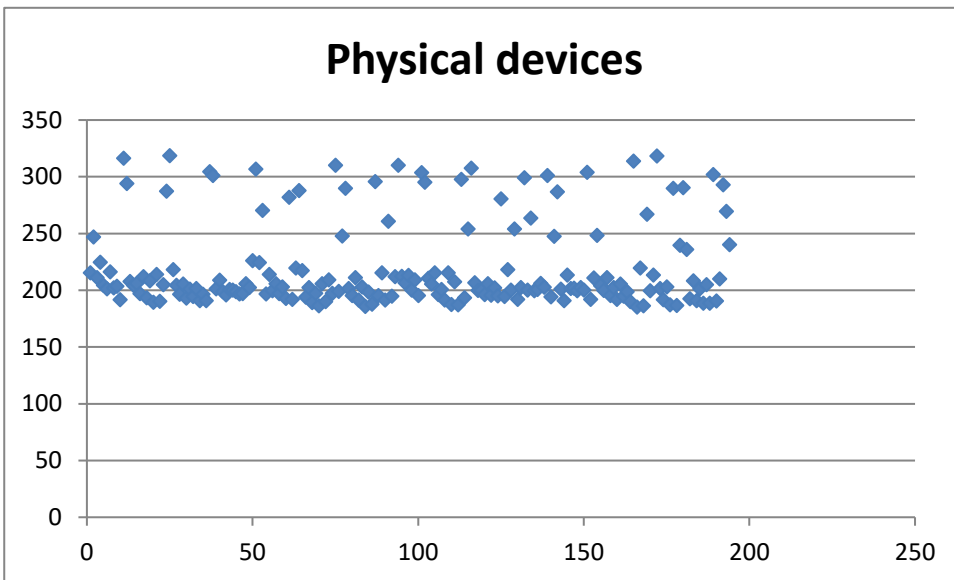


Figure 10-8 Step 4 CPU context switches per second for physical devices

The next metric analysed in CPU idle time which is measured in percentage. Figure 10-9, Figure 10-10, Figure 10-11, Figure 10-12, Figure 10-13, Figure 10-14, Figure 10-15 and Figure 10-16 shows the CPU idle time metric for the various steps and devices.

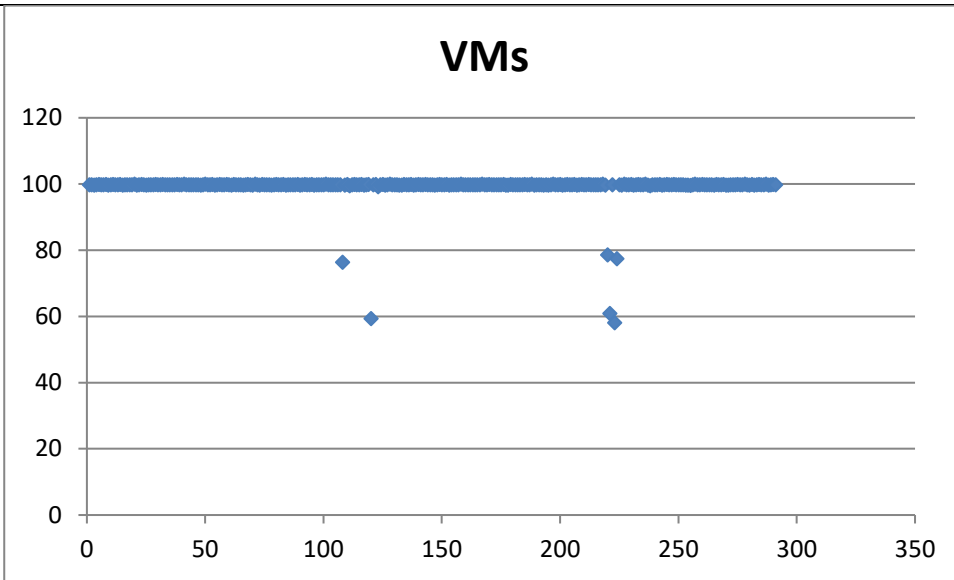


Figure 10-9 Step 1 CPU idle time for VMs

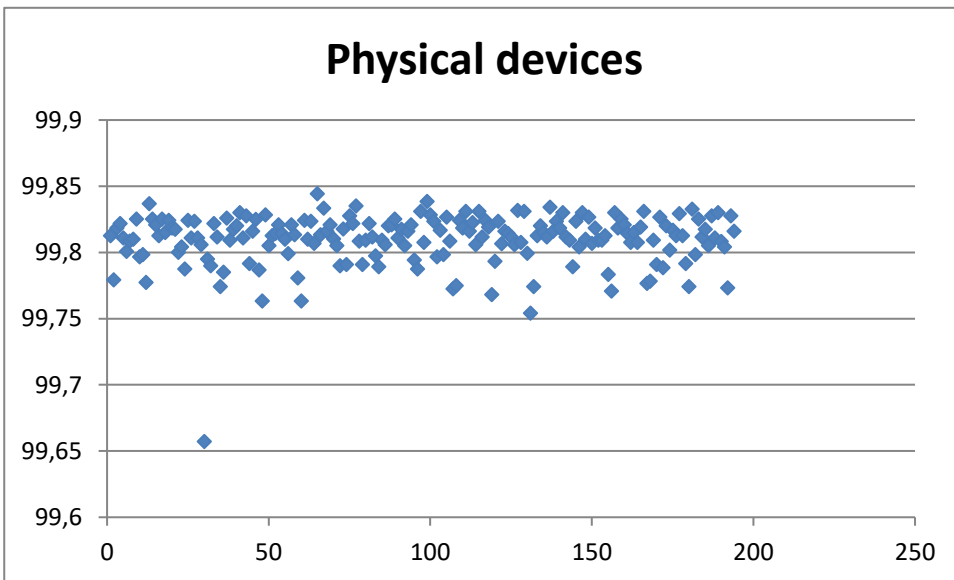


Figure 10-10 Step 1 CPU idle time for physical devices

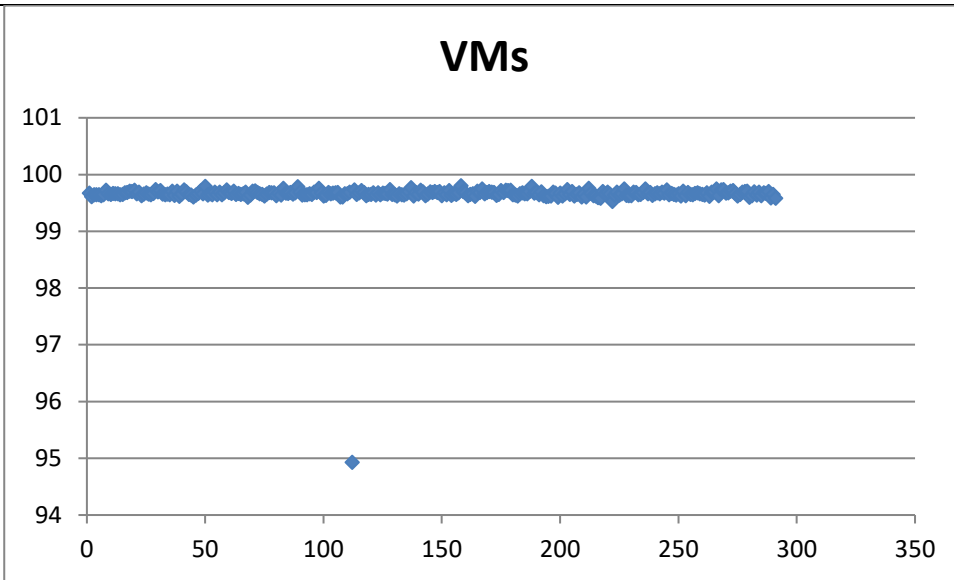


Figure 10-11 Step 2 CPU idle time for VMs

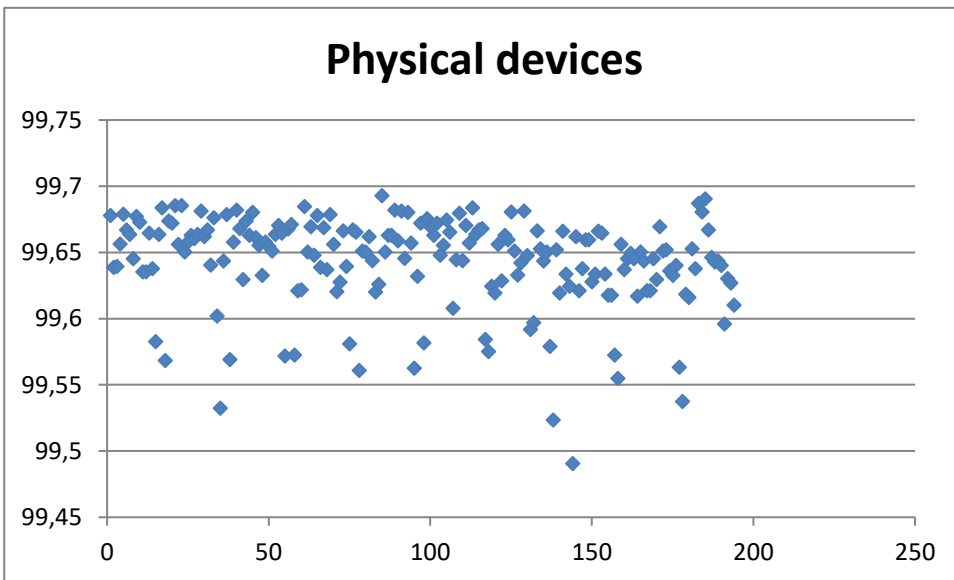


Figure 10-12 Step 2 CPU idle time for physical devices

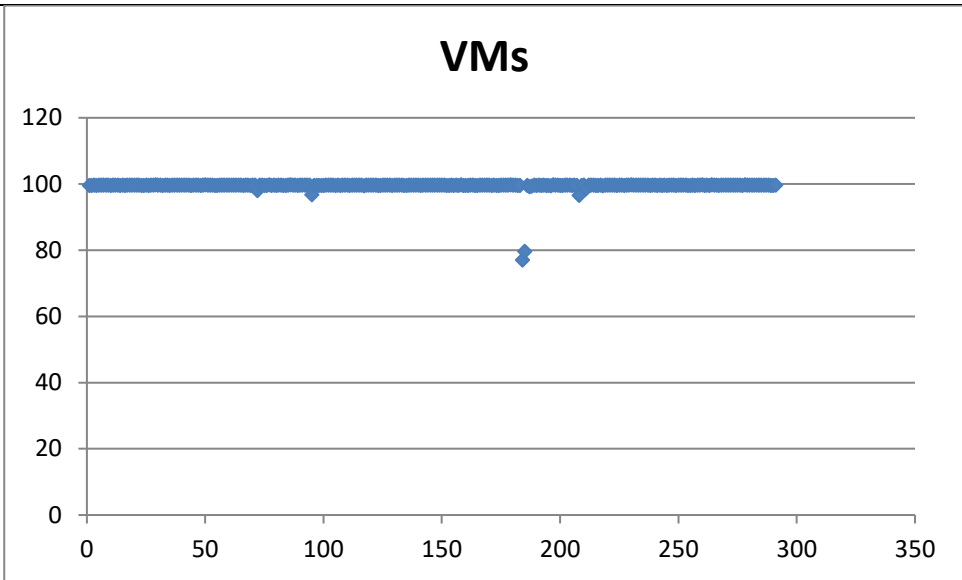


Figure 10-13 Step 3 CPU idle time for VMs

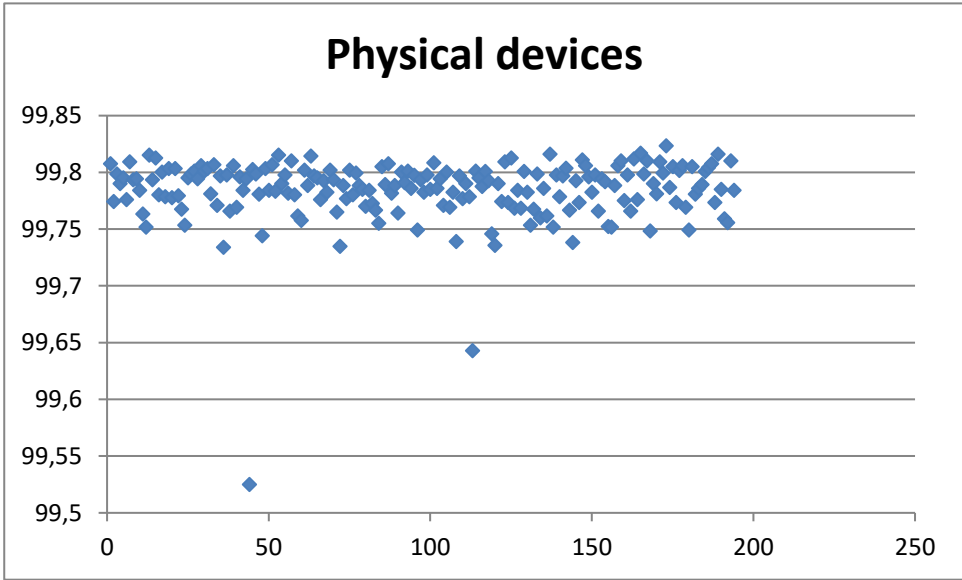


Figure 10-14 Step 3 CPU idle time for physical devices

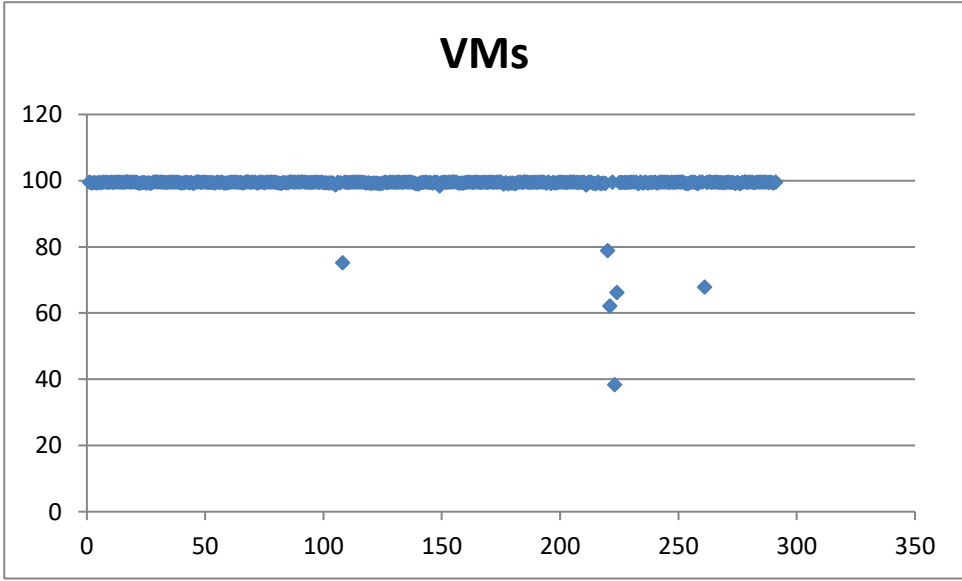


Figure 10-15 Step 4 CPU idle time for VMs

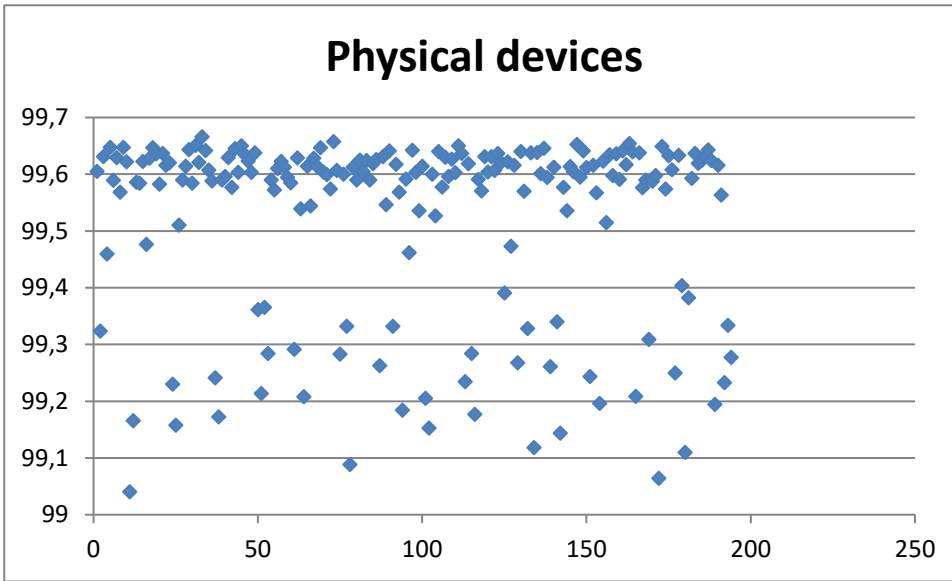


Figure 10-16 Step 4 CPU idle time for physical devices

The next processor-based metric is CPU interrupts per second. Figure 10-17, Figure 10-18, Figure 10-19, Figure 10-20, Figure 10-21, Figure 10-22, Figure 10-23 and Figure 10-24 shows the various CPU interrupts per second graphs for the various steps and devices.

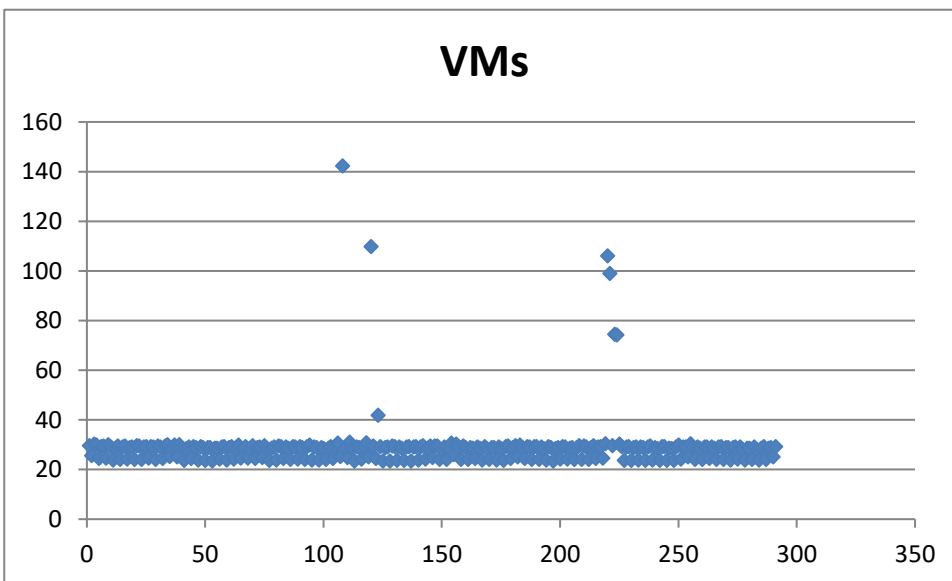


Figure 10-17 Step 1 CPU interrupts per second for VMs

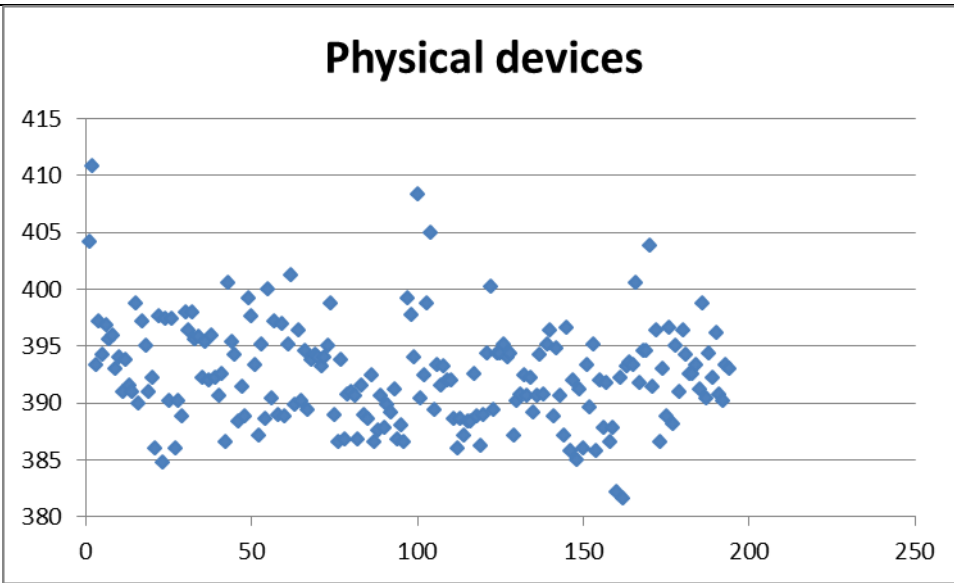


Figure 10-18 Step 1 CPU interrupts per second for physical devices

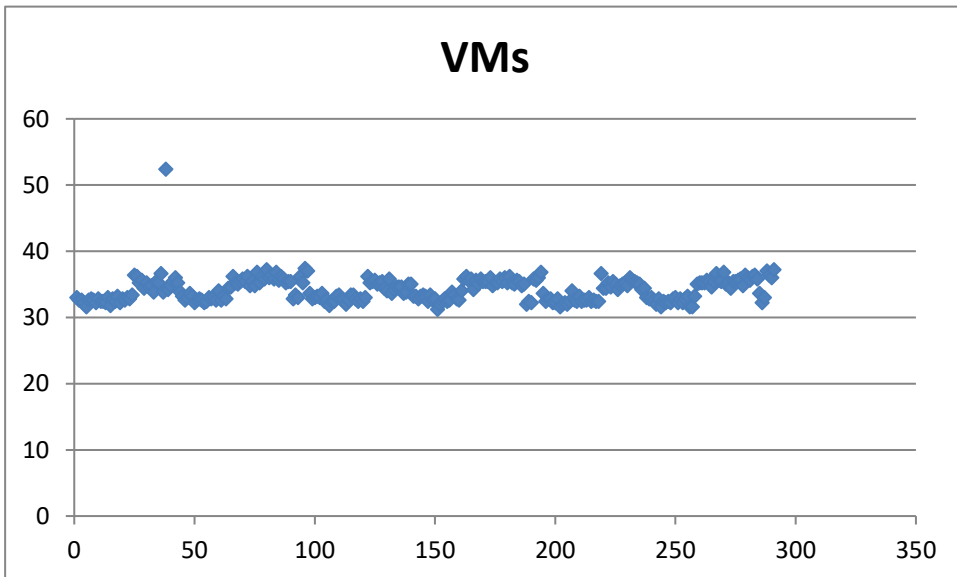


Figure 10-19 Step 2 CPU interrupts per second for VMs

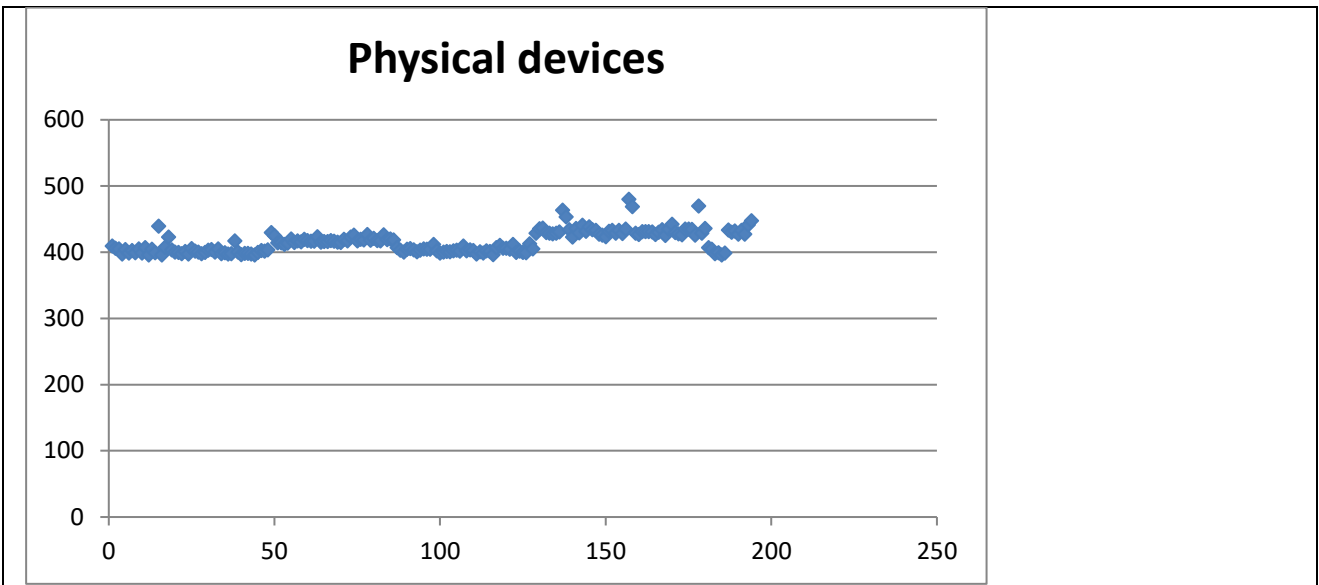


Figure 10-20 Step 2 CPU interrupts per second for physical devices

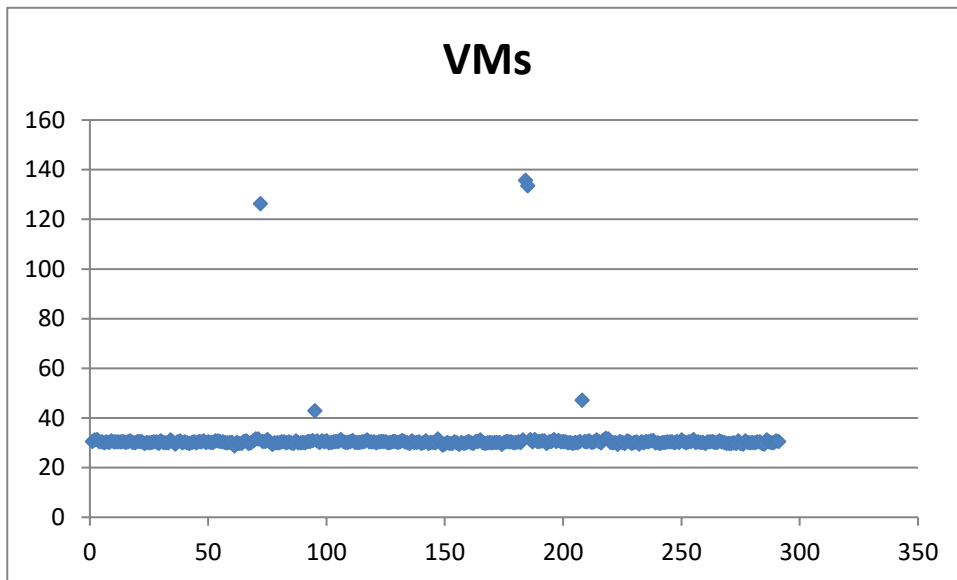


Figure 10-21 Step 3 CPU interrupts per second for VMs

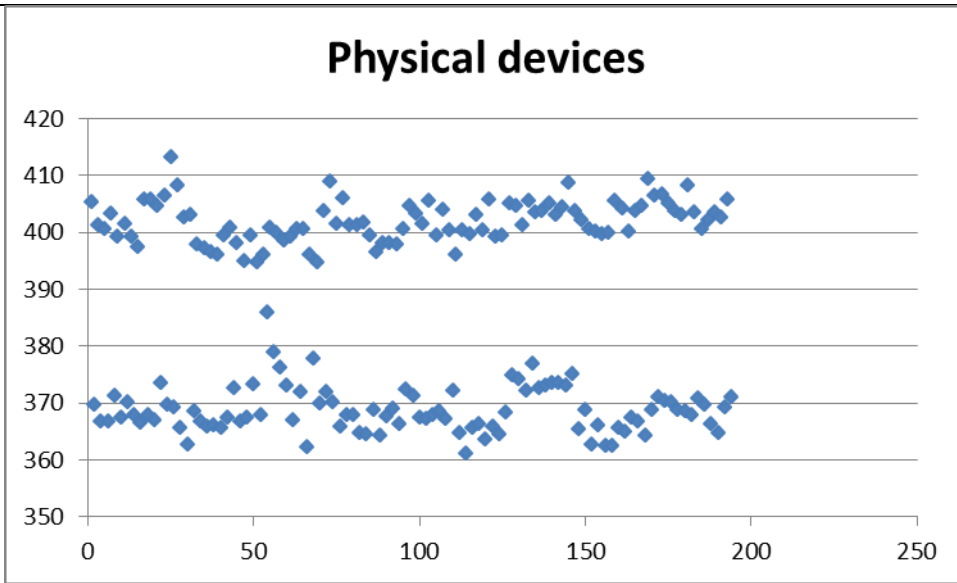


Figure 10-22 Step 3 CPU interrupts per second for physical devices

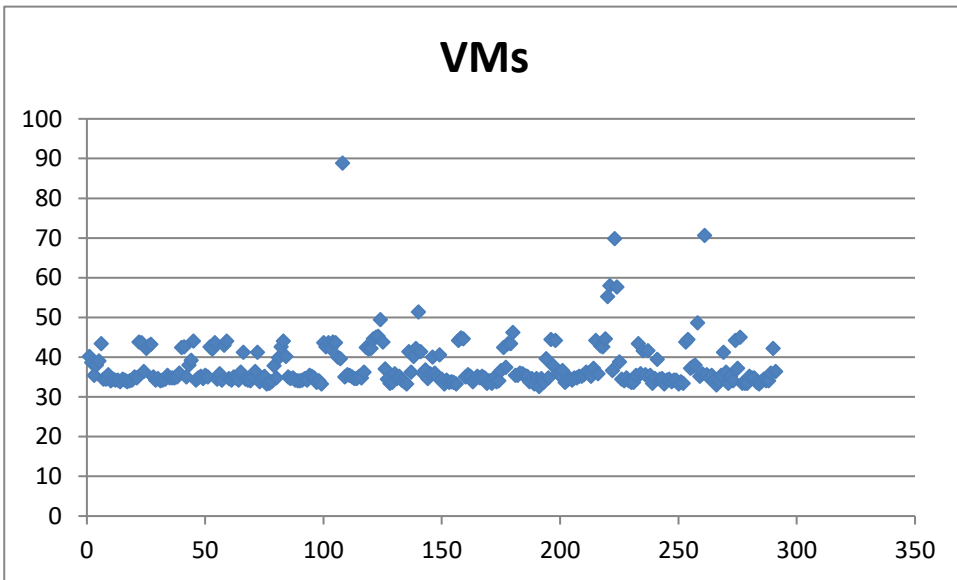


Figure 10-23 Step 4 CPU interrupts per second for VMs

Physical devices

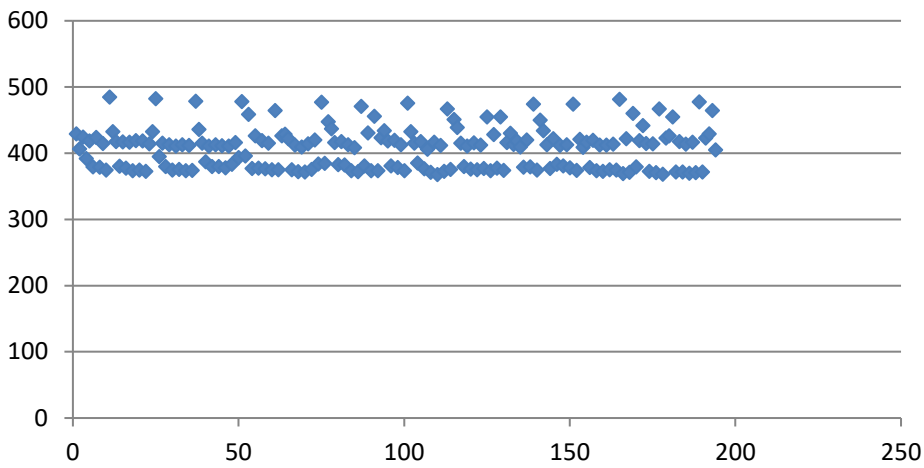


Figure 10-24 Step 4 CPU interrupts per second for physical devices

The next processor related metric is the CPU load 1min average per core. Figure 10-25, Figure 10-26, Figure 10-27, Figure 10-28, Figure 10-29, Figure 10-30, Figure 10-31 and Figure 10-32 shows is the CPU load 1min average per core metric graphs for the various devices.

VMs

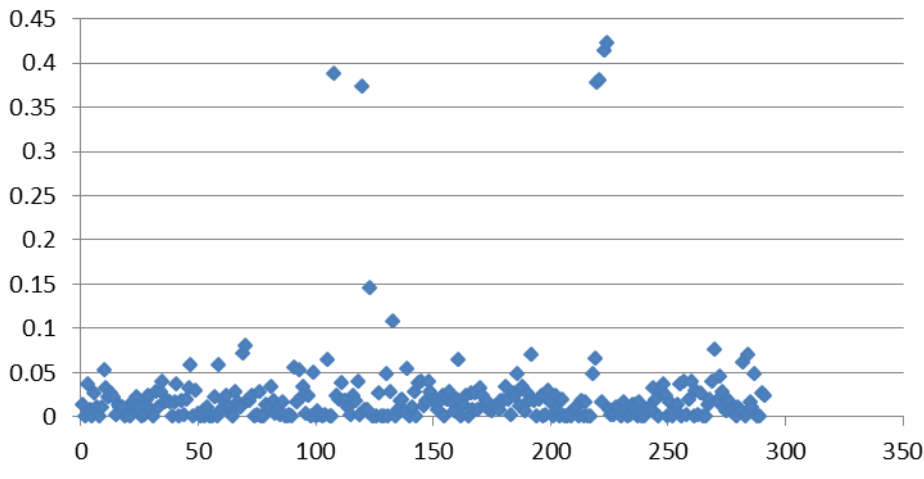


Figure 10-25 Step 1 CPU load 1min average per core for VMs

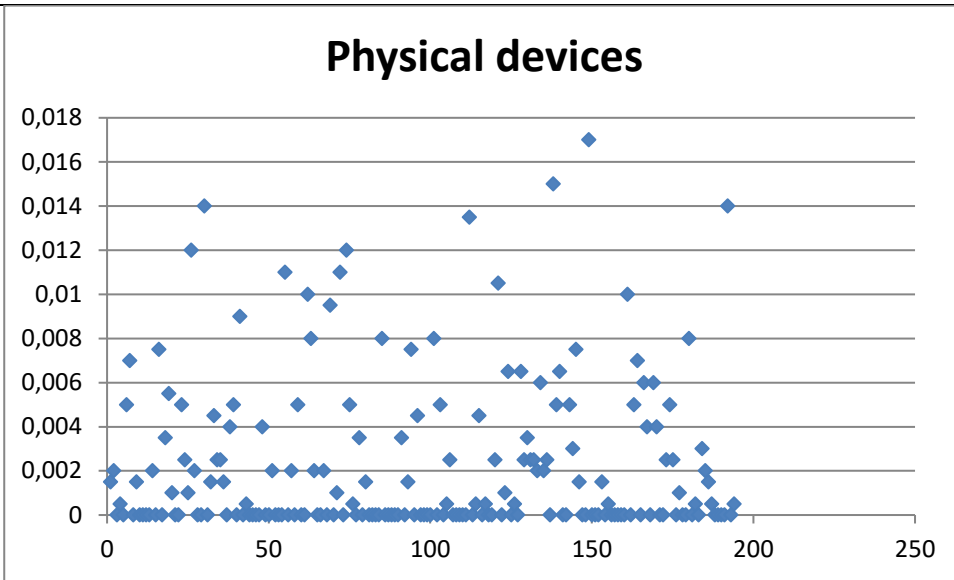


Figure 10-26 Step 1 CPU load 1min average per core for physical devices

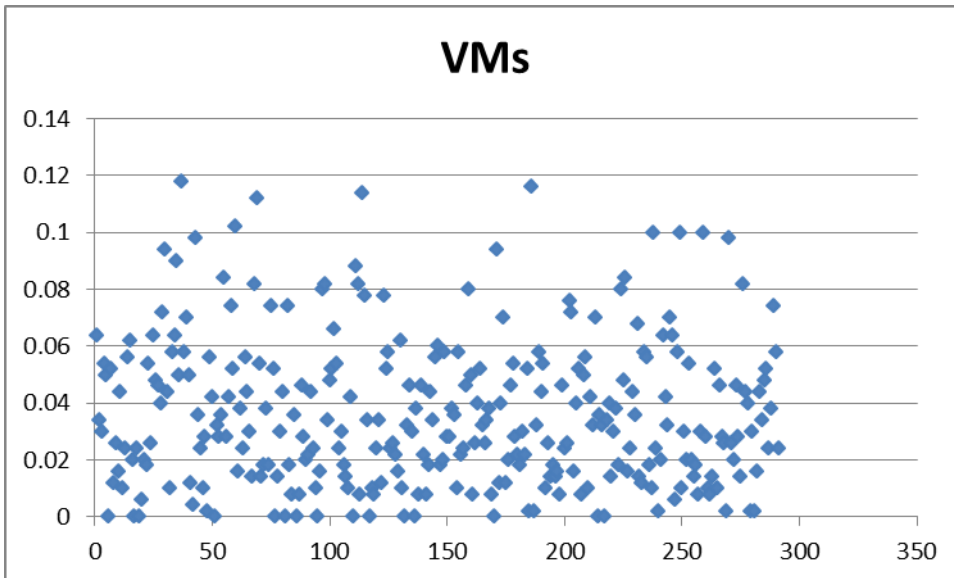


Figure 10-27 Step 2 CPU load 1min average per core for VMs

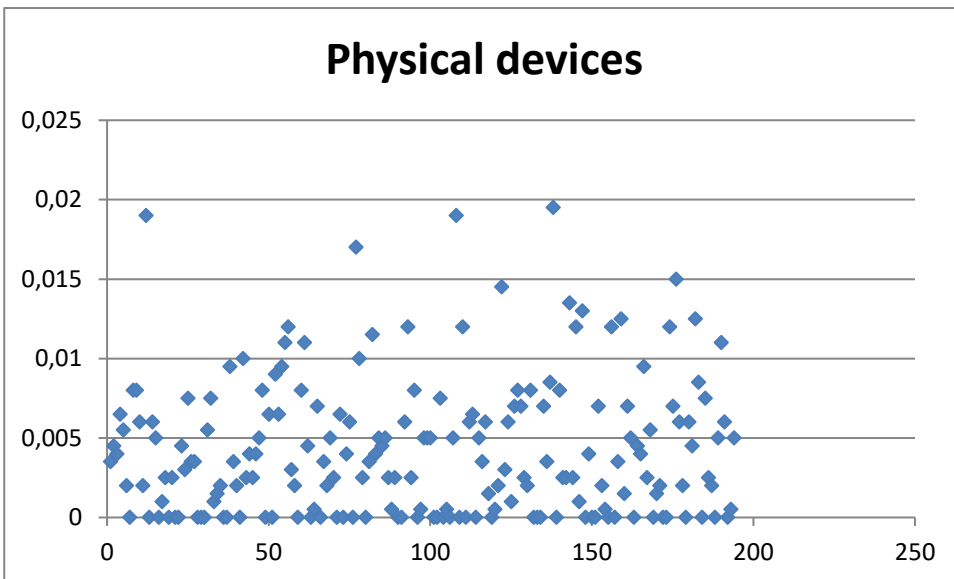


Figure 10-28 Step 2 CPU load 1min average per core for physical devices

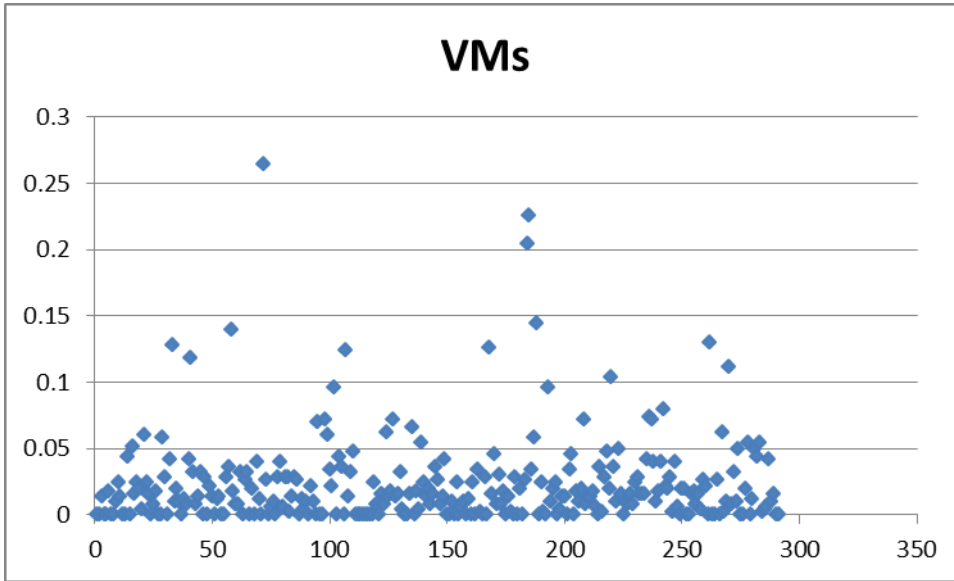


Figure 10-29 Step 3 CPU load 1min average per core for VMs

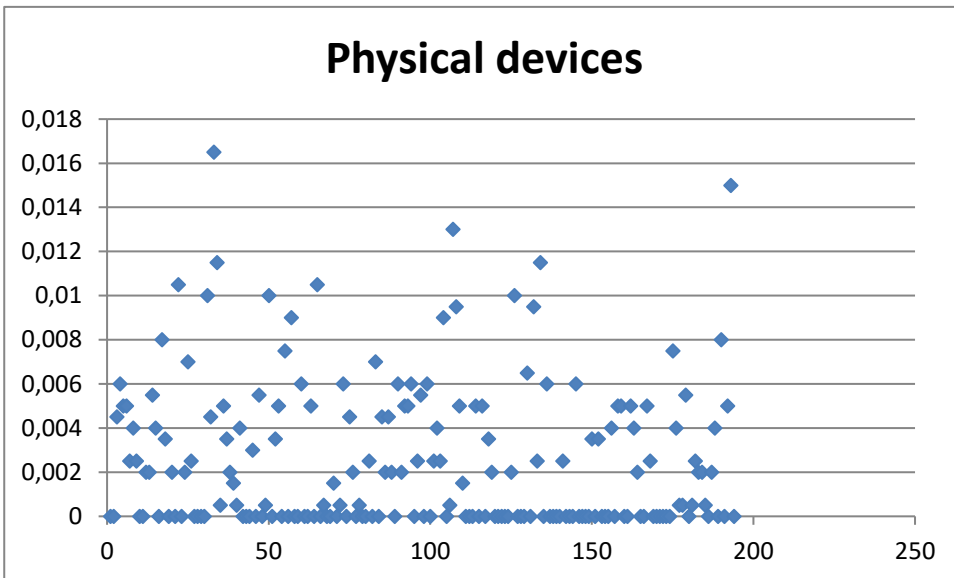


Figure 10-30 Step 3 CPU load 1min average per core for physical devices

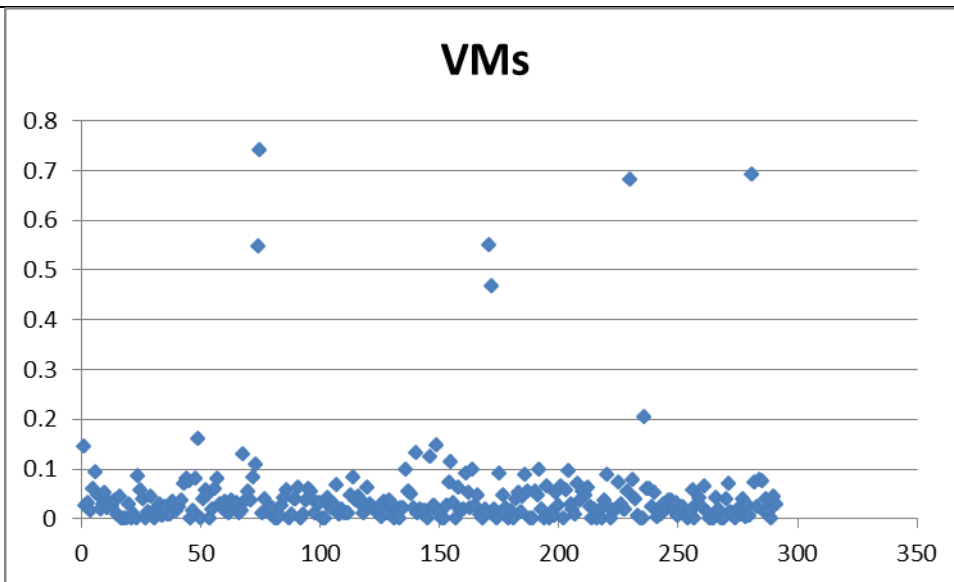


Figure 10-31 Step 4 CPU load 1min average per core for VMs

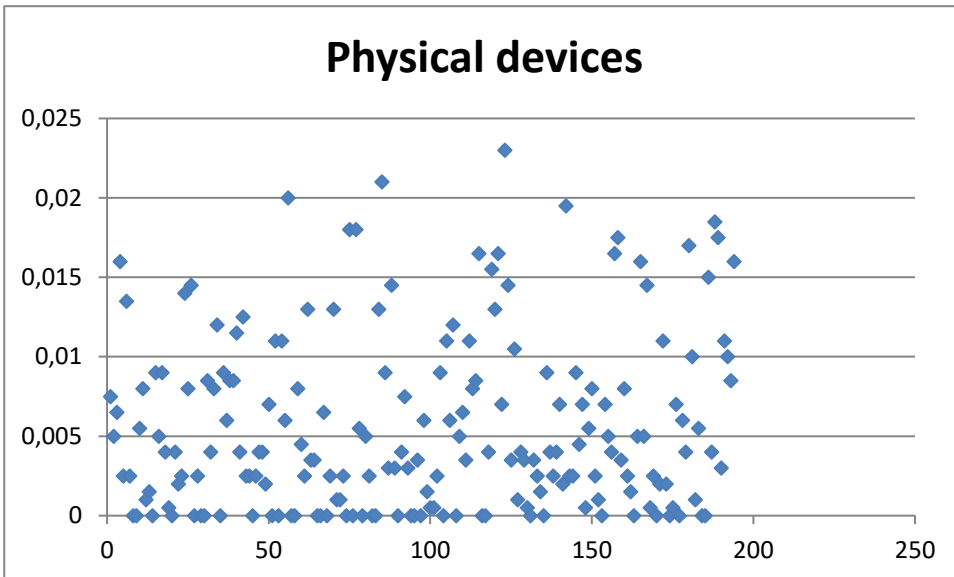


Figure 10-32 Step 4 CPU load 1min average per core for physical devices

The next processor related metric is CPU system time. Figure 10-33, Figure 10-34, Figure 10-35, Figure 10-36, Figure 10-37, Figure 10-38, Figure 10-39 and Figure 10-40 shows the CPU system time for the devices during the various steps of the simulation.

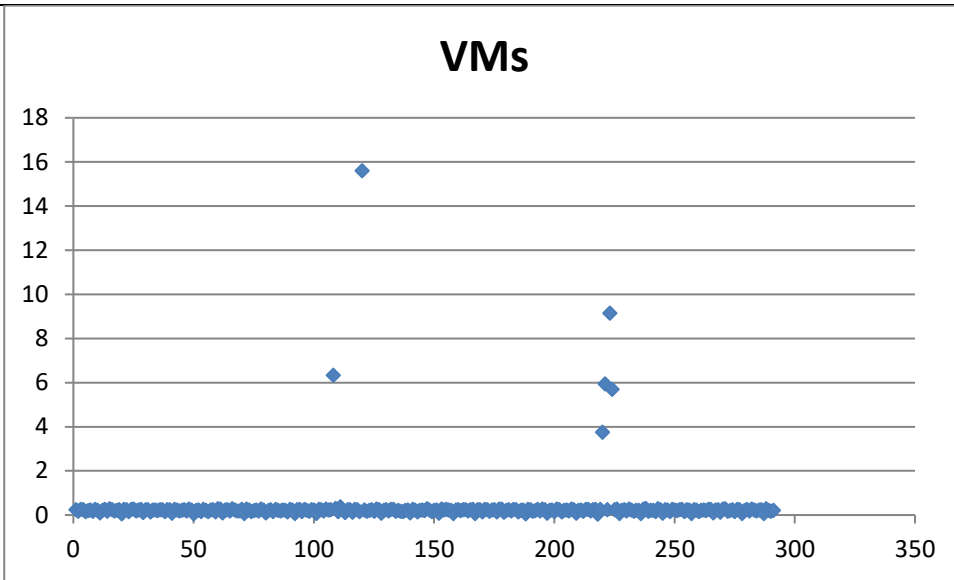


Figure 10-33 Step 1 CPU system time for VMs

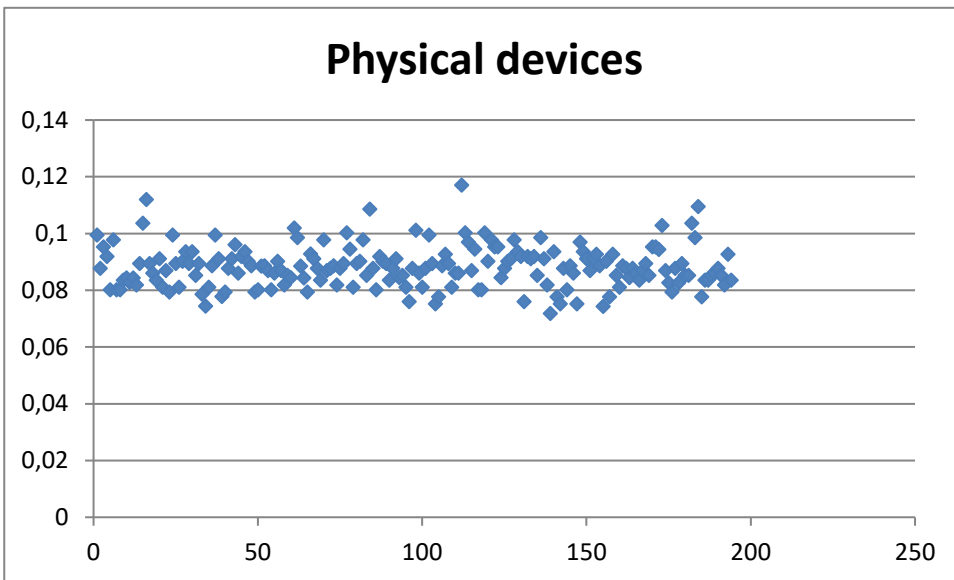


Figure 10-34 Step 1 CPU system time for physical devices

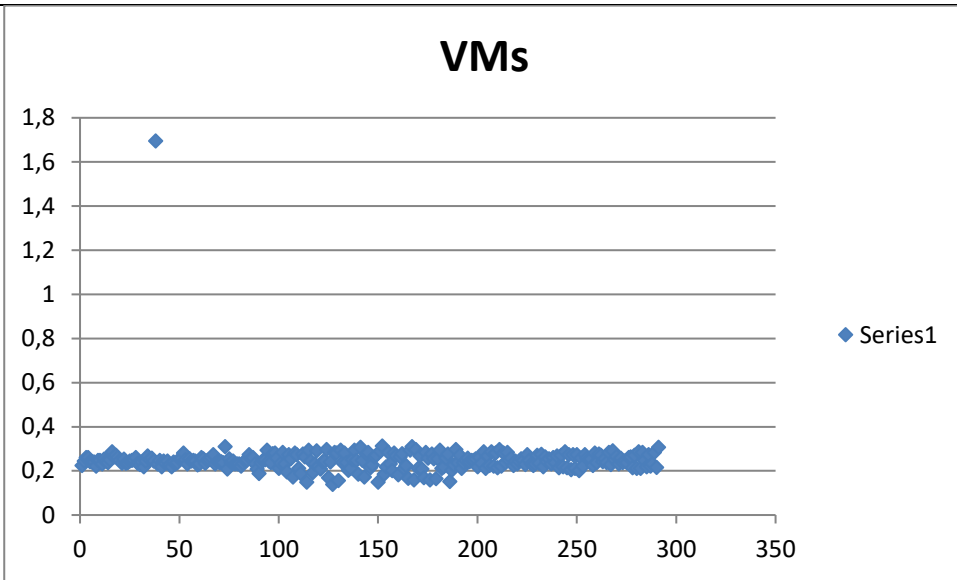


Figure 10-35 Step 2 CPU system time for VMs

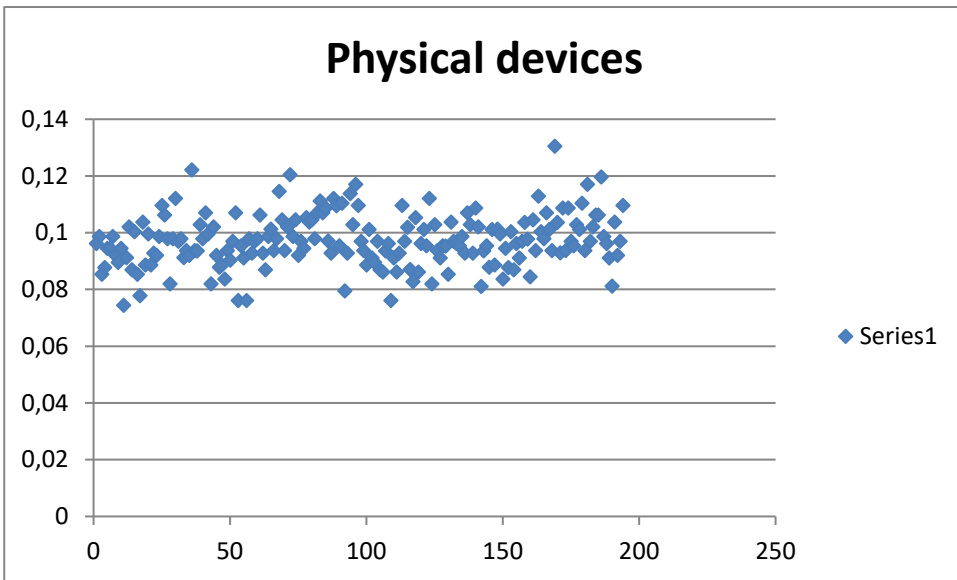


Figure 10-36 Step 2 CPU system time for physical devices

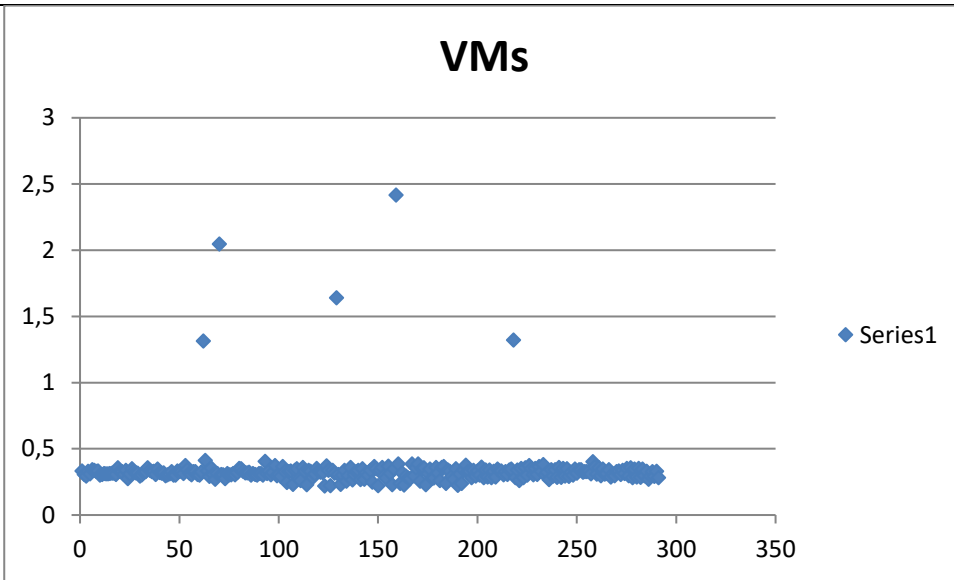


Figure 10-37 Step 3 CPU system time for VMs

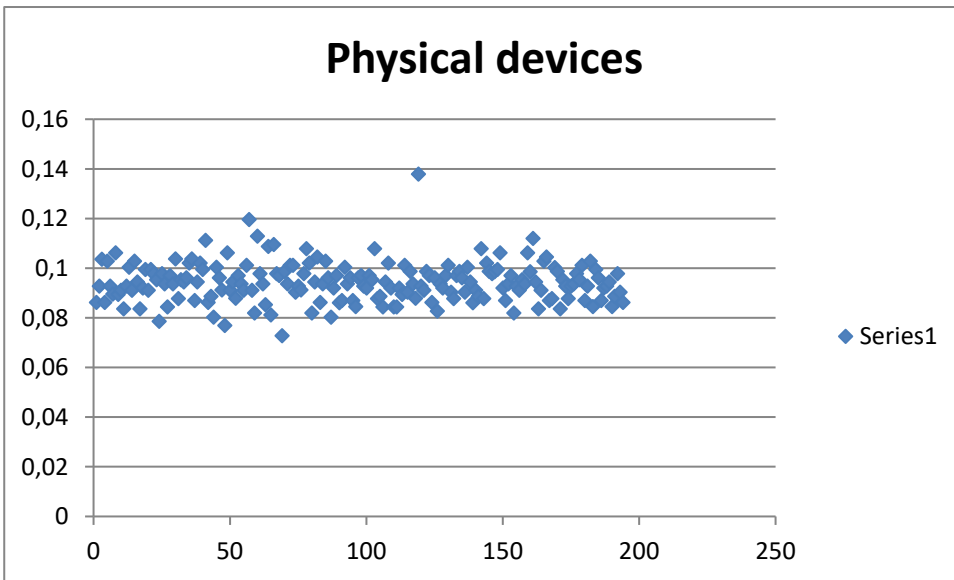


Figure 10-38 Step 3 CPU system time for physical devices

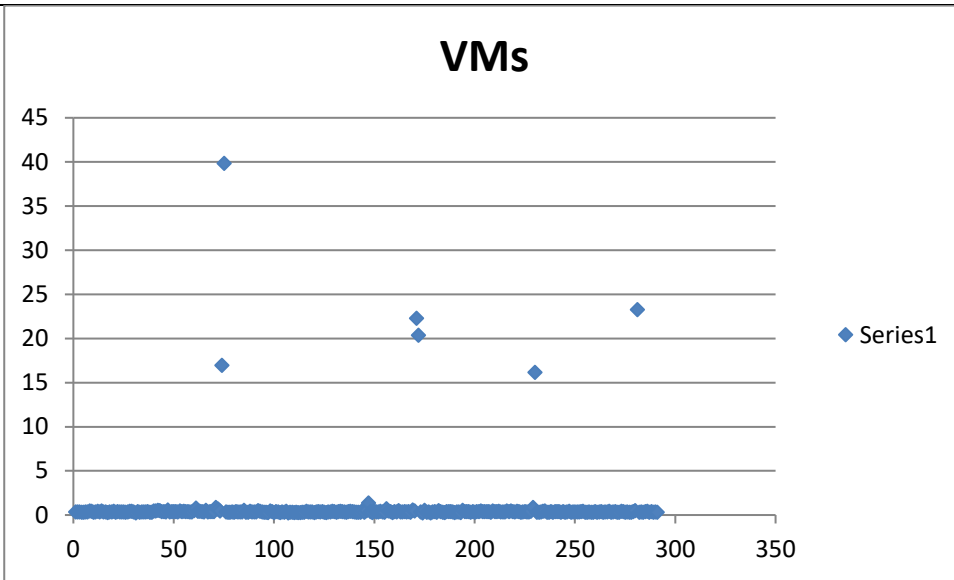


Figure 10-39 Step 4 CPU system time for VMs

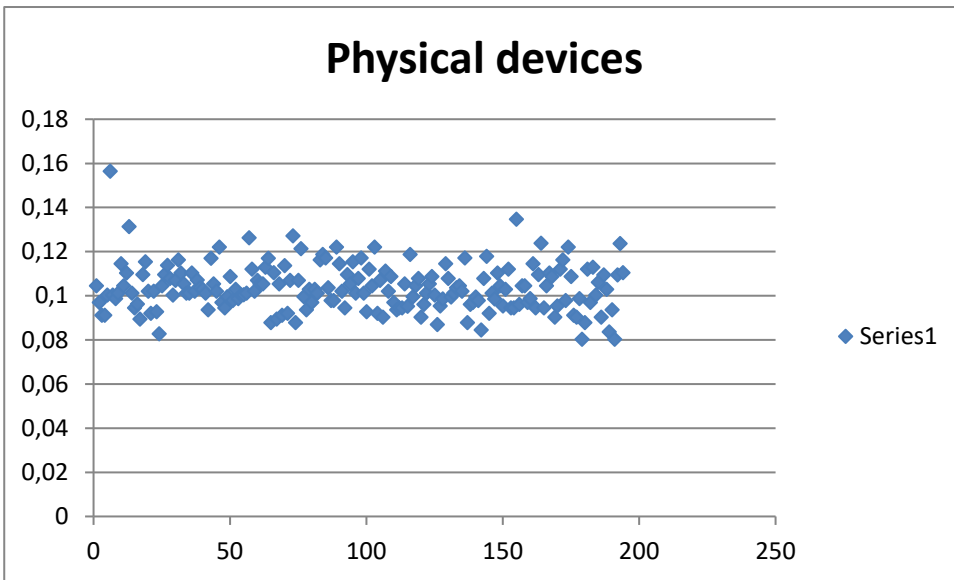


Figure 10-40 Step 4 CPU system time for physical devices

The next metric related to processor performance is the number of processes. Figure 10-41, Figure 10-42, Figure 10-43, Figure 10-44, Figure 10-45, Figure 10-46, Figure 10-47 and Figure 10-48 shows the number of processes metric for the devices during the steps of the simulation.

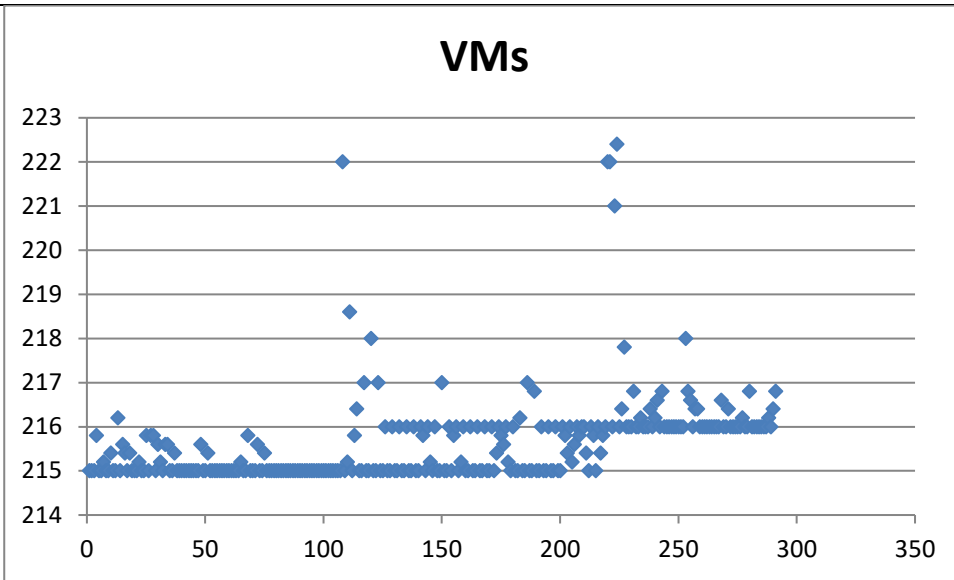


Figure 10-41 Step 1 number of processes for VMs

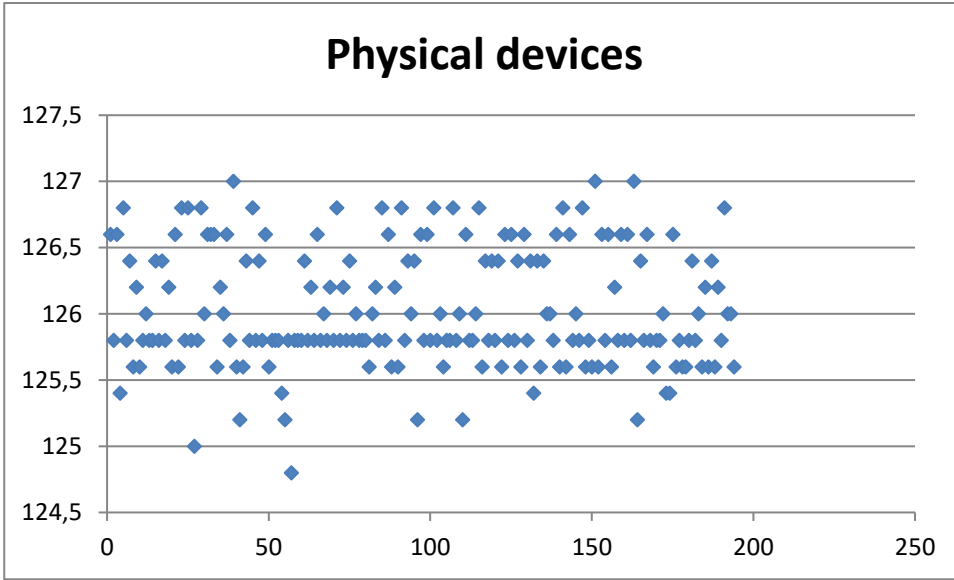


Figure 10-42 Step 1 number of processes for physical devices

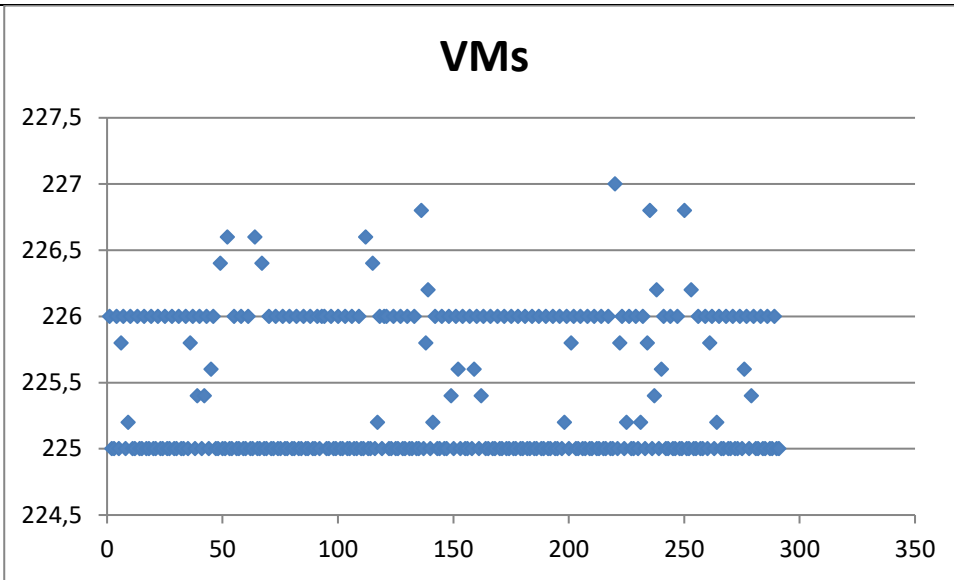


Figure 10-43 Step 2 number of processes for VMs

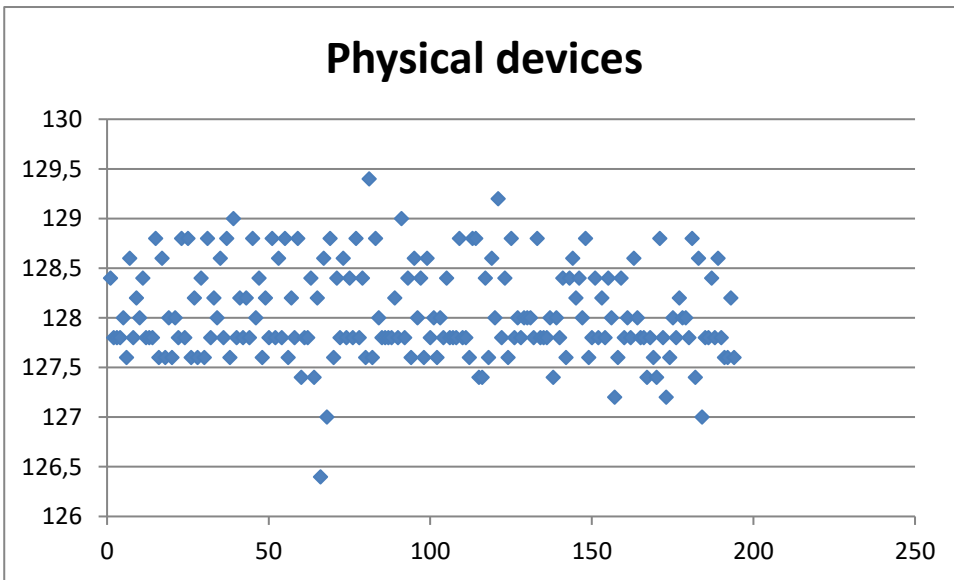


Figure 10-44 Step 2 number of processes for physical devices

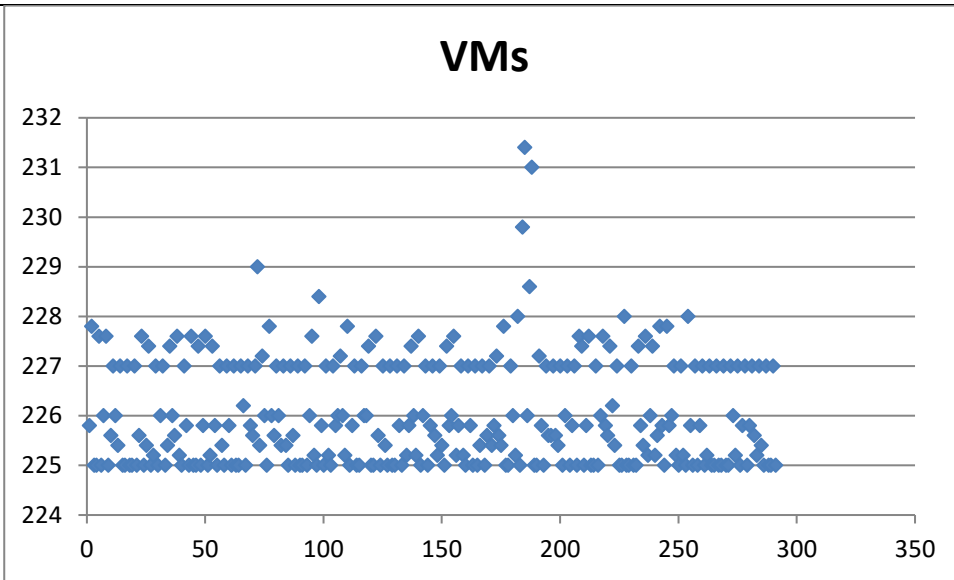


Figure 10-45 Step 3 number of processes for VMs

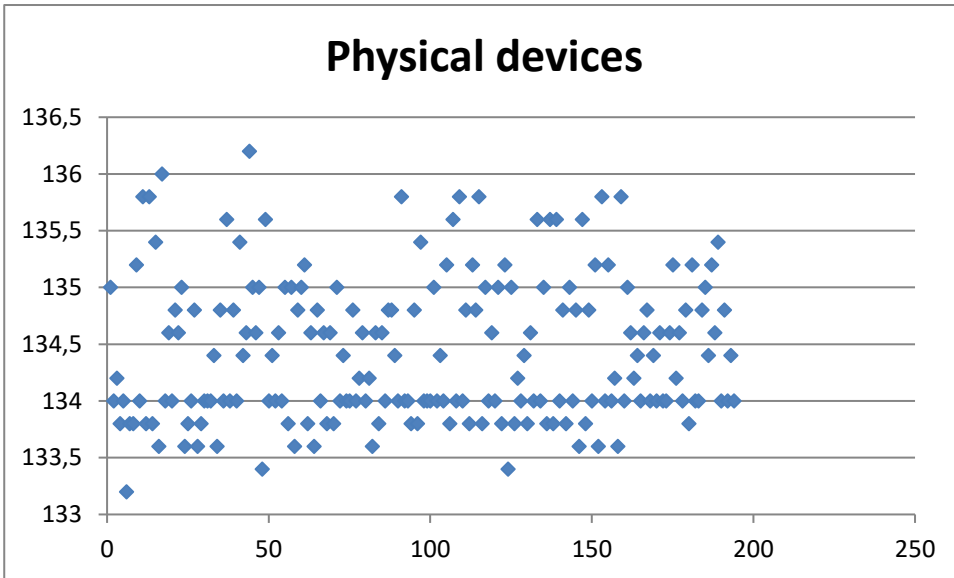


Figure 10-46 Step 3 number of processes for physical devices

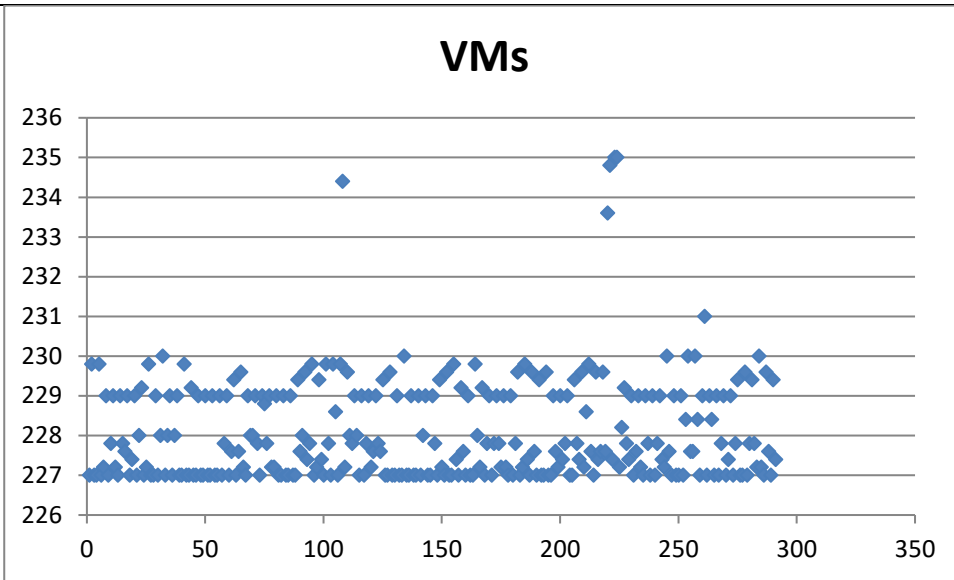


Figure 10-47 Step 4 number of processes for VMs

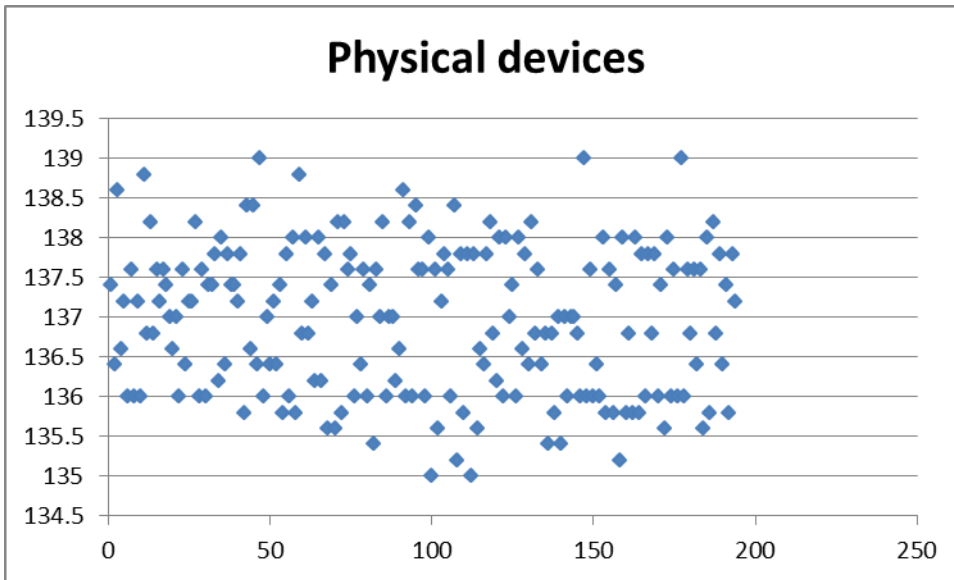


Figure 10-48 Step 4 number of processes for physical devices

The final metric related to the processing resources of the devices is the number of running processes metric. Figure 10-49, Figure 10-50, Figure 10-51, Figure 10-52, Figure 10-53, Figure 10-54, Figure 10-55 and Figure 10-56 show the number of running processes metric for the various steps of the simulation For DFR in IoT.

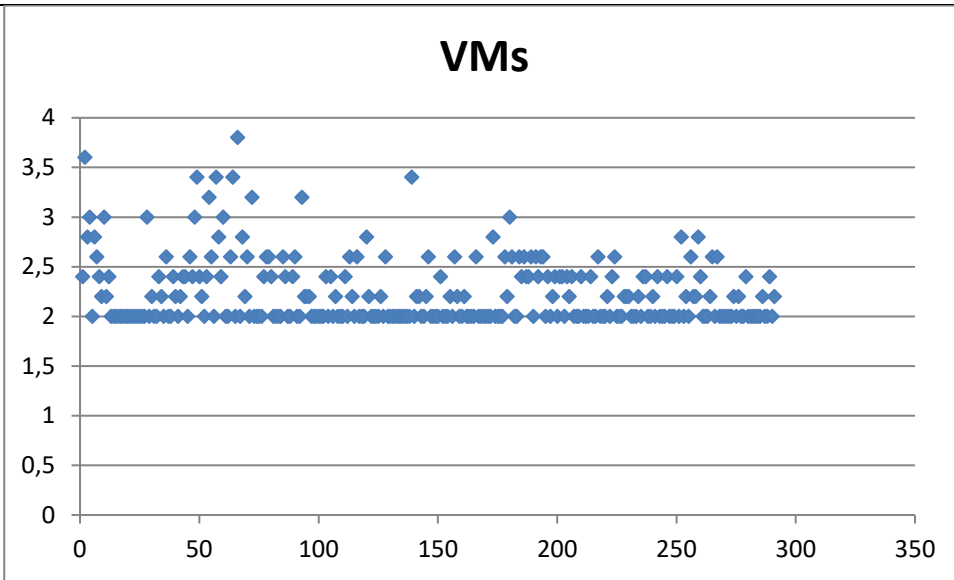


Figure 10-49 Step 1 number of running processes on VMs

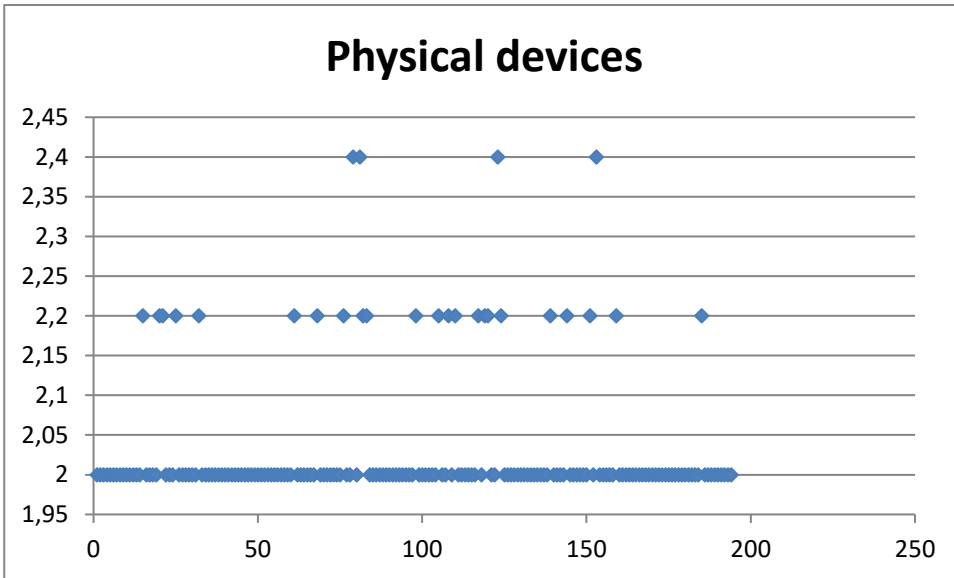


Figure 10-50 Step 1 number of running processes on physical devices

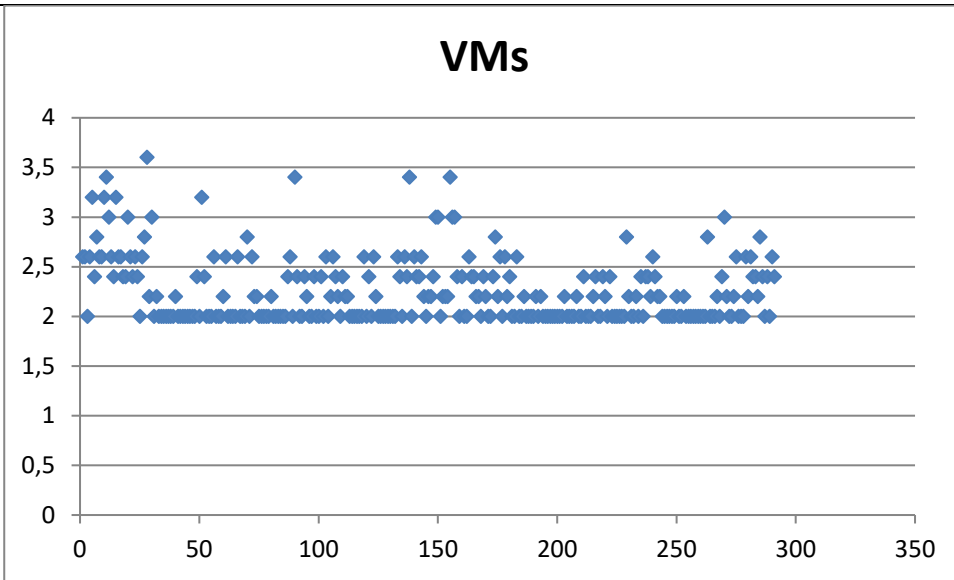


Figure 10-51 Step 2 number of running processes on VMs

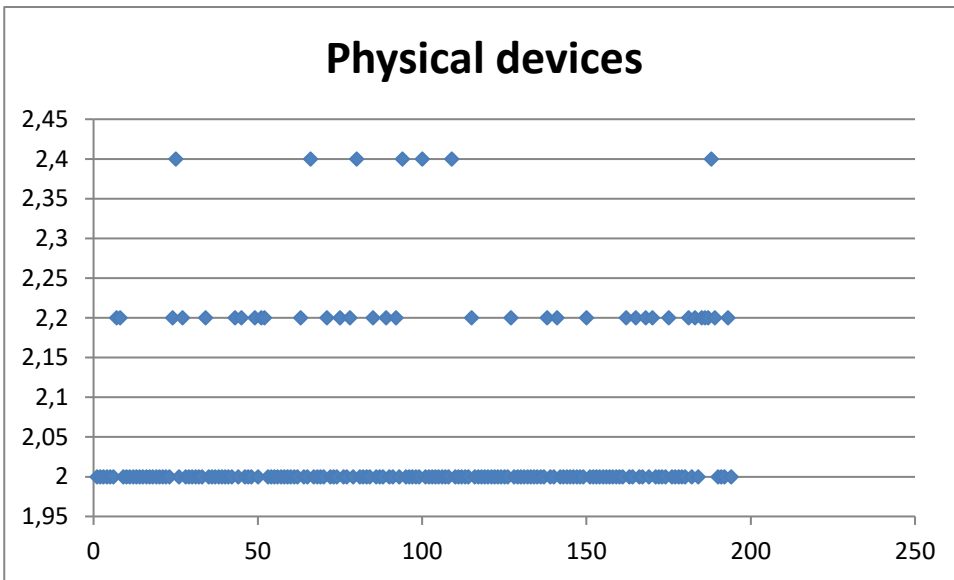


Figure 10-52 Step 2 number of running processes on physical devices

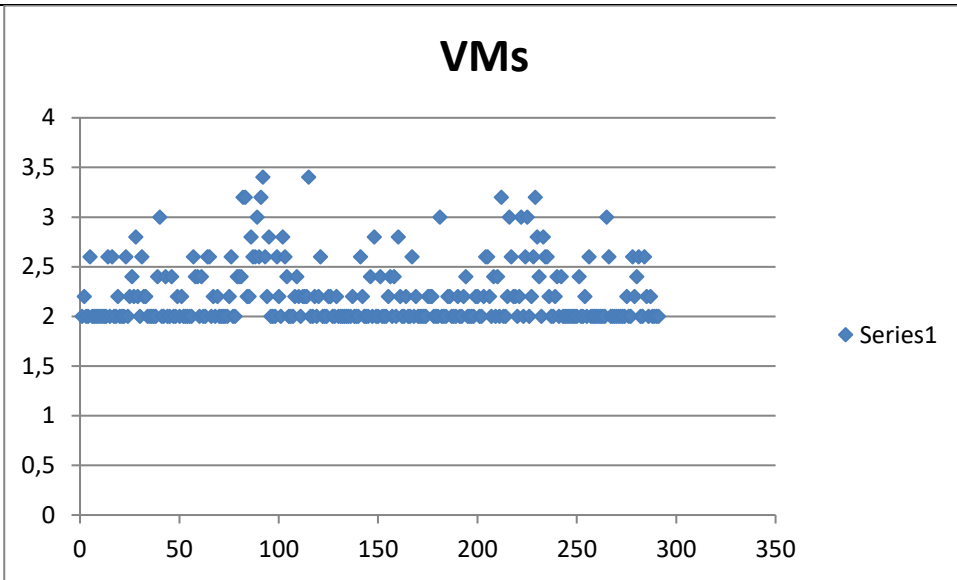


Figure 10-53 Step 3 number of running processes on VMs

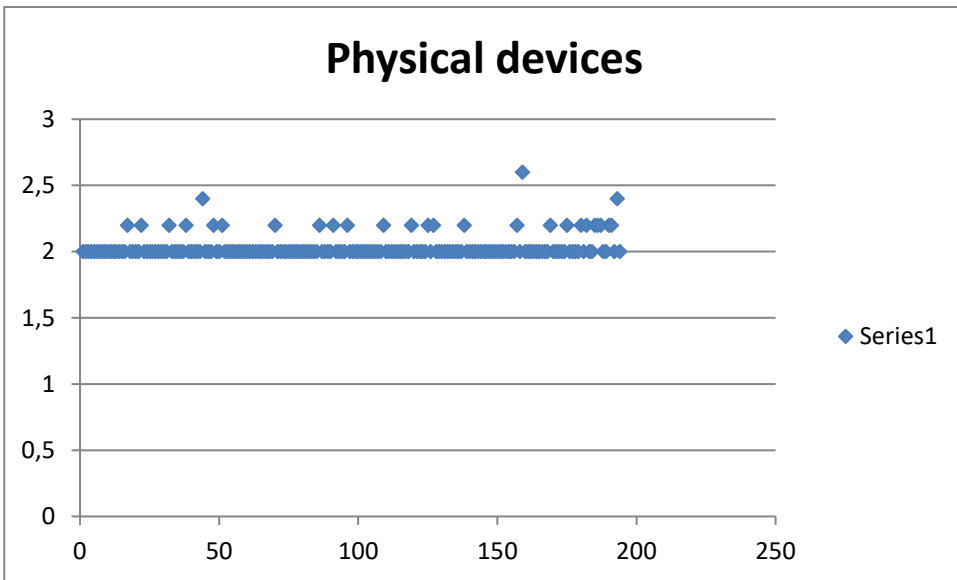


Figure 10-54 Step 3 number of running processes on physical devices

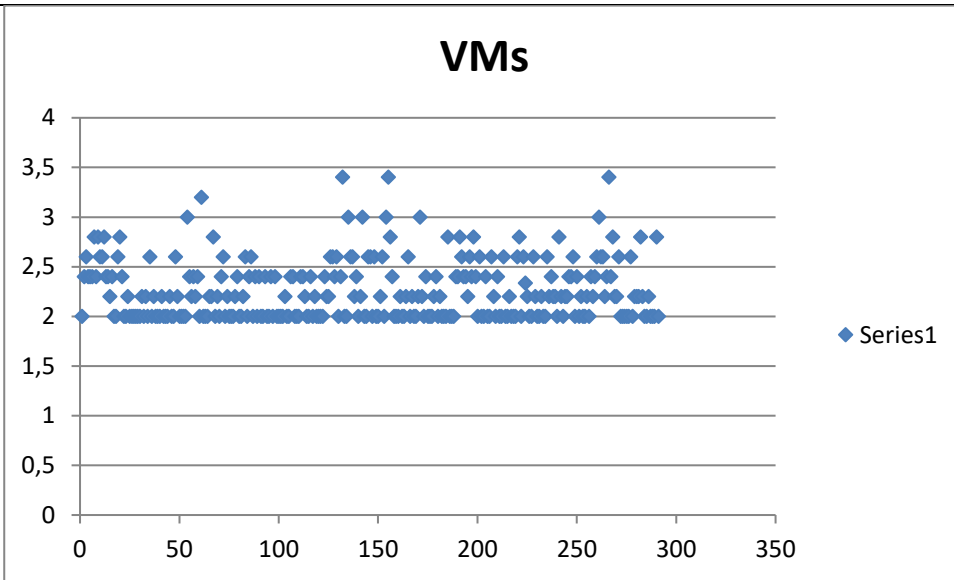


Figure 10-55 Step 4 number of running processes on VMs

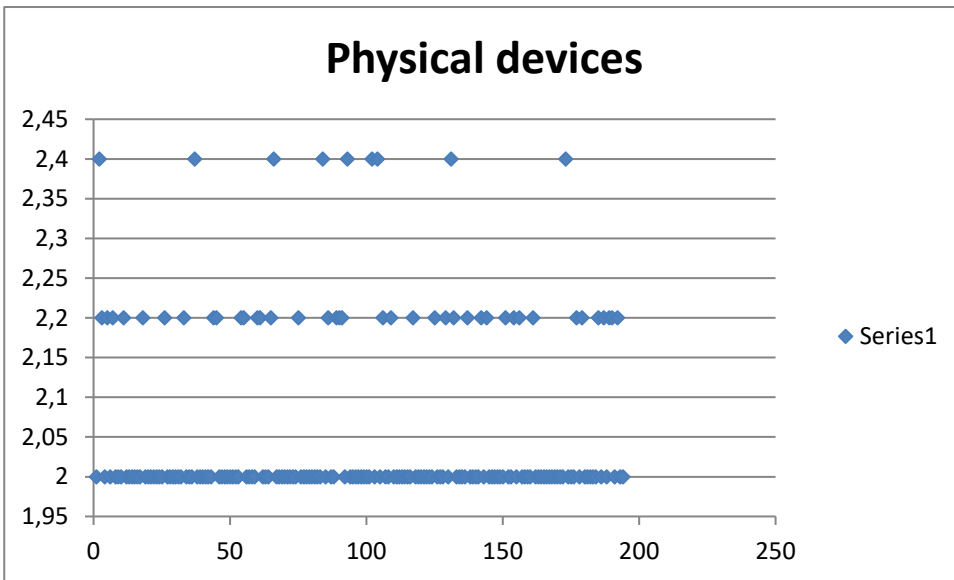


Figure 10-56 Step 4 number of running processes on physical devices

Storage metrics

The free disk space is shown for the various steps of the simulation in Figure 10-57, Figure 10-58, Figure 10-59, Figure 10-60, Figure 10-61, Figure 10-62, Figure 10-63 and Figure 10-64.

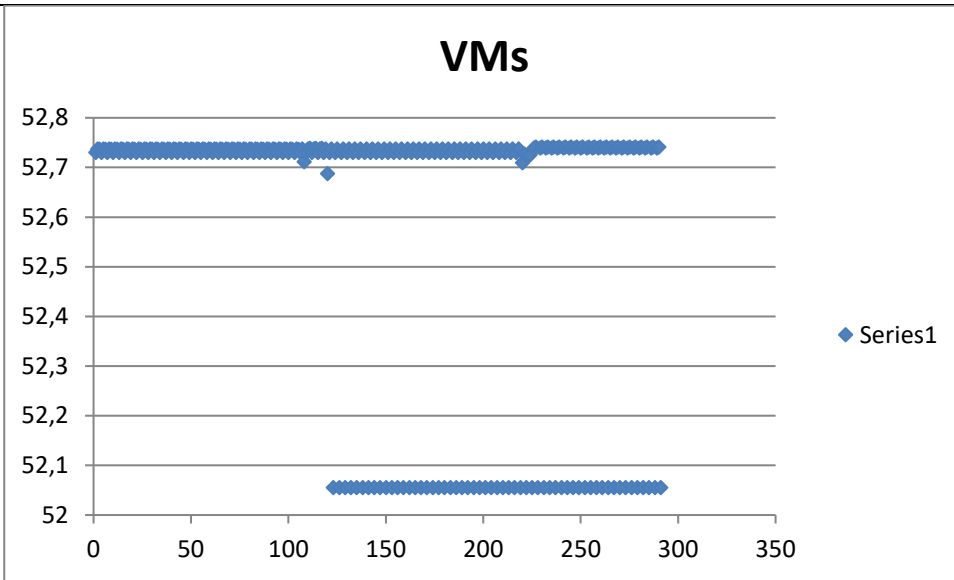


Figure 10-57 Step 1 free disk space in percentage for VMs

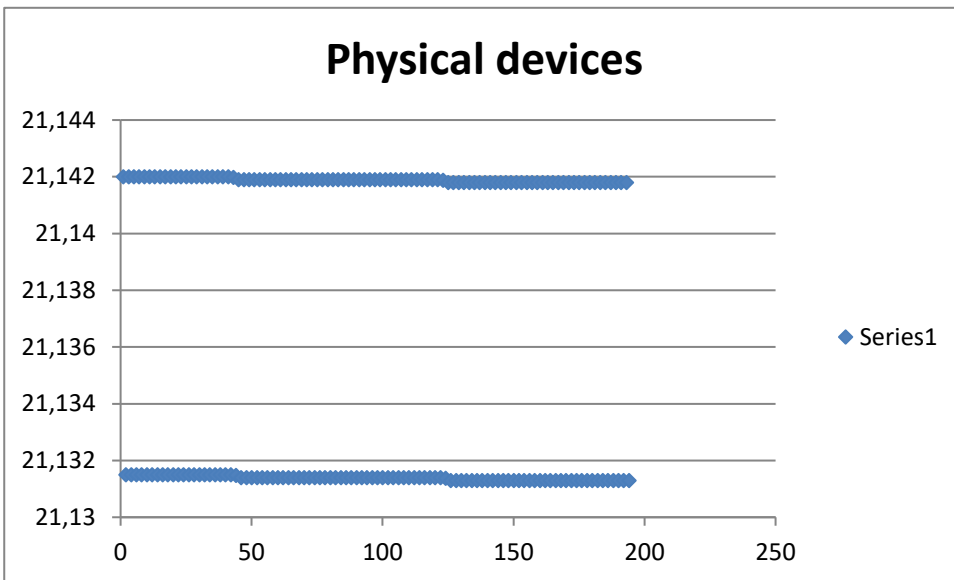


Figure 10-58 Step 1 free disk space in percentage for physical devices

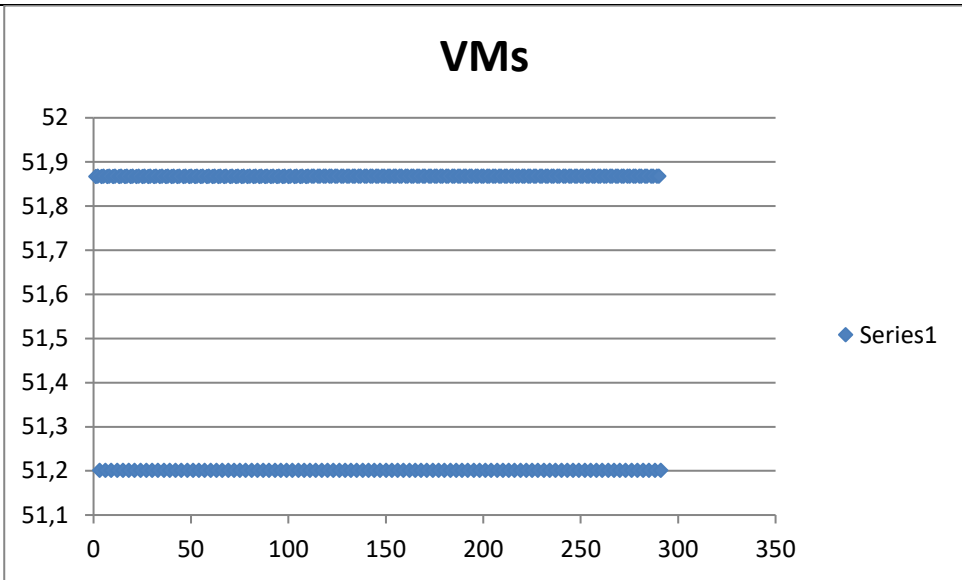


Figure 10-59 Step 2 free disk space in percentage for VMs

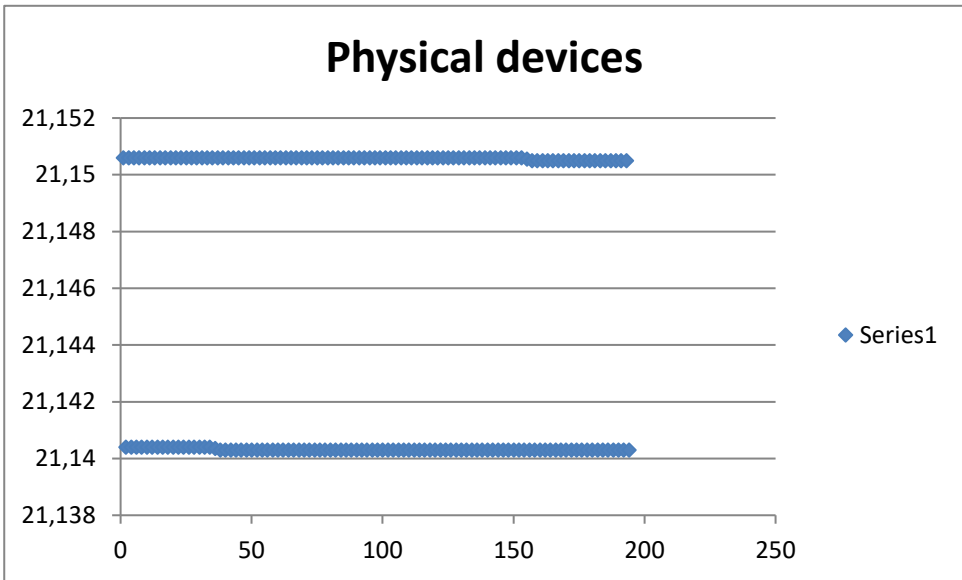


Figure 10-60 Step 2 free disk space in percentage for physical devices

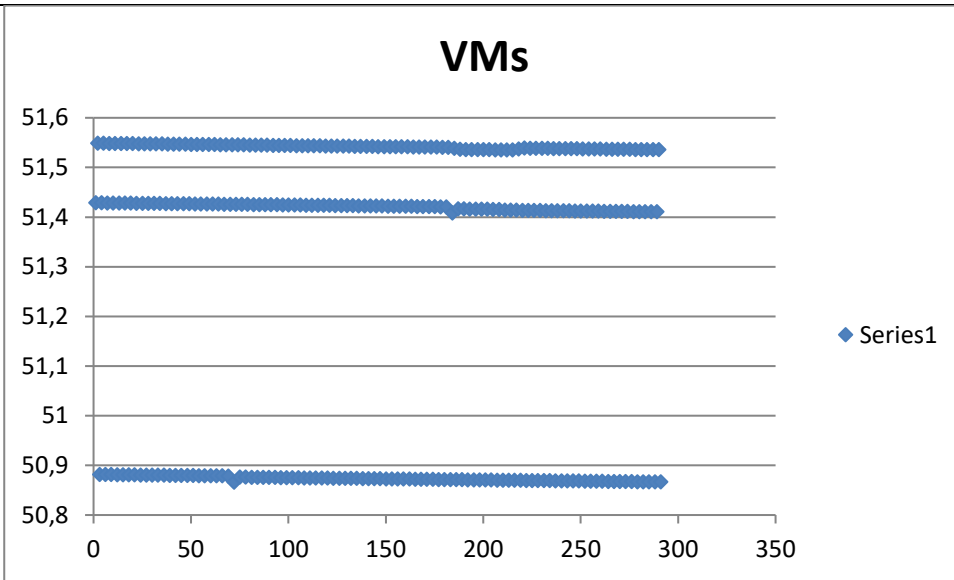


Figure 10-61 Step 3 free disk space in percentage for VMs

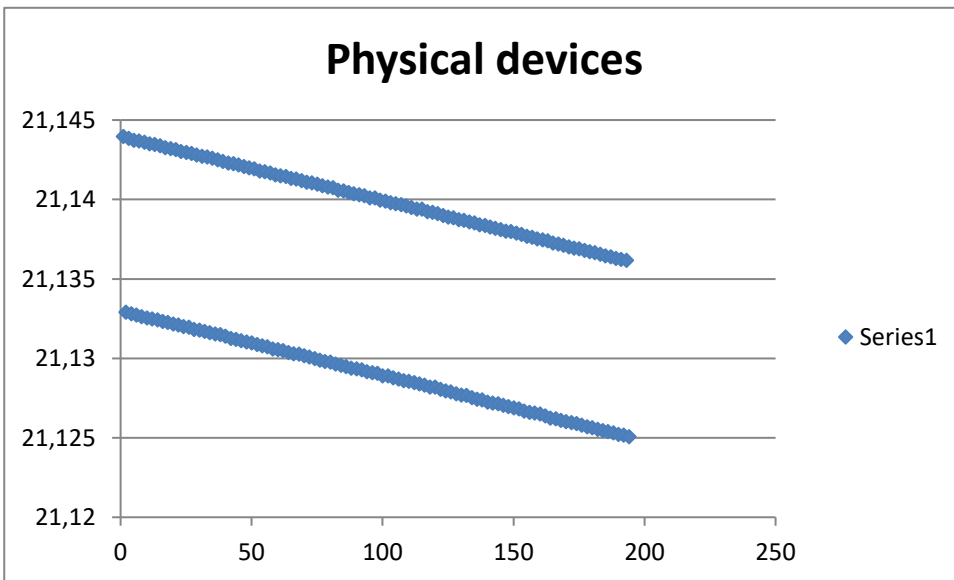


Figure 10-62 Step 3 free disk space in percentage for physical devices

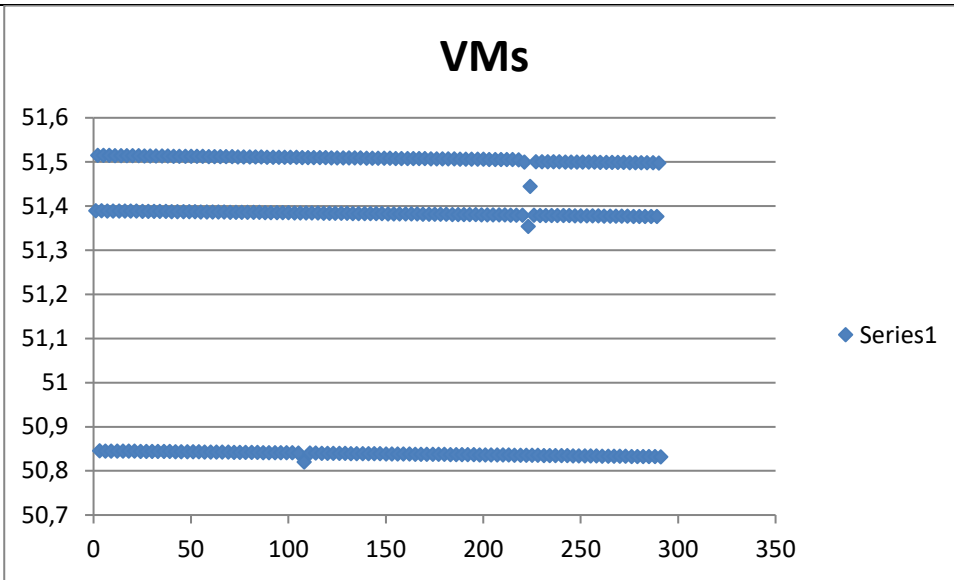


Figure 10-63 Step 4 free disk space in percentage for VMs

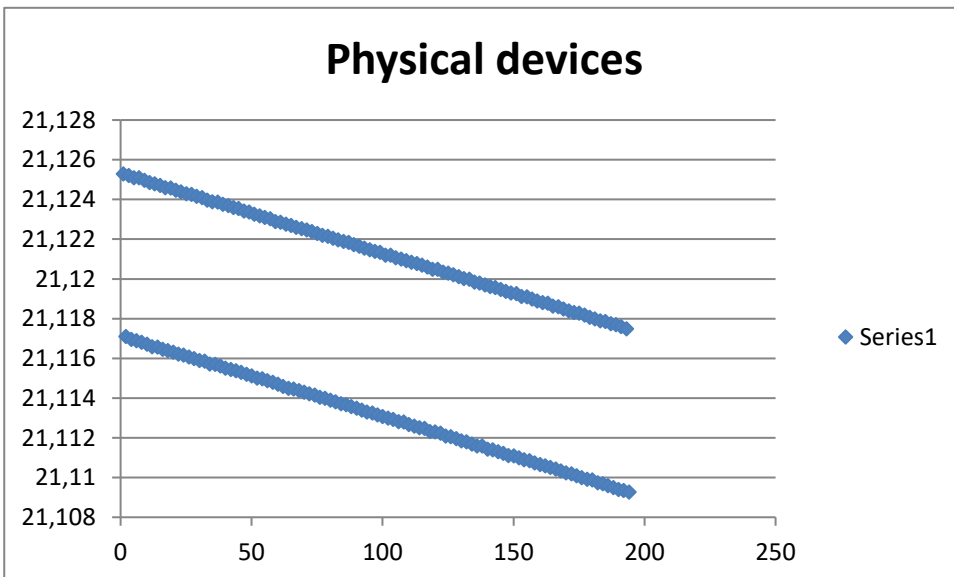


Figure 10-64 Step 4 free disk space in percentage for physical devices

Memory performance metrics

The metrics for the available memory during the various steps of the simulation is shown in Figure 10-65, Figure 10-66, Figure 10-67, Figure 10-68, Figure 10-69, Figure 10-70, Figure 10-71 and Figure 10-72.

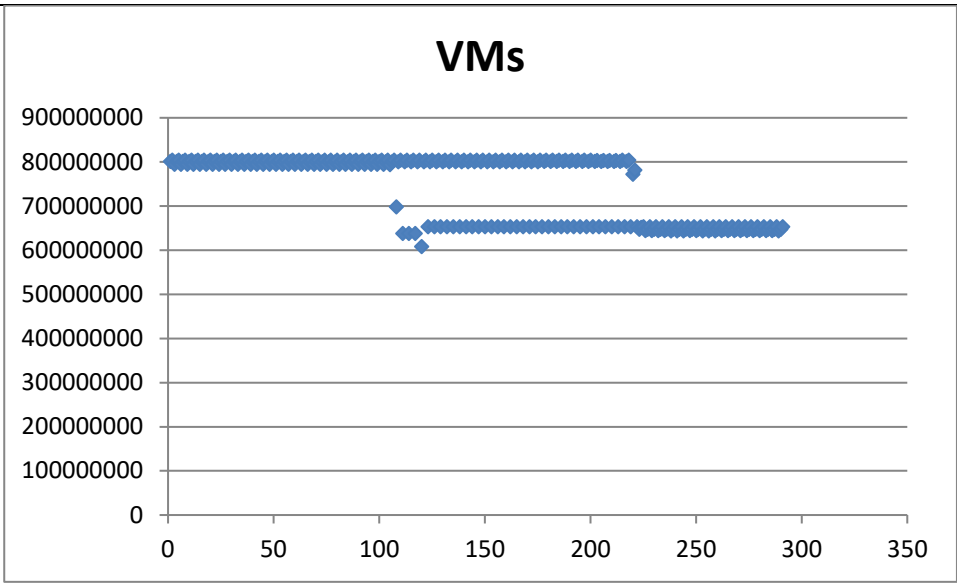


Figure 10-65 Step 1 available memory for the VMs

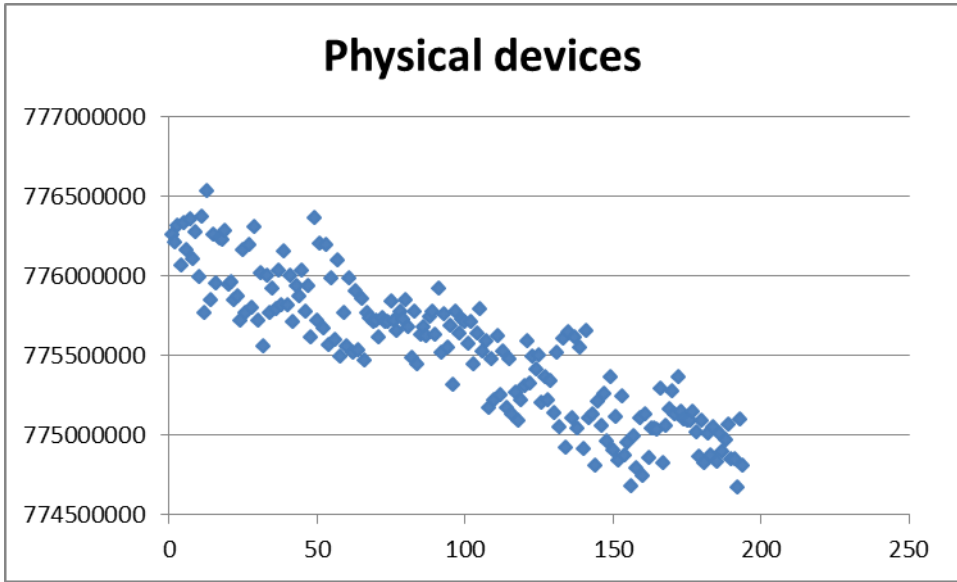


Figure 10-66 Step 1 available memory for the physical devices

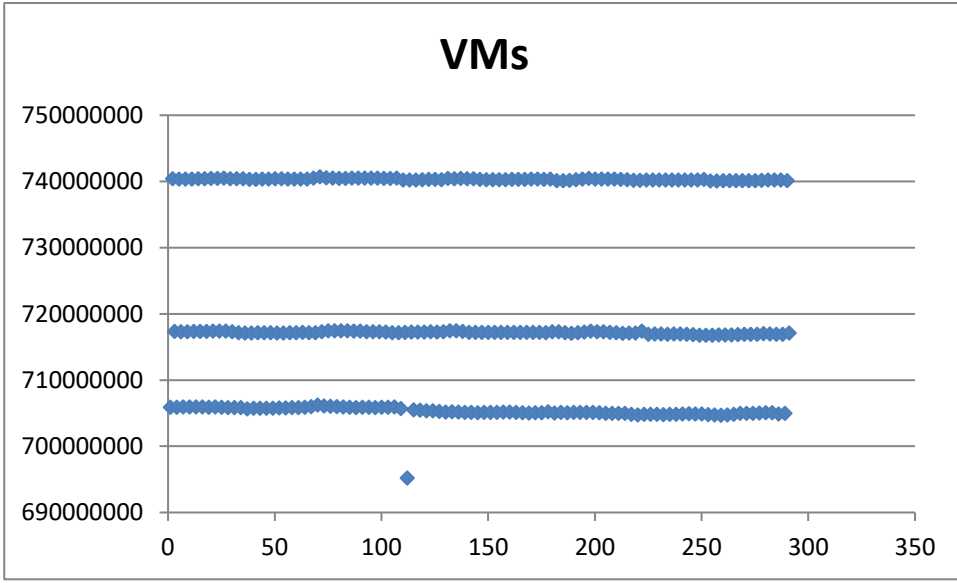


Figure 10-67 Step 2 available memory for the VMs

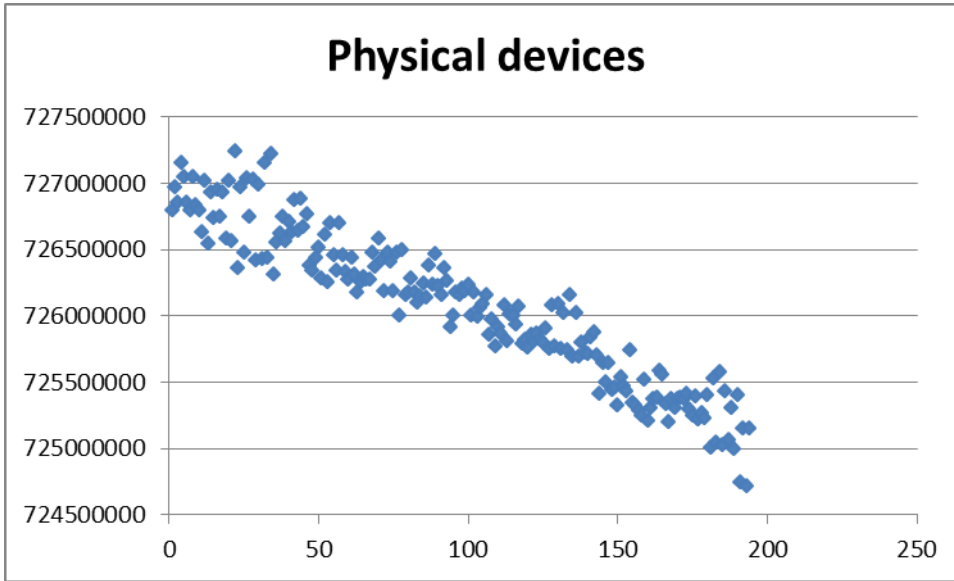


Figure 10-68 Step 2 available memory for the physical devices

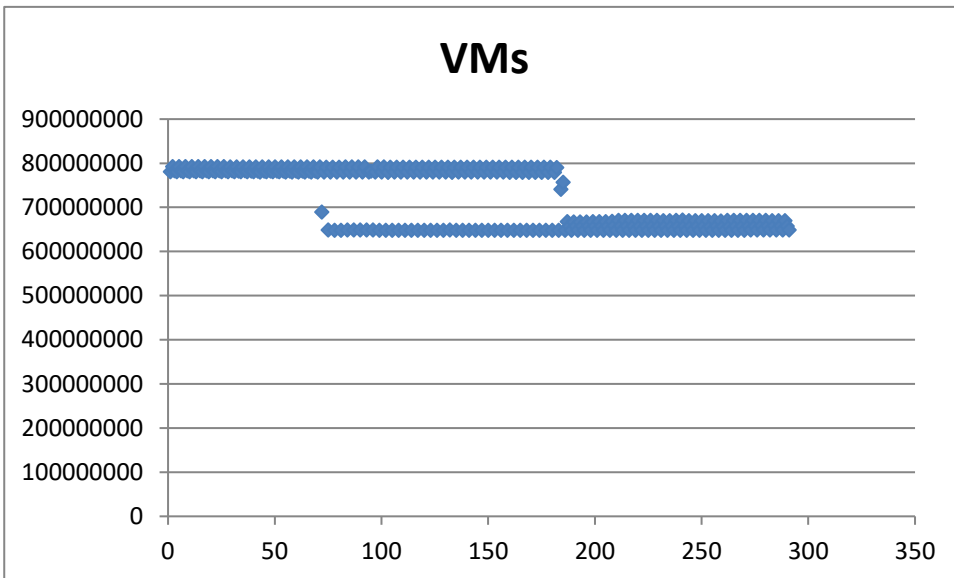


Figure 10-69 Step 3 available memory for the VMs

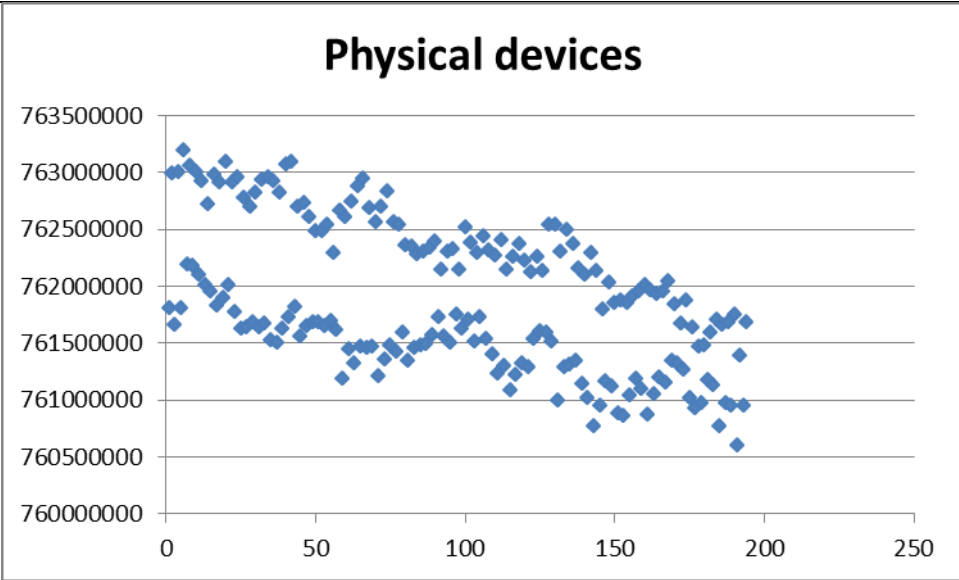


Figure 10-70 Step 3 available memory for the physical devices

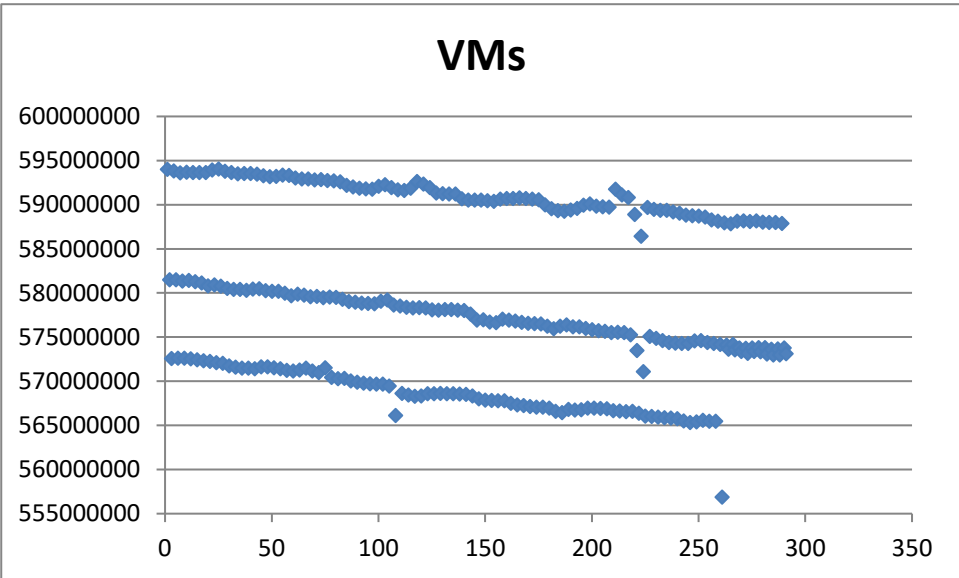


Figure 10-71 Step 4 available memory for the VMs

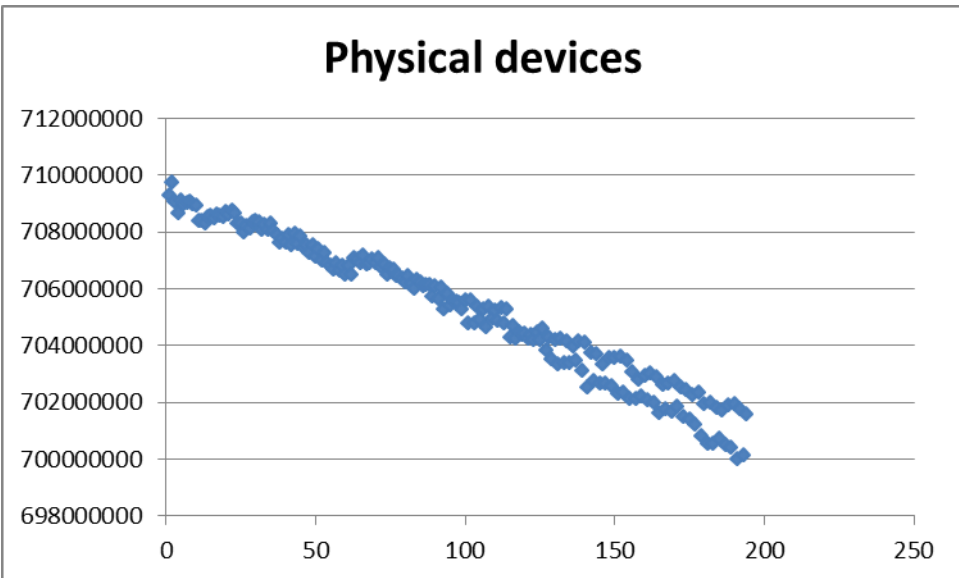


Figure 10-72 Step 4 available memory for the physical devices

The last metric for the memory category is free swap space in percentage. The free swap space in percentage for the various steps of the simulation is shown in Figure 10-73, Figure 10-74, Figure 10-75, Figure 10-76, Figure 10-77, Figure 10-78, Figure 10-79 and Figure 10-80.

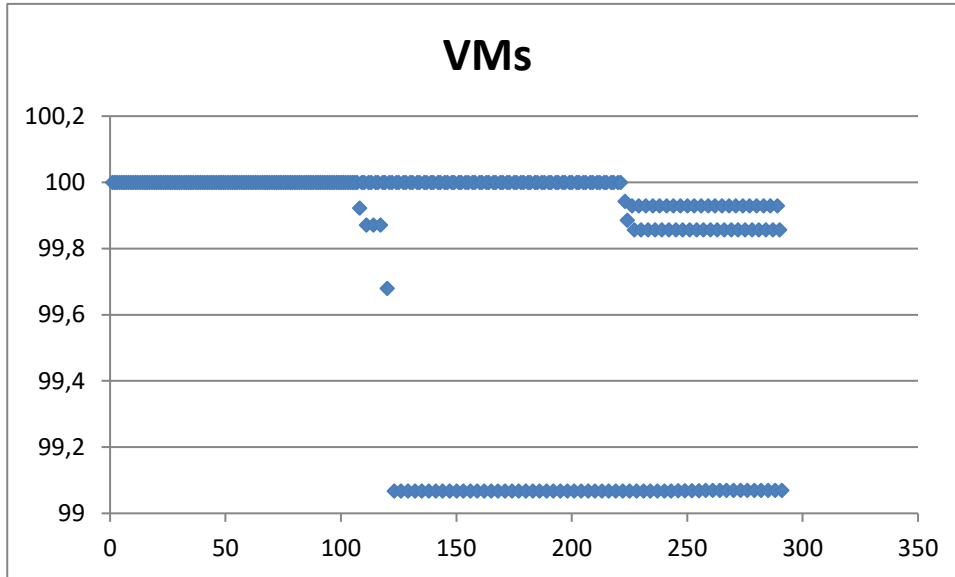


Figure 10-73 Step 1 free swap space in percentage for the VMs

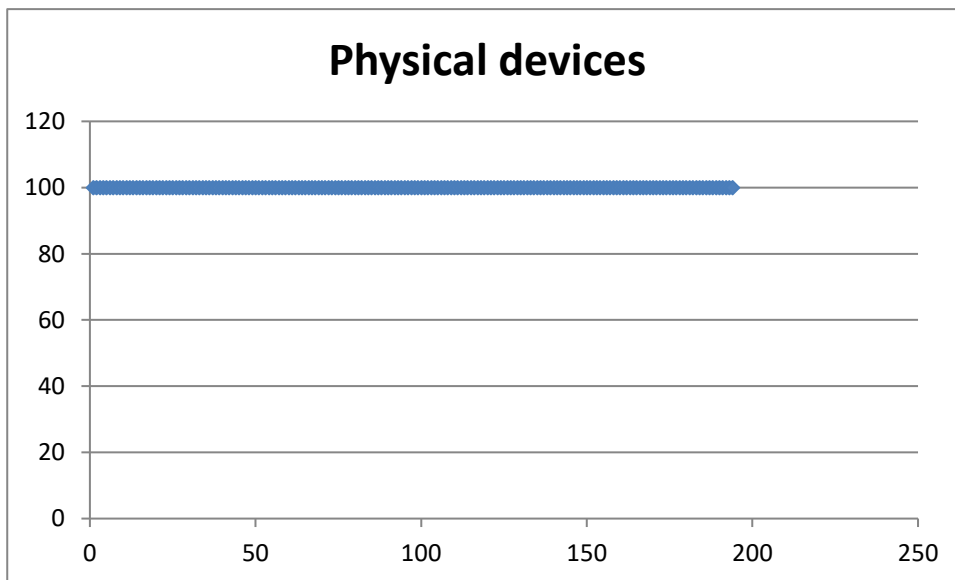


Figure 10-74 Step 1 free swap space in percentage for the physical devices

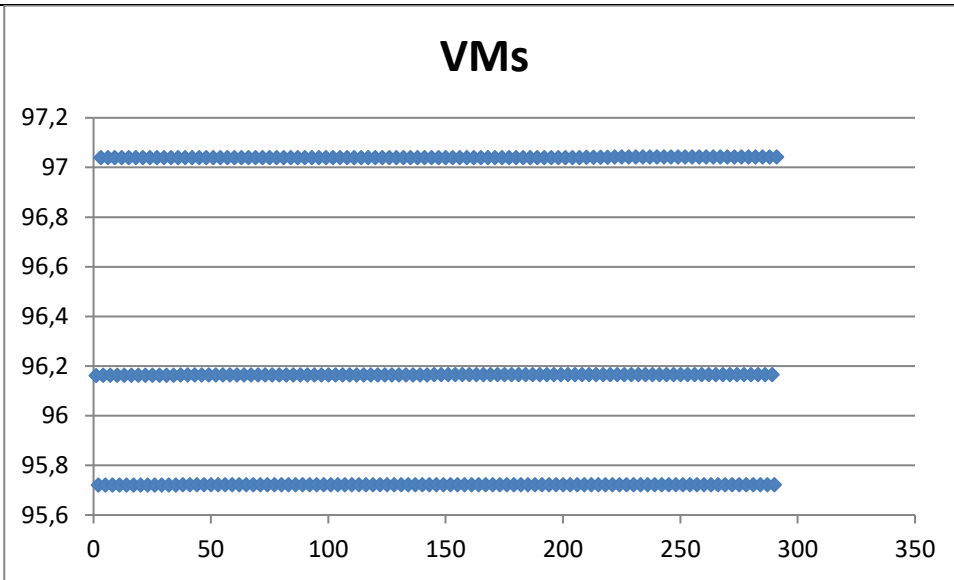


Figure 10-75 Step 2 free swap space in percentage for the VMs

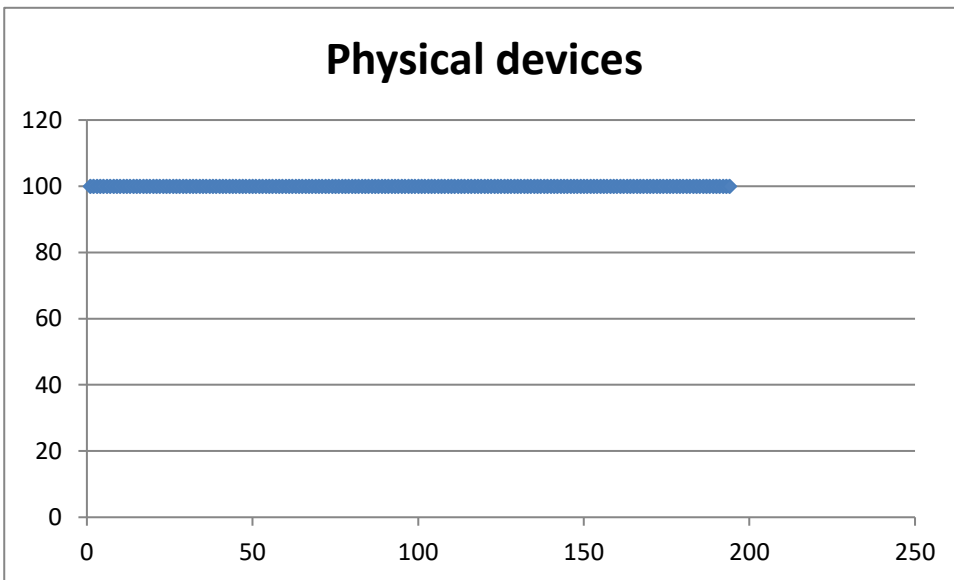


Figure 10-76 Step 2 free swap space in percentage for the physical devices

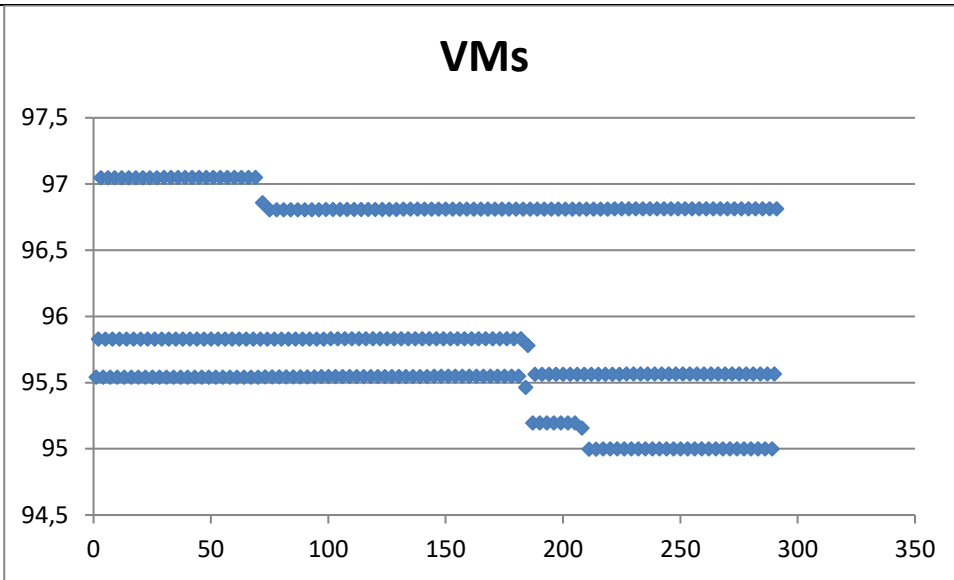


Figure 10-77 Step 3 free swap space in percentage for the VMs

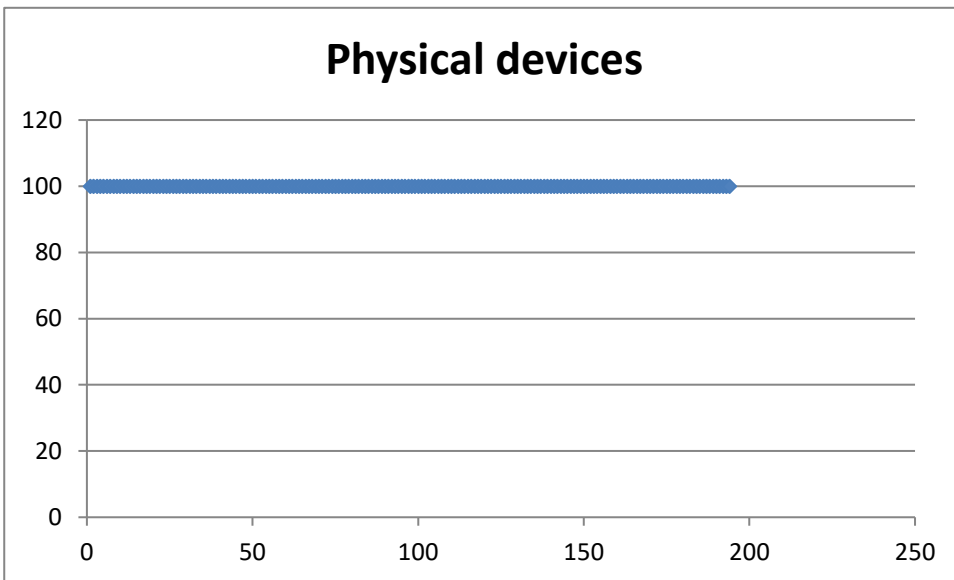


Figure 10-78 Step 3 free swap space in percentage for the physical devices

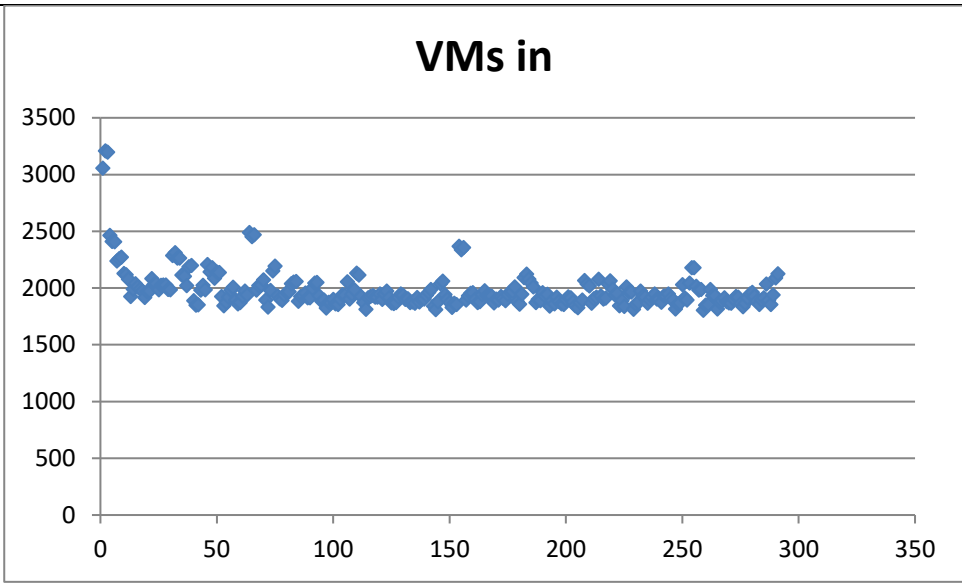


Figure 10-81 Step 1 incoming traffic for VMs

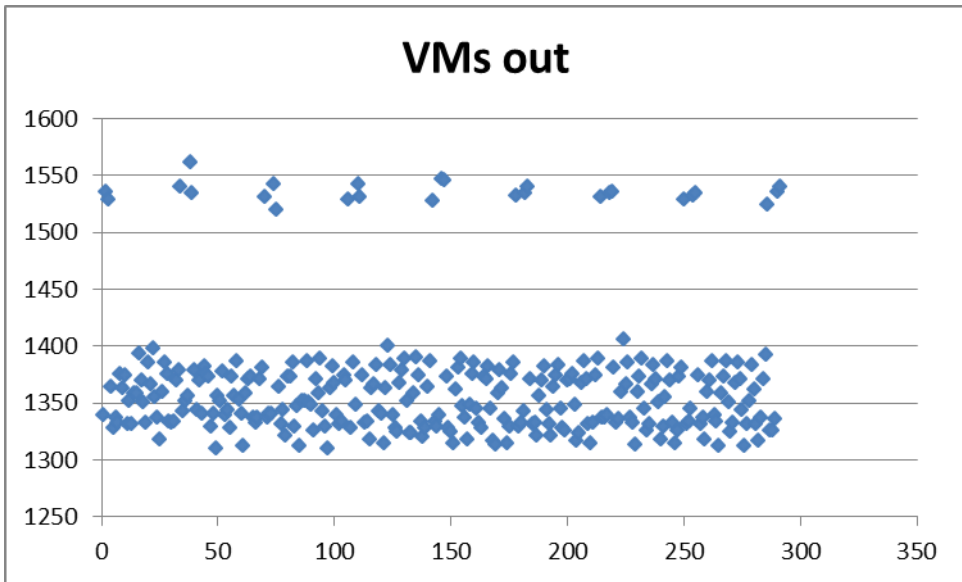


Figure 10-82 Step 1 outgoing traffic for VMs

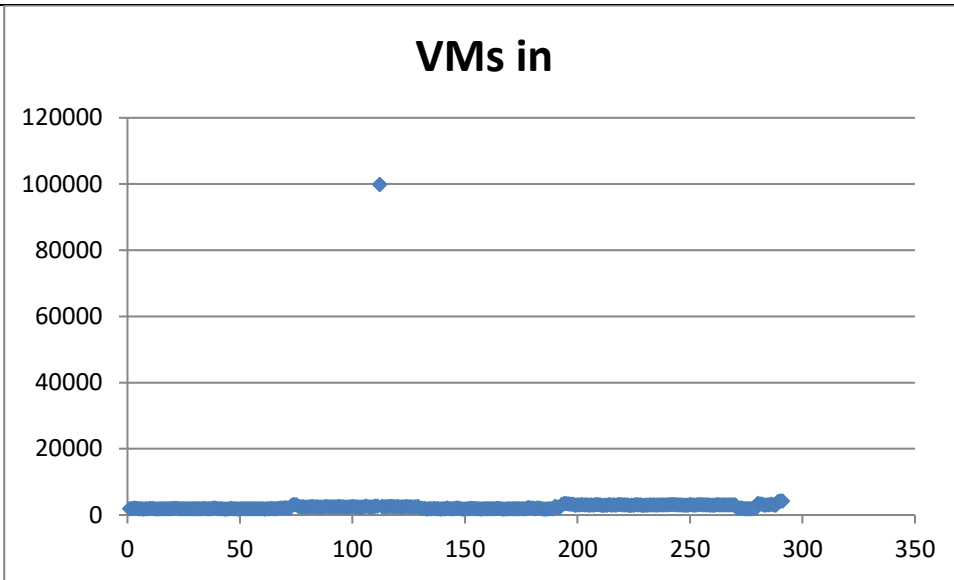


Figure 10-83 Step 2 incoming traffic for VMs

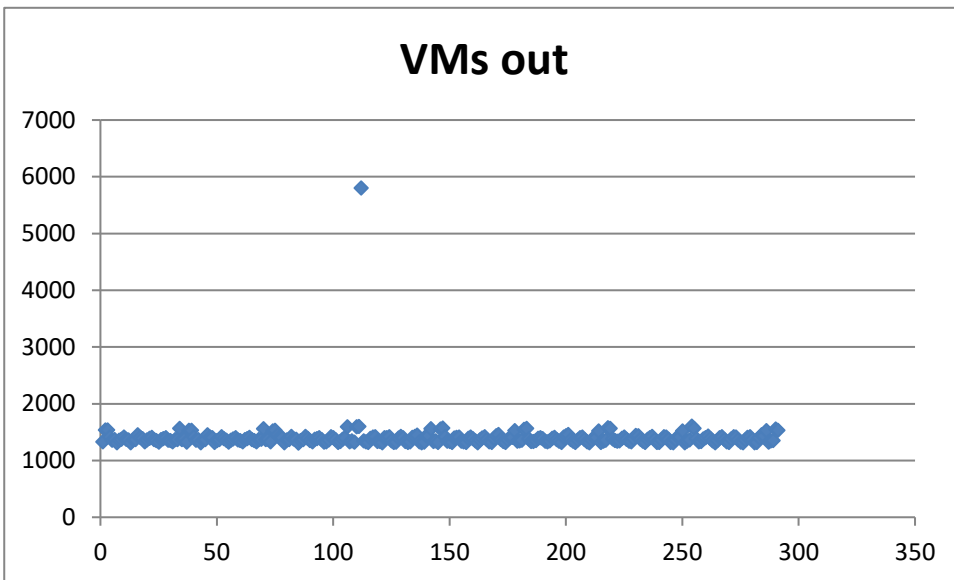


Figure 10-84 Step 2 outgoing traffic for VMs

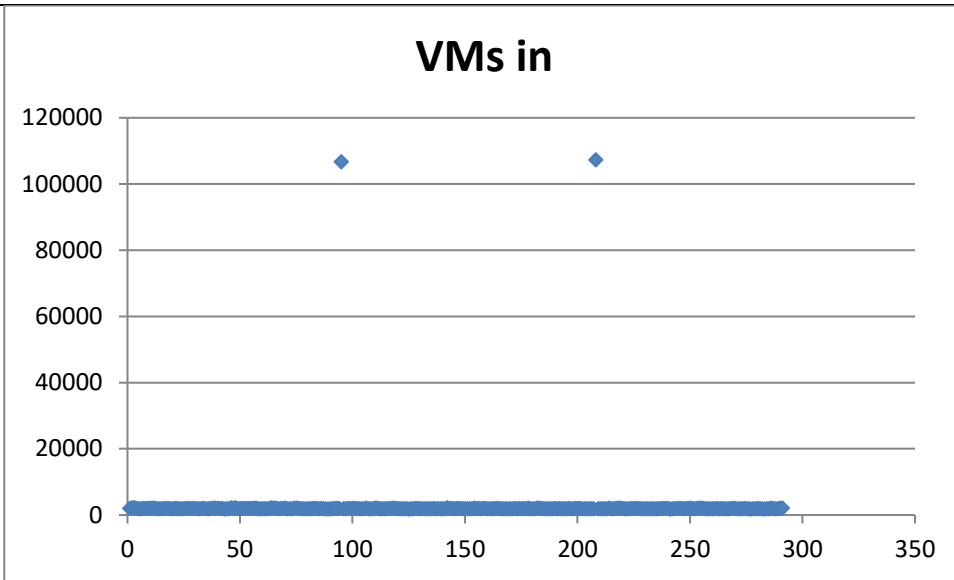


Figure 10-85 Step 3 incoming traffic for VMs

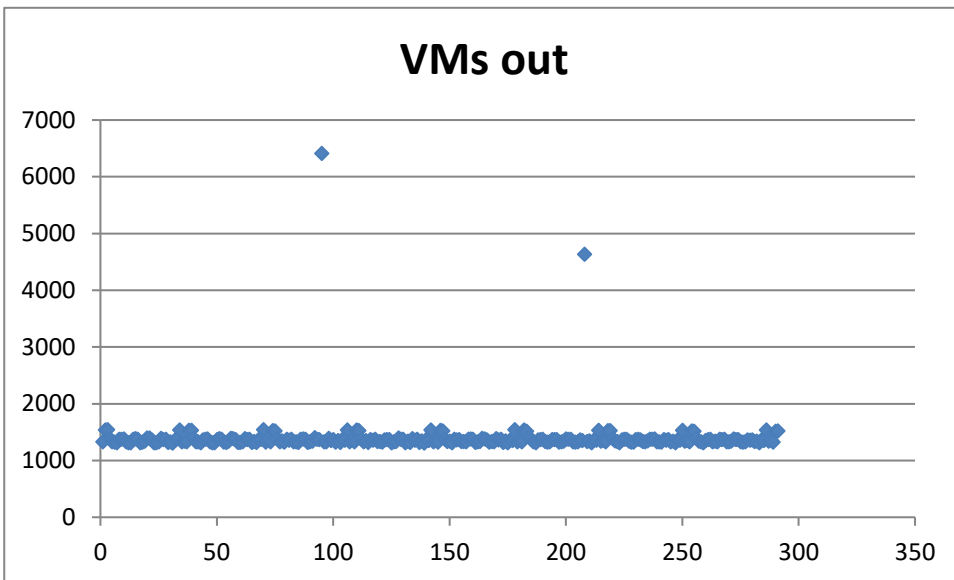


Figure 10-86 Step 3 outgoing traffic for VMs

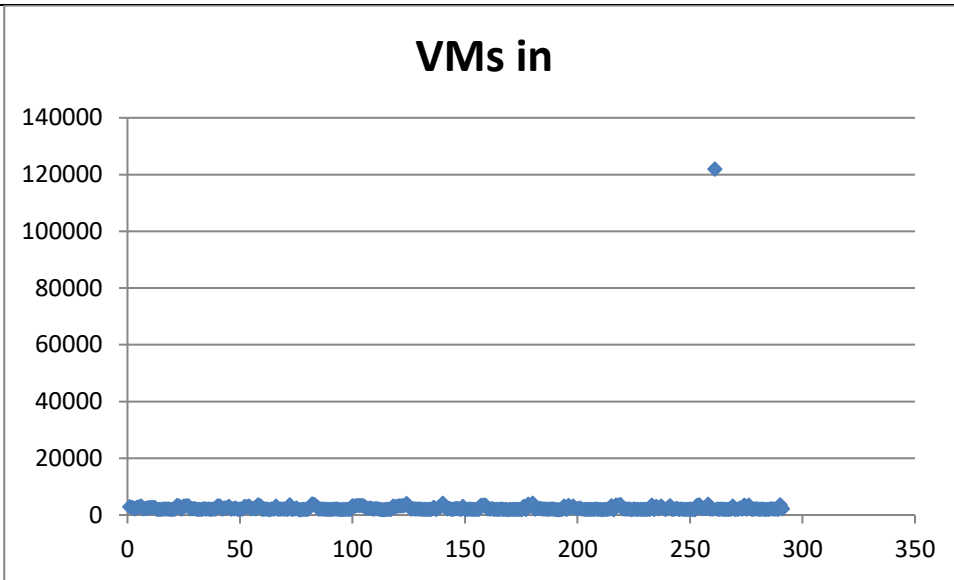


Figure 10-87 Step 4 incoming traffic for VMs

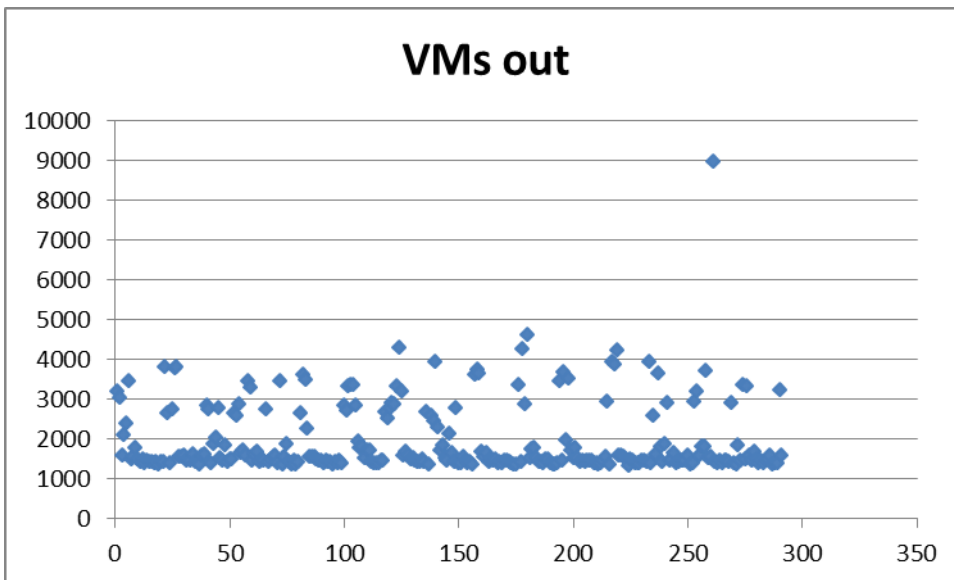


Figure 10-88 Step 4 outgoing traffic for VMs

The network traffic for the physical devices during the various steps of the simulation is also monitored. The network traffic for the physical devices during the various steps of the simulation is shown in Figure 10-89, Figure 10-90, Figure 10-91, Figure 10-92, Figure 10-93, Figure 10-94, Figure 10-95 and Figure 10-96.

Physical devices in

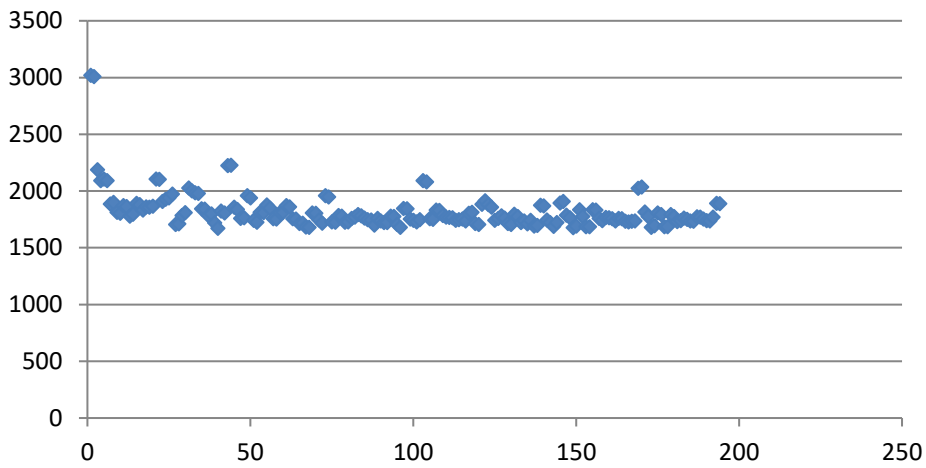


Figure 10-89 Step 1 incoming traffic for physical devices

Physical devices out

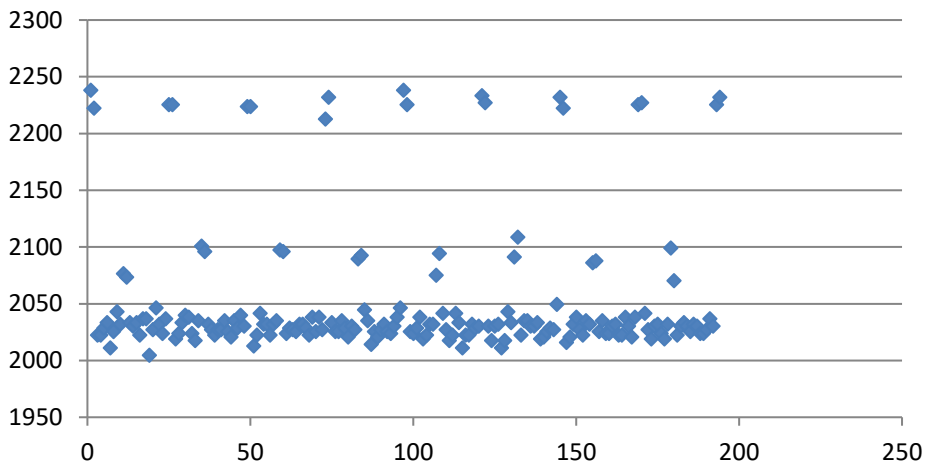


Figure 10-90 Step 1 outgoing traffic for physical devices

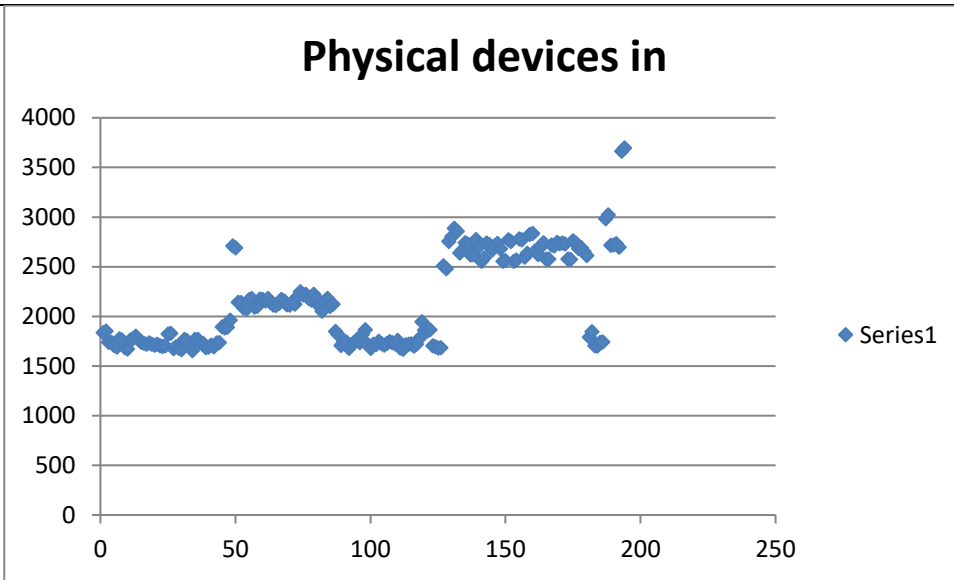


Figure 10-91 Step 2 incoming traffic for physical devices

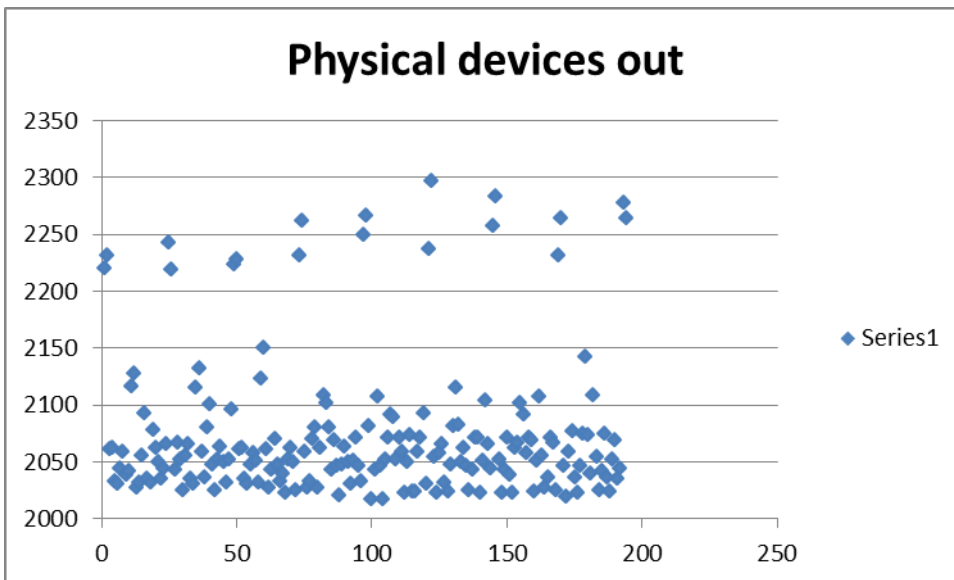


Figure 10-92 Step 2 outgoing traffic for physical devices

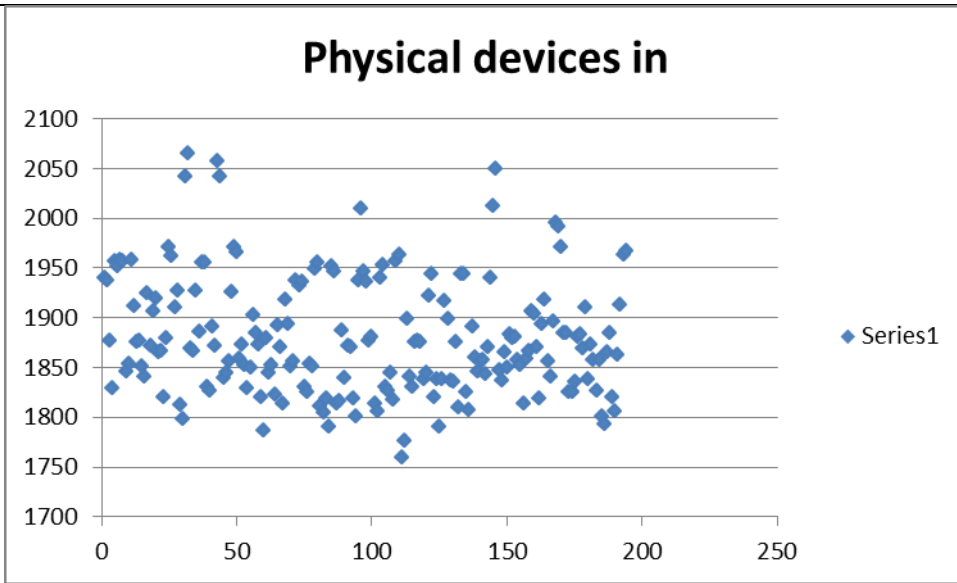


Figure 10-93 Step 3 incoming traffic for physical devices

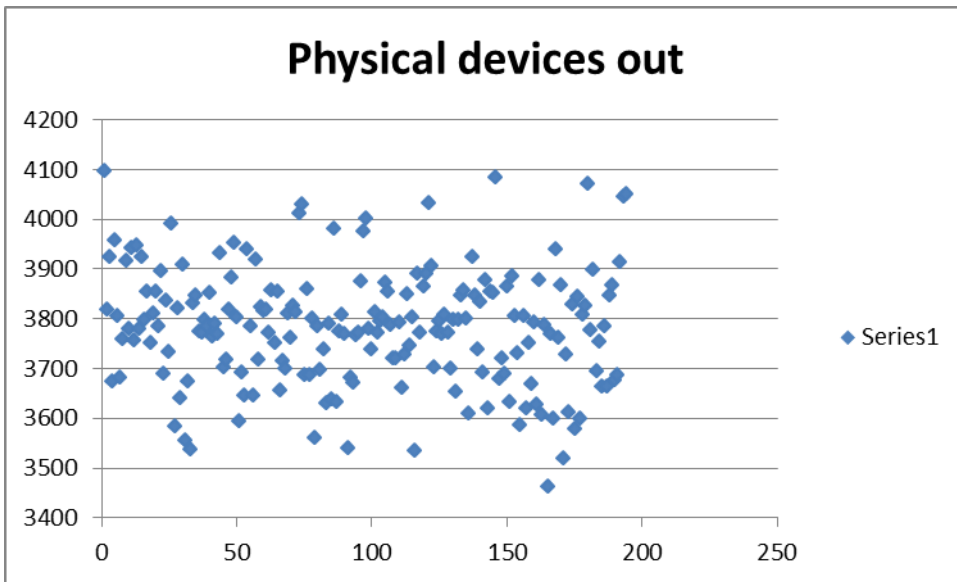


Figure 10-94 Step 3 outgoing traffic for physical devices

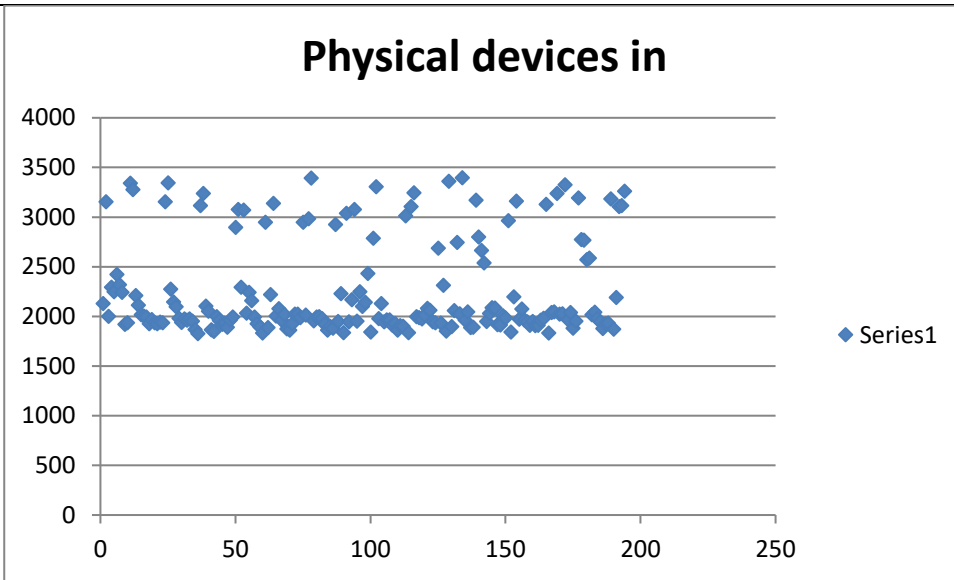


Figure 10-95 Step 4 incoming traffic for physical devices

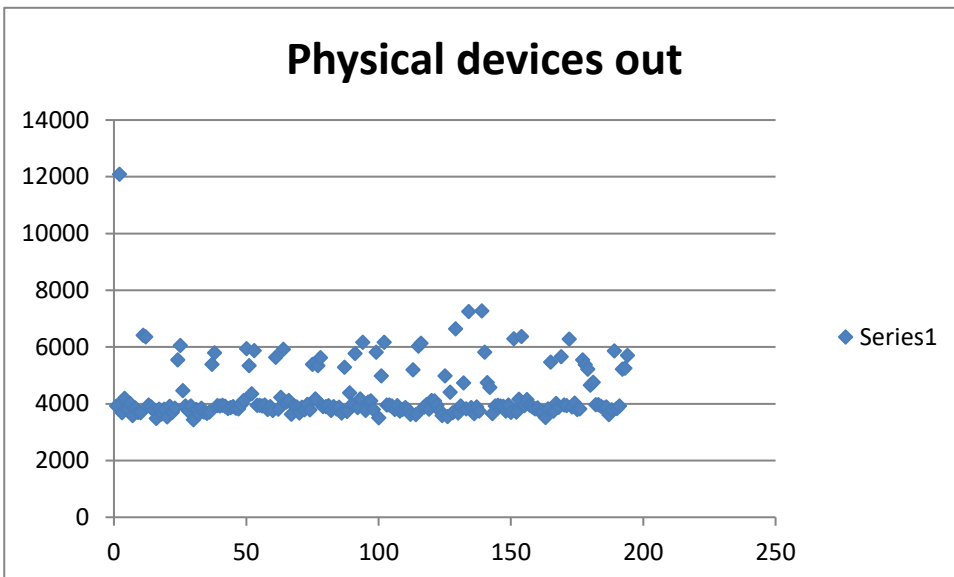


Figure 10-96 Step 4 outgoing traffic for physical devices