The Architecture of a Digital Forensic Readiness Management System

K. Reddy*, H.S. Venter

Information and Computer Security Architectures Research Group,
Department of Computer Science, University of Pretoria, South Africa

Email addresses: kreddy@cs.up.ac.za (K. Reddy) *Corresponding author, hventer@cs.up.ac.za (H.S. Venter)

Abstract

A coordinated approach to digital forensic readiness (DFR) in a large organisation requires the management and monitoring of a wide variety of resources, both human and technical. The resources involved in DFR in large organisations typically include staff from multiple departments and business units, as well as network infrastructure and computing platforms. The state of DFR within large organisations may therefore be adversely affected if the myriad human and technical resources involved are not managed in an optimal manner. This paper contributes to DFR by proposing the novel concept of a digital forensic readiness management system (DFRMS). The purpose of a DFRMS is to assist large organisations in achieving an optimal level of management for DFR. In addition to this, we offer an architecture for a DFRMS. This architecture is based on requirements for DFR that we ascertained from an exhaustive review of the DFR literature. We describe the architecture in detail and show that it meets the requirements set out in the DFR literature. The merits and disadvantages of the architecture are also discussed. Finally, we describe and explain an early prototype of a DFRMS.

Keywords: Digital forensic readiness, Management of digital forensic readiness, Digital forensic management system, Forensic readiness, Management of forensics, Organisational forensic readiness

1. Introduction

Information security is considered an important function in large organisations or enterprises (Ernst & Young 2010). Despite the recognition of its importance, a significant percentage of organisations still experience information security incidents (Richardson 2011). In many such cases it is desirable to perform a digital forensic (DF) investigation into such an incident to determine the identity of the perpetrator and/or the root cause of the incident. At times, depending on the nature of the organisation, it is imperative that such a DF investigation is performed. Militaries and banks are examples of organisations in which such DF investigations are often mandatory when such incidents happen. If the DF investigations that organisations undertake are to be effective, it is desirable for an organisation to be properly prepared prior to any such incidents. This kind of preparedness is known as digital forensic readiness (DFR), which is formally defined as a corporate goal that consists of those actions, technical and nontechnical, that maximise an organisation's ability to use digital evidence (Rowlingson 2004). In large organisations DFR requires the cooperation of people and departments across the organisation (Casey 2005)(Rowlingson 2004). It is for this reason that coordinated approaches to DFR are usually preferable to *ad hoc* approaches (Reddy & Venter 2009)(Reekie & von Solms 2006)(Yasinsac et al. 2003). Coordinated approaches are usually based on implementation frameworks or programmes (Trček et al. 2010)(von Solms et al. 2006).

In coordinated approaches to obtaining an acceptable level of DFR, the coordination of the myriad resources within a large organisation becomes a management challenge (Luoma 2006)(Grobler & Dlamini 2010). If management is unable to meet such a management challenge, it may not be possible to perform a DF investigation at all. Alternatively, it may only be possible to conduct the DF investigation in a less than optimal manner.

In this paper we address the challenge of managing the DFR function within a large organisation. We contribute by proposing a novel architecture for a digital forensic readiness management system (DFRMS). While there are a number of tools and systems within the domain of digital forensics, there is no system that is dedicated to the management of DFR. The DFRMS architecture that we put forward here is therefore novel. The architecture makes use of existing technologies and our contribution does not lie in advancing the state of any of these technologies. Rather, we contribute by 1) proposing the concept of a DFRMS itself, 2) presenting detailed functional requirements for a DFRMS, and 3) providing an architecture that combines existing technologies to achieve a working DFRMS. The functional requirements for the DFRMS are drawn from an analysis of the literature on DFR but also include our own contributions. In addition to this, we offer a proof-of-concept prototype system that is based on the architecture.

The remainder of the paper is structured as follows: Section 2 discusses related work. Section 3 contains the requirements analysis for the DFRMS. The architecture for the DFRMS and our prototype are then presented in Section 4. A discussion of the architecture and an example scenario follows in Section 5. Section 6 contains the conclusion, and finally future work is discussed in Section 7.

2. Related Work

Our review of the literature did not reveal any software or tools that were dedicated exclusively to the management of digital forensic readiness (DFR). We therefore use this section to discuss software or tools that are related to, but are not dedicated to, the management of DFR. It should be noted that we do not discuss DF tools that are used for analysis during investigations. We identified three types of software or existing technology that are directly related to the management of DFR. These are: intrusion detection systems, security event management software, and incident management software. A discussion of each type follows.

2.1 Intrusion Detection Systems

Intrusion detection systems (IDSs) are related to the management of DFR because they enable the monitoring of events on computers and networks. As will be shown in the next section, the monitoring of events is required for DFR. IDSs monitor and analyse the events occurring in a computer system or network and raise an alarm if an intrusion is deemed to have occurred.

Next, we look at security event managers.

2.2 Security Event Managers

Security event managers (SEMs), security information managers (SIMs), and security information and event managers (SIEMs), are all names that are given to security event management software or appliances. All three of these terms are synonymous and the software or appliances typically perform the same function regardless of name (Swift 2006). For the sake of consistency, we use the term *security event managers* (SEMs). SEMs were developed as a result of the inability of IDSs to effectively filter real threats from false alarms and normal system activity (Mehdizadeh 2005). Although they are relatively new, SEMs constitute one of the fastest growing segments of the information security technology market (Deloitte 2010).

SEMs are related to the management of DFR because, like IDSs, they also monitor events or data from multiple sources. SEMs, however, usually perform additional tasks. Swift (2006) lists four important functions that are performed by all SEMs:

- Log Consolidation. Centralised logging to a server is used to consolidate logs.
- Threat Correlation. Artificial intelligence techniques are applied to sort through multiple logs and log entries in order to identify attackers or threats.
- Incident Management. Workflows are defined and stored to determine what happens once a threat is identified. Each such workflow is a suggested path from the initial identification of a threat to the threat's containment and/or eradication. Incident management includes: notification; trouble ticket creation; automated responses, such as the execution of scripts; and lastly, response and remediation logging.
- Reporting. Reports on operational efficiency and effectiveness can be produced, as well as reports tailored for regulatory compliance, ad-hoc enquiries and forensic investigations.

IDSs are different to SEMs as they do not typically perform all of the tasks listed by Swift above, moreover, SEMs may in fact make use of IDSs to perform their functions (Mehdizadeh 2005).

We now discuss incident management software.

2.3 Incident Management Software

Within the context of information security (IS), an incident may be defined as an identified occurrence of a system, service or network state indicating a possible breach of IS policy or failure of safeguards, or a previously unknown situation that may be relevant to security (Kostina et al. 2009). Incidents in an organisation are usually reported to a central point, normally a help desk or service desk (Gupta et al. 2008). Incident management software assists the organisation by facilitating the incident management process, which consists of, *inter alia*, incident detection, classification, analysis/diagnosis and finally repair and recovery (Gupta et al. 2008)(Metzger et al. 2011). It does this by controlling the workflow involved in the incident management process. In order to do this the software also contains "incident records, escalation rules, information about customers and end users, and information about configuration items" (Jäntti 2009).

Since both SEMs and incident management software suggest some or all of the workflow in the incident management process, there is an overlap of functionality between the two. SEMs, however, only deal with IS incidents while most dedicated incident management software deals with IT incidents in general. It is of course up to the organisation itself to determine whether they prefer to deal with IS incidents separately or whether they prefer to adopt a more unified approach.

In the following section, we examine the requirements for a DFRMS.

3. Requirements Analysis

In order to determine the requirements of a DFRMS we examined the literature on DFR. DFR is a relatively new concept in the forensics literature, having first been espoused by Tan (2001). Apart from the seminal work of Tan, perhaps the single greatest contribution to the concept of DFR has been that of Rowlingson (2005). A number of requirements therefore stem from Rowlingson's work. In this section we discuss each of the requirements that are identified by these and other authors. The requirements are discussed below under the headings of 'Monitoring', 'DFR Information' and 'Cost'.

3.1 Monitoring

Tan's initial work identified the logging of network and host activity as being important. Logging is important as it is first necessary to have a record of activity on a network or host in order to be ready to perform a DF investigation. A DFRMS should, therefore, have a monitoring component that is capable of monitoring and logging the activity across a range of hardware and software platforms within an organisation's IT infrastructure. Tan (2001) also recommends that log data should not only be kept securely but that it should also be digitally signed or encrypted so as to prevent tampering. This is important from a DF point of view because the evidentiary value of data diminishes as the likelihood of it being tampered with increases.

A DFRMS, like any system that performs monitoring, must also be able to distinguish between the various elements that it monitors. It should be possible, for example, for a user of DFRMS to choose an element being monitored and to determine that the particular element is a firewall as opposed to a router. A monitoring capability also requires that a DFRMS has the ability to extract the detail of the event messages that it receives. A DFRMS should, for example, be able to distinguish whether a firewall is signalling a port scan or a flooding attack.

The detection of events that constitute a potential or actual incident should be automated and an alarm raised whenever the events are detected (Grobler et al. 2010). Monitoring, however, is only a single dimension within DFR. We now look at other requirements, which we discuss collectively under the title 'DFR Information'.

3.2 DFR Information

A DFRMS needs to be able to store escalation rules, configuration, and any other information that is required for DFR. In this sense, it is analogous to incident management software. The information stored by a DFRMS can be used in two cases: firstly, by the DF personnel who conduct investigations, and, secondly, by incident responders. An example of the first case is a procedure that describes the retrieval of information from a desktop computer. An example of the second case is an escalation procedure that prescribes the steps that must be taken when suspicious activity is noted. Apart from these two general cases, information that pertains to the operations of the DF function must also be stored – for example, information on the DF training of DF personnel.

The importance of monitoring and logging has been mentioned. It is, however, important that, before use, hardware and software should first be configured to log activity adequately (Casey 2005). A DFRMS must therefore contain the necessary configuration procedures for the benefit of the staff that are responsible for configuring hardware and software. In a large organisation, such staff may be from the IT, IT security, or DF departments of the large organisation.

In this paper, and indeed in the architecture itself, we differentiate between incident response and DF teams. We define incident response teams as those individuals that respond the instant an incident is detected. They may be from departments completely outside of DF. Indeed Grobler & Louwrens (2006), and Luoma (2006), recommend that teams should be multidisciplinary and should include, for example, staff with legal expertise. A system administrator who is responsible for stopping an operating system process from executing in response to an incident, is an example of an incident response team member. In contrast, we define DF teams as consisting of individuals with specialised DF skills that are involved in the investigation of incidents. Yasinsac & Manzano (2001), as well as Lamis (2010), note that DF teams and incident response teams should both be defined *a priori*, that is, in anticipation of an incident and not after an incident occurs. If this is not done, valuable time and evidence may be lost while teams are constituted. A DFRMS should therefore be able to store data about such teams for easy accessibility and for automated notification should incidents occur.

Yasinsac & Manzano (2001), as well as Chen et al. (2005) and Rowlingson (2005), also discuss the importance of training the incident response and DF teams. Ahmad et al. (2012) further note the importance of ensuring that lessons from previous incidents form part of this training. Training is also important as untrained staff may compromise or lose evidence through their actions. Consequently, we believe a DFRMS should also have the capacity to record the training undertaken by team members. This will allow managers to determine if teams possess the requisite skills.

Table 1: DFRMS Requirements from the literature

Requirement	Citation / Reason
1. Monitor or log network and host activity.	Tan 2001, p.2
2. Secure storage of logs	Tan 2001, p.20
3. Intrusion detection system	Tan 2001, p.3
4. Distinguish whether hardware or software elements are being monitored	Follows from requirement 1.
5. Automated alarm upon detection of potential or actual incident	Grobler et al. 2010, p.678 and also follows from requirement 3
6. Configuration procedures for monitoring and logging	Casey 2005, p.259 and also follows from requirement 1
7. Investigative teams (DF teams) and incident response teams descriptions	Yasinsac & Manzano 2001, p.292
8. Training requirements and training received	Yasinsac & Manzano 2001, p.292; Chen et al. 2005, p.6; Rowlingson 2005, p.10; Ahmad et al. 2012
9. Business process descriptions	Rowlingson 2005, p.5
10. Organisational DF policies and policies related to DFR	Yasinsac & Manzano 2001, p.292; Rowlingson 2005, p.8; Taylor et al. 2007
11. Suspicion policy	Rowlingson 2005, p.9
12. Law enforcement contact policy	Danielsson & Tjøstheim 2004, p.420; Lamis 2010, p.182
13. Escalation procedure	Yasinsac & Manzano, 2001, p.292; Rowlingson 2005, p.9
14. Incident response procedure	Casey 2005, p.259; Chen et al. 2005, p.4; Rowlingson 2005, p.9
15. Law enforcement contact procedure	Danielsson & Tjøstheim 2004, p.420
16. Organisational structure and staff involved in DFR and incident response	Follows from requirements 7, 8, 10, 11, 12, 13, 14, 15, and Luoma 2006; Grobler & Louwrens 2007, p.20

According to Rowlingson (2005), in order to take a risk-based approach to DFR, it is vital to "define the business scenarios that require digital evidence". Hence, a DFRMS should be able to store descriptions of the business processes DF is involved in. If the business process descriptions are kept up to date, perhaps through a formal system of updates, then DFR management may be in a better position to react to changes in business processes where necessary. In this paper we use the definition of a business process given by Hammer and Champy, which was cited in Lindsay et al. (2003), namely: a set of partially ordered activities intended to reach a goal.

Organisational policy, such as an overall forensics policy, should form the basis for DFR (Yasinsac & Manzano 2001)(Rowlingson 2005)(Taylor et al. 2007). Thus, the staff involved with DFR should have access to the necessary policies to inform their decision making. A DFRMS system should thus be capable of storing the policies that relate to DFR. Besides an overall organisational forensics policy, Rowlingson (2005) advocates a suspicion policy that can be used by monitoring staff to determine what might constitute suspicious behaviour. A suspicion policy should therefore also be included in a DFRMS. The nature of some incidents requires that they be reported to law enforcement for legal or ethical reasons (Lamis 2010), for example, child pornography. A policy offering guidance on when to contact law enforcement should exist (Danielsson & Tjøstheim 2004) and should be stored on a DFRMS.

If a severe incident is detected, or if something seems suspicious per the suspicion policy, an escalation procedure document is necessary. The escalation procedure document offers guidance on how to escalate an incident and the staff to whom it should be escalated to (e.g. an email or telephone notification to the IT Security Manager) (Yasinsac & Manzano 2001) (Rowlingson 2005). An escalation procedure document should therefore be included in a DFRMS.

If an initial investigation into suspicious behaviour yields evidence of an incident this should initiate an incident response procedure (Casey 2005)(Chen et al. 2005)(Rowlingson 2005). An incident response procedure is thus required in a DFRMS. An incident response procedure differs from an escalation procedure because an incident response procedure must be followed every time an incident is detected. By contrast, an escalation procedure is only necessary for those incidents that require escalation.

If the law enforcement contact policy mentioned earlier indicates that it is necessary to contact law enforcement, then a specific law enforcement contact and handover procedure should exist to facilitate this contact. This procedure should be available on the DFRMS, and it should specify report formats, etc. (Danielsson & Tjøstheim 2004) that need to be used for contacting law enforcement personnel.

It is highly likely that the policies and procedures mentioned above will reference staff in the various departments of an organisation by their job titles. For example, a procedure may dictate that the Chief Forensics Officer be notified in the case of incidents deemed to be of the greatest severity. In the same manner, the documentation that defines the DF and incident response teams may also make reference to staff by their job titles. The organisational hierarchy of an organisation should therefore be explicitly defined (Grobler & Louwrens 2007) and include the contact details, job titles and responsibilities for all staff in the hierarchy. This hierarchy should be stored in a DFRMS. It will enable employees using the DFRMS to contact the correct staff as quickly as possible.

As previously mentioned, the requirements that have been discussed are summarised in Table 1 above. In the sub-section that follows we discuss the topic of cost.

3.3 Cost

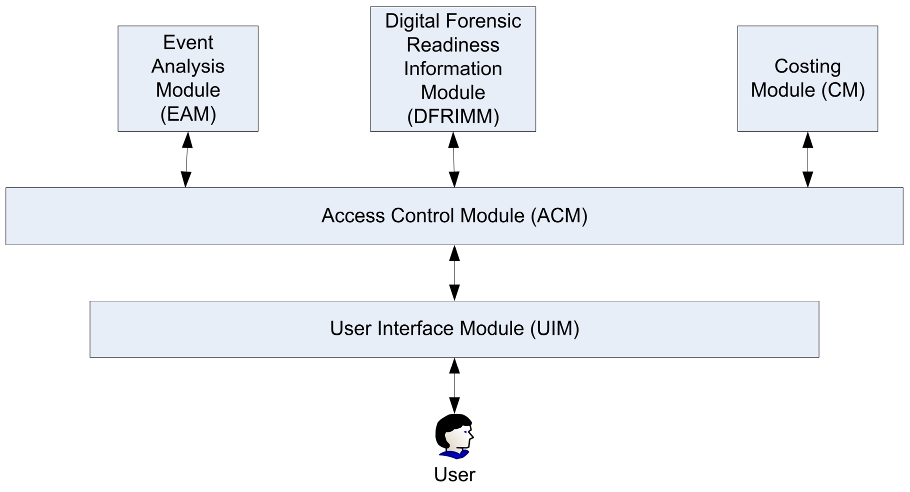
Cost is an important aspect of DFR because DF management typically works with a limited budget. This budget should be spent relative to risk. That is to say, the first priority when spending the budget should be on risks that are 1) greatest in terms of potential loss, and 2) mitigated the most through the use of digital evidence (Rowlingson 2005). A DFRMS should therefore assist staff in determining the cost of DFR measures so that these costs can be weighed against potential losses. Cost is not discussed in any further detail in this paper since, at the time of writing, we have proposed a novel approach to cost and DFR that forms the basis of Reddy et al. (2011).

4. A DFRMS Architecture and Prototype

In this section we present and describe an architecture for a DFRMS together with a proof-of-concept prototype that is based on the architecture. Because it is a proof-of-concept-system, this prototype was not designed to be deployed in a large organisation. We focused rather on developing a system that would be able to accomplish the basic or core functionality. Although it is at an early stage of development, we found that prototype was able to successfully achieve the major functionality required of a DFRMS.

The overall architecture will be presented briefly here and will be followed by a more detailed look at the architectural components in the sub-sections that follow. After each component has been discussed, we shall discuss the implementation of that component in the prototype.

When viewed at the highest level, the architecture consists primarily of five modules, namely: the event analysis module, the DFR information module, the costing module, the access control module, and the user interface module. The architecture is modular in design. This means that the modules are able to function relatively independently from one another. The event analysis module, the DFR information module, and the costing module are all designed to meet the monitoring, DFR information, and cost requirements that were mentioned in the previous section. The access control module, as its name suggests, handles access control for the system, and it is coupled with the user interface module since access rights determine the options available to users in the user interface. Figure 1 below shows a high-level view of the architecture.


Event Analysis Module (EAM)
Digital Forensic Readiness Information Module (DFRIMM)
Costing Module (CM)
Access Control Module (ACM)
User Interface Module (UIM)
User
Figure 1: High-level view of the architecture

In order to implement the architecture, we developed a prototype using the Java programming language and MySQL database. While still a proof-of-concept prototype, it is nevertheless sizeable, and consists of approximately 35 source packages and 225 individual classes.

We begin our discussion of the architecture by examining the event analysis module.

4.1 Event Analysis Module

The event analysis module (EAM) receives events from hardware and software entities in the IT infrastructure. The components that make up the EAM are shown in Figure 2 below. The databases shown in Figure 2 are not necessarily multiple databases, but they are depicted as such for illustrative purposes – that is to say, they are a logical representation, and do not necessarily indicate separate physical databases.

The EAM alerts users based on pre-defined alert definitions. Users with sufficient privileges can create alert definitions that are comprised of a single event or a combination of events. Alert definitions are typically created to indicate suspicious activity or activities that are of interest to DF staff. Alert definitions, once created by users, are stored in encrypted form in a database. Encryption is used so that if the database is compromised, there will be no disclosure of the activities being monitored.

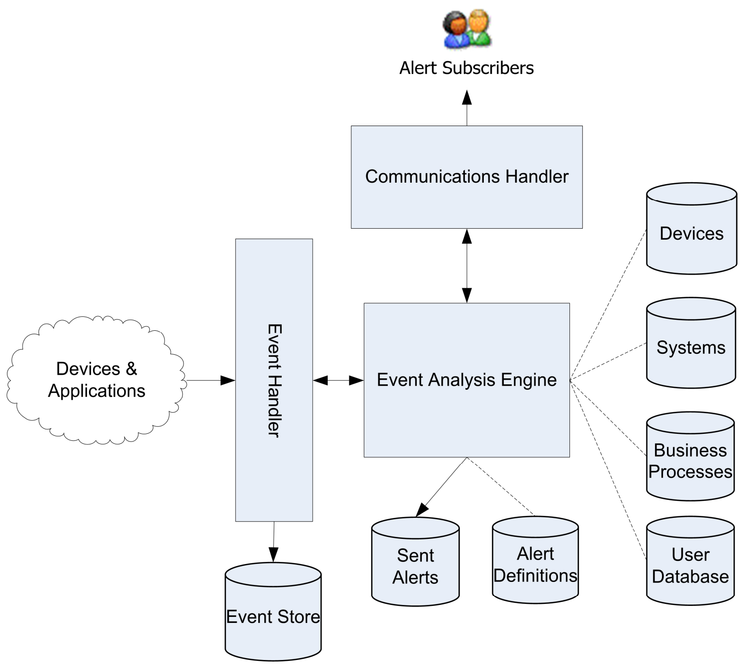


Figure 2 – Figure illustrating components of the Event Analysis Module. Double-sided arrows indicate two-way communication between components. Singe-sided arrows signify one-way communication from the component to the arrow's target. Dashed lines indicate queries that request information from databases.

Alerts are defined with respect to one or more of the following: devices, systems and business processes. A device is typically hardware, for example, a router or a firewall. 'System' refers to software, such as an application system or an operating system. Lastly, 'business process' refers to the series of activities that together comprise a business process. Alerts are only possible for business processes that make use of devices or systems. In such cases, the definition of the business process will include a link or reference to the device or system that is associated with the business process. When any of the devices or systems that have been linked with the business process triggers an alert, the business process alert will also be triggered. The reason for defining alerts with respect to business processes is that certain business processes such as, for example, the accounts payable business process, have a higher risk associated with them. In these cases, those members of management who are associated with the business process will need to be notified about any events that signify potential danger.

In order to receive alerts, it is necessary for users to subscribe to alerts, or else be subscribed to alerts by other users. Only high-level users, such as managers, are allowed to subscribe other users. The databases that store the devices, systems and business processes are shown using database symbols, which are the symbols on the extreme right of Figure 2.

In the architecture, we have not mandated a particular approach to event analysis, such as the typical rule or behaviour-based approaches, or alternative approaches exemplified by Wu et al. (2010). We have left the choice of analysis technique as a design choice, however event analysis must be performed by the event analysis engine (EAE).

In practice dealing with events from hardware or software entities is not trivial for two reasons. Firstly, not all hardware and software entities are designed to explicitly provide event information (Karlzén, 2009). Secondly, where hardware and software entities have been designed to explicitly provide event information, the event information may be formatted according to a number of different standards (Karlzén, 2009)(Swift, 2006). In order to resolve the first problem, it is necessary for the DFRMS to use techniques that are similar to those used by SEMs. SEMs make use of software known as event collectors, or agents, to extract event information from hardware or software entities that do not explicitly provide event information (Nicolett, 2008). In the DFRMS, event collectors or agents must also be used to send event information to the EAM in an appropriate format. To solve the second problem, the DFRMS should cater for all the necessary event information standards that are used by hardware and software entities in a particular organisation.

The EAE does not receive events directly from hardware or software entities. It receives events from a separate sub-module called the event handler. The event handler receives communications directly from hardware and software entities if these entities use supported event information standards. If these entities use standards that are not supported, or if they do not provide explicit event information, the event handler receives event information from their event collectors or agents. The event handler serves as an intermediary in order to separate the communication function from the analysis function.

Apart from monitoring and alerting, a primary function of the EAM, as with a SEM, is to log events (Mitropoulos 2006). To this end, the event handler not only passes events to the EAE, but it stores all events directly in a database in encrypted form. There are three reasons why the event handler also stores events directly in the database. Firstly, if events are passed via the EAE to the database and not directly, a failure in the EAE would result in lost event data. Secondly, not having the EAE store events in the database reduces the computational load on the EAE. Reducing the computational load is important because analysis is computationally intensive. The use of a database for logging also allows for arbitrary queries to be performed against event history during DF investigations. Thirdly, events stored in the database should be free from any processing or alteration that may occur in the EAE. This is in keeping with the principle that forensic data should be acquired without alteration (Wang 2007).

For the sake of abstracting the communication function, alerts are also not sent directly from the EAE but rather through a communications handler. The communications handler allows for alerts to be sent through different forms of communication or channels, such as email or SMS (Liang et al., 2011)(Nguyen & Skrabalek, 2011). Whenever an alert is sent to the communications handler, it is also stored in an encrypted form in a database. This is done so that there will be a record of all alerts that were detected by the EAE. In the event that the communications handler fails, it will still be possible to determine the alerts that were triggered at a specific time by querying the database.

The EAM works in the same way as a SEM. Indeed, a SEM may perform the function of the EAM in the architecture, provided that the SEM meets the access control requirements that are discussed later Section 4.5. In summary, the EAM satisfies requirements 1 to 5 listed in Table 1 above.

4.2 Prototype Implementation of Event Analysis Module

The EAM and all its functionality have a dedicated section of the prototype's graphical user interface (GUI). Users with sufficient access rights can use the EAM to achieve almost all of the functionality listed in the preceding section. Other requirements, such as the encrypted storage of event logs, have also been implemented. In addition, the prototype implementation allows for training requirement and forensic procedure documents to be linked with devices and systems when adding devices and systems to the DFRMS. For example, if a user added a database management system (DBMS) to the DFRMS using the EAM GUI, the user could link a forensic procedure or training requirement document to the DBMS. Selecting the DBMS would then show the procedure documents. In cases where devices or systems are added without being linked to training courses or forensic procedures, a warning indicator light on the GUI is turned on. This allows users to be aware of the potential risk of devices or systems that do not have training or forensic procedure requirements linked to them.

As mentioned, both devices and systems must either be capable of sending events to the EAM, or have the ability to have their event data transmitted to the EAM via event collectors or agents. For the purposes of testing the prototype, we simulated communication with devices and systems. That is, independent programs which simulated devices and systems sent events to the prototype. We used simulation for two reasons. Firstly, to avoid the cost of acquiring systems and devices such as firewalls, routers, etc. Secondly, to avoid the complexity and time required to create event collectors and other software necessary to communicate with systems and devices.

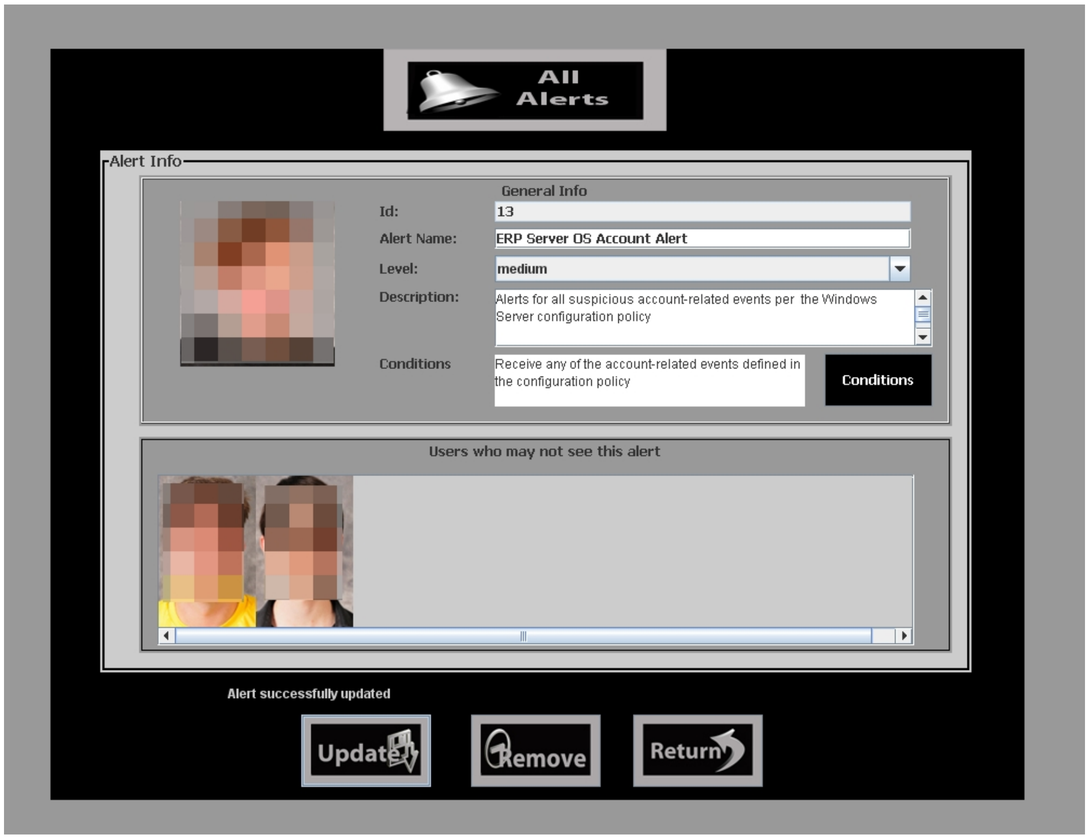
In general, events are handled by the prototype in accordance with the architecture. A specific implementation issue we discuss here is that of event data. The only basic event data that is expected by the EAM is: a unique identifier for the device or system sending the event, the time stamp of the event from the originating device or system, and an identifier for the event itself. In addition, before storing an event in the event store, the event handler adds the time it received the event. Recording both the time stamp at its origin and the time stamp at receipt of an event can help highlight situations where the time-synchronisation between devices/systems and the DFRMS is not correct. This is important from a forensic point of view as time synchronisation between devices/systems within an organisation can be vital when trying to recreate an incident.

When the event handler stores an event in the event store, it adds the appropriate device or system name to the event data before storing the data in the event store. This makes for easier analysis of events in the event store. When querying the event store, for example, a user would be able to see that 'Web Server 1' sent the event 'User Account Lockout' rather than merely seeing that system 'g34f' sent an event 'b34'.

The prototype allows for three levels of alerts. These are:

- Critical: Critical alerts are deemed to be of extreme importance. All users are automatically subscribed to critical alerts. To avoid the accidental deletion of critical alerts, the alerts must first be modified to a lower level before being deleted.
- Medium. These alerts are deemed important, however, they differ from Critical alerts in that: they are subscribed to by individual users and they can be deleted directly.
- Low. Low alerts represent low priority events or events that are predominantly informational by nature.

Figure 4 below shows part of an alert definition. The user assigns the alert a name, a level and a description, and also defines the conditions, or sequence of events required to trigger the alert. The top-most image on the left is of the user that created the alert.



Id: 13
Alert Name: ERP Server OS Account Alert
Level: medium
Description: Alerts for all suspicious account-related events per the Windows Server configuration policy
Conditions: Receive any of the account-related events defined in the configuration policy
Alert successfully updated

Figure 4 – Screen showing part of an alert definition for a medium alert

To use a more specific example of an alert definition, consider a scenario where an attacker exploits a missing OS patch on a server and is therefore able to escalate the privilege of an OS service account to that of an administrator account. Now, the DFRMS can be configured to trigger an alert when the server's OS sends the DFRMS an event signalling the account privilege escalation. To accomplish this in the DFRMS, the user clicks the black ‘Conditions’ button shown in Figure 4, selects the appropriate system, and then selects the events from the system that will trigger the alert.

Although some SEM software uses sophisticated AI techniques to determine when to trigger an alert, these techniques were not the focus of our prototype since the architecture does not specify a particular event analysis technique. Instead, the prototype uses simple pattern matching – if the events received match a pattern associated with an alert, then that alert is triggered. The simple pattern matching engine used in the prototype can be easily replaced by a more sophisticated engine without significant change to the rest of the DFRMS. This is because the prototype complies with the modular nature required by the DFRMS architecture. A UML activity diagram describing how the prototype processes events and raises alerts may be seen in Appendix A.

In the section that follows, we will discuss the DFR Information Management Module.

4.3 DFR Information Management Module

The primary purpose of the DFR information management module (DFRIMM) is to make the information required for DFR purposes available to the appropriate staff. Staff who need to work with the information stored in the DFRIMM are required to become users of the DFRMS. The DFRIMM allows for the management of such DFR-related information through the creation, editing and deletion of the items mentioned in Section 3.2 above, namely: policies, procedures, DF and incident teams, training requirements and organisational structure. These are requirements 6 to 16 in Table 1.) The DFRIMM also has access to the device, system and business processes information used in the Event Analysis Module so that training requirements can be properly managed. A diagram of the DFRIMM components is shown in Figure 3.

In the sub-sections below, we detail how the DFRIMM handles the management of documentation for policies, procedures and organisational structure. With regard to DF teams, incident teams and training requirements, we show how the DFRIMM goes beyond document management and includes other forms of functionality. In addition to the requirements drawn from the literature, we include functionality for leave management and an investigation archive. These will also be discussed in the sub-sections below.

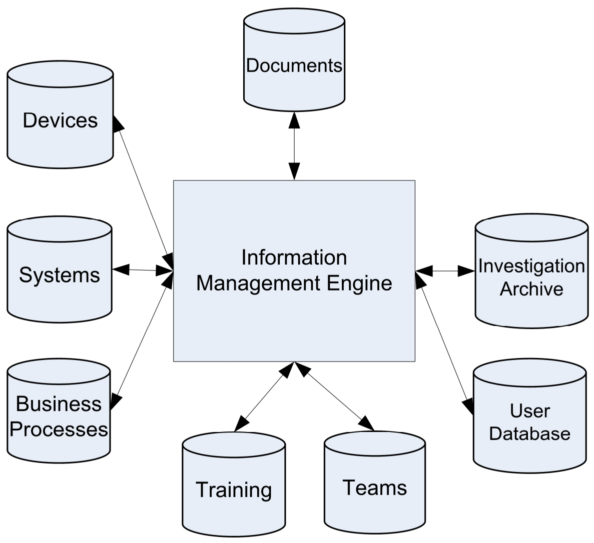


Figure 3 – Figure illustrating components of the DFRIMM.

4.3.1 Management of Documentation

The DFRIMM stores policies, procedures and the organisational structure as electronic documents. In this regard, the DFRIMM serves essentially as a document management system for these documents. Document management may be defined as the automated control of electronic documents through their entire life cycle within an organisation (Cleveland 1995). Cleveland (1995) further notes that document management allows organisations "to exert greater control over the production, storage, and distribution of documents, yielding greater efficiencies in the ability to reuse information" and "to control a document through a workflow process". In large organisations that are likely to use a DFRMS, policies, procedures and organisational structure documents are likely to follow a formal workflow process in their creation and modification, making this functionality particularly useful.

The need for document management stems from the requirements found in the literature, specifically, requirements 10-15 in Table 1. Incident response and escalation procedures, however, require decisions to be taken by staff. We believe that, in addition to the requirements in the literature, it should be optional for the DFRIMM to record such decisions. Recording these decisions makes it possible to determine whether or not procedure was correctly followed. It also allows for review during post-incident analysis or evaluation. Kurowski & Frings (2011) have noted that the documentation of incidents can also prove a valuable asset to DF investigators. We do not to make this functionality mandatory since the recording of decisions may create an administrative overhead to incident and escalation procedures that hampers their effectiveness. Depending on how the decision recording functionality is implemented, it may also force an organisation to adopt a workflow that is not optimal for the organisation. We therefore leave it up to organisations to weigh the pros and cons of this functionality in their own environment.

In the following section, we will discuss training management.

4.3.2 Training Management

The training management functionality in the DFRIMM serves to ensure that management is aware of the various training and skills that staff members possess. Training management functionality records not only the training completed by staff members but also the training currently underway, and the costs associated with the training.

As mentioned, when devices, systems and business processes are added to the DFRMS, they can have training requirements, including a possible null requirement, linked with them. This enables management to ensure that the requisite skills are available for all devices, systems and business processes since management can match the skills required with the skills available. Where skills are not available for devices, systems or business processes, management can then attend to this.

4.3.3 Digital Forensics and Incident Response Team Management

The DFRIMM allows high-level users, such as management, to create DF teams, that is, to specify the DF investigators that are required to work together in a specific team. In order to compose or create a team in the DFRIMM, all members of the team must be represented as staff members in the DFRIMM. Management can then select them from within the DFRIMM. In this way management is able to check their skills and training when creating teams and ensure that the teams are composed appropriately.

The same rationale and process applies to incident response teams. The only difference in the case of incident response teams is that responders are likely to not all be DF staff. A database administrator (DBA), for example, may be part of an incident response team if an incident response procedure requires a database to be shut down. The DBA's manager may therefore also need access to the training management functionality of the DFRIMM to update the DBA's DFR training. As Lamis (2010) points out, incident response must proceed in a forensically safe manner. This implies that incident responders who are chosen from outside of DF staff may need some level of training to appreciate basic DF concepts, such as the chain of evidence.

Leave management, as mentioned, is a feature not found in the literature on DFR, but we suggest its inclusion in the DFRMS and will discuss it next.

4.3.4 Leave Management

The leave management function in the DFRIMM allows management to administer the leave of staff involved in DFR. The reason for managing leave is that DFR can be negatively impacted if staff that possess certain skills are not present when their skills are needed. Consider, for example, the case where there is a single DF staff member who has been trained to extract data in a forensically sound manner from the payroll system. If that staff member is on leave when his specific skills are required, the ability of the organisation to be forensically ready for incidents involving the payroll system will be severely hampered. The leave management functionality therefore brings this increased risk to the attention of management.

When considering leave for the staff member, management can take into account the skills and training that the staff member possesses, as well as the devices, systems, business processes and teams the staff member's absence will affect. This is possible because devices, systems, business processes are linked with skills and training requirements. The leave management functionality of the DFRIMM can then determine what will be affected by linking the staff member's training and skills with the skills and training required for different devices, systems, and business processes. In the example given, it may be safe for a manager may approve the staff member's leave request so that it does not fall within the period of the payroll run.

4.3.5 Investigation Archive

Although the concept of an investigation archive does not appear in the literature on DFR, we nevertheless propose it as an additional feature to be included in the DFRIMM. The investigation archive serves as a secure storage location for potential evidence that DF investigators and/or incident responders may come across in their initial response to an incident. If an incident warrants a full investigation, it is likely that DF analysis tools will be used to for proper analysis. The investigation archive is a convenience to DF investigators and incident responders and is not a replacement for DF analysis tools. The investigation archive is convenient since it allows investigators and responders access to the same evidence or information when making decisions in response to incidents.

The investigation archive is also encrypted and/or digitally signed to prevent the tampering of evidence.

4.4 Prototype Implementation of the DFR Information Management Module

The DFRIMM has its own dedicated section of the prototype's GUI. Within this section of the GUI the functions are grouped into the following three categories, 'General', 'Components' and 'Procedures'. Table 2 summarizes the functionality that is available in each category. The table lists the user interface menu options that are available for each category, and then lists the corresponding functionality in the right-hand column.

Table 2 – Summary of options available from initial DFRIMM screen.

Category:	Menu Option:	Allows For Administration of:
General	Users	Users in the DFRMS
	Teams	DF and incident response teams
	Training	DFR training needs
Procedures	Business Process	Business processes
	Documents	Policy and procedure documents
Components	Devices	Devices the prototype receives events
	Systems	Systems the prototype receives events

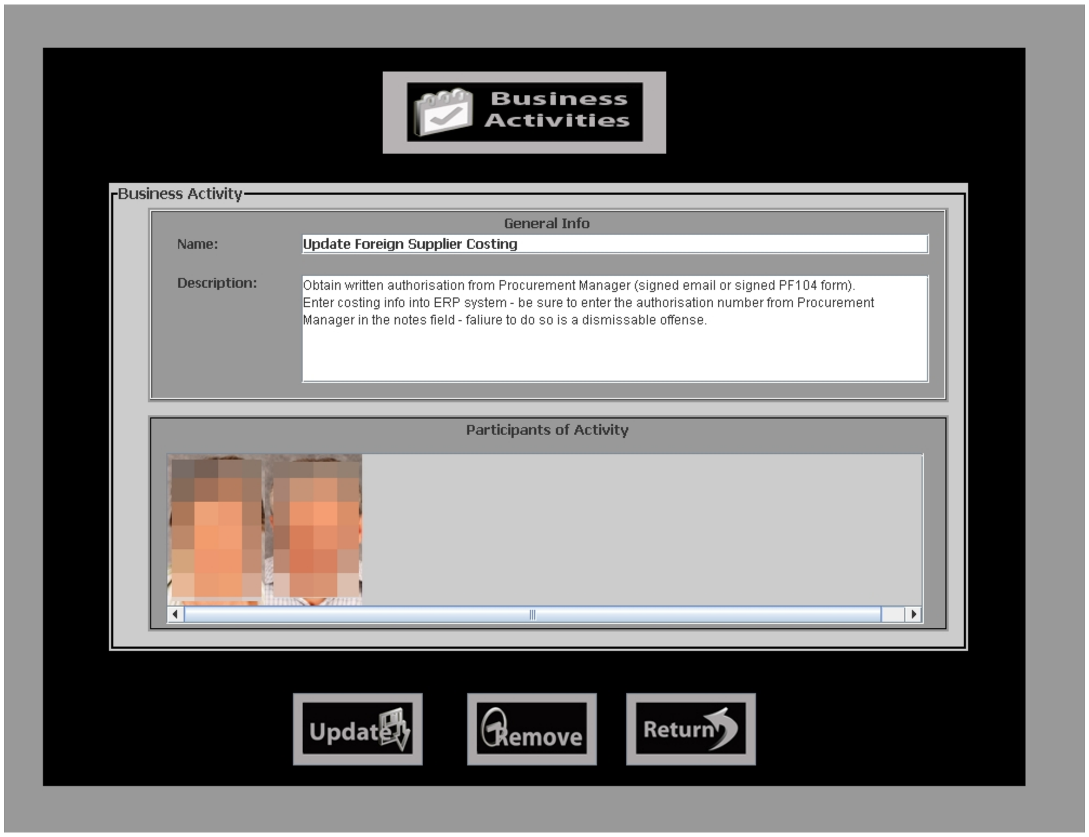
The ‘Components’ category provides for the management or administration of devices and systems. This category makes it possible to add, edit or delete devices or systems and their associated training requirements and forensic procedures. This functionality is available in both the EAM and DFRIMM. In the EAM, devices and systems should have training requirements and forensic procedures linked to them because they are being monitored and possibly involved in potential incidents. In the DFRIMM devices or systems that are not part of any monitoring may be added, for instance, for testing prior to implementation. It is therefore not necessary for training requirements and forensic procedures to be linked to devices or systems in the DFRIMM.

In the EAM or DFRIMM it is not possible to delete or edit a device or system if an alert is based on the device or system. When alerts are edited or deleted, they must first be modified to a lower level and then deleted. Apart from being a security precaution that prevents alerts from being circumvented, this procedure also ensures that devices and systems are properly configured before alerts are based on them.

In the ‘General’ category, it is possible for users with sufficient privileges to administer DFRMS users, as well as DF and incident response teams. Individual users can be added, deleted or have their details edited in DFRMS by users with such administrative privileges. In the same way, such administrative users are able to create, edit or delete DF and incident response teams. When teams are created, users are required to specify a team name, a graphical icon for ease of reference, and a team owner who is responsible for the team.

The last feature that the ‘General’ category makes available is the ability to administer DFR training needs. Users with administrative privileges can therefore add training courses to the DFRMS. The course material itself is not stored or presented in the DFRMS. Rather, the details of a course, such as the course name, a short description of the course, and the cost of the course are added to the DFRMS. Ordinary users may select a course, and the DFMRS will add it to a list of their current courses and display the total cost of all current courses. Once courses have been completed satisfactorily, users with appropriate privileges, such as managers, may change the status of a user’s course to ‘completed’.

The next category in the DFRIMM is the ‘Procedures’ category. This category allows users to administer the business process information that is stored in the DFRMS. Business processes are specified by adding the individual activities that comprise the business process. Figure 5 shows that the business activities themselves each have a name and a detailed description. The detailed description includes a list of those users who contribute to the activity.



Name: Update Foreign Supplier Costing

Description: Obtain written authorisation from Procurement Manager (signed email or signed PF104 form). Enter costing info into ERP system - be sure to enter the authorisation number from Procurement Manager in the notes field - faliure to do so is a dismissable offense.

Figure 5 – Screenshot showing the detail of a business activity

The 'Procedures' category also allows users to administer DFR-related policy and procedure documents. We have already mentioned that forensic procedure documents can be linked to devices and systems. A safeguard included in the prototype is that it is not possible to delete a DF procedure without first unlinking the procedure from the device or system. This is designed to prevent users from deliberately or inadvertently deleting procedure documents, thereby creating a state of unpreparedness when a particular procedure is required but is not available.

It is evident that the DFRIMM is essentially a front-end to a database that stores DFR information and documents. Access to the information is, however, mediated by the access control module which we will discuss in the following section.

4.5 Access Control Module

The access control module (ACM) governs access control for all other modules in the architecture. The ACM is based on an underlying access control model that it uses to determine if users are allowed to access data or execute commands within the DFRMS. We do not require a particular access control model, such as role-based access control, for the DFRMS. The modular design of the architecture makes it necessary for the ACM to be interchangeable with an ACM that is based on a different access control model without significant changes to the other modules in the system.

The particular model chosen for the ACM should nevertheless also be able to cater for the access control requirements that are peculiar to the DFRMS. One of the assumptions in the design of the DFRMS architecture is that the users of the DFRMS may not all be trusted. To illustrate this concept, consider the following scenario. User X has rights to subscribe to alerts. He is suspected of fraud involving a financial application. Since this is the case, an alert has been defined for each time he logs into the financial application server. Since User X can see all such alerts and subscribe to them, he can also see the one that applies to him. This automatic disclosure defeats the objective of defining the alert mentioned above. The ACM therefore needs to be able to hide alerts from users who would otherwise be able to see them. This also implies that the alert definition syntax must include the ability to specify which users should be blinded to the alert. In this scenario, User X may be an IT Security staff member who makes use of the DFRMS. He may even be a DF staff member. This kind of access control requirement is unusual since, in most other monitoring software, such as SEMs, the ability to blind high-level users is not common.

The ACM must not only blind users from seeing specific alerts within a list of alerts, but it must also stop some users from discovering certain features and functionality of the DFRMS. For example, low level users that do not make use of the EAM should not be able to use the user interface to access the EAM and view its capabilities. This may provide less trusted staff with valuable information they could use to circumvent monitoring by the DFRMS. This exclusion of low level users from parts of the user interface is in keeping with the principle of least privilege. It also implies that the ACM may be more tightly coupled with the user interface module than other modules.

Thus far we have discussed the role of the ACM in the EAM. The ACM also plays a role in the DFRIMM and costing module (CM). The ACM controls access to information in the DFRIMM and CM, ensuring that only users with appropriate privileges are able to create, edit or delete documents, teams, cost data, or other relevant information.

4.6 Prototype Implementation of the Access Control Module

The access control module (ACM) in the prototype implements an access control model that is based on a linear rank hierarchy. In the rank hierarchy, users belong to one of four ranks. These ranks are, in order of highest to lowest rank: Chief Forensic Officer, Forensic Officer, Security Officer and External Consultant. Each rank is represented by an integer and the higher the rank used, the greater the integer. Permission to access information or use functionality in the DFRMS is also associated with a rank, which is known as the access rank.

When access to information or functionality in the DFRMS is required by a user, the ACM compares the rank of the user with the access rank. If the user’s rank is greater or equal to the access rank, the ACM grants the requested access. This comparison occurs in a rank comparison method within the ACM. Nevertheless, a ‘pluggable’ architecture is used in the ACM, which allows the rank comparison method to be overridden. If overridden, access comparisons may be performed in a different manner, for example, to deal with custom access control requirements.

The four ranks chosen for the prototype are unlikely to be adequate for an operational DFRMS used in a large organisation. The ranks were chosen merely for proof-of-concept purposes to illustrate the functionality of the ACM. Increasing the number of ranks is trivial, though time-consuming, which is the reason more ranks were not included in the prototype.

We discussed the ability to blind users to the existence of alerts in the previous section. In the prototype it is not possible to blind users from seeing critical alerts because all users are automatically subscribed to these alerts. It is, however, possible to blind specific users from seeing medium and low level alerts. Figure 4 shows a medium alert. The images of the blinded users can be seen in the lower pane. The users' images are pixelated to preserve their privacy.

In the following section we describe the user interface module.

4.7 User Interface Module

The user interface module (UIM), as its name suggests, provides users with a graphical user interface to the other modules. It is the only way in which ordinary, non-administrative users are able to interact with the DFRMS. The modular nature of the architecture requires that the UIM is not tightly coupled with the EAM, DFRIMM, and CM. This architectural requirement abstracts the user interface, or front-end, from the data processing modules, or back-end of the DFRMS.

An important function of the UIM is to record the actions of users in the DFRIMM. This provides an audit trail which can be used as evidence in the event of a user misusing the DFRMS itself. User actions are stored directly in a database in encrypted form.

4.8 Prototype Implementation of the User Interface Module

The user interface module (UIM) makes use of a graphical user interface (GUI). The GUI design in the prototype does not utilise the more traditional, drop-down menu user interface. We opted instead for a user friendly icon-driven GUI that is based on the premise that this may improve user acceptance and ease of use of the DFRMS – as can be seen in the screen shots, large icons and graphical elements featured extensively in the GUI. This premise, though, was not tested as it is out of the scope of this research. We leave this as future work.

In accordance with the architecture we presented earlier in this paper, the GUI module is not coupled with the EAM or DFRIMM. It is, however, dependent on the ACM since it requests a rank comparison from the ACM before it allows a user access to information or functionality. As we noted in the sub-section on the ACM, we have done this to ensure that only those users with sufficient privileges are allowed access to information and functionality.

In the following section, we will discuss the costing module.

4.9 Costing Module

As mentioned earlier in Section 3.3, we have limited the scope of our discussion of the costing module (CM). We suffice it to say that the CM should provide a means by which the cost of DFR measures can be determined. The cost of such measures may include, *inter alia*, the cost of staff, infrastructure, training and business processes. In order to determine such costs, cost information needs to be recorded where necessary. Such cost information may be contained in the databases used by the DFRIMM.

5 Discussion and Scenario

In the preceding section we presented the architecture for a DFRMS and our prototype implementation of a DFRMS. In this section, we will provide a more general discussion of the architecture, prototype, and issues that pertain to the DFRMS as a whole. We will also include a scenario that serves as an example of how a DFRMS can be useful.

5.1 Discussion

In each of the modules discussed in the previous section, the functionality of the modules meets or exceeds the requirements from the literature. Some functionality, such as monitoring, is of obvious importance. Other functionality, such as the storage of the organisational hierarchy, may not seem to be as important. While the space available here does not allow us to explain why each requirement cited from the literature is important, we recognise all of them as being important for the following reason. The individuals and departments involved in DFR, and the interactions between them, can be considered to be a complex system. Whereas it may be possible to adequately manage each of these activities and individuals separately, we contend that the simultaneous management of all of them remains a difficult task that is prone to human error. Cook (2002) points out that "overt catastrophic failure occurs when small, apparently innocuous failures join to create opportunity for a systemic accident. Each of these small failures is necessary to cause catastrophe but only the combination is sufficient to permit failure". The DFRMS therefore helps management to ensure that many small lapses do not escalate into a larger, more significant failure.

The implementation of a DFRMS is not a trivial undertaking. We have already noted that even our limited prototype consists of a sizeable amount of code. Among the lessons we learned when we created the prototype is the importance of planning the design of the databases. A number of databases are shared between the modules in the prototype. The databases that store device and system information are, for example, shared between the DFRIMM and EAM. The DFRIMM will require different information about devices and systems than the EAM, however, since the database is shared, the information required for both modules must be included. It should also be noted that the sophisticated queries that may become necessary during investigations may not be possible if the design of the databases is not carefully considered. Good database designs are also essential if AI techniques are to be used in the EAM or other modules. Without well designed databases certain correlations may not be possible and may therefore limit the effectiveness of certain AI techniques.

The performance of the databases is also important. In our prototype, it was adequate to allow the EAM and DFRIMM to share the database. However, in the case of a fully functional DFRMS in a large organisation, appropriate database technology must be used. Since the EAM may have to handle thousands of events per minute, the database that stores the device and system information needs to be capable of dealing with this number of transactions while at the same time handling requests from the DFRIMM in a timely manner. Alternatively, an organisation may consider using separate databases and periodically synchronising the databases.

The setup of each module requires significant amounts of planning and time. What is more important, however, is that the setup requires the input of non-DF staff. Such staff are likely to be from departments such as IT, IT security, and those departments whose business processes are affected by DFR controls. In fact, staff from multiple departments will not only be needed for setup, but may be needed for the daily use and regular maintenance of the DFRMS. This implies that for a DFRMS to be successfully implemented and used, it will require the buy-in of senior management from all the departments that are affected. The decision to implement a DFRMS therefore has to be made at a high level within the organisation.

As we noted above, the DFRMS architecture is intended for use in large organisations. It is also intended for organisations with a well developed IT infrastructure, and mature IT security and DF programmes. The absence of any of these would make the successful implementation and use of the DFRMS difficult. DF itself must also be of particular importance in an organisation for it to warrant the direct financial cost, as well as the cost in time and administrative overhead that a DFRMS will entail. In this regard, a DFRMS is particularly suitable for large organisations where DF investigations are vital and may need to be undertaken as a matter of law. Such organisations include law enforcement, military and financial organisations. The ACM, which allows for high-level users to be monitored without their knowledge, may be especially useful in these organisations when high-level users need to be investigated.

Although the access control model in the DFRMS may blind users to the fact that they are being monitored, skilled users with the sufficient knowledge and access to IT infrastructure may be able to infer that they are being monitored. Such inferences may be made by intercepting network traffic, or indeed, where network traffic is encrypted, by observing network traffic patterns. Consider the case in which a staff member's logins to an application server are being monitored. The staff member may notice unusual network traffic each time he logs into the server. While it may not be possible to counter such analysis completely in the case of trusted staff, certain measures may be taken to make it more difficult for the staff that are being monitored to perform such analysis. First, communication between DFRMS components, as well as between devices or systems and the DFRMS, should be encrypted where possible. Second, where it is feasible, random traffic can be generated to minimise the effectiveness of traffic pattern analysis. Third, depending on the risk, certain components, modules, or even the DFMRS itself, should be moved to an isolated segment of the organisation's network (Koen & Olivier 2007). This part of the network should be administered by DF staff alone. The reason for isolating the DFRMS is that it may be possible to perform an analysis on components, such as the EAM's communications handler, which may expose specific monitoring due to the alerts that are being sent. Similarly, the communication between the EAE and the alert store and event store databases may also disclose specific monitoring activities. By isolating the DFRMS or its components, it is possible to limit the visibility of such activity to DF staff alone.

In the following section, we will describe and discuss an example scenario.

5.2 Scenario

The following scenario is hypothetical. It was not developed from knowledge of specific incidents at any organisation. It is, nevertheless, sufficiently generic for the purpose of an example scenario.

This scenario involves a newly installed enterprise resource planning (ERP) server. The ERP server stores financial information and, amongst other things, is used for paying suppliers. In the scenario the newly installed server is missing an operating system (OS) patch. Employee X, an employee in the IT Department is paid by a foreign supplier to increase the price of the supplier's goods as listed on the ERP server. This is done because payments are made automatically to the supplier using price data from the ERP server. Employee X exploits the vulnerability exposed by the missing OS patch on the ERP server and gains access to the ERP software on the server. By using the ERP software, ostensibly as a legitimate user, he then proceeds to increase the price of the supplier's goods. An audit that is conducted a month later detects the alteration to the supplier prices. Employee X notices what is happening gets nervous. Within a week he resigns from his job and leaves to a country that is not serviced by a non-extradition treaty. Since the supplier has already been paid, and because it is a foreign supplier, recovery of the overpayment is not feasible.

Upon detection of the price change, the organisation's DF team is notified, and, while they suspect Employee X, they are unable to confirm this suspicion because the ERP server operating system was not configured to log activity appropriately. In addition, no monitoring of event data from the ERP server OS or the ERP software itself was in place. Although the ERP software does not contain specific logs for supplier price changes, it does contain a number of other logs that can be combined to show Employee X's activity. The DF staff, however, do not have sufficient knowledge of the ERP software and only learn of the logs a week after being notified – the same day Employee X leaves the country. The organisation is thus unable to confirm Employee X as the culprit in time.

A primary reason that Employee X was able to access the ERP server was the lack of timely patching of the operating system. The timely patching of an OS is an IT security control and not a DF function. We therefore exclude the reasons for the lack of timely patching from the scope of our analysis in this scenario. A number of failures then occur in this scenario that may have been avoided through a DFRMS based on the architecture proposed in the previous chapter. These are as follows:

1. The ERP server OS was not configured to log appropriate user or network activity. This happened because, in the scenario, no DF procedure for configuring the OS existed for the ERP server. A DFRMS would require that a procedure be specified at the time of adding the server to the DFRMS.
2. No DF staff were sufficiently familiar with the ERP system to immediately use the correct logs. A DFRMS would require the necessary training to be specified when the ERP server is added to the DFRMS. Recall that the prototype has a warning light on the GUI that lights up when a system is not coupled with its corresponding requirement for training. This alerts DF management to any risks due to lack of training and would help mitigate this particular risk in the scenario.
3. No monitoring of event data from the ERP server OS or the ERP software itself was in place. When adding the server to the DFRMS, a user would need to specify the events from the ERP server OS and ERP software that would be received by the DFRMS. This requirement to specify events forces management to consider whether events need to be captured or not, and whether such events need to trigger alerts. Figure 4 shows an alert in our prototype for events that are related to ERP server OS accounts. In this scenario, if management had been duly diligent, it would have been able to configure such an alert by using the DFRMS.
4. The business process for change of supplier pricing was not sufficiently considered. If the business process for change supplier pricing had been captured in the DFRMS, this would have also encouraged management to consider configuring specific logging in the ERP software for supplier price changes, or at least changes to foreign supplier prices. Figure 5 illustrates the business activity for updating foreign supplier costs as it was captured in the prototype.

Each of the failures listed above contributed to the failure to be forensically ready to investigate the incident. This reiterates the principle stated by Cook in the previous section, namely that a large failure is usually the culmination of a series of smaller failures.

6 Conclusion

In this paper we presented the novel concept of a digital forensic readiness management system (DFRMS) that assists in the management of digital forensic readiness (DFR) in large organisations. We determined the functional requirements of a DFRMS from a survey of the literature on DFR. The results of the literature survey were also presented. In addition to the requirements derived from the literature, we also put forward the ideas of leave management functionality and an investigation archive. Furthermore, we proposed and discussed an architecture that could be used to build a DFRMS. An early proof-of-concept prototype DFRMS based on the proposed architecture was also presented. The prototype, although not ready to be deployed in a large organisation setting, demonstrates that the architecture is feasible and that the concept of the DFRMS is potentially useful for organisations that wish to improve their DFR.

Much work on the DFRMS remains for the future, which we now discuss in the following and final section of this paper.

7 Future Work

Future work on the DFRMS architecture presented in this paper would entail the development of a fully functional DFRMS that can be tested in a large organisation. Such empirical testing is required to determine if a DFRMS is truly effective in a large organisation. The DFRMS would have to prove it can function effectively from a technical point of view, that is, it should be able to: interface with the myriad devices and systems, handle the large number of events per second it would receive, and, be able to communicate alerts without unacceptable delays. It should be noted that in a fully operational DFRMS, the incorrect choice of data structure, method or algorithm would have a much larger and more noticeable effect than in the prototype presented here. Since SEM products currently on the market advertise processing thousands of events per second, it is not inconceivable that this would be possible with a DFRMS. Indeed, other technology in SEMs, such as file integrity monitoring and system configuration monitoring, would assist in the DFRMS in its monitoring function if incorporated into the DFRMS. Apart from technical performance, the testing of a DFRMS in a large organisation would involve determining how well the technology is accepted and used by staff. The assumptions that the choice of a non-standard GUI improves user acceptance and efficiency could also be tested. Future empirical research can also be conducted to indicate if the DFRMS adversely affects DFR or other business processes, as this is sometimes the case with information systems that result in changed business processes (Scheer and Habermann 2000).

The use of ontologies to represent devices, systems etc., such as those described by Brinson et al (2006) and Nogueira & Vasconcelos (2008), may be used to good effect in implementation of a DFRMS. A DFRIMM stores a large amount of information about elements such as devices, systems and staff. Ontologies help capture the relationship between these elements and allow for AI techniques to be used to reason about these elements. This may allow for the DFRIMM to proactively highlight non-obvious situations in which DFR may be affected.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful suggestions and comments.

Appendix A

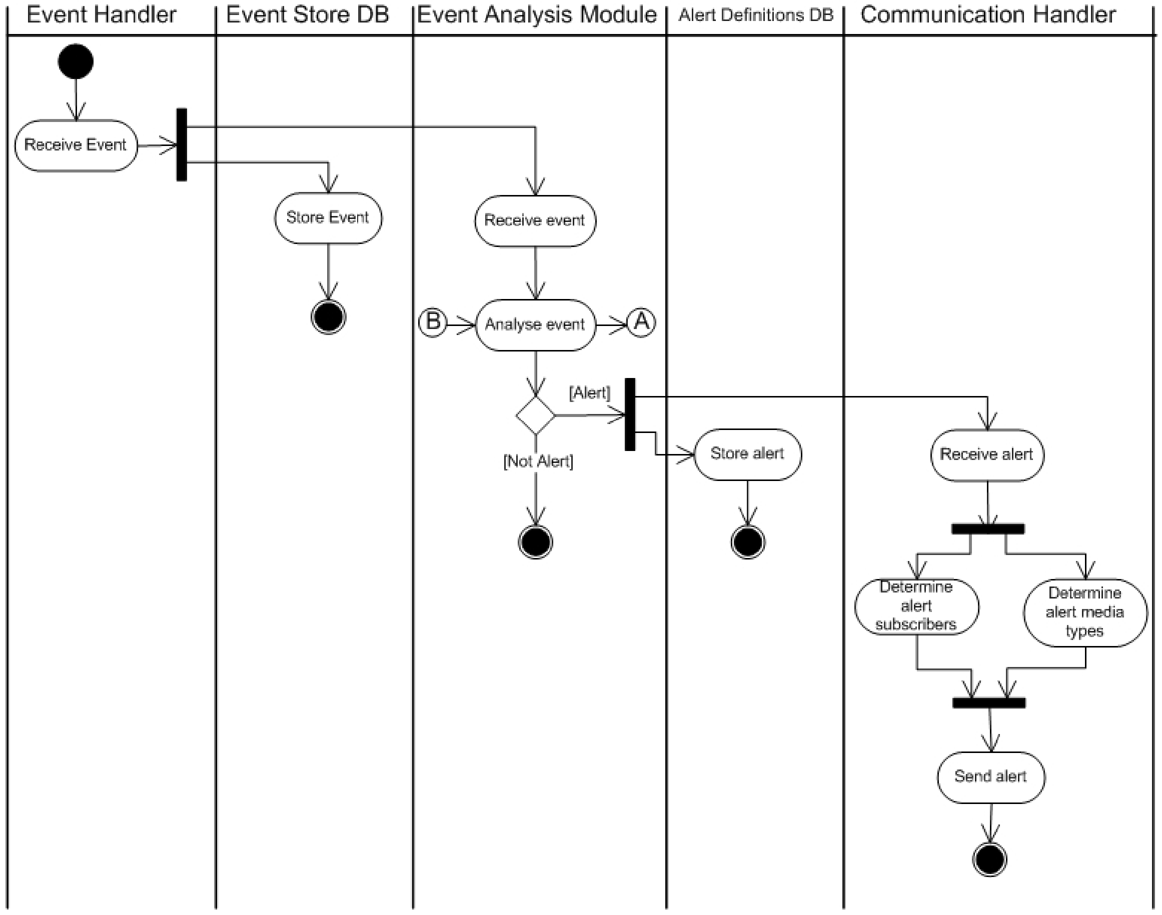


Figure 6 – UML activity diagram showing how the prototype processes events and raises alerts.

See Figure 7 below to follow the references to A and B.

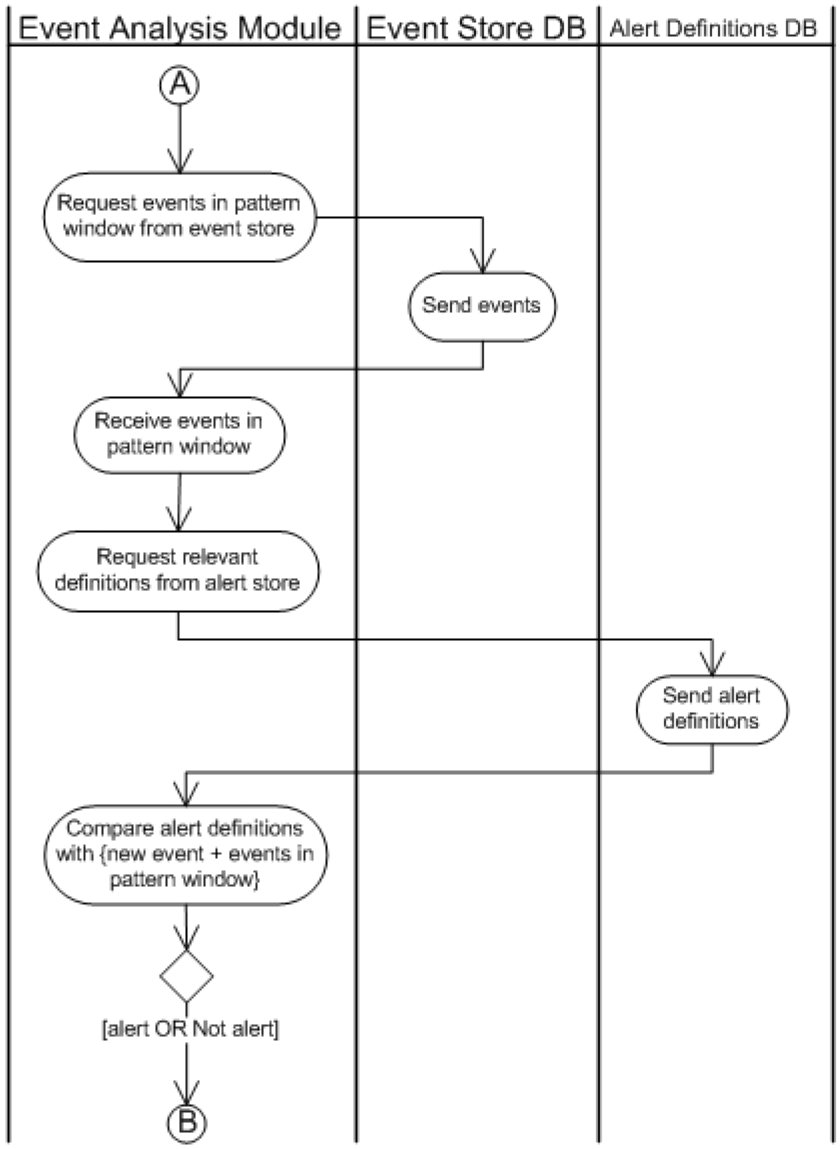

Event Analysis Module
Event Store DB
Alert Definitions DB
A
Request events in pattern window from event store
Send events
Receive events in pattern window
Request relevant definitions from alert store
Send alert definitions
Compare alert definitions with {new event + events in pattern window}
[alert OR Not alert]
B

Figure 7 – UML activity diagram showing event analysis in the EAM.

The pattern window referred to in Figure 7 describes the time period in which the pattern matching algorithm will search for patterns – e.g. a 48 hour pattern window will mean that the pattern matching algorithm will search the event store for events from the previous 48 hours for patterns defined in alerts.

Appendix B

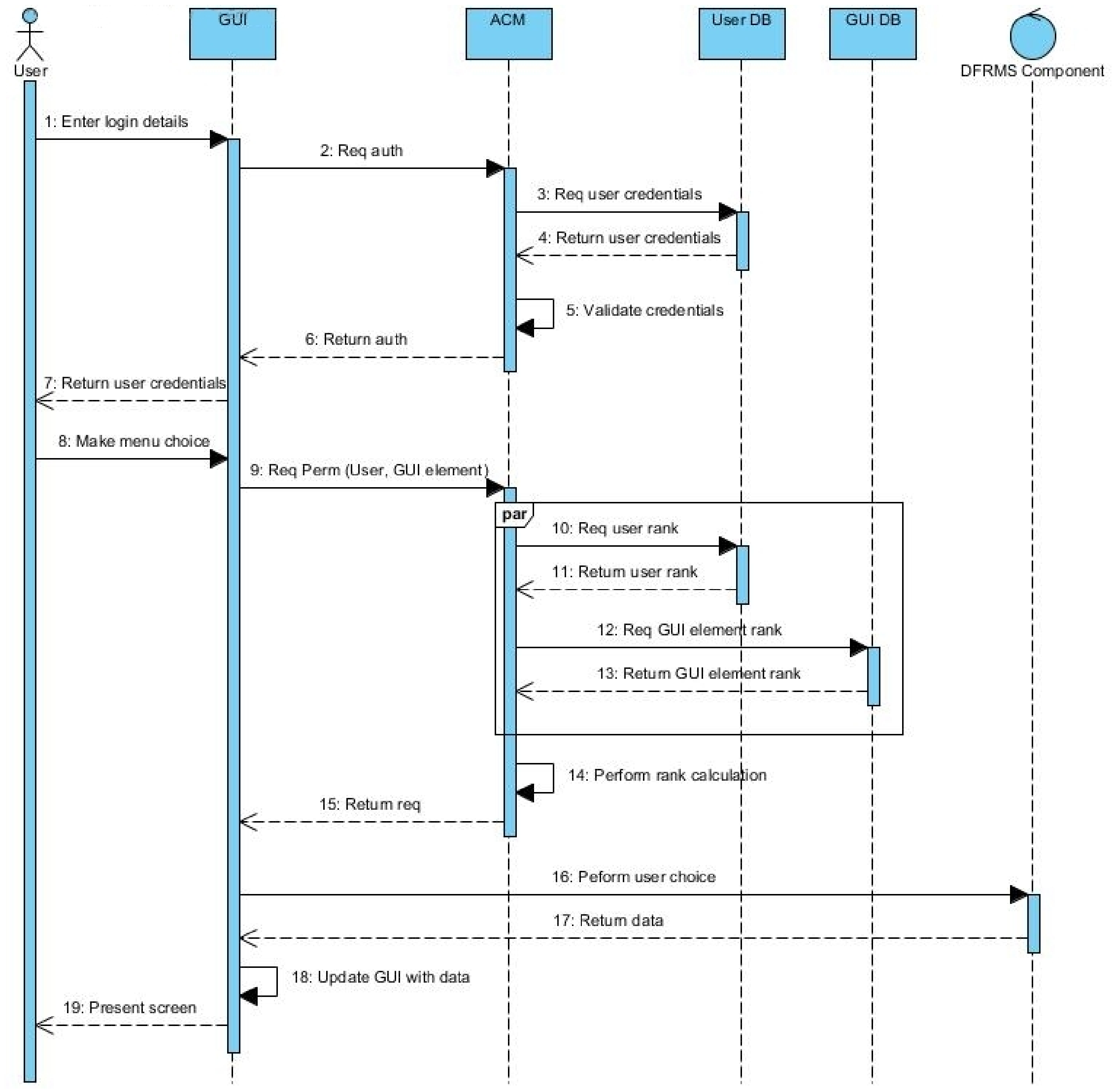


Figure 8 – UML sequence diagram showing a user interacting with the DFRMS.

For the sake of simplicity and to save space it is assumed that all authentication and permission requests shown in the figure are successful.

References

Ahmad, A., Hadgkiss, J. & Ruighaver, A.B., 2012, Incident response teams - Challenges in supporting the organisational security function. Computers & Security. Available at: http://www.sciencedirect.com/science/article/pii/S0167404812000624 [Accessed May 18, 2012].
Brinson, A., Robinson, A. & Rogers, M., 2006. A cyber forensics ontology: Creating a new approach to studying cyber forensics. Digital Investigation, 3S, pp.37–43.

Casey, E., 2005. Case study: Network intrusion investigation - lessons in forensic preparation. Digital Investigation, 2, pp.254-260.
Chen, P.S., Tsai, L.M.F., Chen, Y., Yee, G., 2005. Standardizing the Construction of a Digital Forensics Laboratory. In First International Workshop on Systematic Approaches to Digital Forensic Engineering. Taipei, Taiwan, pp. 40-47.
Cleveland, G., 1995. Overview of Document Management Technology, International Federation of Library Associations and Institutions. Available at: http://archive.ifla.org/VI/5/op/udtop2/udt-op2.pdf [Accessed August 9, 2011].
Cook, R.I., 2002. How Complex Systems Fail. Available at: http://www.npsf.org/members/standup-old/download/articles-howcomplexsystemsfail.pdf [Accessed September 15, 2011].
Danielsson, J. & Tjøstheim, I., 2004. The need for a structured approach to Digital Forensic Readiness - Digital Forensic Readiness and e-commerce. In IADIS International Conference e-commerce 2004. Lisbon, Portugal, pp. 417-421.
Deloitte, 2010. 2010 Financial Services Global Security Study - The Faceless Threat, Deloitte. Available at: http://www.deloitte.com/assets/Dcom-Global/Local%20Assets/Documents/Financial%20Services/dtt_fsi_2010%20Global%20FS%20Security%20Survey_20100603.pdf [Accessed August 11, 2011].
Ernst & Young, 2010. Borderless security - Ernst & Young's 2010 Global Information Security Survey, Ernst & Young. Available at: http://www.ey.com/Publication/vwLUAssets/Global_information_security_survey_2010_advisory/$FILE/GISS%20report_final.pdf [Accessed August 11, 2010].
Grobler, C.P. & Louwrens, B., 2006. Digital Forensics: A Multi-Dimensional Discipline. In Information Security South Africa 2006. Johannesburg, South Africa. Available at: http://icsa.cs.up.ac.za/issa/2006/Proceedings/Research/62_Paper.pdf [Accessed July 19, 2010].
Grobler, C.P. & Louwrens, C.P., 2007. Digital Forensic Readiness as a Component of Information Security Best Practice. In H. S. Venter et al., eds. New Approaches for Security, Privacy and Trust in Complex Environments. IFIP Advances in Information and Communication Technology. Boston, pp. 13-24.
Grobler, C.P., Louwrens, C.P. & von Solms, S.H., 2010. A multi-component view of Digital Forensics. In Fifth International Conference on Availability, Reliability and Security (ARES 2010). Krakow, Poland, pp. 647-652.
Grobler, M.M. & Dlamini, I.Z., 2010. Managing Digital Evidence - The Governance of Digital Forensics. Journal of Contemporary Management, 7, pp.1-21.
Gupta, R., Hima Prasad, K. & Mohania, M., 2008. Automating ITSM Incident Management Process. In 2008 International Conference on Autonomic Computing. Chicago, IL, USA, pp. 141-150.
Jäntti, M., 2009. Defining Requirements for an Incident Management System: A Case Study. In Fourth International Conference on Systems. Cancun, Mexico, pp. 184-189.
Karlzén, H., 2009. An Analysis of Security Information and Event Management Systems - The Use of SIEMs for Log Collection, Management and Analysis. Masters Thesis. Gothenburg, Sweden: Chalmers University of Technology, University of Gothenburg.

Koen, R. & Olivier, M.S., 2007. An Open-Source Forensics Platform. In Southern African Telecommunication Networks and Applications Conference. Sugar Beach, Mauritius. Available at: http://www.satnac.org.za/proceedings/2007/papers/software/Paper%20113%20-%20Koen.pdf [Accessed May 15, 2012].
Kostina, A., Miloslavskaya, N. & Tolstoy, A., 2009. Information Security Incident Management Process. In Second ACM International Conference on Security of Information and Networks. North Cyprus, Turkey, pp. 93-97.
Kurowski, S. & Frings, S., 2011. Computational Documentation of IT Incidents as Support for Forensic Operations. In Sixth International Conference on IT Security Incident Management and IT Forensics. Stuttgart, Germany, pp. 37-47.
Lamis, T., 2010. A Forensic Approach to Incident Response. In Information Security Curriculum Development Conference 2010. Kennesaw, GA, USA, pp. 177-185.
Liang, F. et al., 2011. An Integrated Multi-channel Messaging Model Supporting for Business Collaboration. In 15th International Conference on Computer Supported Cooperative Work in Design. Lausanne, Switzerland, pp. 532–537.
Lindsay, A., Downs, D. & Lunn, K., 2003. Business processes — attempts to find a definition. Information and Software Technology, 45(15), pp.1015-1019.
Luoma, V.M., 2006. Computer forensics and electronic discovery: The new management challenge. Computers & Security, 25(2), pp.91-96.
Mehdizadeh, Y., 2005. Security Event Management. The ISSA Journal, May 2005, pp.18-21.
Metzger, S., Hommel, W. & Reiser, H., 2011. Integrated Security Incident Management — Concepts and Real-World Experiences. In Sixth International Conference on IT Security Incident Management and IT Forensics. Stuttgart, Germany, pp. 107-121.
Mitropoulos, S., Patsos, D. & Douligeris, C., 2006. On Incident Handling and Response: A state-of-the-art approach. Computers & Security, 25(5), pp.351–370.
Nguyen, F. & Skrabalek, J., 2011. NotX Service Oriented Multi-platform Notification System. In Federated Conference on Computer Science and Information Systems. Szczecin, Poland, pp. 313–316.
Nicolett, M., 2008. Critical Capabilities for Security Information and Event Management Technology, Gartner RAS Core Research. Available at: http://www.arcsight.com/collateral/Critical_Capabilities_Report_2008.pdf.
Nogueira, J. & Vasconcelos, W., 2008. Ontology for Complex Mission Scenarios in Forensic Computing. The International Journal of Forensic Computer Science, 3(1), pp.42-50.
Reddy, K. & Venter, H.S., 2009. A Forensic Framework for Handling Information Privacy Incidents. In G. Peterson & S. Shenoi, eds. Advances in Digital Forensics V. Fifth Annual IFIP WG 11.9 International Conference on Digital Forensics. Orlando, Florida: Springer, pp. 143-155.
Reddy, K., Venter, H.S. & Olivier, M., 2011. Using time-driven activity-based costing to manage digital forensic readiness in large organizations. Information Systems Frontiers. Available at: http://dx.doi.org/10.1007/s10796-011-9333-x.
Reekie, C. & von Solms, B., 2006. A Corporate Capital Protection and Assurance Model. In 2006 International Conference on On the Move to Meaningful Internet Systems:

AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET. Montpellier, France, pp. 546-553.
Richardson, R., 2011. 15th Annual CSI Computer Crime and Security Survey, Computer Security Institute.
Rowlingson, R., 2004. A Ten Step Process for Forensic Readiness. International Journal of Digital Evidence, 2(3).
Rowlingson, R., 2005. An Introduction to Forensic Readiness Planning, London, UK: National Infrastructure Security Co-ordination Centre.
Scheer, A. & Habermann, F., 2000. Making ERP a Success. Communications of the ACM, 43(4), pp.57-61.
Swift, D., 2006. A Practical Application of SIM/SEM/SIEM Automating Threat Identification, SANS Institute.
Tan, J., 2001. Forensic Readiness. Available at: http://www.arcert.gov.ar/webs/textos/forensic_readiness.pdf [Accessed September 7, 2010].
Taylor, C., Endicott-Popovsky, B. & Frincke, D.A., 2007. Specifying digital forensics: A forensics policy approach. Digital Investigation, 47, pp.101-104.
Trček, D. et al., 2010. Advanced Framework for Digital Forensic Technologies and Procedures. Journal of Forensic Sciences, 55(6), pp.1471–1480.
von Solms, S., Louwrens, C., Reekie, C., Grobler T., 2006. A Control Framework for Digital Forensics. In M. S. Olivier & S. Shenoi, eds. Advances in Digital Forensics II. IFIP Advances in Information and Communication Technology. Boston, pp. 343-355.
Wang, S., 2007. Measures of retaining digital evidence to prosecute computer-based cyber-crimes. Computer Standards & Interfaces, 29, pp.216-233.
Wu, Q., Gu, Y., Cui, X., Moka, P., Lin, Y., 2010. A graph similarity-based approach to security event analysis using correlation techniques. In IEEE Global Telecommunications Conference 2010. Miami, Florida, USA, pp. 1-5.
Yasinsac, A., Erbacher, R.F., Marks, D.G., Pollitt, M., Sommer, P.M., 2003. Computer Forensics Education. IEEE Security & Privacy, 1(4), pp.15–23.
Yasinsac, A. & Manzano, Y., 2001. Policies to Enhance Computer and Network Forensics. In Proceedings of the 2001 IEEE Workshop on Information Assurance and Security. New York, USA, pp. 289-295.