UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Digital Forensic Readiness in Mobile Device Management Systems

Author
Elsabé Ros

Supervisor
Professor HS Venter

Submitted in fulfilment of the requirements for the degree
Master of Science (Computer Science)
in the
Faculty of Engineering, Built-Environment and Information Technology
at the
University of Pretoria

Abstract

Mobile devices have become very popular, and virtually everyone owns a smart device. As more employees became owners of smart devices, the organisations were put under pressure to allow employees to use their smart devices for work purposes, or alternatively provide employees with smart devices.

Most organisations opted for a Bring Your Own Device policy, where employees use their own smart devices for work purposes, with the organisation reimbursing some of the costs. Adopting such a policy introduced risks into the organisations, since the organisations do not own and do not have direct control over employees' personal devices.

One of the most widely used solutions to this problem is Mobile Device Management (MDM) software, which is installed on employees' devices and prevent them from taking actions that may be harmful to the organisation.

This leads us to the problem statement of this research. Since MDM systems are purely preventative and devices are not owned by the organisation, it is expensive and sometimes impossible for organisations to retrieve potential evidence from the devices when an incident occurs.

This research proposes a model to solve this problem by introducing a digital forensic readiness component into an MDM system. Adding digital forensic readiness to an existing MDM solution reduces costs by collecting evidence when suspicious activity is detected, reducing investigation times and legal costs involved in collecting evidence.

A prototype was created to show that the proposed model could be implemented in practice. The prototype shows how this solution can be utilised to collect data from devices and utilise it in an investigation.

Finally, the research and prototype are critically evaluated, and the benefits and shortcomings of such a solution are presented. The author also addresses privacy concerns arising from the data collection component.

Keywords

mobile, mobile device, forensics, forensic readiness, digital forensic readiness, mobile device management, mdm, bring your own device, byod, android, data collection

Dedication

To the people whose sacrifices made this possible: my parents and grandparents. This research is especially dedicated to my grandfather, Evert Jan Bron.

A man who works with his hands is a labourer; a man who works with his hands and his brain is a craftsman; but a man who works with his hands and his brain and his heart is an artist.
-Louis Nizer

Acknowledgements

My thanks to everyone who made this possible. Specifically:

- My supervisor, **professor HS Venter** - thank you for your guidance and support.
- My parents, **Martin and Louise Ros** for all their support and sacrifices
- My **employers**, for allowing me the freedom to pursue this research
- The **University of Pretoria**, for its financial support
- Finally, the **Heavenly Father** without whom none of this would be possible

Declaration

Contents

List of Figures

List of Tables

PART I

Introduction and problem statement

Part 1 introduces the reader to the problem statement of this research, as well as the motivation and layout. Additionally, the author gives a brief overview of the design science methodology and describes how this methodology is applied in this document.

CHAPTER 1

Introduction

1.1 Introduction

The internet is an insecure place. As organisations started to utilise the internet more and more to conduct business, they implemented security measures that reflected this fundamental insecurity. Since the organisations controlled both the computing devices and the networks they used to connect to the internet, it was possible for the organisations to prevent employees from accessing potentially dangerous resources and exfiltrating sensitive information from the company network.

However, as smartphones became more popular and widespread over time, employers were put under more pressure to either provide employees with work-specific smartphones or allow them to use their own smartphones.

Since most employees already had some form of smart device, most companies adopted a Bring Your Own Device (BYOD) policy, allowing employees to use their personal digital devices, like phones and tablets, for work-related tasks. A BYOD policy is advantageous for both employer and employee, as the employer has fewer costs related to devices and the employee does not have to manage multiple devices.

The BYOD policy, however, did introduce risks into organisations. Since the organisation did not have direct control over employees' devices, the risk of a third-party compromising the organisation became much higher. For example, a malicious actor could infect an employee's device with a piece of malicious software (malware) that exfiltrates sensitive information or an employee's device could introduce malware (like ransomware [10]) into the organisation's network.

In response to the threat of external actors and third-party software, companies put Mobile Device Management (MDM) solutions in place. These systems are installed on an employee's personal smartphone to limit the actions they can take on the device. Employees are typically not allowed to access company resources without the MDM software installed. However, these MDM systems do not include a digital forensic readiness component.

The next section presents the problem statement of this research. After that, the motivation, objectives, methodology and layout of the research is introduced.

1.2 Problem statement

The problem statement of this dissertation is that most MDM systems do not include a digital forensic readiness component, leaving investigators with little to no device-related historical data when an incident does occur.

Mobile Device Management solutions focus on being preventative, stopping users from performing potentially unsafe actions. They do this by preventing users from accessing online resources and applications that have been deemed unsafe. However, adding a digital forensic readiness component has the potential to make the software much more powerful, since it allows for a more detailed investigation after an incident has occurred.

A digital forensic readiness component focuses on gathering data before an incident occurs, allowing investigators and other a stakeholders a much clearer picture of the events that lead up to the incident. Such a digital forensic readiness component does raise some privacy concerns, since it is possible for the digital forensic readiness component to collect highly personal data, but because this problem mainly occurs in large organisations, it can be controlled using privacy policies and will not be addressed in depth in this research, as it has been thoroughly addressed in literature (see for example Kopp [44], Smith and Faley [90] and Watkins et al [99]).

The next section presents the motivation for this research in more detail.

1.3 Motivation

Many companies and organisations deal with sensitive information and, in some cases, information that is protected by law (intellectual property or personal information belonging to customers). Keeping this information from falling into the hands of unauthorised third parties is a complicated task. Many tools and techniques can be deployed to keep external actors from gaining access to this information, but defending against employees (also known as an "insider threat") is more complicated.

An employee might have access to sensitive information belonging to their employer. This problem can be partially mitigated by only giving employees access to the information they need to do their job. Limiting access to information reduces the impact on the organisation if a disgruntled employee decided to access sensitive information for nefarious purposes, but it is still necessary to put controls in place to allow investigators to form a full picture of an incident.

For example, a competitor acquires some sensitive intellectual property (IP). Through an internal investigation, a specific employee is implicated in the theft of this IP. If an MDM with a digital forensic readiness component were in place, the employer would have been able to prove that the employee did, in fact, arrange with the competitor to sell the secrets to them.

In summary, the primary motivation for this study is to allow organisations to collect digital forensic readiness data from the personal devices of people that interact with their sensitive systems and information, in order to effectively investigate any data breaches.

The next section presents the methodology of this research.

1.4 Methodology

In this research, the author makes use of design science to create various artefacts, namely constructs, models and methods, as well as instantiations. First, design science constructs are created, forming the "language" of the research. Then, models and methods are created from the constructs, and finally, instantiations of the artefacts are created to evaluate their efficiency. Chapter 2 presents the design science paradigm in more details and expands on how it is implemented in this research.

Next, the layout of this research is introduced to provide the reader with a birds-eye view of the research.

1.5 Layout

This dissertation is divided into five parts with 14 chapters, as well as three appendices. Part 1 introduces the reader to the research and contains two chapters. Chapter 1 introduces the problem statement, along with a motivation and some objectives for the research.

Chapter 2 introduces the reader to the design science methodology, describing the various artefacts and processes that form part of this methodology. This chapter also describes how a design science methodology is applied throughout this research.

Part 2, the background, contains three chapters covering the background information necessary to this research.

Chapter 3 does a review of the literature around the Bring Your Own Device (BYOD) phenomenon. To understand the problem space, it is necessary to survey the topic of BYOD, since this is the space where the problem is most prevalent. The author defines "BYOD" and presents the advantages and disadvantages of adopting such a policy.

Chapter 4 gives an overview of Mobile Device Management (MDM) systems, defining the concept and presenting the architecture of such a system. MDM systems are used to mitigate some of the risks around BYOD policies, and it is important to understand these risks and their mitigations, before proposing a digital forensic readiness solution that can be added to the MDM software. Additionally, understanding the structure of existing MDM software will make the digital forensic readiness solution more robust, since the understanding will reflect in the design of the solution.

Chapter 5 defines digital forensics and presents the digital forensic process. Additionally, it covers digital forensic soundness and how data may be collected in a forensically sound manner. The author also introduces Digital Forensic Readiness (DFR) and the benefits and drawbacks of implementing DFR in an organisation.

The third part of the research is divided into four chapters that introduces the solution to the problem stated before.

Chapter 6 introduces a model of a solution to solve the stated problem. The components of the model are presented. This chapter also describes data integrity measures and how the data flows through the system.

Chapter 7 introduces and presents the design science methods that enhance the high-level components that have been introduced in the previous chapter.

Chapter 8 defines an architecture for the model that was presented in the previous chapters, showing how such a system will look at a high level. The three main components (mobile client, server and console) are broken down into sub-components, and the program flow for each component is analysed. Additionally, this chapter shows how the data storage components are structured.

Chapter 9 evaluates the proposed model and architecture according to the umbrella standard for digital forensics, ISO 27043, by examining how it enables the various processes in this standard.

Part 4 of the research presents the prototype that was implemented using the solution proposed in Part 3 and contains two chapters.

Chapter 10 starts off the section dedicated to the prototype by defining the requirements that a successful solution will need to adhere to. Business, functional, non-functional and user interface requirements are identified and analysed.

Chapter 11 introduces a prototype of the solution introduced in the previous chapters, showing how the digital forensic readiness may be added to an existing solution. This chapter describes the prototype, showing how the concepts described in the previous chapters can be implemented in practice and using a scenario to demonstrate the prototype to the reader.

The final part of this research, part 5, concludes the research in three final chapters.

Chapter 12 evaluates the solution presented for adherence to the requirements stated in chapter 10. It analyses how well the prototype aligns to the various requirements and where additional work is needed to fulfil the requirements.

Chapter 13 critically evaluates the proposed solution. It analyses the ability of the solution to defend against common threats and identifies the benefits and shortcomings of the solution. It presents the privacy concerns that came up during the course of the research, points the reader to related work and describes the contribution of this research to the state of the art.

Chapter 14 summarises the research, revisits the problem statement and presents opportunities for future work. The research is concluded with a final statement.

The structure of this dissertation is shown graphically in figure 1.1. The next section introduces design science and proposes the methodology used in this research.

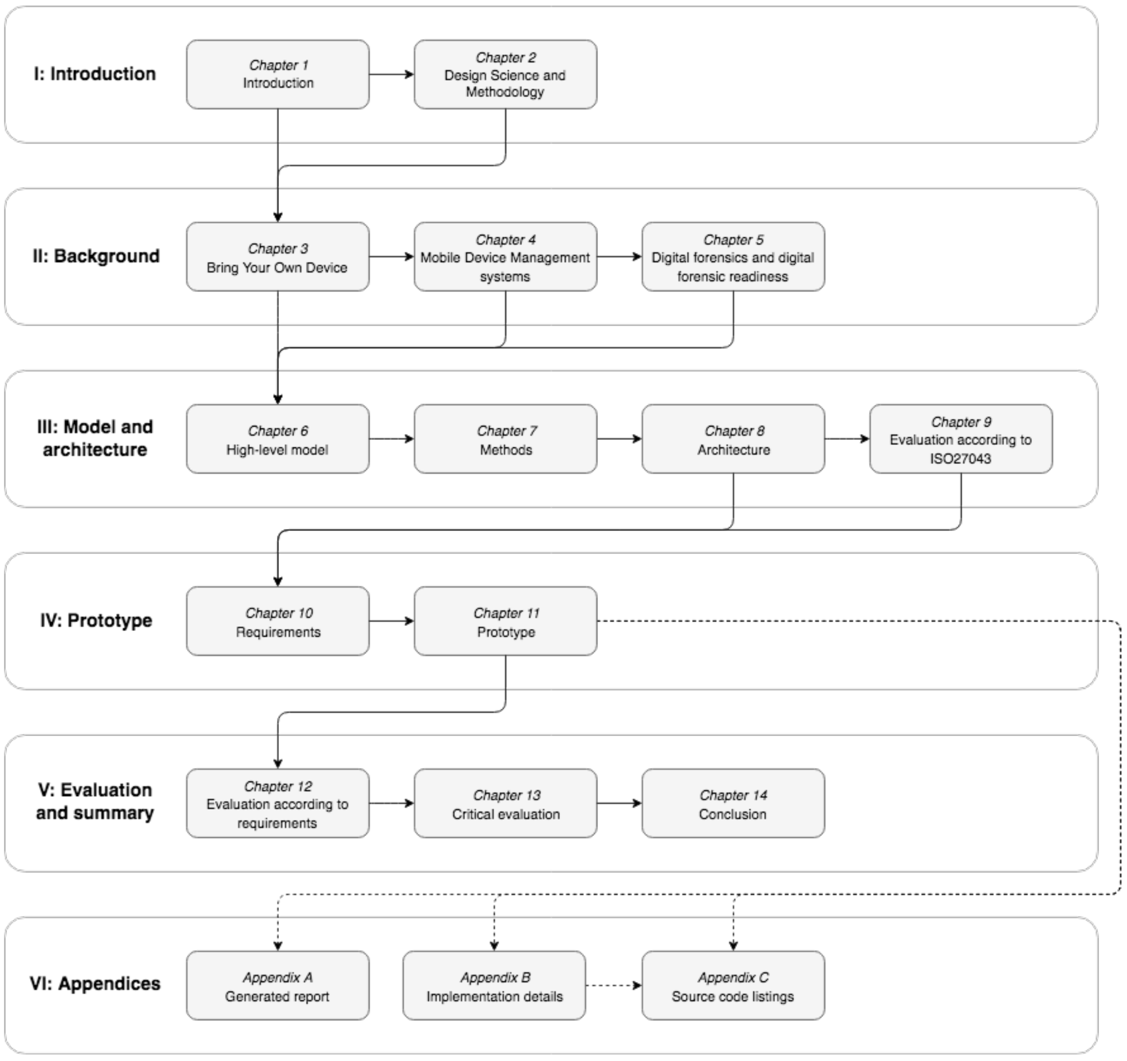


Figure 1.1: Dissertation structure

CHAPTER 2

Design Science and Methodology

2.1 Introduction

As mentioned in the previous chapter, the author makes use of a design science approach to arrive at a solution to the stated problem. This chapter gives an overview of design science and discusses the methodology followed in this research. First, the author discusses and describes design science.

2.2 Design Science

Design Science aims to create innovative and creative solutions [96] to non-natural organisational problems [7]. It creates and evaluates artifacts intended to solve identified problems for organisations [70] in order to create new and innovative solution [39].

Design science has two main processes, namely "Construction" (also called Build and Design) and "Evaluation" [38], and four different artifacts: constructs, models, methods and instantiations [53]. The artifacts are constructed and evaluated during the processes in order to solve an unsolved problem [14] as shown in figure 2.1.

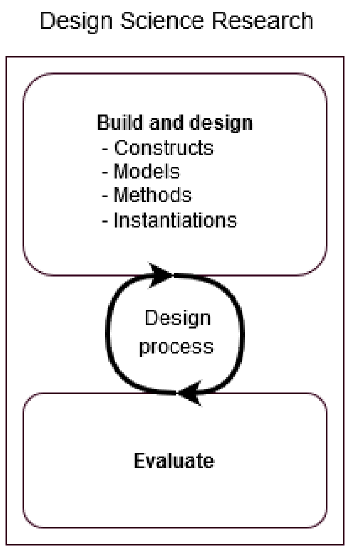


Figure 2.1: Design science process [38]

The rest of this section presents the two processes that are followed and four artefacts that are produced in the design science process.

The two main design science processes are **construction** and **evaluation**. The **construction process**, also called “build,” refers to the construction of the designed artifacts, showing that it is possible for such an artifact to be built [54, 97].

The **evaluation process** develops criteria to evaluate the artefacts produced by the design science process and examines the artefacts that have been produced to assess them according to the criteria, attempting to quantify how well the designed solution works [74].

These two processes both involve the artefacts mentioned previously, namely constructs, models, methods and instantiations. These artefacts are presented next.

2.2.1 Constructs

Constructs, also known as concepts, describes the problem domain. It creates a shared language and knowledge base that allows for a specialised vocabulary for a domain [97].

Theses constructs may be formal or informal, depending on the problem domain, and may include the definition of entities, attributes, relationships and constraints on the system.

These constructs are fundamental to design science, as they define the terminology that is used when discussing and thinking about the problem space. The way the constructs are defined shapes the way the problem is thought about and, by extensions, the resulting solutions.

Once the constructs in a solution has been identified, models of the system can be created. These are presented next.

2.2.2 Models

In design science research, a model defines the relationships between constructs using propositions or statements. The model may be seen as a solution to a task, expressing “how things are” or, in the case of a new system, how things should be [54].

A model shows the structure of reality, but may be vague or inaccurate on the details, using approximations to simplify the representation to be useful.

Once the constructs and models of a system have been defined, the designer can start working towards solving tasks by creating methods. Next, these methods are presented.

2.2.3 Methods

A method is a series of steps that are executed to achieve a task to address a need. Also known as algorithms, methods are based on the constructs and models that were used to define the problem space [97].

Methods make use of constructs and models, using them as input or transitioning between models as it moves through the steps. Methods may also translate between models and constructs.

The method used may also influence the design of the constructs and models used in a design science system. If a particular method is desired, the models and constructs may be defined in such a way that the given method can be used.

Once the constructs, models and methods have been defined, the next step is to create instantiations, which will be introduced next.

2.2.4 Instantiations

The instantiation is the final result of a design science project. This is the "realization of an artifact in its environment" and brings constructs, models and methods into operation [54, 97].

Instantiations shows the feasibility and effectiveness of the other artefacts that have been designed and allows insight into the efficiency of the solution, allowing for the refinement of the other artefacts in the design science solution.

These artefacts that are produced, along with the processes of "build" and "evaluate", allow a designer to create a solution that solves the problem at hand. Next, the author analyses how design science is applied to this research by presenting the methodology of the research.

2.3 Methodology

As mentioned in chapter 1.4 the author makes use of a design science methodology to solve the problem as mentioned in the previous chapter, namely that MDM systems are purely preventative.

During the design science "build and design" process, the author identified and produced the artefacts that was presented in the previous section, namely constructs, models, methods and instantiations.

Constructs in the system are identified by means of a literature review. Part II (chapters 3 to 5) is dedicated to establishing the terminology that is used throughout the research.

The **models** and **methods** for a solution are built and designed in chapters 6 and 8. These chapters build up the relationships between the identified constructs and outline algorithms to achieve the stated tasks.

Finally, chapter 11 presents and **instantiation** of the constructs, models and methods that have been identified, bringing the concept into operation and allowing the author to evaluate the solution.

In addition to the "Build and design" process, design science also makes provision for **evaluating** the artefacts produced from the design process. In chapter 10 the author identifies a number of requirements to evaluate the finished product.

Chapter 9 evaluates the identified **models** and **methods** against the existing umbrella standard for digital forensics, ISO 27043. Finally chapters 12 and 13 evaluates the whole solution as presented on a number of criteria.

2.4 Conclusion

This chapter presented the discipline of design science and how it is applied to this research. Using the design science methodology to approach this research allows the author to apply the scientific method to a non-natural problem.

In the next part of the research, the author gives background information that is required for the solution to be successful, building up the constructs that are required for a successful model and instantiation.

PART II

Background

Part 2 gives a background to the research, presenting Bring Your Own Device, Mobile Device Management systems, digital forensics and digital forensic readiness. This part of the research familiarises the reader with the concepts used throughout the research.

CHAPTER 3

Bring Your Own Device

3.1 Introduction

As mentioned previously, the growing popularity and ubiquitousness of smart devices forced organisations to either issue company-owned devices to employees or to adopt a Bring Your Own Device policy, allowing employees to connect personal devices to the organisation's network and access the organisation's (potentially sensitive) information from those devices.

This chapter describes the concept of Bring Your Own Device and presents the advantages and disadvantages that go along with adopting such a policy, establishing the design science constructs for the BYOD domain.

3.2 What is Bring Your Own Device?

According to Disterer et al. [26], BYOD can be described as " the circumstance in which users make their own personal devices available for company use". A Gartner survey found that, unlike laptops, very few workers (23% of those surveyed) receive corporate-issued mobile devices. This means that the majority of mobile devices in a corporate environment is personally owned [2, 41].

Adopting a policy of BYOD has several advantages for both the organisation and its employees. The next two sections cover the advantages and disadvantages of adopting a BYOD policy. These advantages and disadvantages are listed below and after that the author analyses each of the advantages and disadvantages individually.

Advantages of adopting a BYOD policy

- **One device** Employees have to manage only one device.
- **Costs for employer** The employer does not have to buy a device for all employees
- **Flexibility** Employees can work from other locations
- **Familiarity** Employees can use devices they are already familiar with

Disadvantages of adopting a BYOD policy

- **Different devices** Employers have to cater for a number of different devices.
- **Costs for employee** Employees have to purchase a device that complies with the organisation's requirements.
- **Security** Personal devices are harder to protect and secure than company equipment.

3.3 Advantages of adopting a BYOD policy

Implementing a BYOD policy has a number of advantages, both from the organisation's and the employee's view. This section gives an overview of the advantages that BYOD offers to both organisations and employees.

3.3.1 One device

The main advantage of BYOD from an employee's perspective is the fact that they have to manage only one device, allowing the employee to choose a device with which they are familiar and comfortable. In addition to selecting a familiar device, the employee does not have to expend effort into managing both a personal and a work-related device [26].

3.3.2 Costs

Since the employee uses their personal device for work-related tasks, the organisation does not have to purchase mobile devices for their employees to use [46, 65]. An additional factor here is that people tend to take better care of their own property [89].

3.3.3 Flexibility

Since employees are allowed and even encouraged to access their employer's systems and resources from their personal device, employees can now work from other locations than their office [46, 48].

3.3.4 Familiarity

Employees are familiar with their own devices and will be able to complete tasks faster and more accurately using their personal devices. This way the loss of productivity incurred by learning how to use a new device is avoided [3].

This section presented the advantages of adopting a BYOD policy. When designing a solution to reduce the disadvantages of BYOD, care should be taken not to negate the advantages.

As outlined, there are numerous advantages to adopting a BYOD policy. However, there are also several disadvantages to such a decision. The next section presents some of these disadvantages.

3.4 Disadvantages of adopting a BYOD policy

There are a number of disadvantages to adopting BYOD, both from the employer's and employees' sides. This section analyses some of the disadvantages of adopting a BYOD policy, as listed previously.

3.4.1 Costs for employees

BYOD shifts the costs of the mobile device to the employee. Not all employees may be willing to shoulder the costs for a device and any costs arising from usage of the device, especially if the device is mainly used for work purposes [89].

3.4.2 Different devices

When a company buys devices for their employees, they can pick a device that fits the needs of the organisation. When employees make use of their own devices, the employee has the choice of which device to use. This means that the organisation has to support a large number of different devices with different operating systems and hardware profiles [3, 65], including devices that have constrained resources or old versions of software. This puts a burden on the organisation to ensure that all of these devices can access the organisation's resources.

3.4.3 Security

Organisations take many steps to protect their systems and data from access by unauthorised third parties(for example, disk encryption and virtual private networks). However, employees may not have the same protections on their personal devices. These unprotected devices may give a malicious third party an easier way into an organisation's network. The possibility of physical theft also exists, which can also put sensitive information at risk [45].

Allowing (and even encouraging) employees to access the organisation's systems with their personal devices creates risks that an organisation has to accept before continuing with the adoption of a BYOD policy. The rest of this section introduces the risks involved in allowing employees to use their personal mobile devices.

Mobile devices that are unsecured or not properly secured can lead to sensitive systems and data being compromised or manipulated [26]. This compromise can happen because of a disgruntled employee [3] or by an employee's device being compromised [21].

It is also possible for employees to introduce malware, such as spyware or ransomware, into an organisation's systems [11]. Introduction of malware into an organisation's network can have widespread repercussions, and great care must be taken against malware. These disadvantages, and especially the security risks, need to be kept in mind when designing a solution for organisations that have adopted a BYOD policy.

3.5 Conclusion

This chapter presented BYOD, as well as the advantages and disadvantages that adopting such a policy brings. One of the most widely used solutions to reduce the risks around BYOD is the use of a Mobile Device Management (MDM) solution [26]. The next chapter is dedicated to examining Mobile Device Management. Although Mobile Device Management systems solve some of the challenges that are introduced by a BYOD policy, it does not solve its problems.

CHAPTER 4

Mobile Device Management systems

4.1 Introduction

To mitigate some of the risks that come with personal devices, many organisations deploy a Mobile Device Management system to personal devices before allowing them to access the organisation's network and sensitive resources [26]. Since MDM systems are integral to the problem and, by extension, to the solution, it is important to establish the design science constructs for mobile device management.

This chapter defines Mobile Device Management, followed by an overview of the typical architecture of MDM systems, introducing the components that are present in these systems. After that, the author presents how such a system would be deployed to organisations, describing the phases involved in such a deployment. Finally, the chapter concludes by giving an overview of the threats that can be leveraged against a Mobile Device Management system.

4.2 Defining Mobile Device Management

There are a number of definitions and descriptions of Mobile Device Management in existing literature. Much of the literature drops the word "Mobile" and simply use the phrase "Device Management," since most devices that require such a system are mobile. The author will be using the two phrases interchangeably during this research.

Kravets et al [45] describes Mobile Device Management as the administration, monitoring, integration and securing of mobile devices to optimise security and functionality of the mobile devices while protecting the organisation.

Bui et al. mentions that MDM systems are multi-component software solutions used by the IT departments of an organisation to achieve these goals [16]. The system is applied across multiple device operating systems and service providers. These systems are applied to ensure the security of the organisation's network and data [72], while simultaneously allowing employees to optimally perform their work [78].

These various MDM definitions have been harmonised by the Open Mobile Alliance as part of their initiative to standardize MDM systems [52]. The Open Mobile Alliance defines Mobile Device Management as follows [4]:

> "Device Management refers to the management of Device configuration and other managed objects of Devices from the point of view of the Management Authorities. Device Management includes, but is not restricted to setting initial configuration information in Devices, subsequent updates of persistent information in Devices, retrieval of management information from Devices, execute primitives on Devices, and processing events and alarms generated by Devices."

In addition to the core functions mentioned in the previous paragraphs, many MDM systems also offer additional functionality, like remote administration, asset management, remote lockout and installing updates, both at system and application level [65]. The tools may also offer features such as file synchronisation and in-application tech support [45].

To allow an MDM system to function optimally, it requires a number of different components. The next section describes these components and the communication.

4.3 Architecture of an MDM system

Figure 4.1 shows the four main components in an MDM system. The Open Mobile Alliance identifies three components that are essential to implement a Device Management (DM) system: a **DM server** (1), a **data repository** (2) and a **device client** (3) [4]. Rhee et al. add a **management console** [80] - numbered (4) in figure 4.1. The following subsections give an overview of these four components.

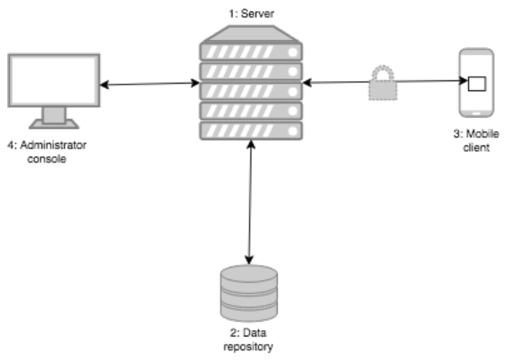


Figure 4.1: Mobile Device Management components

4.3.1 Server

The Device Management server is a software component that interacts with the clients installed on mobile devices [87]. This server is responsible for sending the device management settings to the client and keeping track of registered devices [80]. The initial device management settings, as well as updated settings, can be distributed to the mobile clients either by utilising a direct connection or via a push notification [79].

4.3.2 Data repository

The data repository is a data store that the Device Management server can read and write management configurations to [4]. This data repository can also be used to store the logs from configuration events [16], as well as associations between users and devices [42].

4.3.3 Mobile Client

The device client (also referred to as an agent, mobile device client or mobile client) is the component that runs on the mobile device [28]. This component (also called the agent) is responsible for applying management configurations on the device and transmitting whether or not the settings were applied back to the server [45]. The device client is usually OS-specific since it needs to access API's that are unique to each operating system [91].

4.3.4 Administrator console

The management console is an application that allows a stakeholder to access and manage the device management system and settings [32]. This application is used to modify the device management settings and to fine-tune the MDM system.

Once all the components are in place and configured, the system can be deployed in an organisation. The next section presents the steps involved in successfully deploying an MDM solution across an organisation.

4.4 Deploying an MDM solution

Once an organisation has decided to make use of an MDM system, the system needs to be deployed in the organisation. Rhee et al. [80] identify five discrete steps in the deployment of an MDM system. These five steps are:

- Configuration of the MDM system
- Installation of the mobile client
- Authentication of the mobile client
- Instruction from the server to the mobile client
- Reporting from the mobile client to the server

Each of these five steps is presented in more detail in the following sections.

4.4.1 Configuration of the MDM system

Before the MDM client is installed on the mobile devices, the device management policies are configured from the administrator consoles. Additionally, the devices that will be receiving the client software is registered on the server at this point [80]. This step is mostly executed by the system administrators with input from other stakeholders.

4.4.2 Installation of the mobile client

In this step, the MDM client is distributed to and installed on mobile devices [80]. After installation, the client retrieves the device management configuration from the MDM server.

The Open Mobile Alliance [6] identifies four different methods of receiving (bootstrapping) the initial device management configuration, namely

- **a factory image**. The MDM client and configuration is added directly to the device's factory image.
- **a smartcard**. The MDM configuration is loaded securely from a Smartcard directly onto the devices.
- **initiation by the client**. The MDM client contacts the server and securely loads the MDM settings.
- **initiation by the server**. Before the MDM client is activated, it is registered with the server. Once the MDM client is activated, the server initiates the process of setting up the mobile client.

In practice, the factory image and smartcard methods are not used often, as they require access to physical hardware or proprietary software not owned by the organisation. More often, the organisation will require the user to install the MDM client, after which the configuration of the client will be initiated either by the server or the client.

Once the MDM client has been installed and the MDM client configured, the MDM client needs to authenticate the user that owns the device. This is presented in the next section.

4.4.3 Authentication of the mobile client

After the MDM client has been installed, using one of the mechanisms explored in the previous section, it contacts the MDM server with certain identifying elements (for example, the device's unique identifier - the International Mobile Equipment Identity number, which uniquely identifies every device ever manufactured). If the devices have been registered on the server as part of the configuration step, the server can take the additional step of verifying the identifying information [80].

Additionally, an employee may be required to log in to the system with their company credentials to allow the system to associate the device with a specific user [16]. This process is shown in figure 4.2.

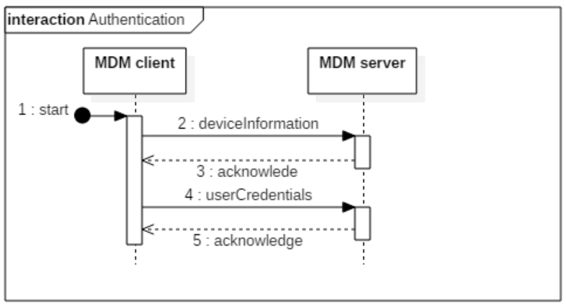


Figure 4.2: MDM Authentication flow

4.4.4 Instruction from the server to the mobile client

The Device Management server sends the management settings to each mobile device client that connects to it. Additionally, the server can also send other commands to the client (for example, remote wipe) [5]. This instruction session is initiated by the client, although the server may use a notification mechanism to prompt the client to initiate a session [6]. The instruction process is shown in figure 4.3 (steps 1, 2 and 3).

4.4.5 Reporting from the mobile client to the server

The Device Management client controls the device based on the management settings it received in the previous step and reports back to the server [5]. This may or may not be part of the instruction journey mentioned in the previous section, depending on the system configuration [6]. Figure 4.3 shows the communication flow between a client and the server if the Reporting step is included in the communication.

The purpose of an MDM solution is to protect the organisation's network and information from malicious third parties. However, the MDM system itself also exposes an attack surface and can be exploited to gain access to the exact resources it was designed to protect. The next section covers some of these threats.

4.5 Threats to MDM systems

MDM systems are usually installed to enhance the security of the devices and to prevent security incidents. However, the MDM system itself can also become an attack surface, and the developers of these systems need to be aware of the most common threats.

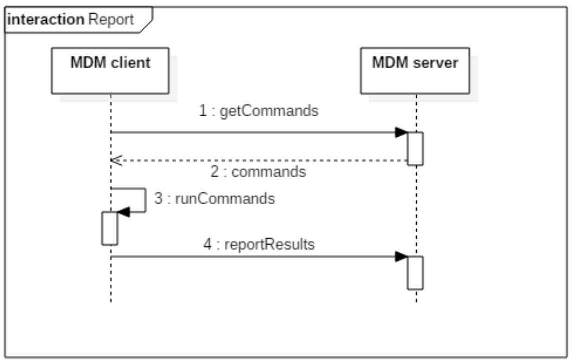


Figure 4.3: MDM Instruction and Report flow

Rhee et al [80] divides the threats to MDM systems into six categories. These six categories are:

- Spoofing
- Tampering
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privilege

Leung [49], adds another to the list, namely

- Malware

and Steiner [92] mentions another threat,

- Users

All of the threats mentioned above attack the confidentiality, integrity or availability of the MDM system. The rest of this section is dedicated to presenting the threats listed above and how they apply to MDM systems.

4.5.1 Spoofing

Malicious third parties can attempt to masquerade as a legitimate entity by replaying old interactions [30]. This entity can pretend to be either the mobile client, to trick the server into disclosing information, or it can pretend to be the server, to get the mobile client to perform some action [49]. It is also possible for a malicious entity to replay some administrator actions to gain access to a higher level of privilege [98].

4.5.2 Tampering

A malicious third party can attempt to modify interactions as they are happening, for example, to modify key values or to inject malicious commands [49].

A malicious entity can also tamper with the hardware, software or operating system in a mobile client to allow them to gain access to the device client software and locally stored data or to prevent the software from functioning correctly [36].

In the case of an MDM system, the user may attempt to circumvent the restrictions of the system by modifying the client software or changing the operating system to prevent the MDM system from functioning correctly.

4.5.3 Repudiation

A malicious user of the system may attempt to hide evidence of their actions by manipulating the data stored on the mobile client or in the data store. An entity may also attempt to modify the system to give them privileges that they are no longer entitled to [80].

It is also possible for a malicious entity to flood the system with data, with the result that essential data and security-related events cannot be easily distinguished from the data originating from the attacker [71].

The MDM system is vulnerable to repudiation since it collects data about a user's actions. If a malicious entity could generate a large amount of data, they could make it very time-consuming and nearly impossible to find the relevant data in the flood of generated data.

4.5.4 Information disclosure

Third parties can gain access to sensitive information by either eavesdropping on communications or by gaining physical access to the device [49]. A malicious entity can also attempt to gain access to the data storage directly [30]. Depending on the intentions of this entity, they may decide to leak this information to other parties [59].

An MDM solution is vulnerable to information disclosure, since the mobile clients are not under the direct control of the organisation at all times. This means that a malicious third party could gain access to information by means of a mobile client.

4.5.5 Denial of service

A malicious entity can manipulate data or communications in an attempt to prevent the MDM system from functioning as intended. This can be achieved by either inducing a system failure [100] or by flooding the system with work [71].

A Denial of Service attack can also be in the form of hindering an ongoing investigation by corrupting, deleting or otherwise tampering with the system's data [80].

4.5.6 Elevation of privilege

A malicious entity (either an external party or a user) can manipulate the data and communications in the system to attempt to gain access to parts of the system that they should not have access to [69]. This can be used to hamper the functionality of the MDM system or the attacker could attempt to remove incriminating evidence.

4.5.7 Malware

Even with all the checks and balances that an MDM system contains, it is still possible that a user will install malicious software on their device. This malware may perform a number of attacks, such as stealing credentials or encrypting the device and demanding a ransom [30].

4.5.8 Users

One of the main threats from users is the possibility of credential sharing. This may or may not be with malicious intent, but it still compromises the integrity of the MDM system [49].

Employees may also try to circumvent the system because of its strictness, perceiving the system as too restrictive. The employees may choose not to access company resources from their personal devices [92].

4.6 Conclusion

This chapter defined MDM and described the components and flows in a typical MDM solution. It also presented the ways that an MDM solution may be threatened.

MDM systems help organisations to protect their data and systems by preventing employees from accessing malicious and compromised resources. However, these systems are purely preventative and do not allow for investigations or a historical overview of device usage. To allow investigators to use evidence from these phones, another mechanism is required.

The standard approach to investigations is to collect digital evidence after an incident occurred. However, it is also possible to collect potential evidence before an incident occurs. This process is known as Digital Forensic Readiness and it, as well as the Digital Forensic process, is presented in the next chapter.

CHAPTER 5

Digital Forensics and Digital Forensic Readiness

5.1 Introduction

With the widespread use of computers, mobile phones and other digital devices, digital evidence is taking a more prominent role in legal proceedings. To deal with this, the field of Digital Forensics was developed. This chapter gives an overview of Digital Forensics, as well as the concept of digital forensic readiness, since the proposed model integrates Digital Forensic Readiness into an existing solution.

5.2 Digital Forensics

This section gives an overview of Digital Forensics, starting with the definition and then giving a summary of the digital forensic process, as laid out by the International Organization for Standardization (ISO). This section also presents the concept of Forensic Soundness.

5.2.1 Defining Digital Forensics

During the *First Digital Forensic Workshop*, Palmer et al [66] formulated the following definition for Digital Forensics, which is generally accepted as the standard definition (see for example Reith et al [77], McKemmish [56], Trenwith and Venter [94] and Carrier [17]):

> "The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorised actions shown to be disruptive to planned operations."

Since digital devices, such as computers and smartphones, are used by the majority of people, it follows that these devices will contain evidence (for example, text messages or emails) when a crime has been committed.

In summary, Digital Forensics is the process of acquiring and interpreting evidence from devices, while following a process that will prevent contamination and allow for the evidence to be admissible in the legal process [23].

There are a number of discrete processes in a Digital Forensic investigation, which will be presented in the next section.

5.2.2 The Digital Forensic process

A number of processes and frameworks for digital forensic investigations has been developed, for example, Kohn [43], which defined a framework for investigations, Martini and Choo [55] focusing on a framework for cloud computing and Cohen [24] creating a framework for network forensics, as well as many others. All these disparate frameworks were unified in ISO 27043 [40], which the author will be using throughout this research.

ISO 27043 [40], the standard for Incident investigation principles and processes, identifies five different types of processes, namely

1. readiness processes
2. initialisation processes
3. acquisitive processes
4. investigative processes
5. concurrent processes

The ISO 27043 is reproduced in figure 5.1. This section presents the standard, starting with the readiness processes. For the reader's convenience, the processes are numbered in both figure 5.1 and the text.

5.2.2.1 Readiness processes

Readiness processes are the processes that are followed to ensure that an organisation can execute a digital forensic investigation when the need arises. These processes ensure that systems are in place and staff are trained to handle forensic investigations correctly [19]. The readiness process is divided into twelve subprocesses:

1. Scenario definition
2. Identification of potential digital evidence sources
3. Planning pre-incident gathering, storage, and handling of data representing potential digital evidence
4. Planning pre-incident analysis of data representing potential digital evidence

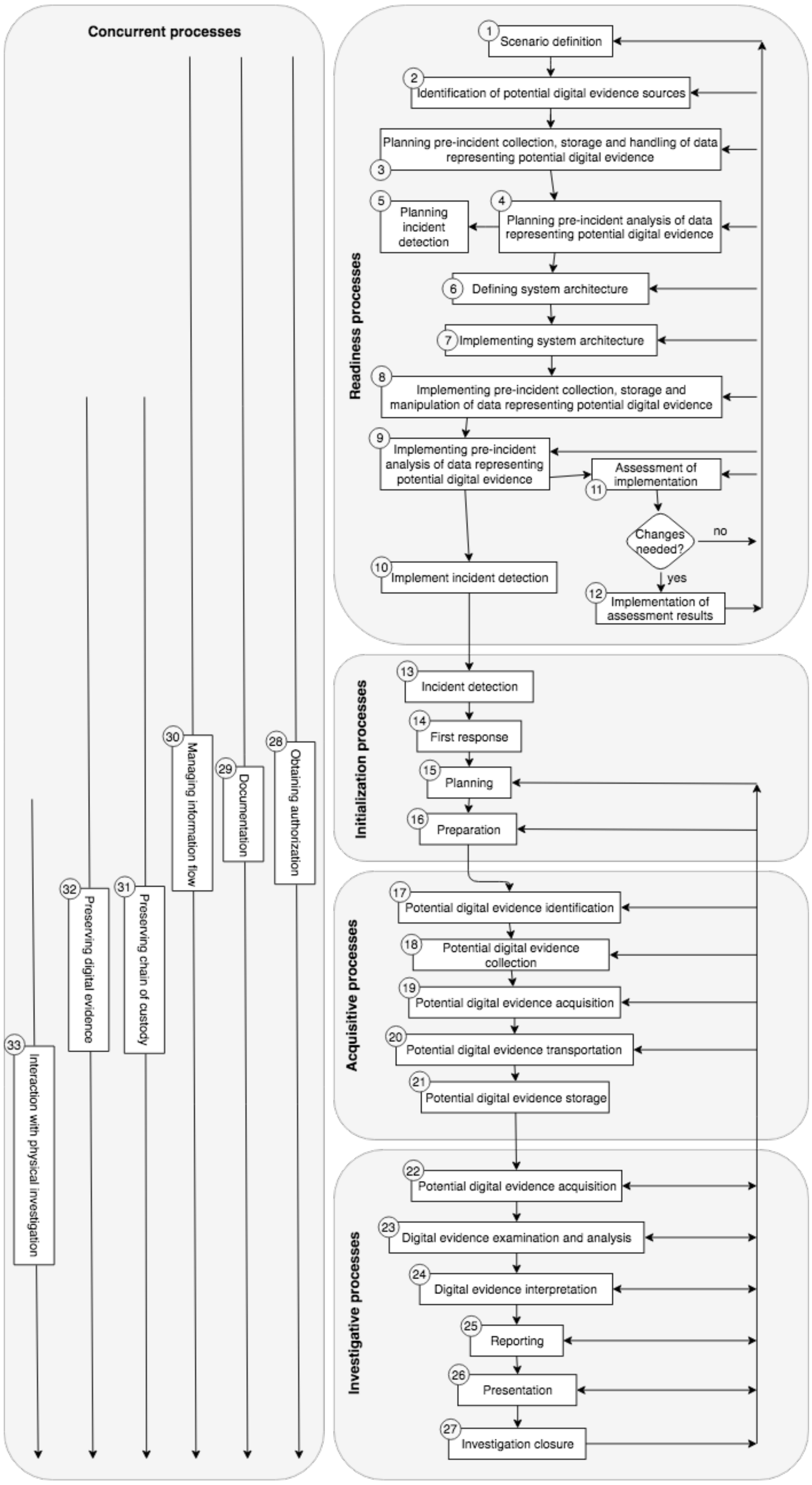

Concurrent processes
Readiness processes
1 Scenario definition
2 Identification of potential digital evidence sources
3 Planning pre-incident collection, storage and handling of data representing potential digital evidence
4 Planning pre-incident analysis of data representing potential digital evidence
5 Planning incident detection
6 Defining system architecture
7 Implementing system architecture
8 Implementing pre-incident collection, storage and manipulation of data representing potential digital evidence
9 Implementing pre-incident analysis of data representing potential digital evidence
10 Implement incident detection
11 Assessment of implementation
Changes needed?
no
yes
12 Implementation of assessment results
Initialization processes
13 Incident detection
14 First response
15 Planning
16 Preparation
Acquisitive processes
17 Potential digital evidence identification
18 Potential digital evidence collection
19 Potential digital evidence acquisition
20 Potential digital evidence transportation
21 Potential digital evidence storage
Investigative processes
22 Potential digital evidence acquisition
23 Digital evidence examination and analysis
24 Digital evidence interpretation
25 Reporting
26 Presentation
27 Investigation closure
28 Obtaining authorization
29 Documentation
30 Managing information flow
31 Preserving chain of custody
32 Preserving digital evidence
33 Interaction with physical investigation

Figure 5.1: ISO 27043 processes [40]

5. Planning incident detection
6. Defining system architecture
7. Implementing system architecture
8. Implementing pre-incident gathering, storage, and handling of data representing potential digital evidence
9. Implementing pre-incident analysis of data representing potential digital evidence
10. Implementing incident detection
11. Assessment of implementation
12. Implementation of assessment results

5.2.2.2 Initialization processes

Initialization processes are the first processes that are executed when a digital forensic investigation commences [95]. This can include processes such as preparing evidence collection kits, obtaining search warrants and developing a collection strategy [13]. There are four initialization sub-processes:

13. Incident detection
14. First response
15. Planning
16. Preparation

5.2.2.3 Acquisitive processes

Acquisitive processes are processes that are executed to identify and obtain sources of digital evidence, as well as processes that are developed to transport and store the collected sources securely [61]. Examples of these types of processes are an investigator walking through the crime scene to identify evidence and making physical copies of evidence [13]. There are five sub-processes in the acquisitive category, namely

17. Potential digital evidence identification
18. Potential digital evidence collection
19. Potential digital evidence acquisition
20. Potential digital evidence transportation
21. Potential digital evidence storage and preservation

5.2.2.4 Investigative processes

Investigative processes are the processes that are executed to extract the evidence from the sources that were identified as part of the acquisitive processes [18]. These processes involve systematic searches, reconstruction based on the evidence, as well as reporting the findings [13]. The sub-processes of the investigative process are:

22. Potential digital evidence acquisition
23. Potential digital evidence examination and analysis
24. Digital evidence interpretation
25. Reporting
26. Presentation
27. Investigation closure

5.2.2.5 Concurrent processes

In addition to the processes mentioned previously, there are also six processes that do not neatly fit into any category, since they happen throughout the investigation. These are processes such as documentation, interacting with other investigations and managing the information flow [64]. The six concurrent process categories are:

28. Obtaining authorisation
29. Documentation
30. Managing information flow
31. Preserving chain of custody
32. Preserving digital evidence
33. Interaction with the physical investigation

This section presented the Digital Forensics process and the different types of processes that make up a Digital Forensic investigation. However, the author has not yet addressed how evidence becomes admissible in legal proceedings, known as forensic soundness. Next, the author presents the concept of forensic soundness.

5.2.3 Forensic soundness

In the previous sections, the author mentioned in passing that data gathered as part of a Digital Forensic investigation may be used in legal proceedings. This concept is known as *forensic soundness.*

McKemmish [56] defines forensic soundness as "[t]he application of a transparent digital forensic process that preserves the original meaning of the data for production in a court of law."

This means that the process used to capture the evidence should be reliable and accurate and that it should be possible to test or verify the process. In addition, the original meaning of the data should be preserved, which means that an investigator or another party should be able to interpret the evidence correctly [56].

Casey [20] mentions that one of the main ways to achieve forensic soundness is by documentation. This documentation should chronicle where evidence was found, how it was collected, who handled it and how and where it was stored. This will allow legal proceedings to determine the integrity and completeness of the evidence.

Digital Forensics as presented in this section is designed to uncover evidence. However, due to the volatile nature of some electronics, evidence may be destroyed by the time the forensic team is called in. In an attempt to address this, the concept of Digital Forensic Readiness was introduced. This is presented in the next section.

5.3 Digital Forensic Readiness

Digital Forensic Readiness falls into the category of *acquisitive processes* and is becoming a more important part of the Digital Forensic strategy of organisations [68], since not having a Digital Forensic Readiness process has both cost and time implications.

Tan [93] found that, without Digital Forensic Readiness in place, 2 hours of intrusion time (i.e. the attacker spent 2 hours on a compromised system) required about 40 hours of billable time from an investigation team. These 40 hours does not include detecting the intrusion, collecting the evidence and restoring the compromised system. Proactive identification of potential intrusions reduces the cost of an investigation [35].

This section defines Digital Forensic Readiness and lists the benefits and drawbacks associated with implementing a Digital Forensic Readiness process.

5.3.1 Defining Digital Forensic Readiness

Grobler et al. [34] defines Digital Forensic Readiness as "...the ability of an organisation to maximise its potential to use CDE [Comprehensive Digital Evidence] whilst minimizing the costs of an investigation". Tan [93] identifies the two goals of Digital Forensic Readiness as

- Maximizing the usefulness of incident evidence data
- Minimizing the cost of forensics during an incident response

That is to say, Digital Forensic Readiness is a process where digital evidence is collected before an incident occurs [76]. Collecting the evidence beforehand allows the organisation to perform forensic investigations quicker and with less expense[60], in addition to a number of other benefits, which is presented in the next section.

5.3.2 Benefits and Drawbacks of Digital Forensic Readiness

Before an organisation decides to implement a Digital Forensic Readiness process, they need to be aware of the benefits and drawbacks related to Digital Forensic Readiness. This section analyses the benefits and costs related to implementing digital forensic readiness.

5.3.2.1 Benefits

There are a number of benefits for an organisation that has implemented Digital Forensic Readiness, namely

- In the case of legal proceedings, the organisation has evidence to protect itself. If, for example, an employee conducted illegal activities using their organisational account, the organisation would be liable unless they possessed evidence incriminating the employee [27].
- Investigations can be conducted quickly and with a lesser impact on business operations, since evidence has already been collected [12].
- It can act as a discouragement against attacks by employees if it is known that the organisation collects this data, since the knowledge acts as a deterrent for unlawful activity [81].
- It demonstrates good corporate governance to governing bodies and stakeholders, protecting the reputation of the organisation and reassuring stakeholders [58].
- It can improve the outcomes of legal actions and business disputes, since the evidence is available immediately and in larger quantities [81].

In addition to all the benefits that a Digital Forensic Readiness process brings, it also has one significant drawback: cost.

5.3.2.2 Cost of implementing digital forensic readiness

The main drawback for organisations looking to implement Digital Forensic Readiness is the large upfront costs associated with such an endeavour. There are several main areas of cost:

- Updating organisational policies, such as fair use policies and terms of employment [12]. Reworking policies can be costly since input from a large number of stakeholders is required. In addition to the costs of reworking the policy, there are also costs related to making employees aware of the new policies.

- Training, especially for the employees that will be implementing and maintaining the Digital Forensic Readiness process [34]. This training could incur large costs since a trainer is required that is knowledgeable about the Digital Forensic Readiness process.

- Installing and implementing tools for gathering and securely storing evidence[81]. The licensing and installation costs for the specialised tools required for digital forensic readiness could incur a large cost for the organisation.

- Preparation for incidents and implementing evidence retrieval capabilities[81]. Making sure that the employees who will be responsible for managing incidents are trained correctly, have processes in place and are ready to react will incur costs for the organisation.

- Obtaining legal advice throughout the process [81]. Legal counsel related to potential incidents, as well as advice on the digital forensic readiness policies, can incur large costs for an organisation.

Considering all the benefits listed above, along with the potential cost of such a program, an organisation needs to make the decision about whether or not it should be implementing digital forensic readiness. Organisations with the financial means will find great advantage in implementing a Digital Forensic Readiness process.

5.4 Conclusion

This chapter gave an overview of Digital Forensics and Digital Forensic Readiness, as well as outlining the drawbacks and benefits around implementing a Digital Forensic Readiness process, listing factors that an organisation will have to consider before deciding to implement digital forensic readiness.

Digital Forensic Readiness allows organisations to gather potential evidence preemptively from devices. This means that investigators will have access to the gathered evidence without requiring physical access to a device.

This concludes the background section of this research. The next section introduces the reader to a high-level model and architecture as a start to the solution to the problem stated in chapter 1.

PART III

Model and architecture

In the third part of the research, the author presents and elaborates on the model and architecture of a solution to the problem stated in chapter 1. In addition to presenting the model and architecture, the author also evaluates the proposed solution based on the umbrella standard for digital forensics, ISO 27043.

CHAPTER 6

A High-level Model for adding Digital Forensic Readiness to a Mobile Device Management (DFR-MDM) System

6.1 Introduction

In the previous chapters, the author covered background information related to the problem statement, namely BYOD, MDM and the concepts of Digital Forensics and Digital Forensic Readiness, creating the design science constructs to be used when defining the model and methods.

This chapter proposes a high-level model for the problem as stated, which is that "most MDM systems do not include a digital forensic readiness component, leaving investigators with little to no device-related historical data when an incident does occur." The author presents a model for adding Digital Forensic Readiness to a Mobile Device Management solution using the design science artefacts defined in the previous chapters.

The first section focuses on the components in the model, presenting both the changes that have to be made to existing MDM components and adding the necessary new components to facilitate digital forensic readiness. The second section presents the measures that the model uses to ensure data integrity, namely utilising digital signatures, using encryption and using checksums.

6.2 Components

This model builds on the standard MDM model as shown in figure 4.1, with four components, namely a server, administrator console, database and mobile client. Adding digital forensic readiness to an MDM system requires changes to all the existing components. There is also one new component, namely a data store. The revised model is shown in figure 6.1.

This section describes the changes that have to be made to the system in order to add Digital Forensic Readiness. First, the author presents the changes that have to be made to the four existing components (the server, administrator console,

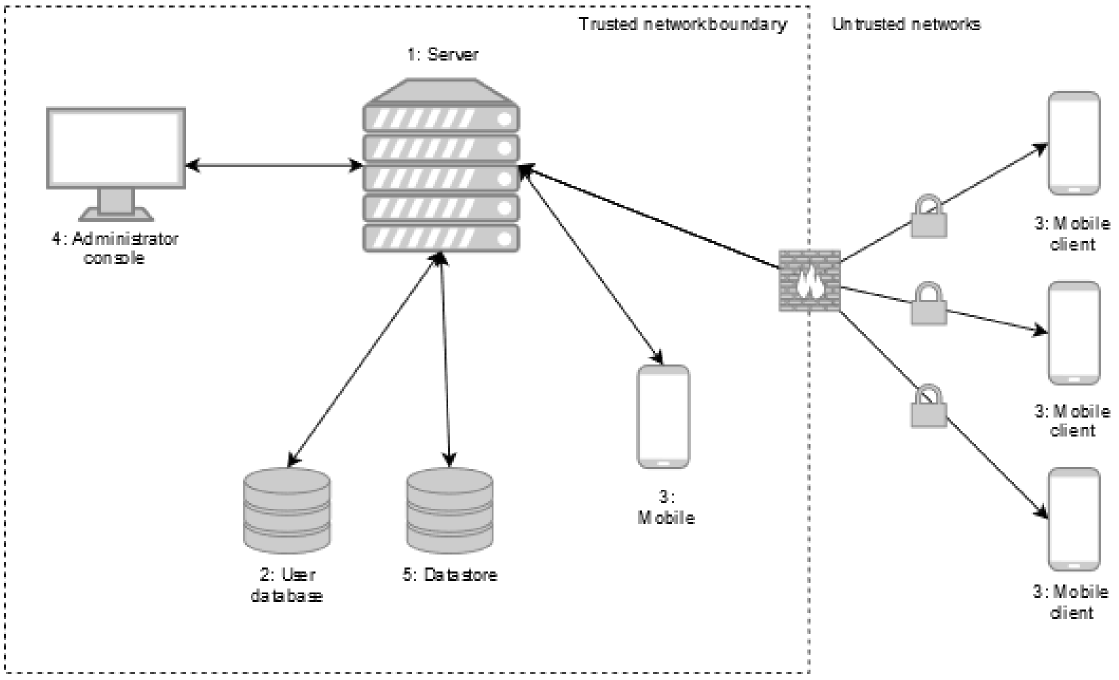


Figure 6.1: Components

database and mobile client) and after that the author explains the new component, the data store.

6.2.1 Server

In addition to the functionality required of the server in an MDM system, namely sending device management settings and keeping track of devices, there are some additional functionality when adding Digital Forensic Readiness to the system.

First, the data collected from devices has to be associated with a specific user, as mentioned in the previous chapter. The registration process on the server is expanded to include device identifiers as well as the public keys that are required to verify uploaded data.

In addition to associating devices with users, the server also accepts the data that is uploaded by the devices. Specifically, the server has to verify the digital signatures of all communications sent by the device, receive both the checksums and the data and verify the data against the checksums.

Finally, the server provides the data collection policies to the mobile device clients. These policies are defined by the system administrators and controls how and when data is collected from mobile devices. The administrators make use of the administration console to define the policies. The console is presented in the next section.

6.2.2 Administrator console

A standard MDM administrator console allows the system administrators to view and change the device management rules, which are applied to all the mobile clients known to the system. To extend the administrator console for Digital Forensic Readiness, the system allows the administrators to view and edit device collection policies, as well as viewing the users associated with the mobile devices.

In addition to providing administration capabilities to system administrators, the administration console also enables digital forensic investigations. It allows digital forensic investigators to combine data and findings in a report that can be presented as evidence.

The investigation section of the console allows digital forensic investigators to start investigations, select users to investigate and associate pieces of evidence with an investigation. Finally, the console produces a coherent report once an investigation has been completed.

In order to ensure a high degree of trust, audit logs of all actions taken by the administrators and investigators are kept. These logs can be added to investigation reports as further evidence of the integrity of the report.

All the data that can be viewed by the administrators and digital forensic investigators are stored in a database. In the next section, the author presents the changes made to the database to support the digital forensic readiness process.

6.2.3 Database

To allow the system to ensure the integrity of the collected data, the database that previously stored only the device management rules now also stores data related to users, their keys and the identifiers of the devices associated with the users. The mechanism to ensure data integrity is discussed later in this chapter and the structure of the database is presented in greater detail in the next chapter.

The next section outlines the changes to the mobile client, which generates most of the data that is stored in the database.

6.2.4 Mobile client

In addition to the standard MDM functionality, namely blocking access to dangerous resources and enforcing security policies, the mobile client (also referred to as a device client) requires a number of changes to achieve the goal of adding Digital Forensic Readiness.

The responsibility of the mobile client is to enforce the data collection policies specified by the system administrators and upload the data that has been collected to the server in a forensically sound manner.

To achieve this goal, the device client detects when the user accesses a resource that is deemed high-risk by the collection policy. After that, the device starts gathering and sending the data to the server, using encryption and checksums to preserve the integrity of the data. This process is described in more detail later in this research.

The final component is the data store, which is used to store the data gathered by the mobile client. The data store is presented in the next section.

6.2.5 Data store

As mentioned earlier in this chapter, the data store is the one component that is not present in a traditional MDM system.

Since the data that can be collected from the device clients are varied and may contain diverse data points, a relational database is not a good fit for storing these types of data. Instead, the model makes use of a more unstructured data store, which will allow for the storage of the disparate data points collected from the device clients.

Since this store contains data that may be used in legal proceedings, it is imperative that it can be proven to be correct. The structure of the data store is presented in more detail in the next chapter.

This section outlined the components in the model and the changes made to them to support the goal of adding digital forensic readiness. The next section presents the mechanisms that the model uses to ensure data integrity.

6.3 Ensuring data integrity

To preserve the integrity and digital forensic soundness of the MDM system and the data it collects, the storage of the data and the communications between the server and the client have to be secured. This section presents the three mechanisms that the model uses to ensure data integrity, namely

- Encryption
- Digital signatures
- Checksums.

6.3.1 Encryption

To ensure that third parties cannot intercept or tamper with data while it is in transit, all communication is encrypted. This model makes use of both symmetric and asymmetric encryption. **Symmetric encryption** makes use of one shared secret key that is used to both encrypt and decrypt the data while **asymmetric encryption** makes use of one key to encrypt the data and another key to decrypt it [71].

This model makes use of a widely used mechanism where asymmetric cryptography is used to establish a session key and symmetric cryptography to secure communications once a session key has been established.

Specifically, the client generates a symmetric key and encrypts it using the server's public key and transmits the encrypted value to the server. The server then decrypts the key using its private key. Once both parties are in possession of the symmetric key, they encrypt traffic using a symmetric encryption algorithm. Since the server's private key is known only to the server, the symmetric key cannot be decrypted by any parties eavesdropping on traffic and therefore the traffic is safe, since the server's private key is only known to the server [73].

The next section presents digital signatures, which also makes use of asymmetric keypairs but to ensure integrity rather than secrecy.

6.3.2 Digital signatures

A digital signature is a mechanism that serves the same purpose as a traditional signature. Only the sender can produce it, but other parties can easily recognise it as being produced by the sender. Pfleeger and Pfleeger [71] identifies two primary conditions that a signature must meet, namely that it should be *unforgeable* and *authentic*, i.e. only the sender should be able to produce the signature and any receiver should be able to verify that the sender created the signature. Additionally, it should not be *alterable* or *reproducible*, i.e. once a message has been signed, changing the message will invalidate the signature and the re-sending of a previous message will be detected by the receiver, preventing replay attacks and attempts to submit false data to the system.

This model makes use of an asymmetric keypair to sign messages sent from the client to the server. The client signs the message using its private key, and the server verifies the message using the client's public key (section 7.2 outlines how the server come to be in possession of the client's public key). This mechanism ensures that the signatures are unforgeable and authentic since only the client has access to the private key that is used to generate the signatures.

The next section presents the final element mechanism that the model uses to ensure integrity and authenticity, namely checksums.

6.3.3 Checksums

Checksums, more accurately known as cryptographic hash functions, are used to ensure the integrity of a piece of data. The checksum of the data is computed and stored along with the data. Then, whenever the integrity of the data has to be verified, the checksum of the data can be recomputed and compared with the stored value. The checksum function is designed in such a way that a change of even a single bit will result in a completely different checksum value [71]. Making use of checksums ensures data integrity and prevents tampering and the submission of altered data.

This section presented the three mechanisms that the model use to ensure security and integrity, namely encryption, digital signatures and checksums.

6.4 Conclusion

This chapter proposed a high-level model for adding digital forensic readiness to an MDM system. It outlined changes to existing components in the MDM system, as well as new components that has to be added. It presented the mechanisms used by the model to ensure the integrity of the data, namely encryption, checksums, and digital signatures.

As mentioned in chapter 2, once the models of a system have been defined, the methods can be created. These algorithms are presented in the next chapter.

CHAPTER 7

Methods in a DFR-MDM system

7.1 Introduction

Design science methods are a series of steps that are executed to achieve a particular task. These methods are based on the constructs and models which has been outlined in the previous chapters.

In the previous chapter, the author defined a high-level model with various components. This chapter introduces the methods that make use of those components to add digital forensic readiness to a Mobile Device Management system.

There are five main methods that need to be executed for the proposed model to achieve the goal as stated in chapter 1. These five processes are:

1. Device registration
2. Acquiring an authentication token
3. Loading policies
4. Collecting data about user activity
5. Uploading collected data

This section describes the processes listed above, showing how the components interact through these different processes to achieve the stated goals. The first interaction is associating a device with a user through device registration, which is presented next.

7.2 Device registration

The first process that is executed happens when a user logs into a new device for the first time. During this process, the device is associated with the user, and the required keys are exchanged. To ensure the integrity of the data that is exchanged during this process, it has to be executed on a trusted network. This process is shown in figure 7.1.

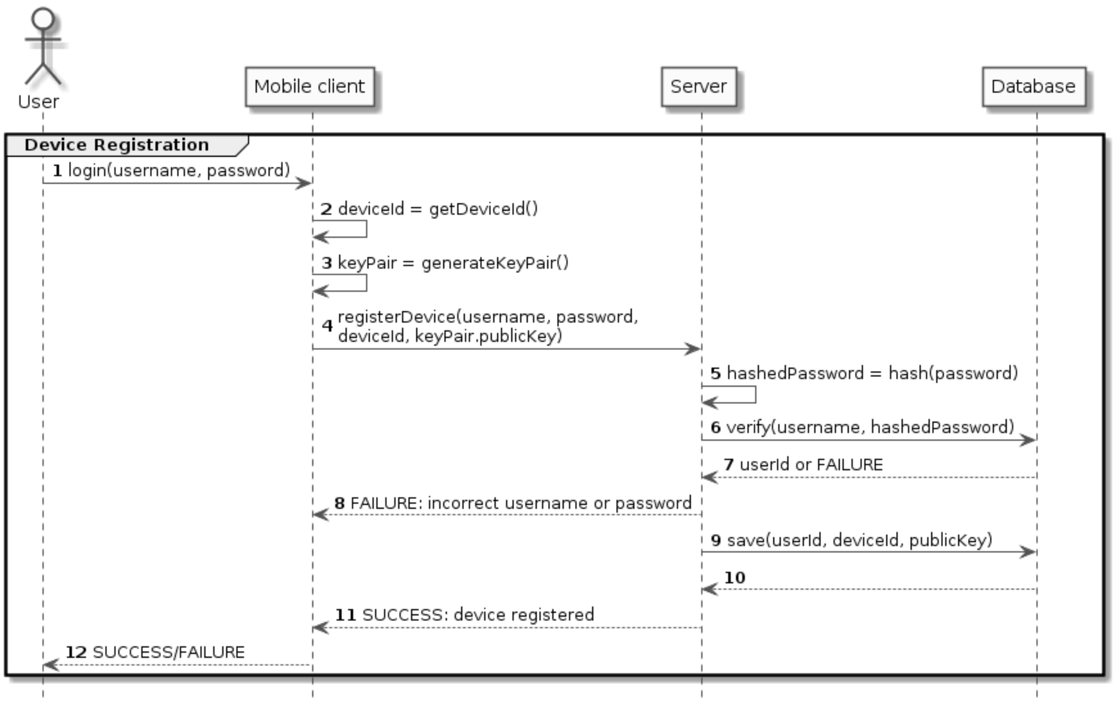

User
Mobile client
Server
Database
Device Registration
1 login(username, password)
2 deviceId = getDeviceId()
3 keyPair = generateKeyPair()
4 registerDevice(username, password, deviceId, keyPair.publicKey)
5 hashedPassword = hash(password)
6 verify(username, hashedPassword)
7 userId or FAILURE
8 FAILURE: incorrect username or password
9 save(userId, deviceId, publicKey)
10
11 SUCCESS: device registered
12 SUCCESS/FAILURE

Figure 7.1: Registration Sequence

First, the mobile client prompts the user for their username and password (fig 7.1 step 1). The user's credentials are required to link the mobile device to the user on the MDM system.

After the user has been identified, the client obtains a unique identifier for the device (this can either be randomly generated or obtained from the device) (fig 7.1 step 2) and generates a public/private keypair (fig 7.1 step 3). The device stores this keypair securely. In most modern devices the task of generating and storing this keypair is delegated to the operating system because the operating system can execute the process in a protected environment. As mentioned previously, this generated keypair is used to secure all other communication between the device client and the server.

The device client sends the user credentials, a unique identifier and public key to the remote system (fig 7.1 step 4), encrypted using SSL/TLS. Since this process is executed in a trusted environment, encrypting traffic using SSL/TLS is sufficient. The server verifies the user's login credentials (fig 7.1 steps 5, 6 and 7). If a user's credentials are incorrect, the user is notified of the fact (fig 7.1 step 8).

Once the user's credentials have been verified, the system associates the given device identifier and public key with the user (fig 7.1 steps 9 and 10) and notifies the user that the registration has been successful (fig 7.1 steps 11 and 12).

Once a device has been registered, the client software on the device can start to perform its functions. To prevent malicious third parties from interfering, the mobile client needs to be authenticated by the server before the server accepts any other data. Since the mobile client runs in the background, it cannot prompt the user for credentials every time, and another authentication method is required. This authentication process is presented in the next section.

7.3 Acquiring an authentication token

Any communication between the client and the server must be authenticated to prevent third parties from tampering with the data in the system. Since one of the requirements of this mobile client is to interfere with normal usage of the mobile device as little as possible, the client cannot prompt the user for their username and password every time it attempts to connect to the server.

Authentication of interactions when the device client is running in the background is achieved through the use of an authentication token. This token is generated when the user logs in, and the token is valid for a specific period. Once the period of validity has expired, the user is prompted to log in, and a new token is generated. Since the token on the device client and the token stored on the server has to match for authentication to succeed, it is also possible to invalidate a token from the server, should it be necessary.

The sequence for the authentication process is shown in figure 7.2. Note that the first time this token is generated, is directly after the registration process and as such, the user will not be prompted for their login credentials again.

The first step in generating an authentication token for a user is prompting them for their username and password (fig 7.2 step 1). The mobile client then retrieves the private key that has been stored on the device during registration (fig 7.2 step 2) and encrypts the user's credentials (fig 7.2 step 3). The encrypted credentials and the device identifier (fig 7.2 step 4) are then used to request a new authentication token from the server (fig 7.2 step 5).

Once the request for the token reaches the server, the server retrieves the correct public key (fig 7.2 steps 6 and 7) and decrypts the user's credentials (fig 7.2 step 8). The server verifies the user's credentials (fig 7.2 steps 9, 10 and 11) and notifies the user if they are incorrect (fig 7.2 step 12).

If the user's credentials are correct, the server will generate a new authentication token (fig 7.2 step 13), store it in the database (fig 7.2 steps 14 and 15) and pass the token back to the mobile client (fig 7.2 step 16). The mobile client then stores the token (fig 7.2 step 17) and notify the user that the process has been completed (fig 7.2 step 18).

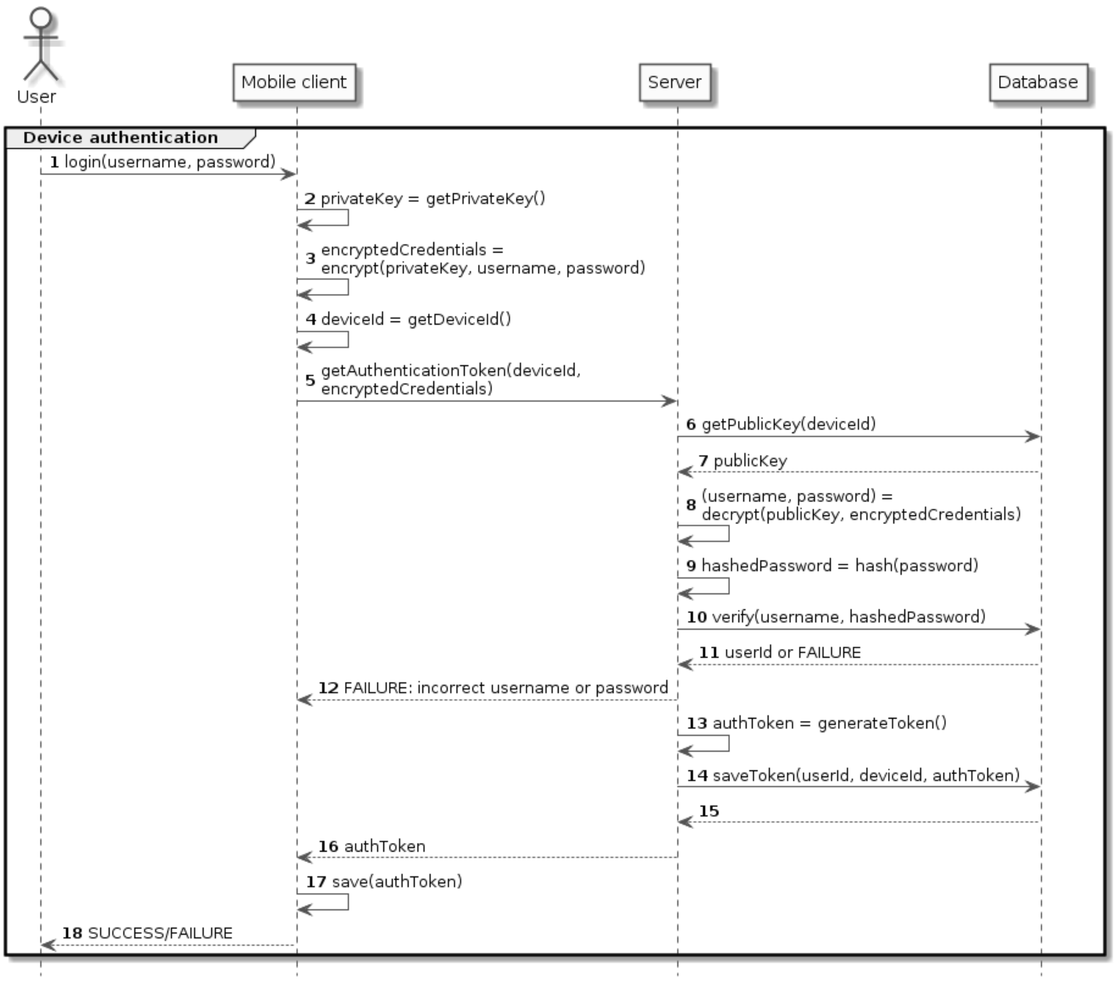

User
Mobile client
Server
Database
Device authentication
1 login(username, password)
2 privateKey = getPrivateKey()
3 encryptedCredentials = encrypt(privateKey, username, password)
4 deviceId = getDeviceId()
5 getAuthenticationToken(deviceId, encryptedCredentials)
6 getPublicKey(deviceId)
7 publicKey
8 (username, password) = decrypt(publicKey, encryptedCredentials)
9 hashedPassword = hash(password)
10 verify(username, hashedPassword)
11 userId or FAILURE
12 FAILURE: incorrect username or password
13 authToken = generateToken()
14 saveToken(userId, deviceId, authToken)
15
16 authToken
17 save(authToken)
18 SUCCESS/FAILURE

Figure 7.2: Authentication Sequence

After the device client has acquired a valid authentication token; it can run in the background without any input from the user of the device. All other actions it performs, namely loading policies, gathering data and uploading the gathered data happens automatically and without notifying the user. The next section presents the first of these three processes, namely the loading of new policies.

7.4 Loading policies

One of the strengths of an MDM system is the fact that the management rules can be updated. The new rules can then be downloaded and applied by the mobile clients. This functionality can be extended to the data gathering rules, and the process for loading new data gathering policies is shown in figure 7.3.

The process of downloading a new policy can be started in one of two ways:

1. The server notifies the client that a new policy is available. The mobile client then downloads the new policy as soon as it can do so.

2. The mobile client periodically checks with the server for policy updates. This is done to ensure that the new policies are downloaded even if the notification from the server never reaches the client.

Once the process of downloading a new policy has been started in one of the two ways described (fig 7.3 step 1), the mobile client retrieves the device identifier and the private key (fig 7.3 steps 2 and 3), as well as the authentication token that has been generated as part of the authentication process (fig 7.3 step 4). The token is then signed using the retrieved private key (fig 7.3 step 5) and the mobile client sends a request for policy updates to the server, passing the server the device identifier, authentication token and the digital signature (fig 7.3 step 6).

When the server receives a request for a policy update, it retrieves the public key associated with the device identifier (fig 7.3 steps 7 and 8), notifying the mobile client if this operation fails (fig 7.3 step 9). Once the public key has been retrieved, the server verifies the signature that has been sent by the mobile client (fig 7.3 step 10) and then identifies the user based on their unique authentication token (fig 7.3 steps 11 and 12), again notifying the mobile client if the operation fails (fig 7.3 step 13).

Once the server has gone through all the steps described to verify that this request is legitimate, it retrieves the policy associated with the requesting user (fig 7.3 steps 14 and 15) and sends it back to the mobile client (fig 7.3 step 16). Finally, the mobile client stores the new policy (fig 7.3 step 17) and applies the new policy (fig 7.3 step 18).

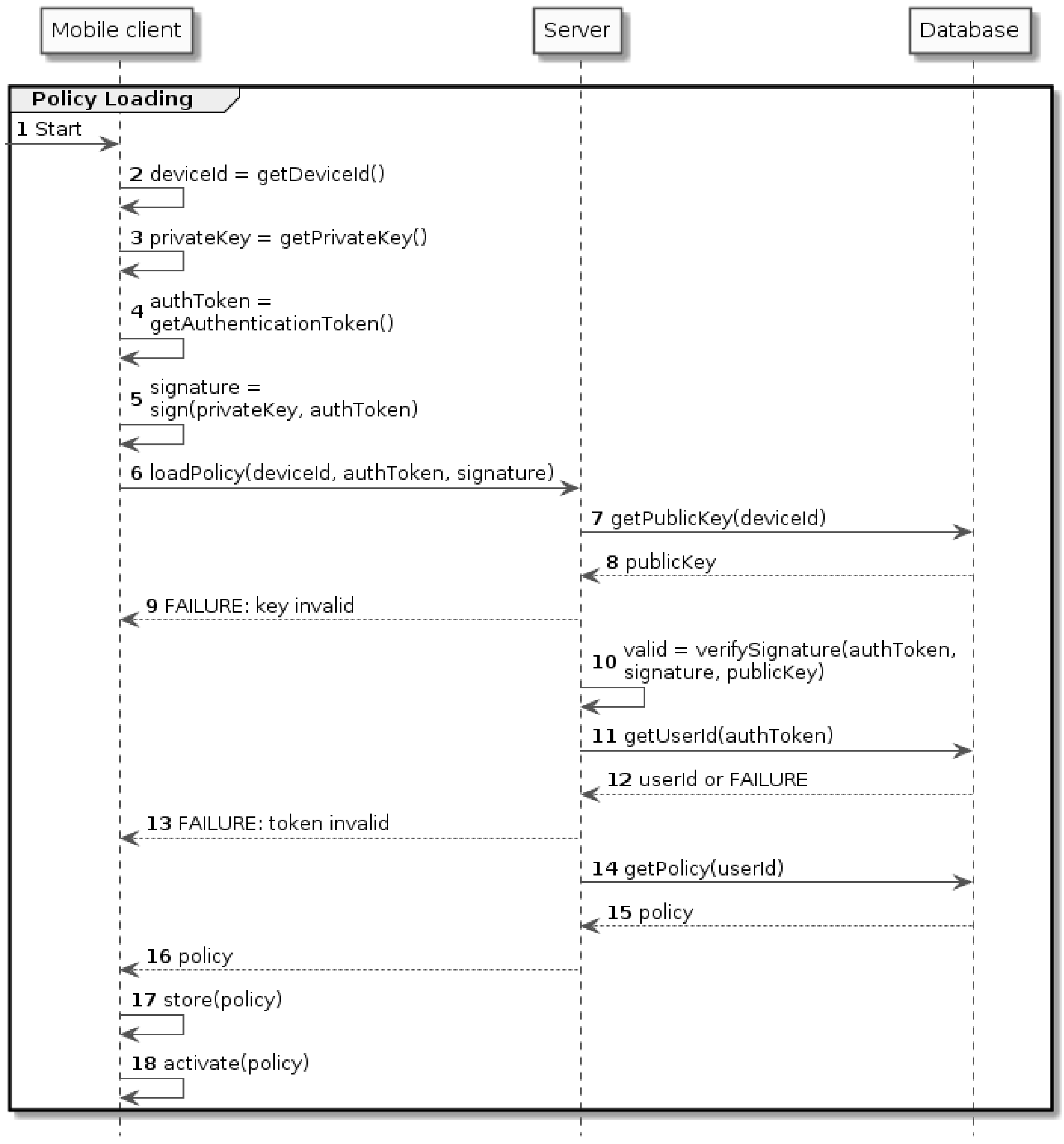

Mobile client
Server
Database
Policy Loading
1 Start
2 deviceId = getDeviceId()
3 privateKey = getPrivateKey()
4 authToken = getAuthenticationToken()
5 signature = sign(privateKey, authToken)
6 loadPolicy(deviceId, authToken, signature)
7 getPublicKey(deviceId)
8 publicKey
9 FAILURE: key invalid
10 valid = verifySignature(authToken, signature, publicKey)
11 getUserId(authToken)
12 userId or FAILURE
13 FAILURE: token invalid
14 getPolicy(userId)
15 policy
16 policy
17 store(policy)
18 activate(policy)

Figure 7.3: Policy Loading Sequence

Once the mobile client has gone through the process of registration and downloading the data collection policy, it can start monitoring the activity of the user on the device. The next section presents the process of gathering data about the user's activity.

7.5 Collecting data about user activity

When a mobile client detects that a user is accessing a resource that is flagged in the policy, it starts to collect data on the user's activities. Which activities are detected is controlled by policy and may include actions like accessing certain websites (for example, many organisations choose to monitor usage of cloud storage solutions like Dropbox). The process of collecting data is shown in figure 7.4.

When the data collection process starts (fig 7.4 step 1), the client loads the policy that specifies which data it should collect (fig 7.4 step 2) and schedules the data upload to happen at some later time (fig 7.4 step 3). Then, while the conditions that have been specified in the policy are being met, the client performs a data collection loop.

The first step in the data collection loop is to retrieve the relevant data from the device's Operating System (fig 7.4 steps 4 and 5). Once the client has retrieved the data, it calculates the checksum om the data (fig 7.4 step 6), generates a unique identifier for the data (fig 7.4 step 7) and saves the data locally (fig 7.4 step 8). After saving the data locally, the client prepares for upload.

After collecting the data, the mobile client immediately uploads the checksum of the data for integrity purposes. To do this, the client retrieves the device's identifier (fig 7.4 step 9), the private key (fig 7.4 step 10) and the authentication token (fig 7.4 step 11) that has previously been saved. The mobile client then signs the authentication token using the private key (fig 7.4 step 12) and sends the signature, along with the device identifier, data identifier and checksum to the server (fig 7.4 step 13).

When the server receives the checksum upload, it retrieves the public key that is associated with the device identifier (fig 7.4 steps 14 and 15) and verifies the digital signature (fig 7.4 step 17). If the public key could not be found or if the signature is invalid, an error is returned to the client (fig 7.4 steps 16 and 18). If the checks pass, the server saves the device identifier, data identifier and the checksum to the data store (fig 7.4 steps 19 and 20). Finally, the server returns the result of the operation back to the client (fig 7.4 step 21).

After the checksum has been uploaded; the client still needs to upload the data that has been collected. The uploading of data is done in batches to reduce the load on the server and network traffic from the client. The data upload process is presented in the next section.

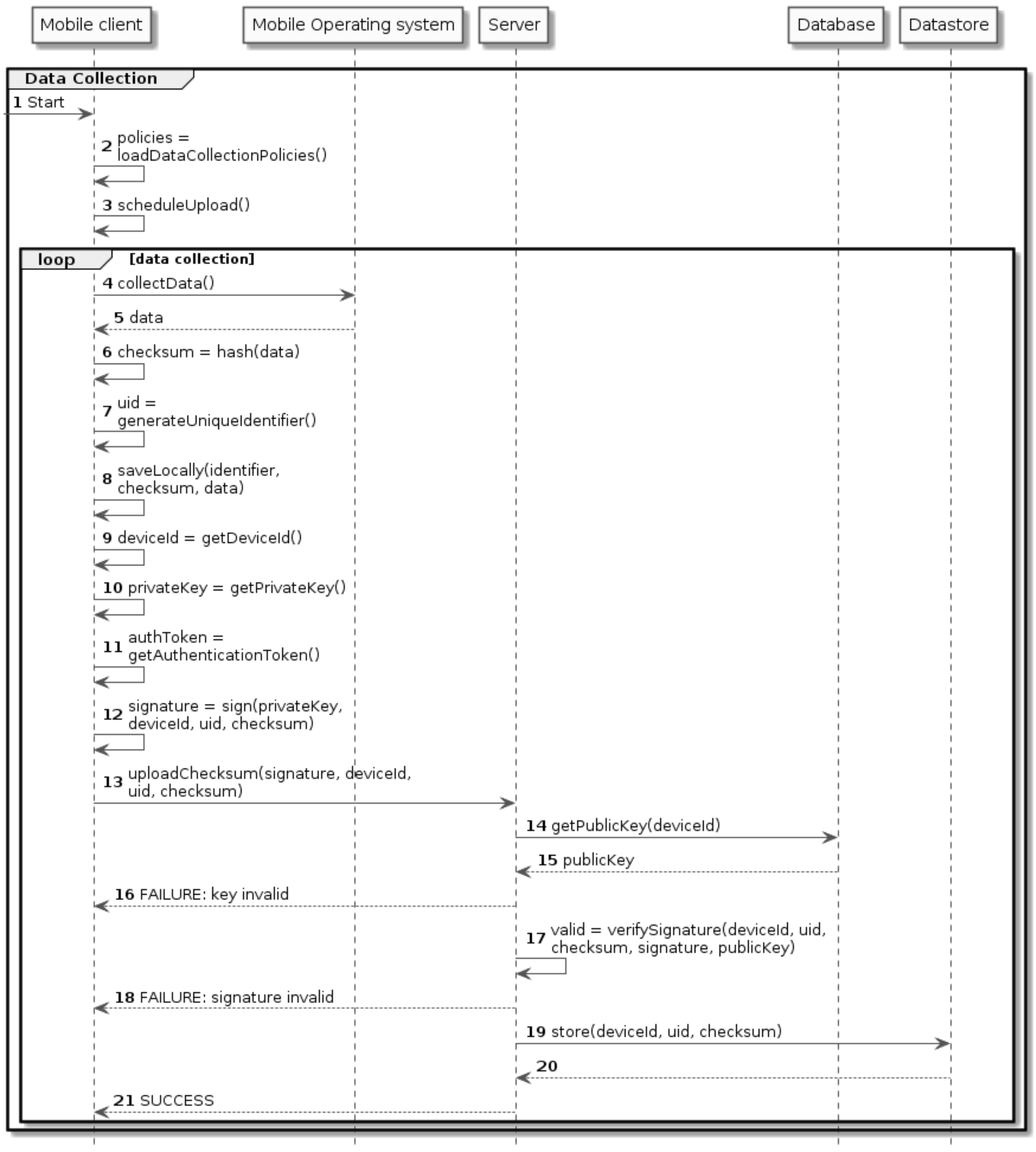

Mobile client
Mobile Operating system
Server
Database
Datastore
Data Collection
1 Start
2 policies = loadDataCollectionPolicies()
3 scheduleUpload()
loop [data collection]
4 collectData()
5 data
6 checksum = hash(data)
7 uid = generateUniqueIdentifier()
8 saveLocally(identifier, checksum, data)
9 deviceId = getDeviceId()
10 privateKey = getPrivateKey()
11 authToken = getAuthenticationToken()
12 signature = sign(privateKey, deviceId, uid, checksum)
13 uploadChecksum(signature, deviceId, uid, checksum)
14 getPublicKey(deviceId)
15 publicKey
16 FAILURE: key invalid
17 valid = verifySignature(deviceId, uid, checksum, signature, publicKey)
18 FAILURE: signature invalid
19 store(deviceId, uid, checksum)
20
21 SUCCESS

Figure 7.4: Data Collection Sequence

7.6 Uploading collected data

Periodically, a few times per day, the client checks if there are data that has to be uploaded to the server. If there are, the data upload process shown in figure 7.5 is executed. The figure assumes one set of data, but if multiple sets of data are present, they are batched to reduce network traffic.

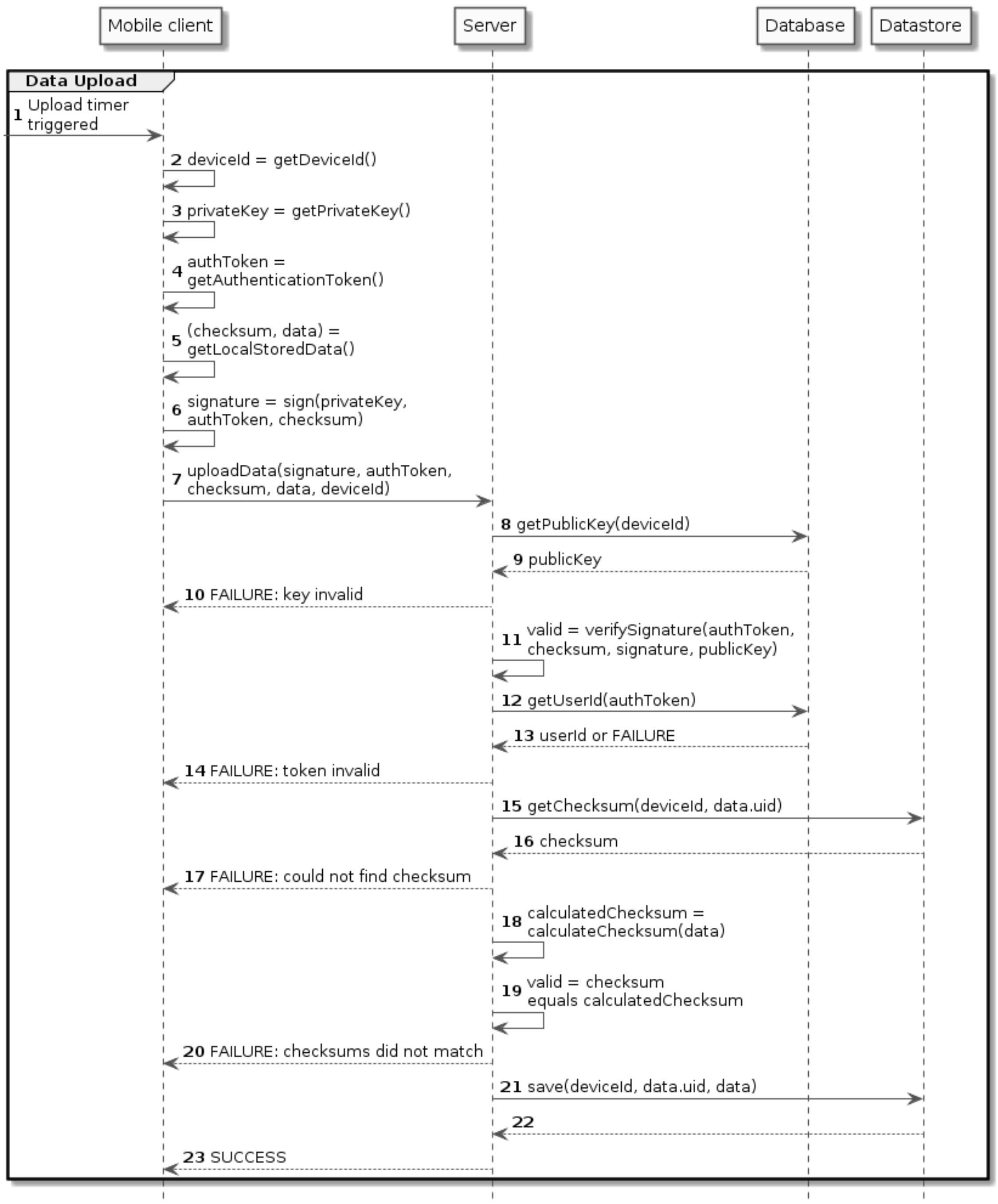

Mobile client
Server
Database
Datastore
Data Upload
1 Upload timer triggered
2 deviceId = getDeviceId()
3 privateKey = getPrivateKey()
4 authToken = getAuthenticationToken()
5 (checksum, data) = getLocalStoredData()
6 signature = sign(privateKey, authToken, checksum)
7 uploadData(signature, authToken, checksum, data, deviceId)
8 getPublicKey(deviceId)
9 publicKey
10 FAILURE: key invalid
11 valid = verifySignature(authToken, checksum, signature, publicKey)
12 getUserId(authToken)
13 userId or FAILURE
14 FAILURE: token invalid
15 getChecksum(deviceId, data.uid)
16 checksum
17 FAILURE: could not find checksum
18 calculatedChecksum = calculateChecksum(data)
19 valid = checksum equals calculatedChecksum
20 FAILURE: checksums did not match
21 save(deviceId, data.uid, data)
22
23 SUCCESS

Figure 7.5: Data Upload Sequence

When the data upload process is started (fig 7.5 step 1), the mobile client retrieves the device identifier, public key, authentication token (fig 7.5 steps 2, 3 and 4) and the data that has to be uploaded with its checksum and data identifier (fig 7.5 step 5). The mobile client then creates a signature using the private key, the authentication token and the checksum (fig 7.5 step 6) and uploads the data to the server (fig 7.5 step 7).

When the server receives a data upload request, it first retrieves the public key that is associated with the given device (fig 7.5 steps 8 and 9) and sends a failure back to the client if the key could not be retrieved (fig 7.5 step 10). It then uses the public key to verify the signature created by the device (fig 7.5 step 11) and retrieves the user associated with the given authentication token (fig 7.5 steps 12 and 13). If the signature or user could not be verified it returns a failure to the client (fig 7.5 step 14).

Once the device and user has been verified, the server checks the integrity of the data. First, it checks that the checksum sent by the client exists in the database (fig 7.5 steps 15 and 16) and returns an error if it could not find it (fig 7.5 step 17). After that, it recalculates the checksum (fig 7.5 step 18) and verifies that the checksum sent by the client matches the checksum that was calculated (fig 7.5 step 19), again sending an error to the client if there are any problems (fig 7.5 step 20). Finally, it saves the received data (fig 7.5 steps 21 and 22) and returns a success message to the client (fig 7.5 step 23).

This section covered the processes that the model executes to achieve its goal of enabling digital forensic acquisition as part of an MDM system.

7.7 Conclusion

This chapter presented the five main methods that need to be executed to collect and store potential digital forensic evidence collected from the device clients. These methods describe how the data is collected and stored to facilitate the digital forensic process.

In addition to the model, there are more details that are required to implement this model. The architecture for such a system is outlined in the next chapter.

CHAPTER 8

Architecture of a DFR-MDM system

8.1 Introduction

The previous chapters outlined a high-level model and methods for the implementation of digital forensic readiness in Mobile Device Management systems, exploring data integrity mechanisms and showing data flow. However, the high-level is not detailed enough to be used to create a prototype.

This chapter fleshes out the proposed solution by presenting an architecture that can be used to apply the proposed model to a system. As mentioned in the previous chapter, there are four main components, namely:

- a mobile client,
- a server,
- a database and
- a data store.

This chapter presents how each of these components can be architected to achieve the stated goal of digital forensic readiness in the system. In this process, the author defines the design science methods. The first component that the author introduces is the mobile client.

8.2 Mobile client

As mentioned in the previous chapter, after the mobile client application is installed on a device, the user logs into the client and registers the device to their account. After registration, the mobile client runs on the employees' mobile devices and, in addition to the standard MDM functionality, it monitors the user's activities. If the application detects suspicious activities, it collects and transmits the data to the server in a forensically sound manner. The mobile application's monitoring and collection activities are controlled via data collection policies that are downloaded from the server.

This section is divided into two subsections. The first subsection outlines the various components that form part of the mobile application. The second subsection demonstrates the program flow through the use of activity diagrams.

8.2.1 Mobile application components

The mobile client has a number of components, as seen in figure 8.1. This subsection explores each of these components in more detail.

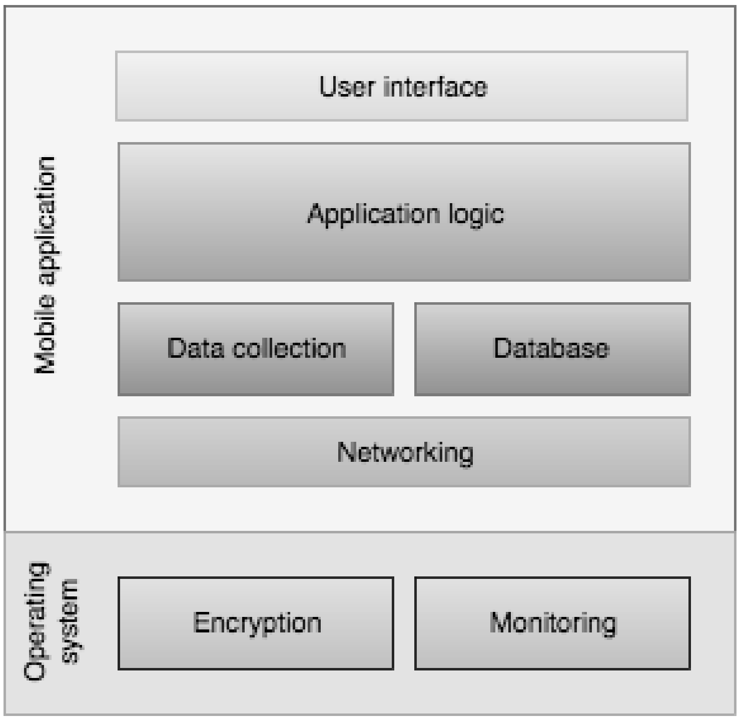


Figure 8.1: Mobile Client components

The **user interface** presents information to the user, as well as prompting the user for details where necessary. In the case of the mobile client, the user interface prompts the user for their login details, walks them through the registration of their device and displays information on the application's functionality.

The user interface is driven by the **application logic**. This component checks for the state of the application and takes the appropriate action. These actions include checking for registration status and checking if the user is logged in. This component also checks for policies and applies the policies to the monitoring component.

The **data collection** component is responsible for collecting and storing the data in the local **database**. This component integrates closely with the application logic and monitoring components to collect the specified data.

As mentioned in the previous the chapter, the client stores the data locally before uploading it to the server in batches. This is the purpose of the **database** component. This component can also be used to store miscellaneous pieces of data that is required by the client to operate successfully.

In order to allow the mobile client to connect to the server, a **networking** component is required. This component controls all communication with the server, including login, registration, downloading policies, uploading checksums and batch uploading data.

The final two components, namely the **encryption** and **monitoring** components runs in the Operating System (OS) space. This is because these two components require privileged access to perform their functions.

The **encryption** component is responsible for all the cryptographic functionality in the application. Its functionality includes the encryption of all network traffic, the calculation of checksums and the signing of data to verify integrity. To prevent cryptographic functionality from being compromised by third parties, the actual cryptographic functions are executed in OS-space, not in application-space. All modern mobile operating systems have support for executing cryptographic functions in OS-space.

The final component is the **monitoring** component. This component tracks the activities of the user and notifies the data collection component. This component also runs in OS-space, since it needs privileged access to events that happen in the operating system and other applications.

This section presented the components that are present in the mobile application. The next section shows the program flows that the mobile client executes to achieve the stated goals.

8.2.2 Mobile application program flow

There are three main application flows that in the application. These application flows are started by three distinct events:

- when the application is installed,
- when suspicious activity is detected and
- periodically.

Each of these events has distinct program flow associated with it. This section introduces those program flows, starting with the flow that happens after application installation.

8.2.2.1 On application installation

Right after installation, the application starts up, and the flow that is shown in figure 8.2 is executed. This flow is responsible for registering the device to a user and doing the initial download of the data collection policies.

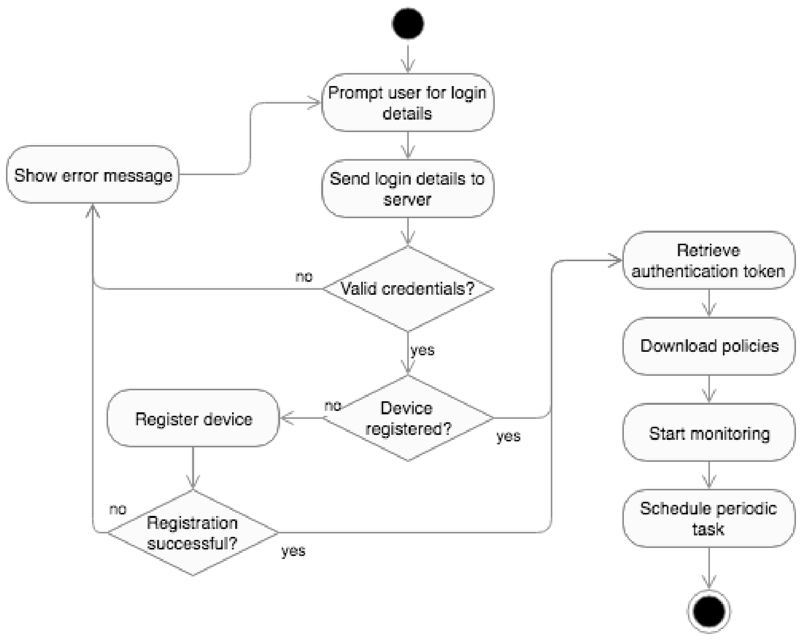


Figure 8.2: Application installation Activity Diagram

First, the user is prompted to enter their credentials, which is then sent to the server to be verified. The response from the server will indicate whether or not the device has been registered before. If the server does not recognise the credentials, an error message will be displayed to the user.

If the device has not been registered to a user, the user will be asked if they want to register this device to their account. If the registration fails, an error message is displayed to the user. If registration succeeds, the process continues.

Once the client has verified that the device has been registered to the user, the mobile client will obtain an authentication token and download the data collection policies that are associated with this user. Once the policies have been downloaded, the client starts the monitoring process and schedule a periodic task to upload any collected data. The next section will present the program flow that happens when the monitoring component notifies the client of suspicious activity.

8.2.2.2 On detection of suspicious activity

The monitoring component runs in the background, observing the user's activity. When it detects suspicious activity, it starts the process shown in figure 8.3.

First, the mobile client assembles the data into a format that is shared with the server and calculates the checksum of the data. The client generates a universally unique identifier [47] for the data and stores the identifier, the data and the calculated checksum locally. In addition to these key pieces of data, the client also stores

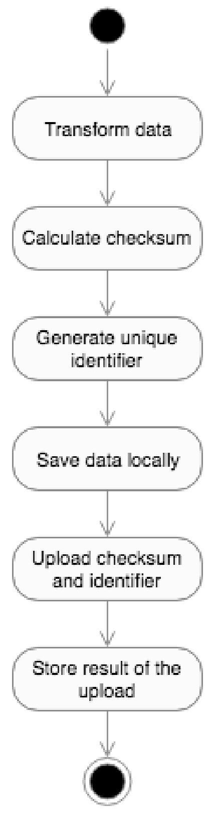

Transform data
Calculate checksum
Generate unique identifier
Save data locally
Upload checksum and identifier
Store result of the upload

Figure 8.3: Suspicious activity detection Activity Diagram

the device timestamp of when the data was collected, as well as the identifier of the policy that triggered this collection, for audit purposes.

Next, the checksum of the data and the unique identifier is uploaded to the server, using the authentication token that was generated during the previous process. The full checksum upload process was presented in section 7.5.

Finally, the result of the checksum upload (success or failure) is stored locally, and the mobile client ensures that a periodic task is scheduled to upload the collected data, as well as any checksums that have not been uploaded successfully. This periodic program flow is presented next.

8.2.2.3 Periodically

Periodically, a background process is started to verify that the application is still in possession of a valid authentication token and to upload any collected data. This process usually runs a few times a day and the exact period can be configured.

This process performs the following tasks in parallel:

- validating the authentication token,
- updating policies,

- uploading any checksums that have not been uploaded and
- uploading any data that has been collected,

as shown in figure 8.4. Next, the author presents each of these processes separately, starting with the process that validates the authentication token. The author does not discuss checksum upload again, as it has been covered earlier in this chapter.

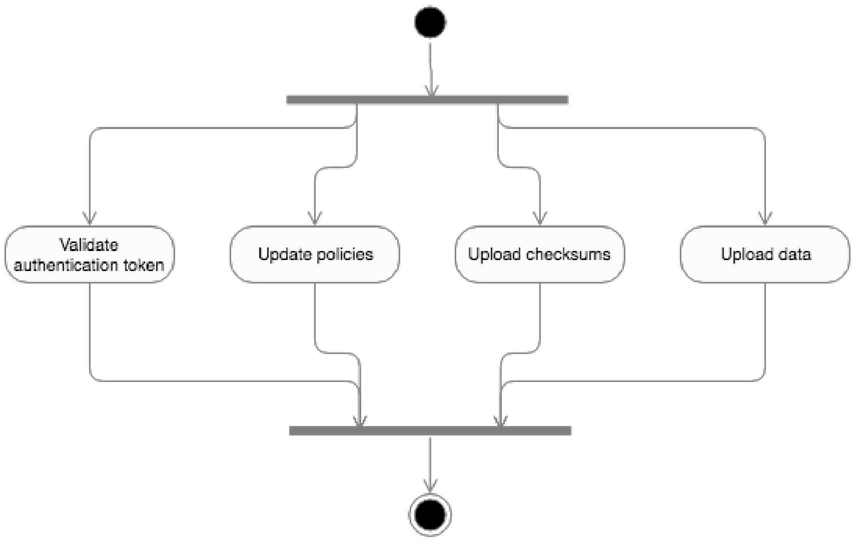


Figure 8.4: Periodic activation Activity Diagram

Authentication token validation

Industry best practises are to expire an authentication token after some period of time, to limit its usefulness should it be acquired by an unauthorised third party [50]. To do this, the mobile client prompts the user for their credentials, in order to get a new authentication from the service. This process is shown in figure 8.5.

First, the mobile client checks the expiry date of the token. If the expiry date is within a specified period, it initiates the process of refreshing it.

In order to refresh the authentication token, the mobile client prompts the user for their credentials. Once the user enters their credentials, the mobile client retrieves a new authentication token from the service. If this process fails, an error message is displayed to the user, and the user is prompted for their credentials again.

The next process, executed at the same time as the credential refresh, is the periodic updating of the data collection policies, which is presented next.

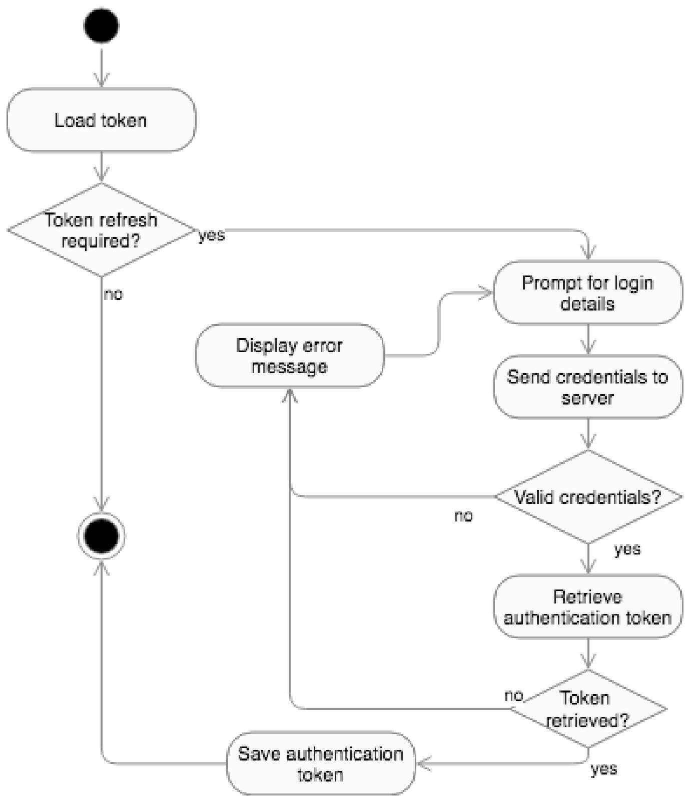


Figure 8.5: Validating and refreshing authentication token Activity Diagram

Updating policies

Since the mobile client keeps a local copy of the data collection policies, it needs to periodically check if the administrators have updated the policies. This is done by re-downloading the policy periodically, as shown in figure 8.6.

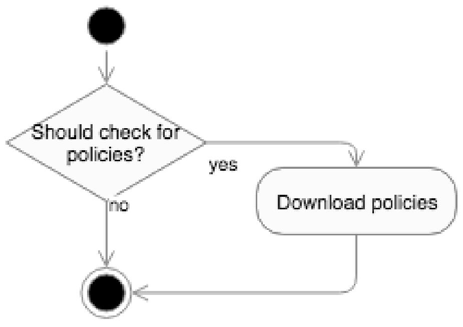


Figure 8.6: Refreshing data collection policies Activity Diagram

The final of the parallel processes is the data upload process, which is presented next.

Uploading data

The last process that is executed in parallel is the uploading of the collected data. This process is shown in figure 8.7.

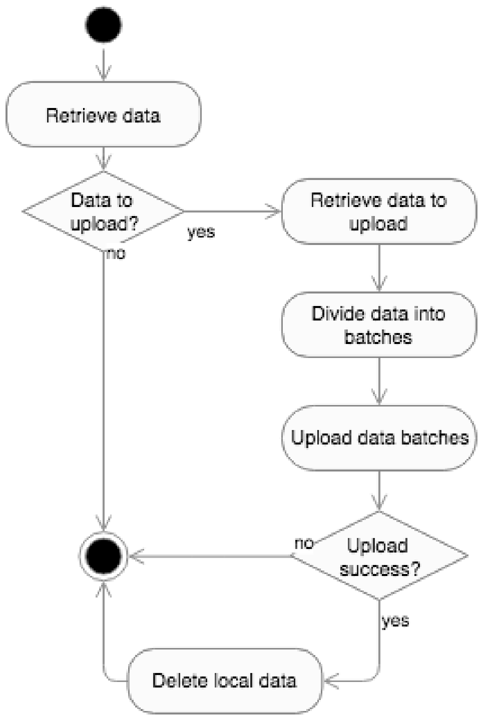


Figure 8.7: Uploading data Activity Diagram

The mobile client loads the data that has been collected, but not yet uploaded, from the local database and combines the data into batches to reduce network traffic. These batches of data are then uploaded to the server, as described in section 7.6. If the data upload fails, another upload will be attempted at a later time. If the data upload succeeded, the locally stored data is deleted.

This section presented the architecture of the mobile client that runs on the users' devices. These mobile clients collect data and transfer it to a server for further processing. The architecture of the server is outlined in the next section.

8.3 Server

The server receives requests from the mobile clients installed on the devices, processes these requests and store the results either in the database or the data store (presented later in this chapter). There are six distinct functions that the server performs, namely:

- logging the user in

- registering a device
- generating the authentication token
- loading the data collection policies for a user
- receiving checksums from the mobile clients
- receiving data from mobile clients.

In addition to these functions, the server also has some functionality that is used by more than one of the functions listed above, namely:

- a connection to the database,
- a connection to the data store,
- verifying hashes,
- verifying signatures
- and verifying the authentication token.

The server is visualised in figure 8.8. The top section of the figure shows the six distinct functions that the server performs and the bottom sections show shared functionality.

The rest of this section presents the components as shown in the figure, starting with the shared components.

8.3.1 Shared components

The shared components contain functionality that is used by more than one of the processes. As mentioned, there are five shared components, two of which are connections to external components (the database and data store), which are presented later in this chapter. The other three shared components are used for signature verification, hash verification and authentication token verification and are introduced next.

8.3.1.1 Authentication token verification

Before any of the functionality (except login and device registration) can be executed, the server first needs to verify that the authentication token that is passed with the request is valid. The process of verifying an authentication token is shown in figure 8.9.

First, the server looks up the authentication token that is associated with the device that is making the request. If the authentication token does not exist, it returns an error to the caller.

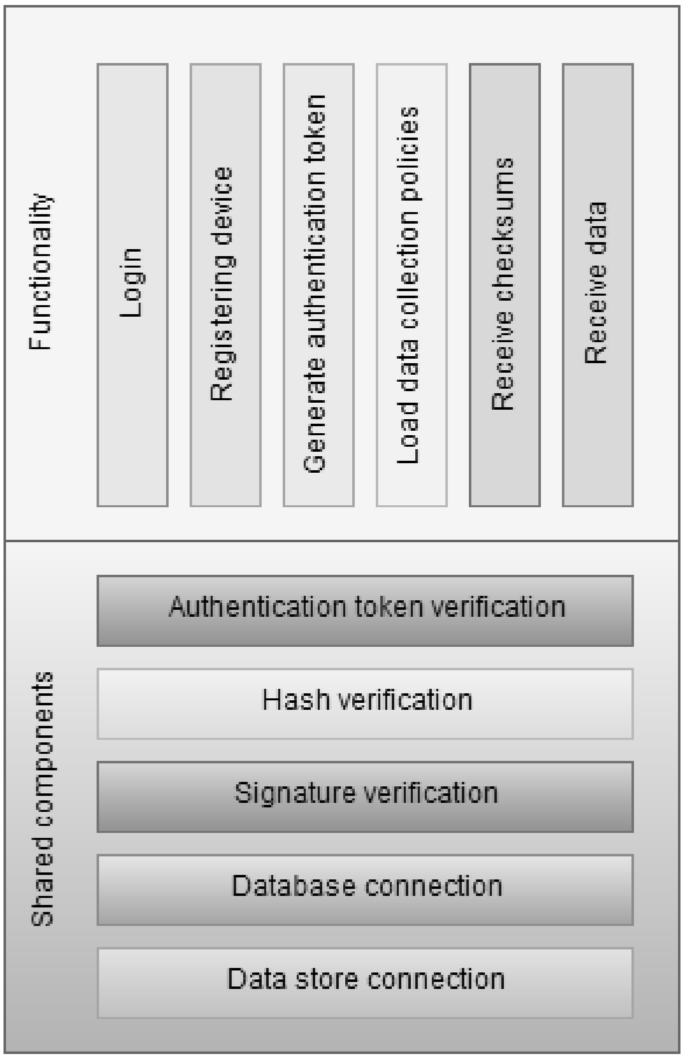


Figure 8.8: Server architecture

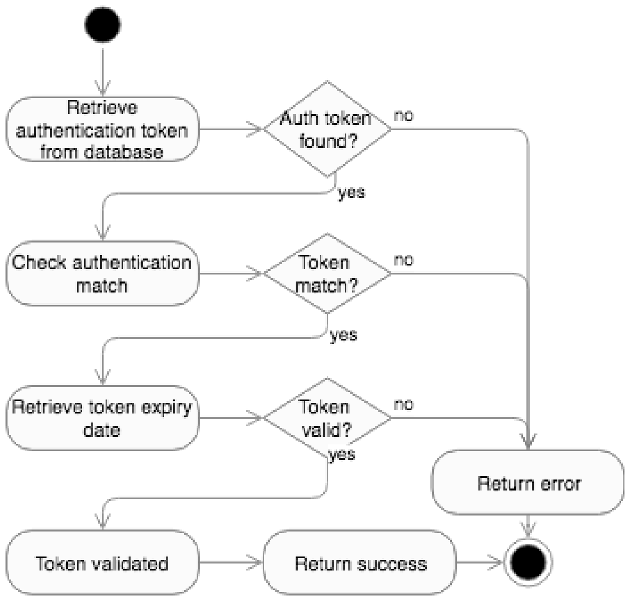


Figure 8.9: Authentication token verification

If the authentication token does exist, the server compares the authentication token received from the device with the authentication token stored in the database. If these do not match, an error is returned to the caller.

Finally, the server checks the expiry date of the authentication token and returns an error if the token is expired. If all of these checks pass, the server returns a success.

Verifying the authentication token is the first part of the process of verifying that a request is legitimate. The second part is verifying that the signature is correct, which is presented in the next section.

8.3.1.2 Signature verification

During the device registration process, the mobile client sends a public key to the server, which is then stored in the database. All subsequent requests from the client are signed using the corresponding private key. This mechanism allows the server to verify that a request was sent by the device in possession of the private key, as shown in figure 8.10.

To verify a signature, the server retrieves the public key corresponding to the device sending the request and decodes it from the format that it is stored into a format used by the signature algorithm. Then, it loads the public key, the signature and the plain text that was initially signed and verifies that the signature was created with the corresponding private key.

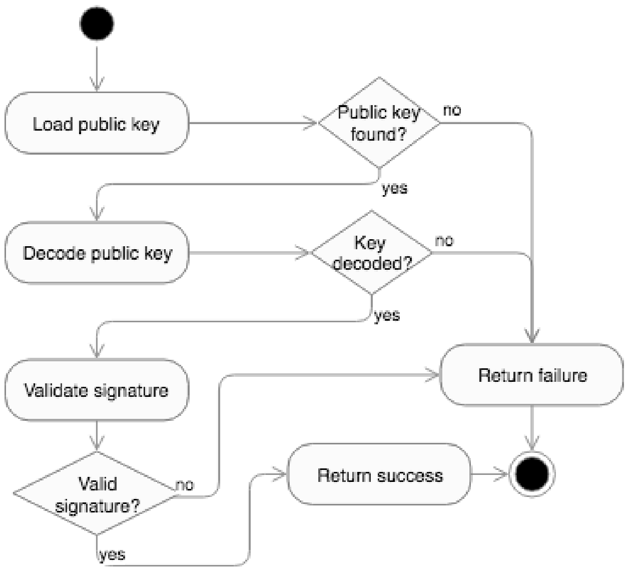


Figure 8.10: Signature verification

If any of these steps cannot be completed, the server returns an error to the caller. If the process succeeds, a success state is returned.

Verification of the signature is the second part of verifying the origin of a request. The final shared component is used for hash verification, which is used to verify the integrity of a request.

8.3.1.3 Hash verification

When data is uploaded to the server, it is imperative that the data has not been tampered with. To achieve this, the checksum of the data is calculated and compared with a checksum that has been sent to the server at an earlier time, as shown in figure 8.11.

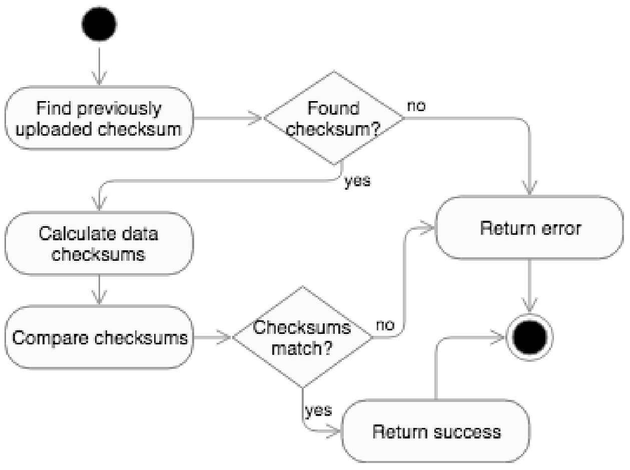


Figure 8.11: Hash verification

First, the server looks up the checksum that has been previously sent to it. Next, it calculates the checksum of the data it has just received. Finally, it compares the two checksums with each other. If the previous checksum cannot be found, or if the checksums do not match, an error is returned to the caller.

This section covered functionality that is shared by all the processes in the server. The next section presents the communication that the server receives from the mobile client.

8.3.2 Functionality

As mentioned previously, the server performs six different functions that are invoked by a request from the mobile client. These six functions are presented next, starting with the login functionality.

8.3.2.1 Log in

The first step when the mobile client is installed on a new device is to log into the client. The mobile client passes the user's username and password to the server and the process shown in figure 8.12 is started. Alternatively, the mobile client can also pass the server an authentication token. The server returns the resulting status to the device client.

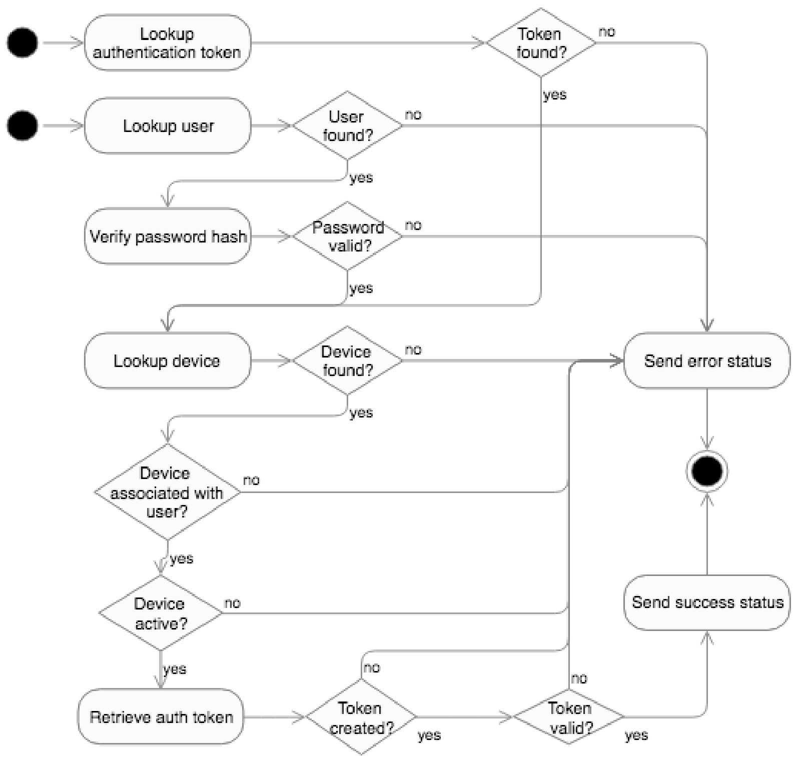


Figure 8.12: Login

First, the user is looked up in the database, and their password is compared with the stored password hash. Alternatively, if an authentication token was supplied, the authentication token is looked up. If the user cannot be found in the database, the password hashes do not match, or the authentication token is not found, the server sends an `Invalid credentials` message to the mobile client.

If the user's credentials are valid, the server looks up the identifier of the device that is sending this request. If the device making the request is not found in the database, the server notifies the mobile client with a `Device not found` message. If the device is registered to another user, the mobile client is told that `Device belongs to someone else`. Finally, if the device has been marked inactive by an administrator, the server returns `Device inactive`.

Once the server has checked that the device is active and belongs to the user attempting to log in, it checks if there is an authentication token associated with this device. If no authentication token can be found, the server notifies the mobile client with an `Auth token not created`. Alternatively, if a token exists, the server can either send a `Auth token expired` or `Auth token valid`, depending on the token's expiry date.

At this point, the mobile client may initiate another process, like registration, or go to running in the background, depending on the response from the server. On initial installation, the next process to be invoked is the registration of a device, which is introduced next.

8.3.2.2 Registering a device

Before the mobile client can start collecting data, the device that it is installed on needs to be registered to a user. The mobile client sends the user's credentials, the device identifier and the public key to the server, which initiates the device registration process shown in figure 8.13.

First, the user's credentials are verified in the same way as described in the previous section and an error message is returned if the credentials do not match.

After that, the public key is decoded from the format that is used for data transmission and stored in the database, along with the device identifier. This device is then associated with the logged in user.

Once a device has been registered to a user, an authentication token can be generated, which will allow the mobile client to access server functionality without requiring the user's credentials. This process is presented next.

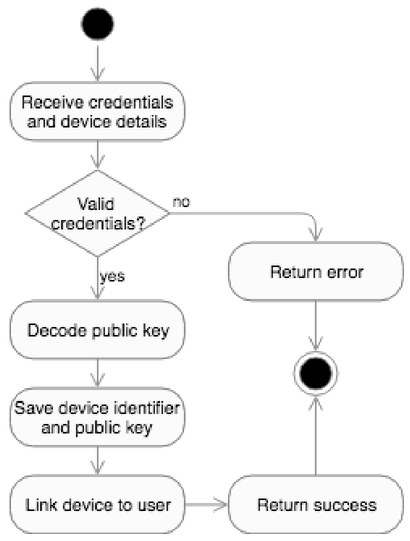


Figure 8.13: Device registration

8.3.2.3 Generating an authentication token

To allow the mobile client to function in the background, the system makes use of a token. This token needs to be refreshed periodically, which will kick off the process shown in figure 8.14.

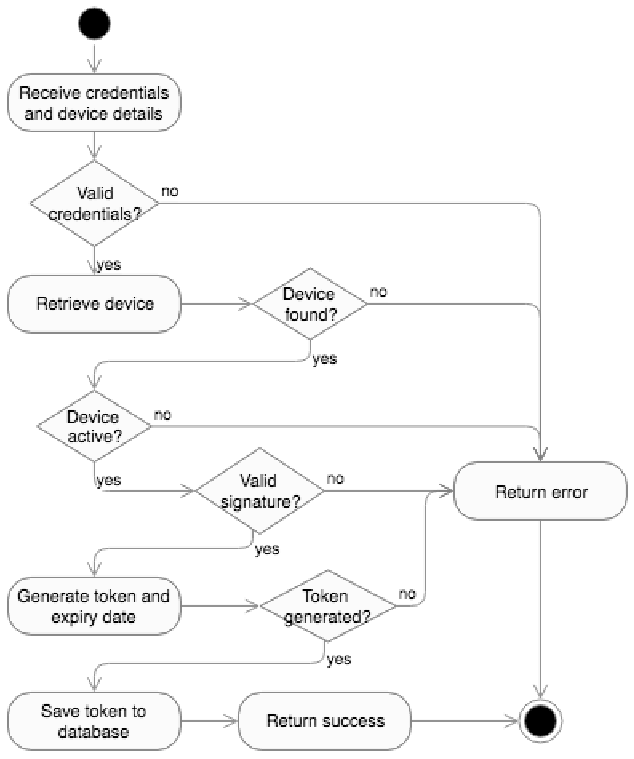


Figure 8.14: Generate authentication token

As with the previous two processes, the first step in this process is to verify the user's credentials. Once that has been done, the device details are retrieved from the database, and the server verifies that the device has not been marked as inactive by an administrator.

Next, the server verifies the request signature against the public key received in the previous process by following the steps described in section 8.3.1.2. If any of the checks fails, an error is returned to the caller.

If all the checks pass, the next step is to generate the authentication token and set the expiry date. The token is generated randomly, and the expiry date is set to some time in the future. Finally, both these values are stored in the database and then returned to the caller.

Once the mobile client is in possession of a valid token, it can execute the remaining processes, including loading the data collection policies, which is presented in the next section.

8.3.2.4 Loading data collection policies

Before the mobile client can start collecting data, it needs to know what data to collect. Which data should be collected is determined by retrieving the policies from the server. When a mobile client initiates the data collection policy download, the process shown in figure 8.15 is started.

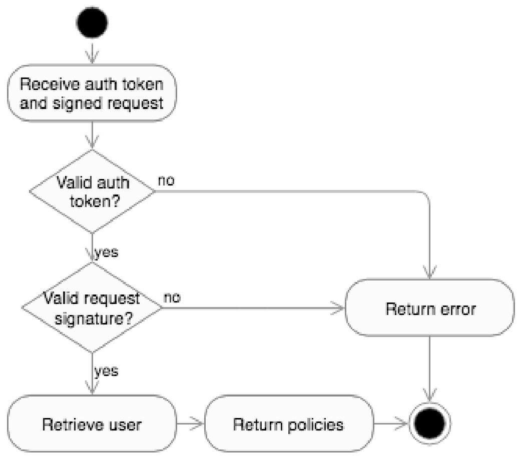


Figure 8.15: Load data collection policies

Similarly to the other processes, the first step is to verify the authentication token that has been sent. After that, the server verifies the request's signature against the public key that has been stored as part of device registration.

Once the request has been verified, the server retrieves the user associated with the authentication token and then looks up the policies that have been associated with the user (the process of associating policies with a user is presented later in this chapter). Finally, the processes are transformed into a format compatible with the mobile client and returned to the caller.

Once the policies has been downloaded, data collection can start on the mobile client. The next point where the client and the server interacts is when data checksums are uploaded. This is presented next.

8.3.2.5 Receiving checksums

As mentioned previously, the mobile client uploads the checksums as it collects the data. When a checksum upload request is sent to the server, the process shown in figure 8.16 is started.

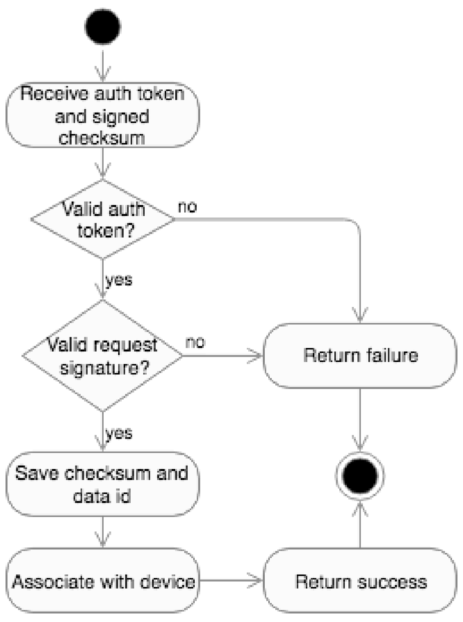


Figure 8.16: Receive checksums

As with the previous processes, the first steps in this process is to verify the authentication token and the signature associated with the request and return an error to the caller if the checks do not succeed.

Once the request has been verified, the server stores the checksum value, the data identifier, and the collection date into the data store. After this, it associates the checksum entry with the device identifier.

The final process in the list of server processes is the process of receiving data from the mobile client. This process is introduced next.

8.3.2.6 Receiving data

Periodically, the mobile client will send a batch of collected data to the server to process. When such a request is received, the process shown in figure 8.17 is started.

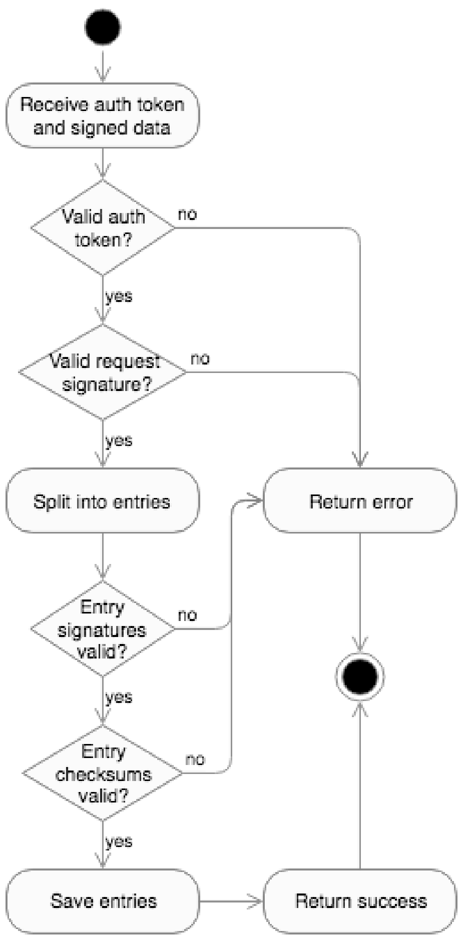


Figure 8.17: Receive data

Again, the first steps in this process is to verify the authentication token and the signature associated with this request. Additionally, the public key associated with the device making the request is retrieved from the database.

Next, the batch that was sent by the mobile client is split into separate data entries and every data entry is processed separately.

For every entry, the entry's signature is verified against the device's public key. Once the signature has been verified, the server looks up the checksum that has previously been associated with this data identifier and the checksum of the data is verified as described in section 8.3.1.3. Finally, the data entry is saved in the datastore. If the checksum verification failed, the data entry is flagged.

This section described the architecture of the server that is connected to by the mobile clients. The next section presents the database that is utilised by the server to store its data.

8.4 Database

In order for the system to function properly, the data needs to be stored somewhere. This system makes use of two types of storage, a relational database and a document store. This section introduces the relational database, visualised in figure 8.18, which stores data related to users, devices, policies and audit events.

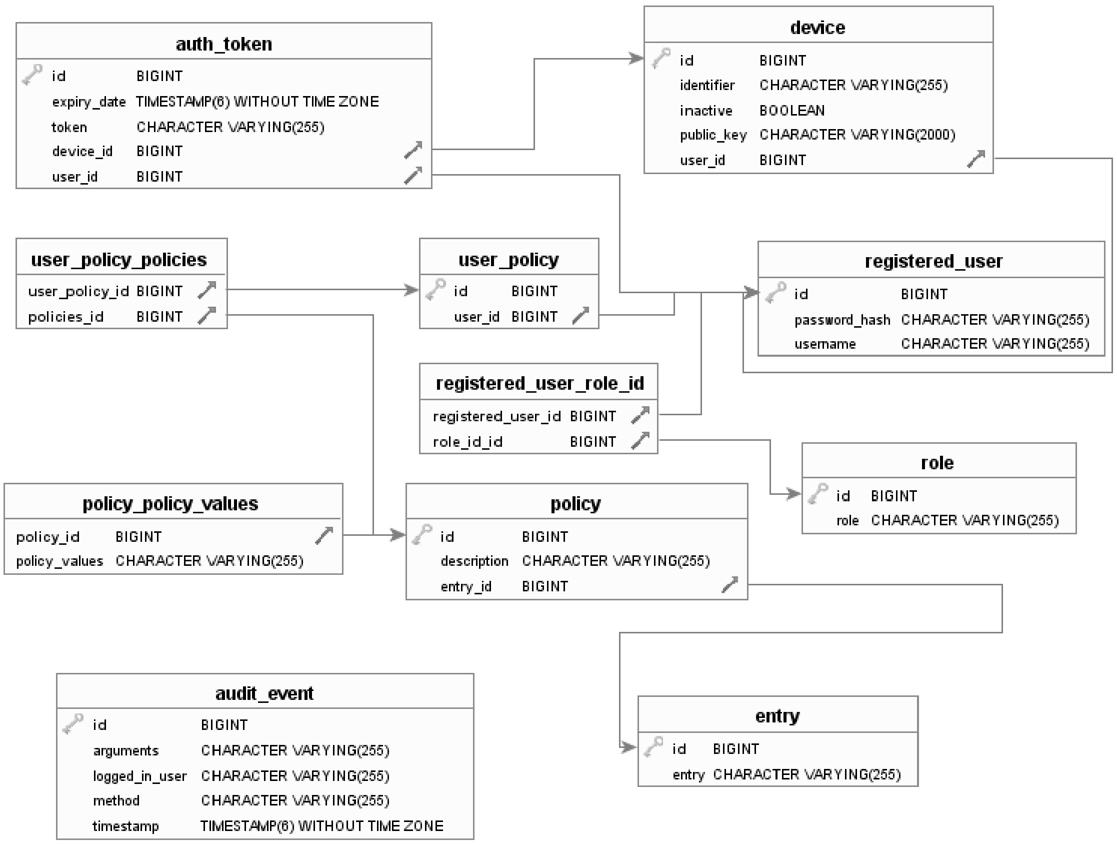

auth_token
id BIGINT
expiry_date TIMESTAMP(6) WITHOUT TIME ZONE
token CHARACTER VARYING(255)
device_id BIGINT
user_id BIGINT
device
id BIGINT
identifier CHARACTER VARYING(255)
inactive BOOLEAN
public_key CHARACTER VARYING(2000)
user_id BIGINT
user_policy_policies
user_policy_id BIGINT
policies_id BIGINT
user_policy
id BIGINT
user_id BIGINT
registered_user
id BIGINT
password_hash CHARACTER VARYING(255)
username CHARACTER VARYING(255)
registered_user_role_id
registered_user_id BIGINT
role_id_id BIGINT
role
id BIGINT
role CHARACTER VARYING(255)
policy_policy_values
policy_id BIGINT
policy_values CHARACTER VARYING(255)
policy
id BIGINT
description CHARACTER VARYING(255)
entry_id BIGINT
audit_event
id BIGINT
arguments CHARACTER VARYING(255)
logged_in_user CHARACTER VARYING(255)
method CHARACTER VARYING(255)
timestamp TIMESTAMP(6) WITHOUT TIME ZONE
entry
id BIGINT
entry CHARACTER VARYING(255)

Figure 8.18: Database diagram

The details about the users are stored in the **registered_user**. This table contains the user's username, password hash and a unique id for the user that is used to link the user to entries in other tables. The database also contains a **role** table, shown in table 8.1 that contains all possible roles that a user can have and a **registered_user_role_id** table that links a user to one or more roles.

The **device** table contains the details of devices that have been registered to users, including the public key associated with the device and the device's unique

id	**role**
1	ADMIN
2	USER
3	FORENSIC

Table 8.1: Roles table

identifier. Additionally, it contains a flag indicating whether this device has been marked as inactive.

The **auth_token** table stores the details of the authentication tokens that have been generated by the system. An entry in this table is linked to both a device and a user and contains the value of the token and the expiry date of said token.

The majority of the tables that the author has not mentioned yet is related to the storage of policies. The **entry** table contains a list of types that a policy can be, as shown in table 8.2.

id	**entry**
1	DNS
2	CONNECT
3	INSTALL
4	PASSWORD

Table 8.2: Entry table

The **policy** table contains the policy definitions, linked to the **entry** table and assigning each policy a unique identifier. Since each policy may have multiple values assigned to it, the values are stored in the **policy_policy_values** and linked to the policy using the policy's unique identifier.

Finally, the policies have to be linked to users. Since there is a many-to-many relationship between users and policies, two intermediate tables, namely **user_policy_policies** and **user_policy**, are used to assign policies to a user.

The final table in this database is the **audit_event** table, which is used to store the audit events generated by the administrator console (presented later in this chapter). These entries contain the user who performed the action, the action they performed and the arguments that were sent to that action. The entry also contains a timestamp indicating when this action was taken.

This section presented the relational database that is used to store the well-structured data that is required and generated by the solution. However, some of the data is not suitable for a relational database and is instead stored in a document store. This document store is introduced in the next section.

8.5 Data store

Some of the data that is generated by the solution is not suitable for a relational database, as the data is unstructured or too complex to be efficiently modelled in a relational fashion. Additionally, some data should be stored separately from the main database to allow for redundancy and robustness in the system. This section shows the structure of the data store for the four pieces of data that is stored here, namely

- checksums,
- collected data,
- reports and
- audit checksums.

Figure 8.19 shows how storage is structured for the **checksum** and **collected data** objects. The **checksum** structure contains a unique identifier, the calculated checksum, the identifier of the device that sent this checksum, the data identifier generated by the device and the type of the policy that triggered this data collection. The model also contains the data collection timestamp and the timestamp of when the data was uploaded.

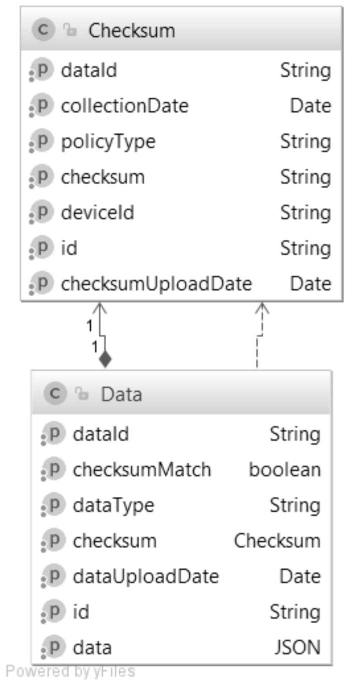
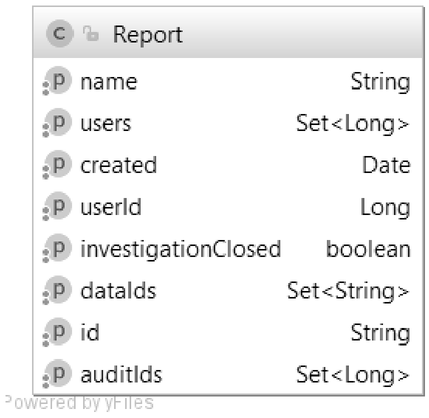


Figure 8.19: Data store

The **collected data** structure contains a unique identifier, a link to the **checksum**, as well as the data identifier generated by the device and a flag indicating if the server could verify that the checksums matched at the time of upload. It also contains the actual data that has been uploaded, the media type of the uploaded data and the date that the data was uploaded to the server.

The **report** structure shown in figure 8.20 contains all the details of a report (reporting is covered in more detail later in this chapter). The details include the name of the report, a list of the identifiers of users that are being investigated, the identifier of the user that initiated the investigation and the date the investigation was initiated. Also, it contains a list of identifiers of the data entries that are relevant to the investigation, as well as identifiers of audit entries and a flag indicating whether the investigation has been completed.



Report	
name	String
users	Set<Long>
created	Date
userId	Long
investigationClosed	boolean
dataIds	Set<String>
id	String
auditIds	Set<Long>

Figure 8.20: Report store

The last set of data that is stored in the document store is the **audit checksums**, shown in figure 8.21. These entries are not stored in the document store because they are unsuited to relational databases, but rather to ensure integrity should a malicious third party access the relational database. If the audit logs are modified in any way, the datastore will still contain the identifiers of the audit logs, as well as checksums that could be used to verify integrity.

This section presented the document store that is used to store data that is not well suited to the primary relational database. At various points during this section and the previous one, the author mentioned functionality related to administration and digital forensic investigations. The component enabling these are covered next.

8.6 Application console

The final component in the solution is the console. This component has two main functions, namely administration and forensic investigations, with some shared

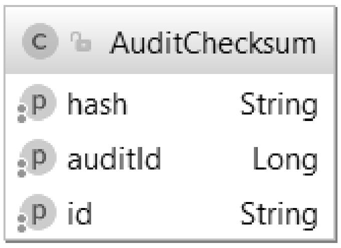


Figure 8.21: Audit checksum store

components. The components of the console can be seen in figure 8.22, showing the shared components at the bottom and the administration and forensic investigation components at the top. This section covers these components in more detail. First, the author presents the shared components.

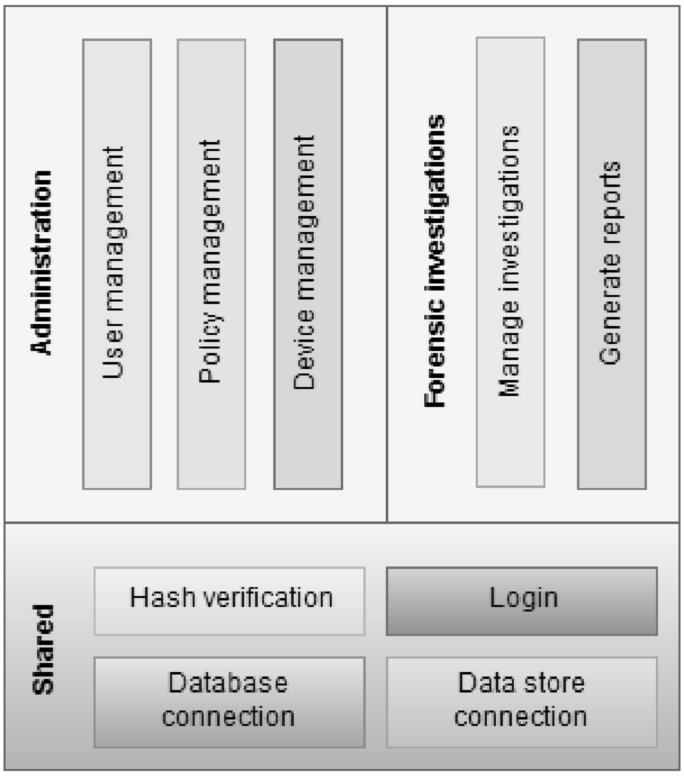


Figure 8.22: Console components

8.6.1 Shared components

There are four shared components in the console, namely

- hash verification,
- login,

- a database connection and
- a datastore connection.

This section does not cover the database and datastore connection, as these have been presented previously in this chapter. The hash verification component functions the same as the server's hash component outlined in section 8.3.1.3.

The **login** component prompts the administrator or digital forensic investigator for their username and password and verifies their credentials. Additionally, it makes sure that the user has the correct authorisation to access the functionality they are trying to access. This process is shown in figure 8.23.

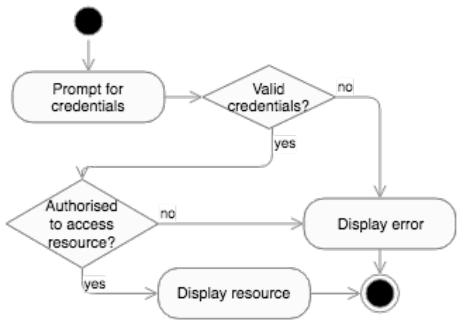


Figure 8.23: Console login process

The shared components enable the functionality of the administrator console and the digital forensic investigations. Next, the author presents the administrator console.

8.6.2 Administration console

The administrator console is used by the system administrators to manage users, as well as their associated policies and devices. This console also gives an administrator an overview of the system. The rest of this section presents the administration console, along with mockups on how such a console could be designed.

The policy management section of the console allows the administrators to create new policies and edit existing policies, adding and removing the collection details, as shown in figure 8.24. This allows the administrators to adjust the system to circumstances.

The user management section of the administrator console gives the administrator the ability to add, view and edit users. To add a user, the administrator enters a username and a temporary password, which is then written to the database using the database connection. A mock-up is shown in figure 8.25

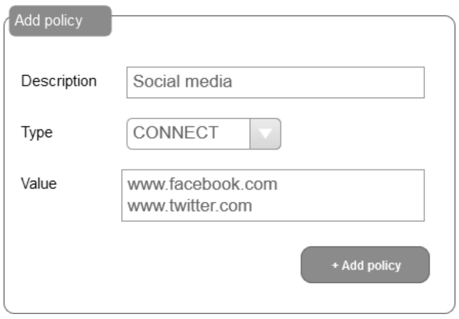


Figure 8.24: Mockup of adding a new policy

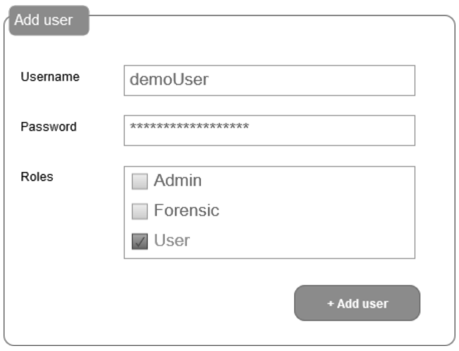


Figure 8.25: Add a new user

Once a user has been created, the administrator can view and edit their details, as well as viewing their devices and associated policies. The administrator can deactivate a device, should it become necessary (figure 8.26).

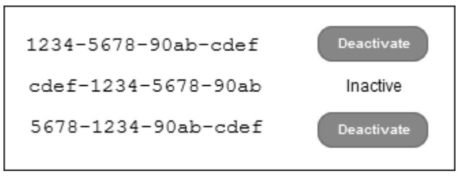
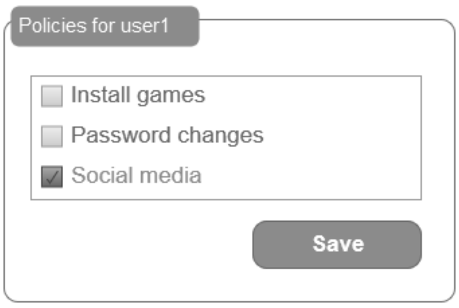


Figure 8.26: Deactivate a device

Additionally, the administrator can associate and remove policies from a user's profile, which will dictate the data that the mobile client collects and sends back to the server. Having this type of control over the policies associated with a user allows the administrators to fine-tune the data collection based on the specific circumstances or statutory requirements.

The system administrators maintain the users, policies and the association between the two. However, they do not have direct access to the data that has been gathered from the mobile devices, since this is the purview of the digital forensic investigators. The next section presents the console that is used by the investigators to conduct digital forensic investigations.


Policies for user1

- [ ] Install games
- [ ] Password changes
- [x] Social media

Figure 8.27: Change policies associated with user

8.6.3 Digital forensic investigations

When an incident occurs, and an investigation is started, the digital forensic investigators need to have access to the data that has been gathered in order to reconstruct the events that are being investigated. This section describes the digital forensic investigation component.

Figure 8.28 shows the process that investigators go through in the process of completing an investigation.

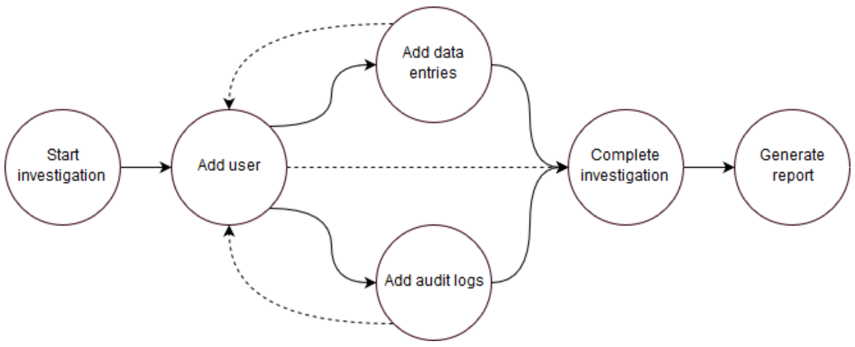


Figure 8.28: Report steps

First, the digital forensic investigator starts an investigation, entering all the particulars of the investigation into the system. Next, the investigator selects the users whose data has to be investigated and add them to the active investigation.

Once users have been added to the investigation, the investigator can browse data records and audit logs associated with those users and flag relevant entries to be added to the report.

After the investigator has completed their investigation, they close the investigation, after which no more modification is allowed. The system generates a report containing all the data that they have selected as relevant, as well as a document outline for writing up the investigation.

This section presented the console component, both for system administrators and digital forensic investigators. The next section concludes this chapter.

8.7 Conclusion

This chapter presented the architecture of a solution implementing the high-level model outlined in chapter 6, covering how each of the components can be architected to achieve the stated goal of adding digital forensic readiness to an MDM system.

This concludes this section of the research. The next section evaluates the proposed model and architecture according to the international umbrella standard for digital forensic investigations, ISO 27043.

CHAPTER 9

Evaluation according to ISO 27043

9.1 Introduction

Part of the design science process is to evaluate the artefacts that have been produced by the designer of the solution. The previous chapters presented constructs, models and methods, as well as an architecture to be used to solve the stated problem.

This chapter evaluates the design science model and methods, as well as the presented architecture, according to the International Standard ISO 27043 that governs Incident investigation principles and processes [40].

The standard identifies five different types of processes, namely:

1. readiness processes,
2. initialization processes,
3. acquisitive processes,
4. investigative processes and
5. concurrent processes.

The full digital forensic process, as defined in ISO 27043 is reproduced in figure 9.1. This figure has been modified from the original to highlight where the solution proposed by this research enables and interacts with the standard.

As highlighted in yellow in figure 9.1, the prototype enables the digital forensic process in three of the five stages, namely readiness and investigative, as well as some of the concurrent processes. The rest of this chapter presents how the solution enables the highlighted processes, as well as showing how it does not interact with the other process. Processes that are not interacted with are indicated with an asterisk (*). First, the author covers the concurrent processes.

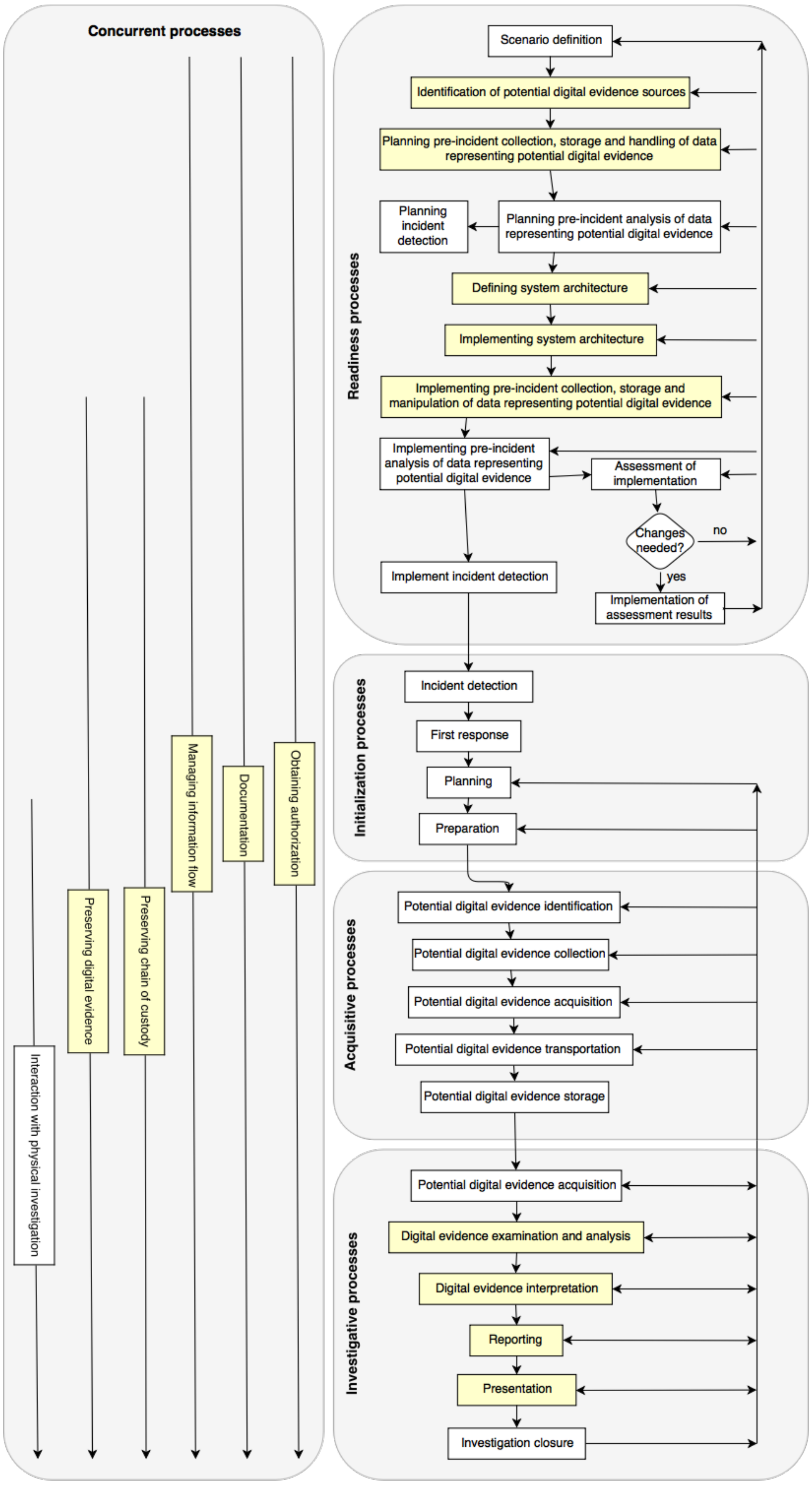

Concurrent processes
Interaction with physical investigation
Preserving digital evidence
Preserving chain of custody
Managing information flow
Documentation
Obtaining authorization
Readiness processes
Scenario definition
Identification of potential digital evidence sources
Planning pre-incident collection, storage and handling of data representing potential digital evidence
Planning incident detection
Planning pre-incident analysis of data representing potential digital evidence
Defining system architecture
Implementing system architecture
Implementing pre-incident collection, storage and manipulation of data representing potential digital evidence
Implementing pre-incident analysis of data representing potential digital evidence
Assessment of implementation
Changes needed?
no
yes
Implementation of assessment results
Implement incident detection
Initialization processes
Incident detection
First response
Planning
Preparation
Acquisitive processes
Potential digital evidence identification
Potential digital evidence collection
Potential digital evidence acquisition
Potential digital evidence transportation
Potential digital evidence storage
Investigative processes
Potential digital evidence acquisition
Digital evidence examination and analysis
Digital evidence interpretation
Reporting
Presentation
Investigation closure

Figure 9.1: ISO 270143 processes [40] with highlighted processes

9.2 Concurrent processes

Concurrent processes are applied throughout an investigation since they are applicable to multiple stages. These processes embody the principles of the digital forensic process [40].

The solution proposed in the previous chapters covers five of these processes, namely preserving digital evidence, preserving the chain of custody, documentation, managing information flow and obtaining authorisation. The solution does not enable the process of interacting with the physical investigation, since evidence is collected before the investigation is initiated. The rest of this section lists the digital forensic concurrent processes and presents how the solution interacts with them.

Preserving digital evidence. The solution preserves the digital evidence it collects from a device by uploading it to a trusted server using various mechanisms to ensure the integrity of the data, proving that it has not been tampered with.

Preserving chain of custody. The solution preserves the chain of custody for digital evidence by documenting the process of sending evidence between components. This documentation allows for a clear view of how the evidence proceeded through the system.

Managing information flow. The solution enables the management of information flow by attaching metadata to the collected evidence, showing its flow through the system and verifying the integrity of the data at critical points.

Documentation. Since the solution records all significant events attached to a piece of evidence, the process can be traced and reproduced. Additionally, the solution allows investigators to produce this event log in the form of a document. The solution also produces other documentation, presented in the investigative processes section of this chapter.

Obtaining authorisation. To make use of the solution, authorisation needs to be obtained from various parties. To install the mobile component on a user's device, the permission of the user needs to be obtained. Authorisation needs to be obtained to update the policies on the system and, finally, investigations need to be authorised before being started.

Interaction with physical investigation*. The solution does not interact with the physical investigation, as all the evidence are collected before the investigation starts. When an investigation is initiated, the evidence is already present in the system.

The processes presented in this section interacts with other processes throughout the entire digital forensic process, as can be seen in figure 9.1. The first set of non-concurrent processes when the digital forensic process is initiated are the readiness processes, which are outlined next.

9.3 Readiness processes

Readiness processes are executed to set up an organisation for a digital forensic investigation. When an investigation is required, the organisation can maximise the potential of the digital evidence while minimising the costs related to the investigation [40].

The solution proposed in the previous two chapters enables a number of readiness processes, namely identifying potential digital evidence, the planning and implementation of pre-incident collection, storage and manipulation, and the definition and implementation of a system architecture.

This section presented the processes that this solution enables, as well as why the solution does not interact with the other processes. Processes that the solution does not interact with are marked with a star.

Scenario definition.* The proposed solution does not cover the scenario definition process, as scenarios will be specific to each organisation's threat profile.

Identification of potential digital evidence sources. To allow for the collection of potential evidence, potential evidence sources were identified. This is required since the evidence collection happens on a device that is not under the control of the organisation.

Planning pre-incident collection, storage and handling of data representing potential digital evidence. The previous chapters planned in detail how the solution would go about the collection, storage and handling of the potential evidence that the system is set up to collect.

Planning incident detection.* Currently, the proposed solution does not support automated incident detection. This may be addressed in future work.

Planning pre-incident analysis of data representing potential digital evidence.* Currently, the solution does not support automated pre-incident analysis. This may be addressed in future work.

Defining system architecture. In the architecture chapter, the author defined the system's architecture, detailing the composition of the components and how they interact with each other.

Implementing system architecture and pre-incident collection, storage and manipulation of data representing potential digital evidence. A prototype was implemented to prove the feasibility of this solution. In the next part of this research, the author presents this prototype in more detail.

Implementing and assessment of pre-incident analysis of data representing potential digital evidence and incident detection.* As mentioned before, the presented solution does not currently support the automated analysis of data and incident detection.

After an organisation has completed the readiness processes, investigations will be less expensive and will be completed quicker. The other categories of processes deal with completing the actual investigation.

The proposed solution does not enable the initialisation and acquisitive processes, as it operates during the readiness and investigative phases of an investigation only. Next, the enabling of investigative processes are presented.

9.4 Investigative processes

Investigation processes are executed when an incident occurs that requires investigation. They include analysis of evidence, reporting the results and presenting said results to interested parties [40].

The solution touches almost all of the investigative processes, namely evidence examination and analysis, evidence interpretation and reporting, as well as the presentation of the conclusions. There are also two processes that the solution does not interact with, which is again marked with a star.

Potential digital evidence acquisition.* The solution collects potential digital evidence as part of the *readiness processes*. At the point where an investigation is launched, the evidence has already been collected. As such, the solution does not touch this process.

Digital evidence examination and analysis. The solution as proposed in the previous chapters allow investigators to view the data that has been collected, as well as the metadata related to the collection and processing of the data.

Digital evidence interpretation. The solution allows investigators to view the data and metadata. The investigators can also compare different sets of data or view a number of datasets in order to establish a pattern.

Reporting. Once an investigator has selected the datasets that are relevant to the investigation, the solution has the ability to generate a report that encompasses the data, allowing the investigator to add their interpretation.

Presentation. The solution creates a report during the reporting process. This report contains prompts to be completed by the investigator. Once the investigator has completed the prompts, the report should be usable during the presentation process.

Investigation closure.* The closure of an investigation is a human process and the proposed solution is not involved in this process.

This section presented the investigative processes as outlined in ISO 27043, showing in which ways the proposed solution enables the various processes. The next section concludes this chapter.

9.5 Conclusion

This chapter the evaluated the design science artefacts that have been produced so far against the umbrella standard for digital forensics, ISO 27043. Figure 9.2 shows the revised diagram containing only the processes that are relevant to this solution. This diagram is used in the final prototype to indicate the process to the administrators and investigators.

The next part of this research creates the final design science artefacts that are still missing, namely the instantiations. This is done by first outlining the requirements to evaluate the instantiations and then creating the instantiations.

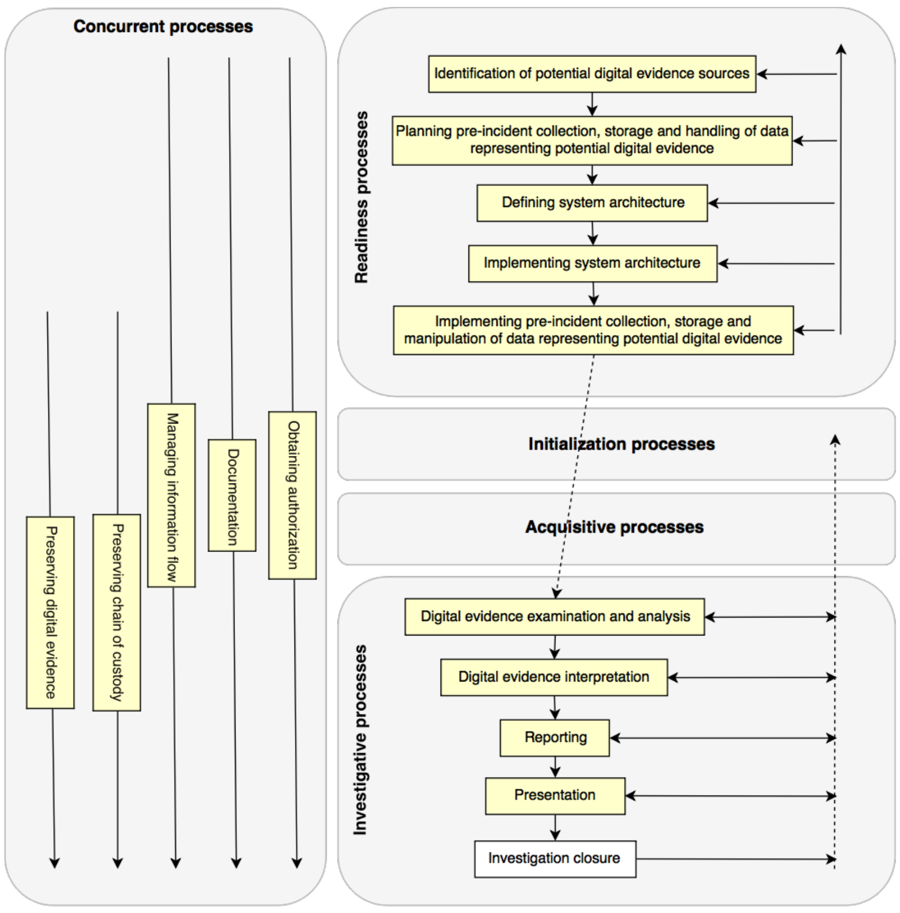

Concurrent processes
Preserving digital evidence
Preserving chain of custody
Managing information flow
Documentation
Obtaining authorization
Readiness processes
Identification of potential digital evidence sources
Planning pre-incident collection, storage and handling of data representing potential digital evidence
Defining system architecture
Implementing system architecture
Implementing pre-incident collection, storage and manipulation of data representing potential digital evidence
Initialization processes
Acquisitive processes
Investigative processes
Digital evidence examination and analysis
Digital evidence interpretation
Reporting
Presentation
Investigation closure

Figure 9.2: Modified figure showing only relevant processes

PART IV

Prototype

In this part, the author presents a prototype based on the model and architecture outlined in the previous chapters. First, the requirements for such a prototype is identified, and after that the prototype implementation is presented.

CHAPTER 10

Prototype Requirements

10.1 Introduction

The previous chapters have covered background information relevant to the problem statement, namely that most Mobile Device Management systems do not include a digital forensic readiness component, leaving investigators with little to no device-related historical data when an incident does occur. The author also described a high-level model and architecture for a solution to the stated problem.

Before a final prototype can be created, it is necessary to have a set of requirements that will allow for the evaluation of the proposed instantiations [75] as part of the evaluation process of design science. This chapter identifies the requirements that the instantiations have to adhere to for it to solve the problem.

Shrivathsan [88] identifies five different types of requirements, namely:

- **Business requirements** High-level goals of the organization.
- **Market Requirements** These requirement drill into the business requirements and outline market needs.
- **Functional Requirements** Covers the functionality of the product in detail.
- **Non-functional requirements** These are requirements that cannot be mapped directly to functionality but are necessary for the product to function.
- **UI Requirements** These are requirements related to the user interface of the product.

The rest of this chapter is dedicated to defining the requirements for the product. The author does this by dividing the requirements into the categories as mentioned above, starting with the business and market requirements.

10.2 Business and market requirements

Business and market requirements are the high-level goals of the organization commissioning the project [88].

The main business requirement for this solution is to **reduce the costs of digital forensic investigations**. Gathering data before the incident occurs has both cost and time implications as mentioned in chapter 5.

An additional requirement is to have **more complete data for investigations**. More complete data will allow the organisation to conduct more thorough investigations and will also speed up investigations since investigators will not have to search for data.

These are the two business and market requirements that the author identified. The next section presents the functional requirements that define the product from a user perspective.

10.3 Functional requirements

Lightsey [51] defines functional requirements as ”what the system must accomplish or must be able to do.” These requirements translate directly into what the user can see. The author split the functional requirements into three sections: mobile requirements, administration requirements and investigation requirements. First, the author defines the functional requirements related to the mobile device.

10.3.1 Mobile functional requirements

There are four functional requirements related to the mobile client.

The mobile device should be associated with a user. Since the data gathered is for the purpose of attributing actions to employees, a device has to be linked to an employee. If the data collected from the device cannot be linked to an employee, it has very little value in an investigation.

The mobile device should collect data about the user's actions. Data collection is the primary functional requirement in the system. When a user accesses a potentially dangerous resource, the mobile device should collect data about their actions, to simplify future investigations.

The device's data collection actions should be controlled by the administrators. Since the threat landscape changes rapidly, the administrators should be able to change which data is being collected to keep up with new threats and disable collection rules that are no longer relevant.

The device should upload the collected data to the investigators in a forensically sound manner. Since the investigators may not have direct access to the device at the point when an investigation happens, the mobile device should transmit the data to some system where the investigators can access it at any point in the future.

During the exploration of the functional requirements for the mobile device, the author mentioned that the administrators should control some of the functionality. Next, the author elaborates on the functional requirements as they relate to the administrators.

10.3.2 Administration functional requirements

The administrators maintain the users and the policies on the system. This section presents the functional requirements that will enable the administrators' tasks.

Administrators should be able to create data collection policies. First, the administrators should be able to define the policies that the mobile devices use to collect system data. As mentioned before, the threats to an organisation changes, as well as device functionality, changes over time, which necessitates policy changes.

Administrators should be able to create new users on the system. As new employees join the organisation, the administrators should have the ability to add them to the solution.

Administrators should be able to change the policies assigned to a user. As the role of an employee in the organisation changes, the policies associated with the user may change too. Administrators should have the ability to assign new policies to a user, as well as removing policies that are no longer relevant.

Administrators should be able to view devices associated with a user. Once a user has been registered on the system and has devices associated with their username, the administrators should be able to view the devices to allow them to troubleshoot any issues.

Administrators should be able to mark devices as inactive. When a user is no longer using a device, the administrators should mark it as inactive on the system, to prevent data from old devices from being collected.

Administrators are not allowed to view the data that is collected by the system, as that could be a privacy risk. The only users that can access the data are the digital forensic investigators. The requirements for those are presented next.

10.3.3 Investigation functional requirements

The final steps in the digital forensic process are related to the investigation of an incident. This section explores the functionality that is required for a digital forensic investigator to complete an investigation successfully.

The digital forensic investigator should be able to start an investigation. The digital forensic investigator should be able to start an investigation on the system, allowing them to associate users and data with the investigation, given that they have the correct administrative and judicial permissions to access the data.

The digital forensic investigator should be able to add users to an investigation. The solution should allow the digital forensic investigator to associate users with an investigation, effectively placing the user and their data under investigation.

The digital forensic investigator should be able to view the data related to the users under investigation. Once a user has been placed under investigation, the digital forensic investigator should be able to view all data related to the user and add any relevant data to the investigation.

The digital forensic investigator should be able to mark an investigation as complete. Once the digital forensic investigator is satisfied that they have completed their investigation and that all relevant users and data have been added to the investigation, they should have the ability to mark the investigation as complete. Once an investigation has been completed, the system should not allow any changes to it.

The digital forensic investigator should be able to generate a report from a completed investigation. Once an investigation has been completed, the system should allow the digital forensic investigator to generate a report containing all the relevant findings.

Mobile	**Administration**	**Investigation**
Associate with user	Create collection policies	Start investigation
Collect data	Create new users	Add users
Controlled by administrators	Change assigned policies	View data
Forensically sound	View devices	Complete investigation
	Deactivate devices	Generate report

Table 10.1: Functional requirements summary

This section presented the functional requirements that a system should adhere to in order to solve the stated problem. These functional requirements are summarized in table 10.1. The next section analyses the non-functional requirements of the system.

10.4 Non-functional requirements

Non-functional requirements are constrains placed on a system [82], such as availability. Pandey et al [67] defines non-functional requirements as system properties and Chung and Leite [22], as well as Nuseibeh and Easterbrook [63], mentions that non-functional requirements could also be described as "quality" attributes or requirements.

Roman [82] identifies six types of non-functional requirements, namely interface, performance, operating, lifecycle, economic and political requirements. Roman's definition of interface requirements has significant overlap with the UI requirements as mentioned above and are presented in section 10.5. Additionally, the author could not identify any significant economic, political and lifecycle requirements, and as such this section covers performance and operating requirements, starting with the performance requirements.

10.4.1 Performance requirements

Performance requirements places bounds on the system's performance. These could be related to processing time, reliability, security and resource usage [82]. The author identified a number of performance requirements on the mobile device, server and administrator interface.

Mobile data collection should run in the background. Since the purpose of the client running on the mobile device is to collect data on the user's actions, the mobile client should run unobtrusively, unless it requires input from the user.

Mobile data collection should not consume excessive resources. Since the resources on a mobile device are very constrained, the software running on the mobile device should be resource efficient, as not to impact the functionality of other applications negatively.

The server should always be available. Since the mobile devices can upload data to the server at any time, the server should always be available to receive requests from mobile clients.

The server should complete operations in a reasonable time. Since a potential large number of mobile devices may be connected to the server at any time, the server should complete operations efficiently and as quickly as possible.

The server should ensure the identity of the mobile device. To prevent a malicious party from corrupting the data in the system, the server should verify the identity of the mobile client before accepting any data from it.

The administration console should be available during business hours. Since all the users making use of the console will do so during business hours, it is imperative that the administration console be available during those times.

The administration console should be access controlled. Only users who have been given access to the system should be allowed to access the system.

Access levels should be verified before a user is allowed access to any sensitive functionality. To prevent horizontal privilege escalation, the user's access level should be checked before they are allowed access to any functionality.

This section explored the performance requirements that the author identified. The next section presents the operating requirements.

10.4.2 Operating requirements

Operating requirements are requirements related to the operation of a system, for example distribution of components, skill level considerations and availability [82].

Mobile data collection should not interrupt the user. Since the mobile client is supposed to run in the background, it should not interrupt or impact the user unless absolutely necessary.

The server should be accessible from anywhere. The users will carry their mobile devices around with them and as such, the mobile device should be able to connect to the server from anywhere.

The administration console should be accessible inside the organisation's network. Since the administrators and digital forensic investigators will be accessing the console while at work, the console should be accessible from anywhere inside the organisation's network.

Performance	**Operating**
Run in background	Don't interrupt user
Resource efficient	Server accessible from anywhere
Server availability	Console available in network
Server performance	
Mobile identity	
Console availability	
Access control	
Verify access levels	

Table 10.2: Non-functional requirements summary

This section presented the non-functional requirements that a system should adhere to in order to solve the stated problem. These functional requirements are summarized in table 10.2. The next section explores the user interface requirements of the system.

10.5 UI requirements

UI requirements specify how the User Interface of a system should behave [88], in order to give the user of a system the best experience possible [103]. Roman [82] defines it more broadly as "the ways the component and its environment interact". This section presents the User Interface requirements of the solution.

The mobile client should have a login screen. The mobile client should have a screen to prompt the user for their credentials when it is required.

The administration console should have a login screen. The console should prompt the user for their credentials before they are allowed to access any resources.

The console should have a screen for viewing and editing policies and users. The console should allow the administrators to view and edit the user and policy data that has been stored in the system.

The console should have a screen for managing investigations. To allow a digital forensic investigator to complete an investigation, the console should have a screen allowing them to manage their investigations.

UI
Client login screen
Console screen
View and edit users
View and edit policies
Manage investigations

Table 10.3: UI requirements summary

This section presented the user interface requirements for the solution. These requirements are summarised in table 10.3.The next section concludes this chapter.

10.6 Conclusion

This chapter identified and explored a number of requirements, including functional, non-functional, UI and business requirements. These requirements are used to assess whether or not the proposed prototype solves the problem as stated in chapter 1.

Now that the author has defined the requirements for a prototype, the next chapter ”demonstrates” the prototype that has been created by the author.

CHAPTER 11

Prototype Implementation

11.1 Introduction

The previous chapters proposed a high-level model and architecture for a solution to add digital forensic readiness to an MDM system. The chapters outlined the components, the interaction between the components and how the components would be structured in such a system, defining the design science constructs, model and methods.

To show that this model is viable, the author implemented a prototype of the system, creating a design science instantiation of the artefacts that have been defined. This chapter presents the prototype that has been developed, starting in the next section.

11.2 Prototype demonstration

The author demonstrates the prototype by utilising a fictional scenario. At every step, the author describes the scenario (inside the framed blocks), with the points at which the parties interact with the prototype **highlighted in bold** and after that shows how the prototype would function in that scenario. First, an introduction to the scenario:

ABC Incorporated is hiring a new manager. This manager will work with high-profile clients and as such have access to some sensitive information related both to ABC Inc and their clients. The successful candidate will be required to use their personal device for work purposes. After advertising the job and some rigorous interviews, the organisation decides to hire Mr X.

The use of the prototype starts when a new employee is appointed. The first of the actions that have to be taken on the prototype may be performed before the employee starts their employment.

The human resources department (HR) notifies the system administrator that Mr X will be joining the company. The system administrator, Lucy, registers Mr X on all the systems, **including the MDM system**. Since he is the first manager that will be using his personal device to access confidential information, she also sets up **the data collection policy** for managers.

The first thing that the system administrator sees when she accesses the administrative console of the prototype, is the login screen (figure 11.1). Once she has entered her credentials and the system has verified them, she is allowed access and sees a landing page, showing the current users and policies that has been defined in the system, which can be seen in figure 11.2.

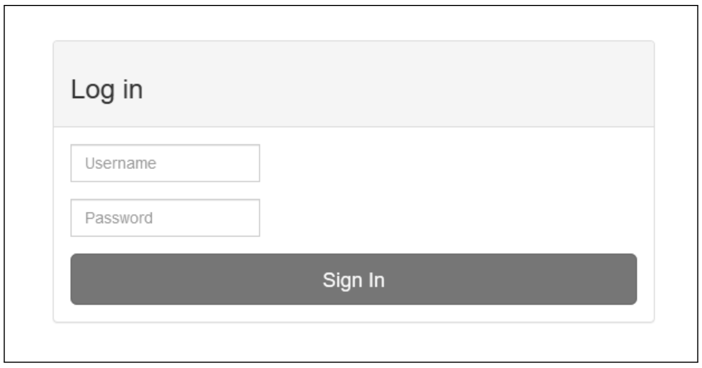


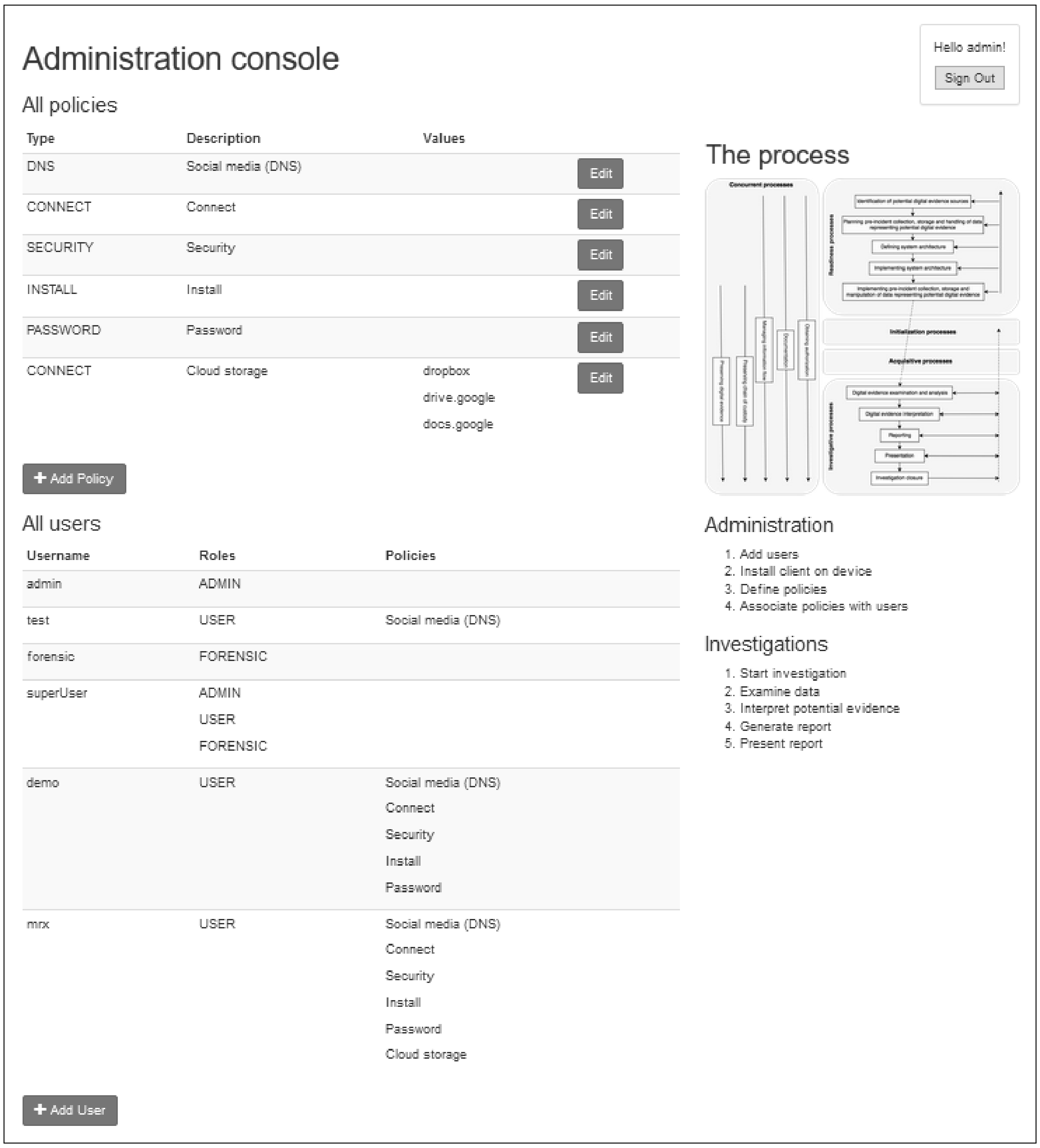
Figure 11.1: Administration console login

To create new credentials for Mr X, the system administrator clicks on the "Add user" button on the landing page (seen in figure 11.2) and enters the details of the user, including the username, a password and the roles that the user will have on the system. These roles may be

- **ADMIN** a system administrator
- **USER** a normal employee
- **FORENSIC** a digital forensic investigator

or any combination of the above. The system administrator selects the *USER* role for Mr X, as he is neither a system administrator or an investigator. The screen to add a new user can be seen in figure 11.3.

At this point in the process, the user for Mr X has been defined, and Mr X can log into the mobile application once it is installed on his device. However, no policies have been associated with Mr X, and as such, any devices associated with him will not send any data back to the server.



Hello admin!

Sign Out

Administration console

All policies

Type	Description	Values	
DNS	Social media (DNS)		Edit
CONNECT	Connect		Edit
SECURITY	Security		Edit
INSTALL	Install		Edit
PASSWORD	Password		Edit
CONNECT	Cloud storage	dropbox drive.google docs.google	Edit

+ Add Policy

All users

Username	Roles	Policies
admin	ADMIN	
test	USER	Social media (DNS)
forensic	FORENSIC	
superUser	ADMIN USER FORENSIC	
demo	USER	Social media (DNS) Connect Security Install Password
mrx	USER	Social media (DNS) Connect Security Install Password Cloud storage

+ Add User

The process

Concurrent processes
Readiness processes
Identification of potential digital evidence sources
Planning pre-incident collection, storage and handling of data representing potential digital evidence
Defining system architecture
Implementing system architecture
Implementing pre-incident collection, storage and manipulation of data representing potential digital evidence
Initialization processes
Acquisitive processes
Investigative processes
Digital evidence examination and analysis
Digital evidence interpretation
Reporting
Presentation
Investigation closure

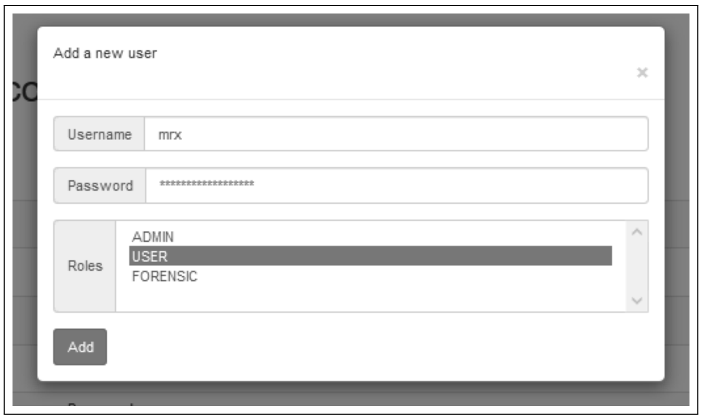
Administration

1. Add users
2. Install client on device
3. Define policies
4. Associate policies with users

Investigations

1. Start investigation
2. Examine data
3. Interpret potential evidence
4. Generate report
5. Present report

Figure 11.2: Administration landing page


Add a new user

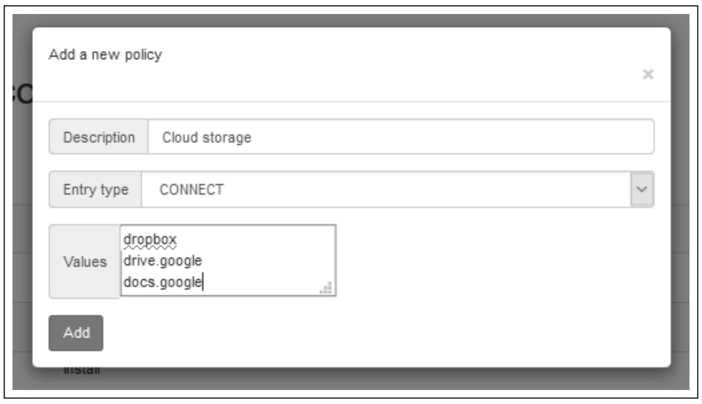
Username	mrx
Password	******************
Roles	ADMIN USER FORENSIC

Figure 11.3: Add a new user

As mentioned Mr X is the first manager to be registered on the new MDM prototype. This means that all the policies for middle management have not yet been defined. Specifically, ABC Inc does not allow managers to store company documents on cloud storage, because of the privacy laws in their country. The system administrator creates a policy which instructs the mobile client to monitor for connections to cloud storage providers.

To create a new policy, the system administrator clicks on the "Add policy" button on the landing page (figure 11.2). The administrator gives the policy a description and selects the type of the policy, in this case, *CONNECT*, since the events that the prototype has to monitor for is connection events. The other policy types that may be selected are *DNS* (monitors for DNS lookup events), *INSTALL* (monitors application installations) and *PASSWORD*, which monitors for password changes.

Next, the administrator enters the values that should trigger the data collection on the mobile device (figure 11.4). In this case, the administrator enters a list of cloud storage providers.

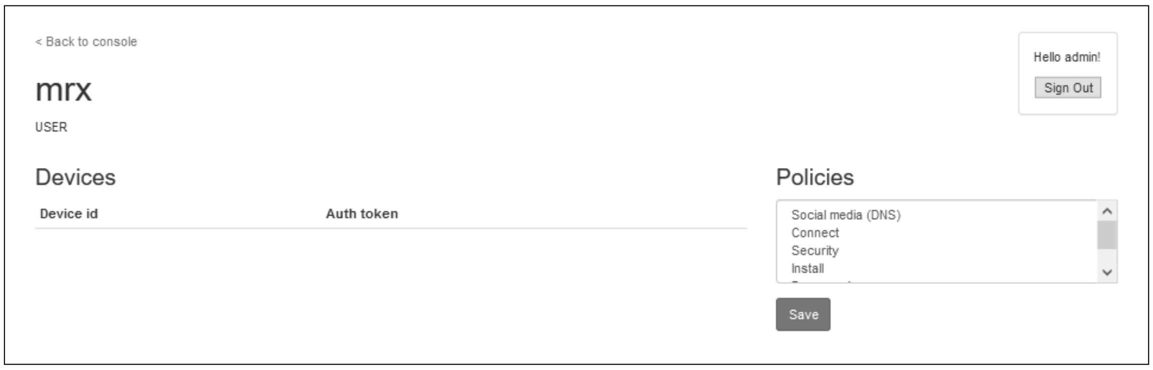

Add a new policy

Description	Cloud storage
Entry type	CONNECT
Values	dropbox drive.google docs.google

Figure 11.4: Add a new policy

After clicking on “Add”, the policy definition is stored in the database and is ready to be associated with a user. The next step in the process is for the administrator to associate the policy with Mr X.

To do this, the administrator clicks on the user’s entry on the landing page (figure 11.2) to access the user details page (figure 11.5) and selects the policies that have to be associated with the user, in this case, the “Cloud storage” policy that has just been defined. The administrator also selects some other policies that apply to all users and clicks on the “Save” button.



mrx

USER

Devices

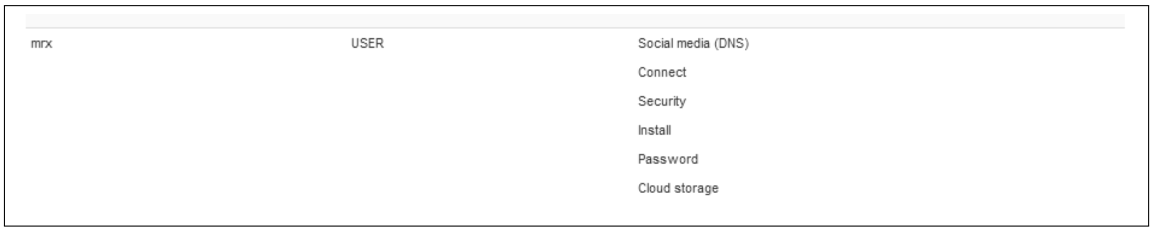
Device id	Auth token

Policies

Social media (DNS)
Connect
Security
Install

Figure 11.5: User details page

When the administrator goes back to the landing page, they can see that there are now policies associated with Mr X’s account (figure 11.6).



mrx	USER	Social media (DNS) Connect Security Install Password Cloud storage

Figure 11.6: The user has policies associated with it

This completes the actions that can be taken before a new employee has to interact with the system. The next steps can only be completed once the employee and their device are physically present.

When Mr X joins the company, he goes through the normal induction process with HR and undergoes the necessary training. Once he is ready to begin work, he is taken to the system administrator. She **installs the mobile client** on his personal device, setting up device permissions in the process and asks him to **log in** using his newly created credentials.

When Mr X comes to her office, the system administrator assists him in installing the application on his Android device. After the installation is complete, she makes sure to give the application the correct permissions to allow for monitoring.

After the application is successfully installed, the system administrator asks Mr X to log into the application using the credentials that have been created on the system previously and provided to him during his onboarding with HR. The mobile application login screen is shown in figure 11.7.

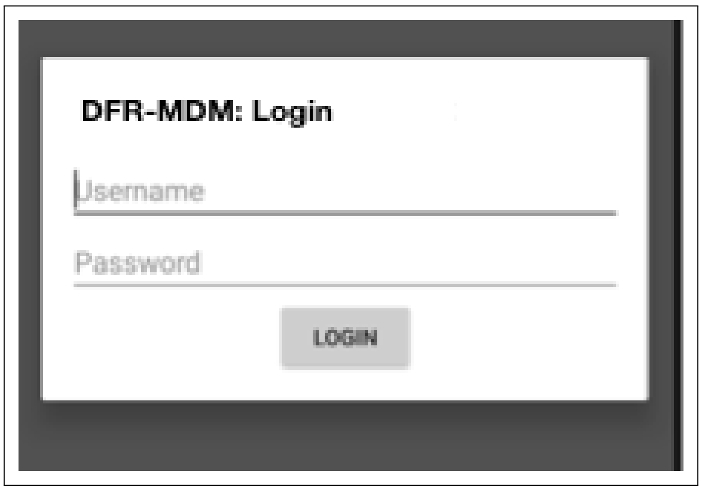
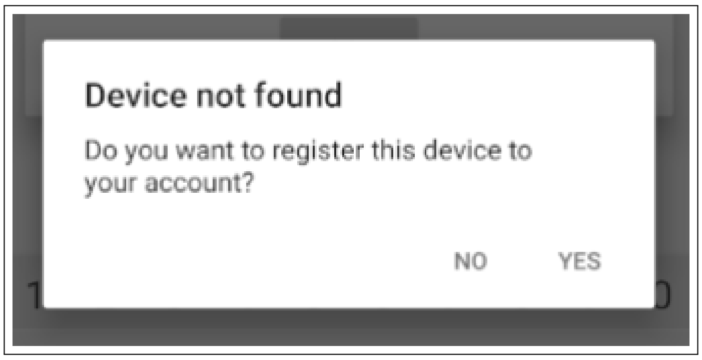


Figure 11.7: Mobile login

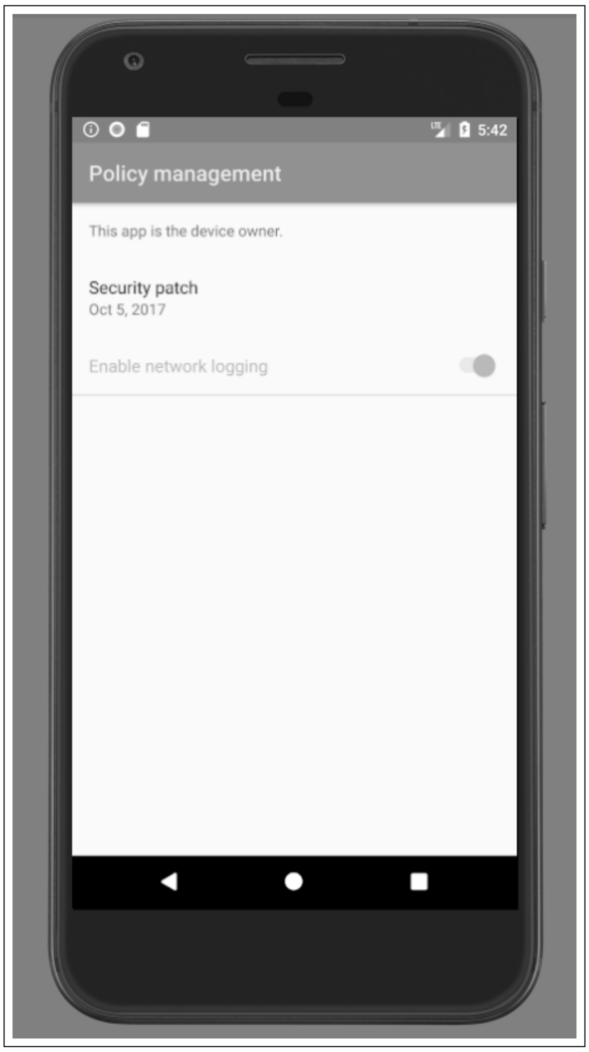
The application verifies his credentials and checks if the device is registered to him, as described in section 7.2. In this case, it is not as it is the first time he is logging in.


Device not found

Do you want to register this device to your account?

Figure 11.8: Mobile registration prompt

The application prompts Mr X to register the device to his account (see figure 11.8 for the prompt). When Mr X taps “Yes”, the application registers the device and redirects him to the landing page that can be seen in figure 11.9).



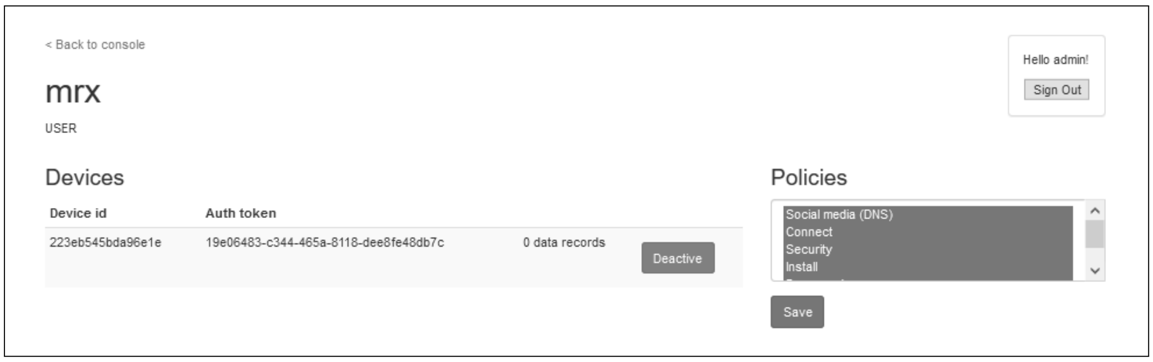
This app is the device owner.

Security patch
Oct 5, 2017

Enable network logging

Figure 11.9: Mobile landing page

Once the device redirects to the landing page, the administrator can verify that the device has been registered to Mr X by logging into the console and clicking on his username. This will bring up the screen shown in figure 11.10, where the user's details and all devices associated with their account is shown. The administrator can check that device is shown in the list of devices associated with Mr X. Since Mr X has only registered one device, only this device is displayed on the console.



< Back to console

mrx

USER

Hello admin!

Sign Out

Devices

Device id	Auth token		
223eb545bda96e1e	19e06483-c344-465a-8118-dee8fe48db7c	0 data records	Deactive

Policies

Social media (DNS)
Connect
Security
Install

Save

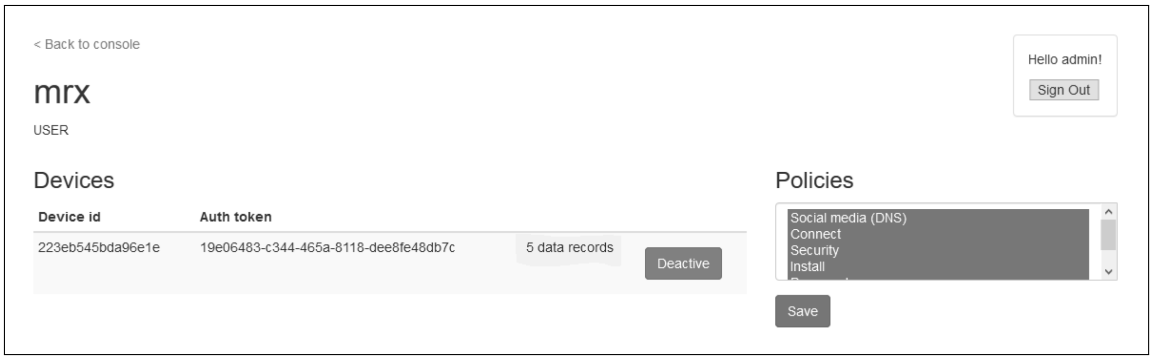
Figure 11.10: User has a mobile device associated with them

At this point, the user has been configured, the device has been linked to their username, and the application is running in the background. The application downloads the policies that are assigned to Mr X and the device starts monitoring.

As Mr X is making use of his mobile device, the device client is monitoring his actions in the background. If the device client detects activities defined by the policies associated with the user, the client collects the data and uploads it to the server.

After a few months, Mr X's boss start receiving alarming reports from Mr X's employees concerning the way that he does business. After conferring with HR, they ask the system administrator to **check the number of data entries** collected from Mr X's device.

While the system administrator cannot see the collected data, she can see the number of data entries that are associated with a user. This is done by clicking on the name of the user on the landing page to access the user's details. Next to the devices associated with the user, the administrator can view the number of data entries that have been collected from that device (figure 11.11).


< Back to console

mrx

USER

Devices

Device id	Auth token		
223eb545bda96e1e	19e06483-c344-465a-8118-dee8fe48db7c	5 data records	Deactive

Policies

Social media (DNS)
Connect
Security
Install

Figure 11.11: Data count is displayed next to the device

The system administrator logs into the prototype's administration console and pulls up Mr X's profile. She monitors the amount of data entries that have been collected from Mr X's device and sends the numbers through to HR. After some discussion with Mr X's manager and the forensic department, it is decided to assign an investigator and **open an investigation** into Mr X's activities.

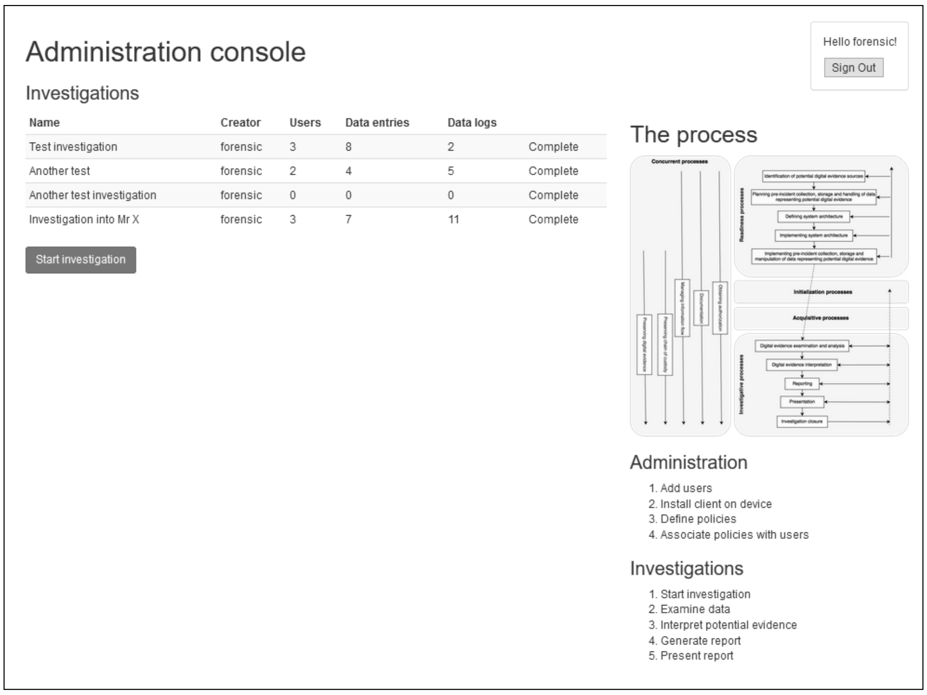
At this point, the digital forensic investigator takes over. The prototype allows the investigator to conduct their investigation by viewing data entries and compiling all of the entries into a report.

The digital forensic investigator accesses the prototype's investigation console through the same login screen as the system administrators, which can be seen in figure 11.1. When a user logs into the console, the prototype verifies the role that has been assigned (see section 8.4) and gives them access to the correct functionality. Once the digital forensic investigator is logged into the system, they land on the forensic landing page shown in figure 11.12.

This page displays a list of investigations that the digital forensic investigator has conducted and allows them to continue with previous investigations, start a new investigation or view the details and reports of investigations that have already been completed. This page also displays the digital forensic process and the functionality that is available to the different types of users.

To start a new investigation, the investigator clicks on the “Start investigation” button. On the next screen, that can be seen in figure 11.13, they enter a name for the investigation and clicks on the “Start investigation” button. This creates an entry for the investigation in the data store.

After starting the investigation, the investigator lands on the page shown in figure 11.14. This page shows an overview of the investigation and allows the investigator access to the data that the system has collected. This same page is displayed if an investigator continues with an investigation that has been previously started.


Administration console

Hello forensic!

Investigations

Name	Creator	Users	Data entries	Data logs	
Test investigation	forensic	3	8	2	Complete
Another test	forensic	2	4	5	Complete
Another test investigation	forensic	0	0	0	Complete
Investigation into Mr X	forensic	3	7	11	Complete

The process

Concurrent processes
Preserving digital evidence
Preserving chain of custody
Managing information flow
Documentation
Obtaining authorization
Readiness processes
Identification of potential digital evidence sources
Planning pre-incident collection, storage and handling of data representing potential digital evidence
Defining system architecture
Implementing system architecture
Implementing pre-incident collection, storage and manipulation of data representing potential digital evidence
Initialization processes
Acquisitive processes
Investigative processes
Digital evidence examination and analysis
Digital evidence interpretation
Reporting
Presentation
Investigation closure

Administration

1. Add users
2. Install client on device
3. Define policies
4. Associate policies with users

Investigations

1. Start investigation
2. Examine data
3. Interpret potential evidence
4. Generate report
5. Present report

Figure 11.12: Forensic landing page

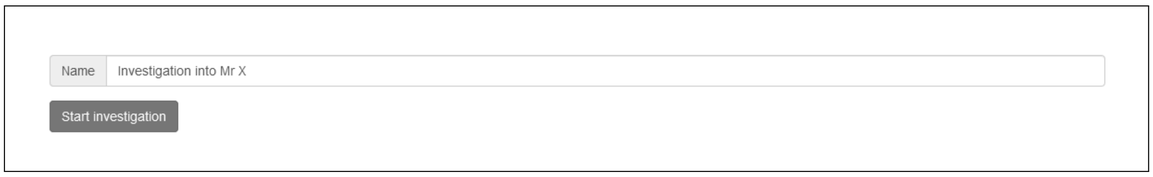
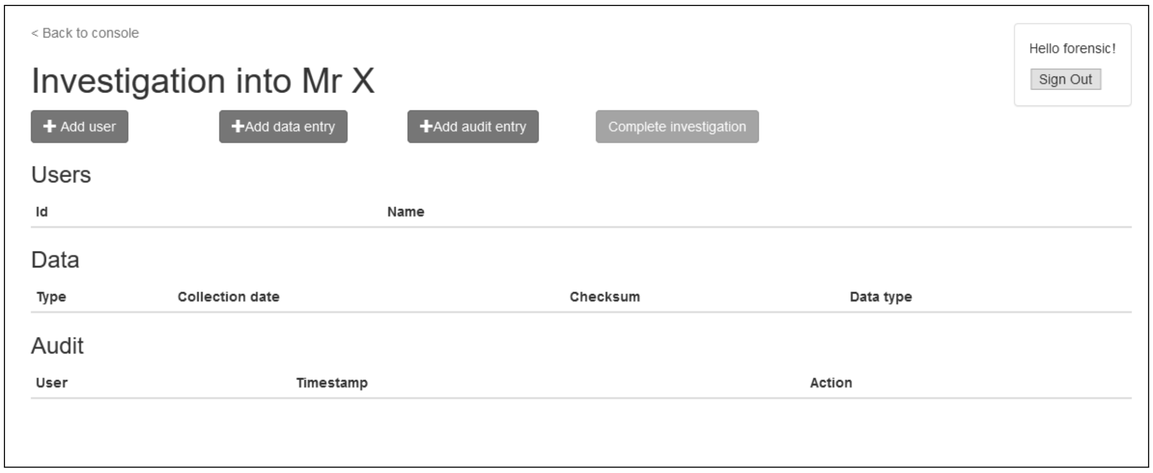

Name: Investigation into Mr X

Figure 11.13: Start a new investigation


< Back to console

Investigation into Mr X

Hello forensic!

Users

Id	Name

Data

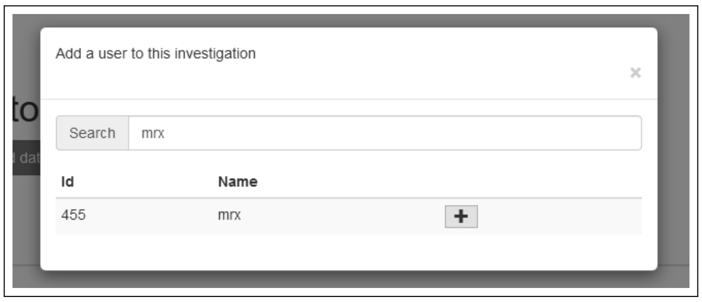
Type	Collection date	Checksum	Data type

Audit

User	Timestamp	Action

Figure 11.14: Investigation overview page

The first action that the digital forensic investigator takes, is to add Mr X to the investigation by clicking on the "Add user" button on the investigation page that can be seen in figure 11.14. The prototype prompts the investigator to search for a user. After searching, the investigator can click on the "Plus" button next to the user to add them to the investigation. The user addition screen can be seen in figure 11.15.



Id	Name
455	mrx

Figure 11.15: Add user to investigation

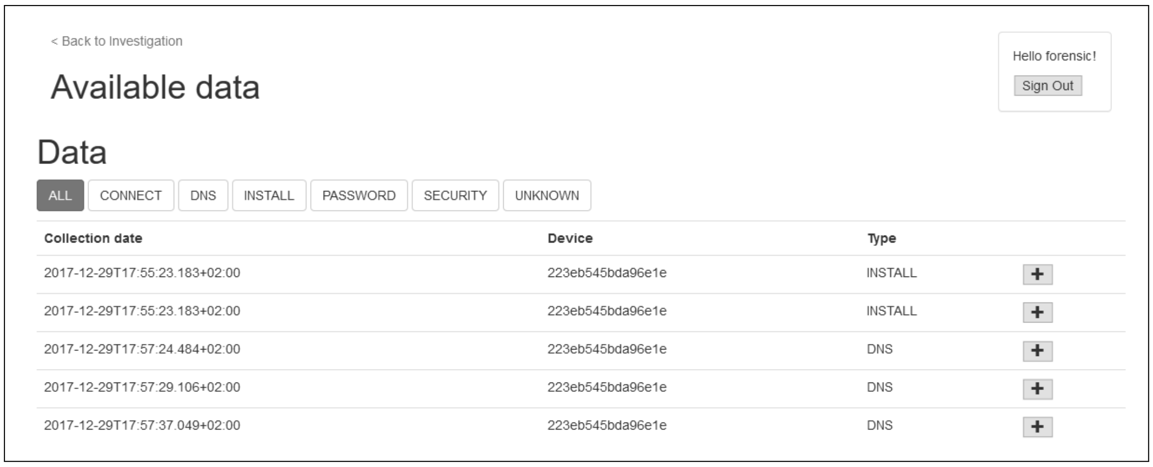
Once the investigation has been started, and a user added, the digital forensic investigator can proceed with their investigation. They do this by examining the data that has been collected from the devices of the users involved in the investigation.

Once the investigator has started the investigation, they **look at the data collected** from Mr X's device and **add any relevant entries** to the investigation that has been created in the prototype.

To look at and add data entries to an investigation, the digital forensic investigator clicks on the "Add data entry" button on the main investigation page that can be seen in figure 11.14. The prototype then shows the digital forensic investigator a page with all the data entries available. The user can filter the data entries by clicking on the filters at the top of the table. This screen can be seen in figure 11.16.

The investigator can click on any of the data entries to view the details of that entry, as shown in figure 11.17. This page shows the details of the data collected, namely the unique data identifier (data id), the type of the policy that triggered this collection, which device it was collected from and the checksum of the data, as well as the format in which the data was transmitted to the server.

Additionally, this page shows the full dataset that was sent from the device, in the format specified in the *data format* field. Finally, it shows a full log of the data: when it was collected, when the checksum and the data was uploaded, as well as showing when and how the data was verified using the checksums. The prototype verifies the checksums at key points in the process to show that the data has not been tampered with.

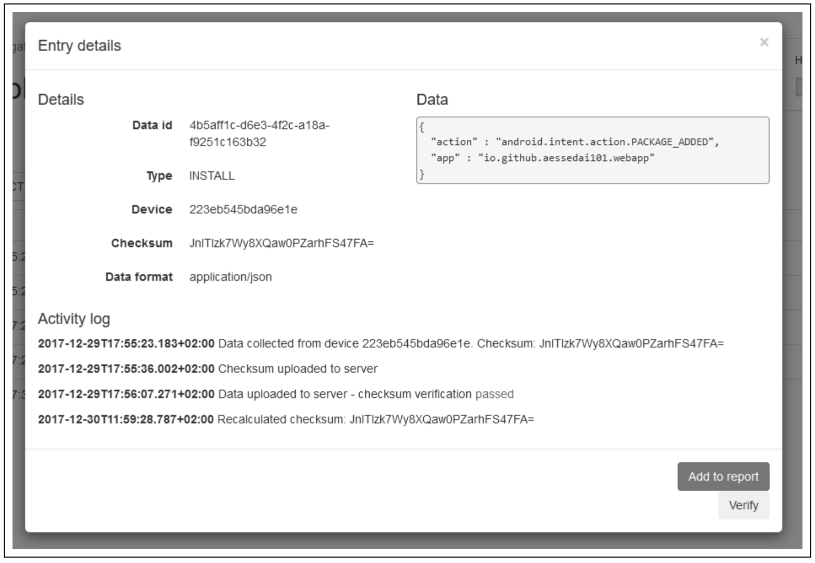


Available data

Data

Collection date	Device	Type	
2017-12-29T17:55:23.183+02:00	223eb545bda96e1e	INSTALL	+
2017-12-29T17:55:23.183+02:00	223eb545bda96e1e	INSTALL	+
2017-12-29T17:57:24.484+02:00	223eb545bda96e1e	DNS	+
2017-12-29T17:57:29.106+02:00	223eb545bda96e1e	DNS	+
2017-12-29T17:57:37.049+02:00	223eb545bda96e1e	DNS	+

Figure 11.16: Available data



Entry details

Details

Data id	4b5aff1c-d6e3-4f2c-a18a-f9251c163b32
Type	INSTALL
Device	223eb545bda96e1e
Checksum	JnlTlzk7Wy8XQaw0PZarhFS47FA=
Data format	application/json

Data

```
{
  "action" : "android.intent.action.PACKAGE_ADDED",
  "app" : "io.github.aessedai101.webapp"
}
```

Activity log

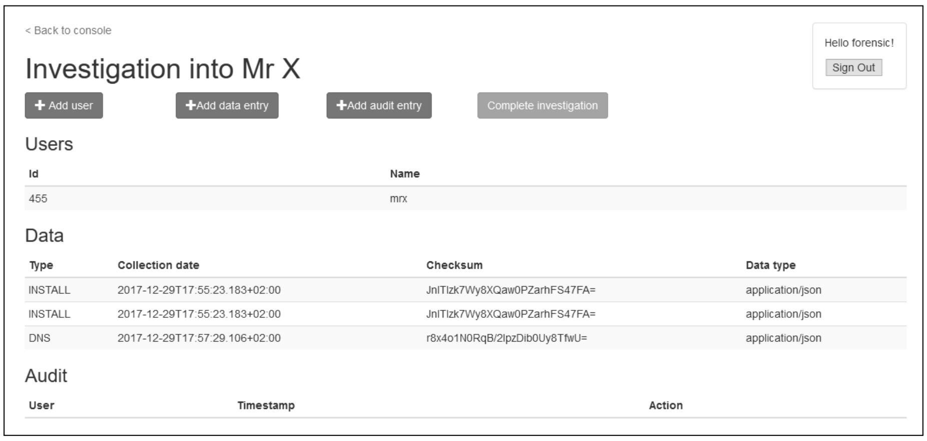
```
2017-12-29T17:55:23.183+02:00 Data collected from device 223eb545bda96e1e. Checksum: JnlTlzk7Wy8XQaw0PZarhFS47FA=
2017-12-29T17:55:36.002+02:00 Checksum uploaded to server
2017-12-29T17:56:07.271+02:00 Data uploaded to server - checksum verification passed
2017-12-30T11:59:28.787+02:00 Recalculated checksum: JnlTlzk7Wy8XQaw0PZarhFS47FA=
```

Figure 11.17: Data entry details

The investigator can tap on the “Verify” button to have the prototype verify that the checksums calculated throughout the process do indeed match. In addition to checking the checksums, the prototype also recalculates the checksum for added verification. This gives the investigator, as well as any disciplinary or judicial processes, confidence that the data has not been tampered with.

Once the investigator has examined a data entry and has determined that it is relevant to the investigation, they can either click the “Plus” button on the data overview page (shown in figure 11.16) or the “Add to report” button on the data details page (shown in figure 11.17) to add the data entry to the investigation.

When an investigator has added a data entry to the investigation, the prototype displays the data entry on the investigation overview page, along with the selected users and other data entries. Figure 11.18 shows an open investigation with three data entries added to it.



Investigation into Mr X

Users

Id	Name
455	mrx

Data

Type	Collection date	Checksum	Data type
INSTALL	2017-12-29T17:55:23.183+02:00	JnlTlzk7Wy8XQaw0PZarhFS47FA=	application/json
INSTALL	2017-12-29T17:55:23.183+02:00	JnlTlzk7Wy8XQaw0PZarhFS47FA=	application/json
DNS	2017-12-29T17:57:29.106+02:00	r8x4o1N0RqB/2lpzDib0Uy8TfwU=	application/json

Audit

User	Timestamp	Action

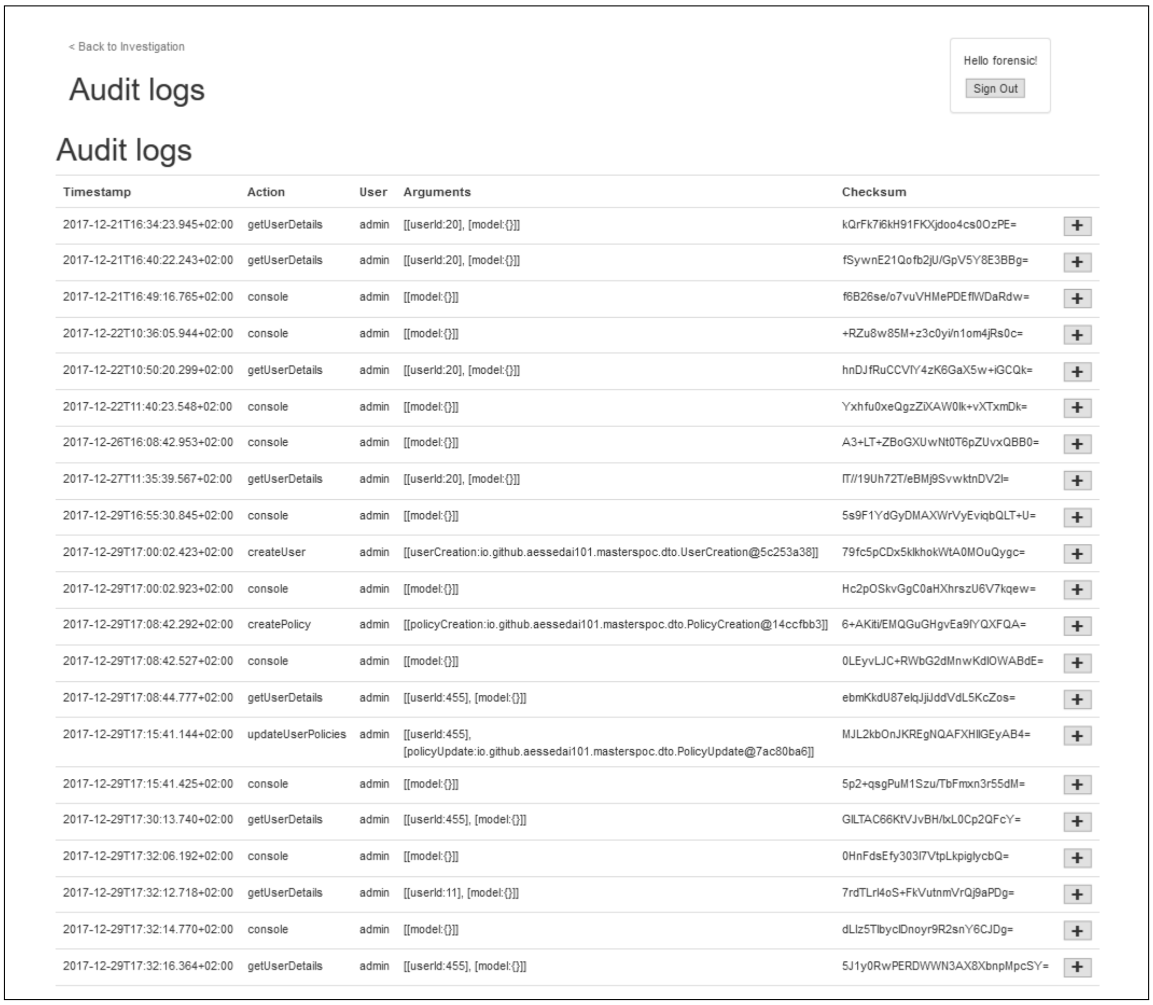
Figure 11.18: Investigation with data entries

While the investigation is in progress, HR discovers that a system administrator and another employee were working with Mr X. The accomplices are **added to the investigation** and the investigator examines their data records as well. Since one of the employees under investigation is a system administrator, the investigator can also **add audit logs** of the system administrator’s actions to the report.

To add another user to the investigation, the investigator follows the steps described previously, by tapping on “Add user” button, searching for the user and using the “plus” button to add the user to the investigation. This process is repeated for the second user, as explained before.

To add relevant audit details to the investigation, the investigator taps on the “Add audit entry” button on the main investigation page that can be seen in figure

11.18. This brings up the page that can be seen in figure 11.19, which displays a table of the actions that the users under investigation have performed on the administrator console, in the form of audit logs. The investigator looks through the audit logs and taps the "Plus" button to add any entries that they deem relevant to the investigation.



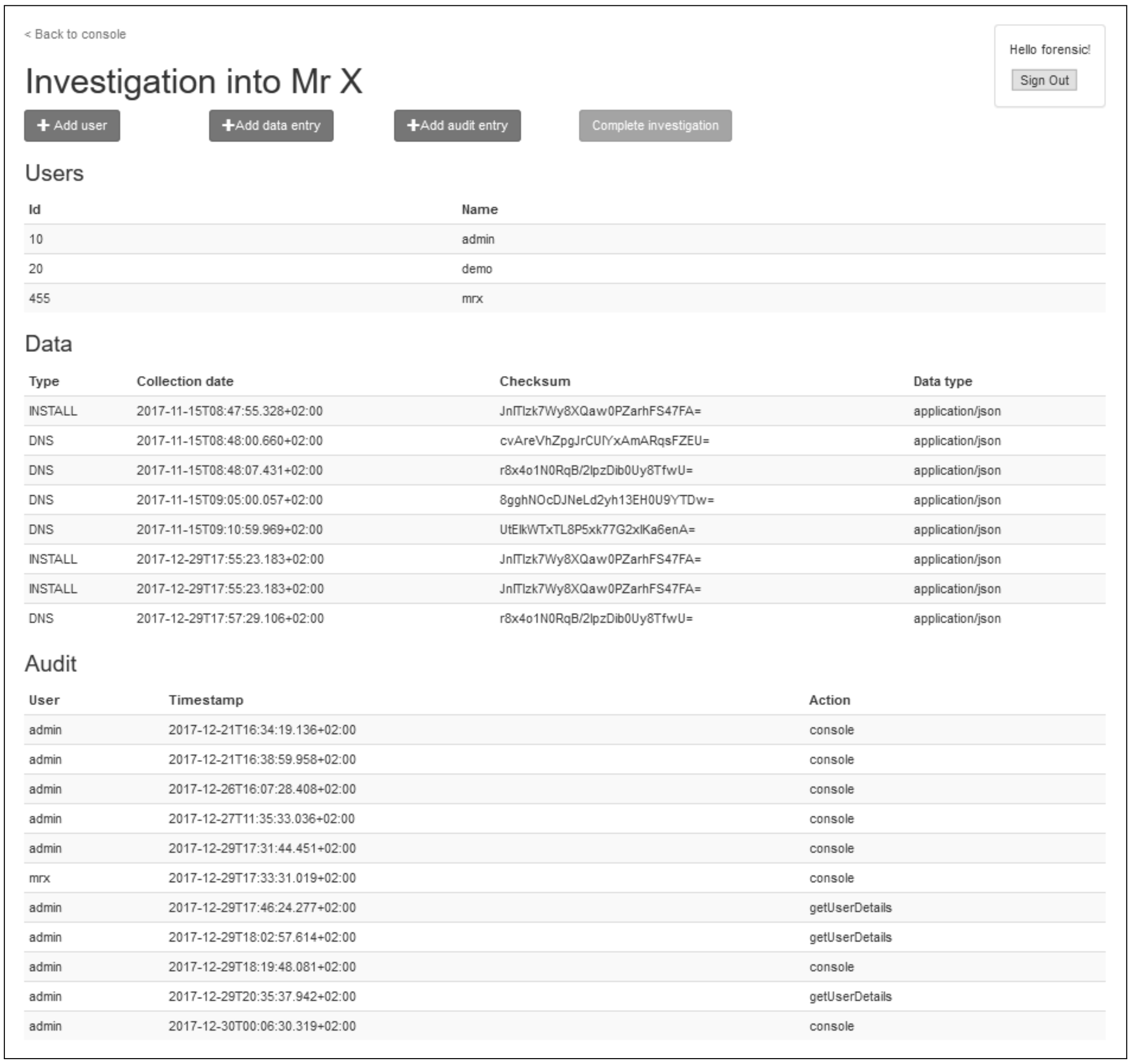
Audit logs

Audit logs

Timestamp	Action	User	Arguments	Checksum	
2017-12-21T16:34:23.945+02:00	getUserDetails	admin	[[userId:20], [model:{}]]	kQrFk7i6kH91FKXjdoo4cs0OzPE=	+
2017-12-21T16:40:22.243+02:00	getUserDetails	admin	[[userId:20], [model:{}]]	fSywnE21Qofb2jU/GpV5Y8E3BBg=	+
2017-12-21T16:49:16.765+02:00	console	admin	[[model:{}]]	f6B26se/o7vuVHMePDEfIWDaRdw=	+
2017-12-22T10:36:05.944+02:00	console	admin	[[model:{}]]	+RZu8w85M+z3c0yi/n1om4jRs0c=	+
2017-12-22T10:50:20.299+02:00	getUserDetails	admin	[[userId:20], [model:{}]]	hnDJfRuCCVlY4zK6GaX5w+iGCQk=	+
2017-12-22T11:40:23.548+02:00	console	admin	[[model:{}]]	Yxhfu0xeQgzZiXAW0lk+vXTxmDk=	+
2017-12-26T16:08:42.953+02:00	console	admin	[[model:{}]]	A3+LT+ZBoGXUwNt0T6pZUvxQBB0=	+
2017-12-27T11:35:39.567+02:00	getUserDetails	admin	[[userId:20], [model:{}]]	lT//19Uh72T/eBMj9SvwktnDV2I=	+
2017-12-29T16:55:30.845+02:00	console	admin	[[model:{}]]	5s9F1YdGyDMAXWrVyEviqbQLT+U=	+
2017-12-29T17:00:02.423+02:00	createUser	admin	[[userCreation:io.github.aessedai101.masterspoc.dto.UserCreation@5c253a38]]	79fc5pCDx5klkhokWtA0MOuQygc=	+
2017-12-29T17:00:02.923+02:00	console	admin	[[model:{}]]	Hc2pOSkvGgC0aHXhrszU6V7kqew=	+
2017-12-29T17:08:42.292+02:00	createPolicy	admin	[[policyCreation:io.github.aessedai101.masterspoc.dto.PolicyCreation@14ccfbb3]]	6+AKiti/EMQGuGHgvEa9lYQXFQA=	+
2017-12-29T17:08:42.527+02:00	console	admin	[[model:{}]]	0LEyvLJC+RWbG2dMnwKdlOWABdE=	+
2017-12-29T17:08:44.777+02:00	getUserDetails	admin	[[userId:455], [model:{}]]	ebmKkdU87elqJjiJddVdL5KcZos=	+
2017-12-29T17:15:41.144+02:00	updateUserPolicies	admin	[[userId:455], [policyUpdate:io.github.aessedai101.masterspoc.dto.PolicyUpdate@7ac80ba6]]	MJL2kbOnJKREgNQAFXHIlGEyAB4=	+
2017-12-29T17:15:41.425+02:00	console	admin	[[model:{}]]	5p2+qsgPuM1Szu/TbFmxn3r55dM=	+
2017-12-29T17:30:13.740+02:00	getUserDetails	admin	[[userId:455], [model:{}]]	GILTAC66KtVJvBH/lxL0Cp2QFcY=	+
2017-12-29T17:32:06.192+02:00	console	admin	[[model:{}]]	0HnFdsEfy303I7VtpLkpiglycbQ=	+
2017-12-29T17:32:12.718+02:00	getUserDetails	admin	[[userId:11], [model:{}]]	7rdTLrl4oS+FkVutnmVrQj9aPDg=	+
2017-12-29T17:32:14.770+02:00	console	admin	[[model:{}]]	dLlz5TlbyclDnoyr9R2snY6CJDg=	+
2017-12-29T17:32:16.364+02:00	getUserDetails	admin	[[userId:455], [model:{}]]	5J1y0RwPERDWWN3AX8XbnpMpcSY=	+

Figure 11.19: Adding audit entries

The investigator repeats the process of adding users, selecting relevant data entries and adding audit entries until they are satisfied with the evidence that they have collected. At this point, the investigation overview looks similar to the investigation depicted in figure 11.20 with the investigation containing a number of users, a set of data entries and audit records, if an administrator or investigator is under investigation.



Investigation into Mr X

Users

Id	Name
10	admin
20	demo
455	mrx

Data

Type	Collection date	Checksum	Data type
INSTALL	2017-11-15T08:47:55.328+02:00	JnlTlzk7Wy8XQaw0PZarhFS47FA=	application/json
DNS	2017-11-15T08:48:00.660+02:00	cvAreVhZpgJrCUlYxAmARqsFZEU=	application/json
DNS	2017-11-15T08:48:07.431+02:00	r8x4o1N0RqB/2lpzDib0Uy8TfwU=	application/json
DNS	2017-11-15T09:05:00.057+02:00	8gghNOcDJNeLd2yh13EH0U9YTDw=	application/json
DNS	2017-11-15T09:10:59.969+02:00	UtElkWTxTL8P5xk77G2xlKa6enA=	application/json
INSTALL	2017-12-29T17:55:23.183+02:00	JnlTlzk7Wy8XQaw0PZarhFS47FA=	application/json
INSTALL	2017-12-29T17:55:23.183+02:00	JnlTlzk7Wy8XQaw0PZarhFS47FA=	application/json
DNS	2017-12-29T17:57:29.106+02:00	r8x4o1N0RqB/2lpzDib0Uy8TfwU=	application/json

Audit

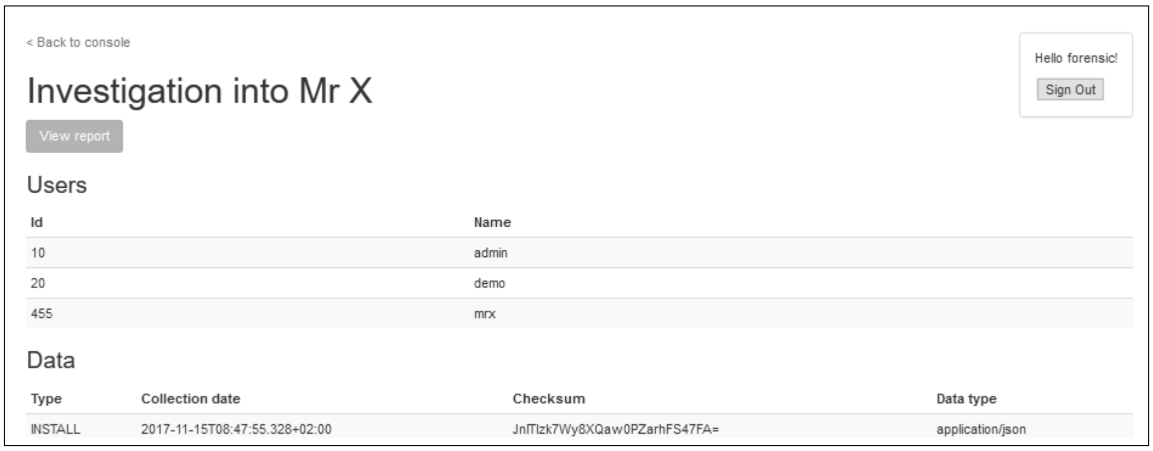
User	Timestamp	Action
admin	2017-12-21T16:34:19.136+02:00	console
admin	2017-12-21T16:38:59.958+02:00	console
admin	2017-12-26T16:07:28.408+02:00	console
admin	2017-12-27T11:35:33.036+02:00	console
admin	2017-12-29T17:31:44.451+02:00	console
mrx	2017-12-29T17:33:31.019+02:00	console
admin	2017-12-29T17:46:24.277+02:00	getUserDetails
admin	2017-12-29T18:02:57.614+02:00	getUserDetails
admin	2017-12-29T18:19:48.081+02:00	console
admin	2017-12-29T20:35:37.942+02:00	getUserDetails
admin	2017-12-30T00:06:30.319+02:00	console

Figure 11.20: Investigation covering multiple users, data entries and audit records

Once the investigator is satisfied with the investigation and all relevant data entries have been added, they can **complete the investigation** on the system and **generate a report**. This report contains all the data entries that the investigator has highlighted, as well as a report outline. At this point, the digital forensic investigation may not be complete yet - just the part of the investigation that happens on the prototype.

The digital forensic investigator is satisfied that the part of their investigation that happens on the DFR-MDM system is complete and that all relevant data about Mr X and his accomplices have been added to the investigation created on the prototype. To mark the investigation on the prototype as complete, the investigator clicks on the "Complete investigation" button on the investigation overview page that can be seen in figure 11.20.

Once the investigation is marked as complete, the investigator can no longer change anything within the investigation, such as adding a user, adding a data entry or adding audit records. The actions that would allow the investigator to do that are replaced with one new action: a "View report" button. The completed investigation page can be seen in figure 11.21.



Investigation into Mr X

Users

Id	Name
10	admin
20	demo
455	mrx

Data

Type	Collection date	Checksum	Data type
INSTALL	2017-11-15T08:47:55.328+02:00	JnlTlzk7Wy8XQaw0PZarhFS47FA=	application/json

Figure 11.21: Closed investigation

Once an investigator has completed an investigation, their job is not complete. They need to report the findings of their investigation to the stakeholders, who may be law enforcement, management or another party. To expedite this process, the prototype generates a report structure and includes the data that the investigator flagged.

The generated report outline contains five sections, with the first four sections containing a prompt for the report writer:

- **Introduction** An introduction to give the reader an overview of the purpose of the report.

- **Investigation** A section that the investigator can use to describe the investigation they performed.
- **Findings** The investigator describes the findings that have emerged from the investigation described in the previous section
- **Conclusion** A section for the investigator to conclude their report.
- **Appendix** An appendix containing all the data that the investigator has added to the investigation, grouped by user.

A complete example of such a generated report can be seen in appendix A.

Once the prototype has generated the report outline, the investigator writes the report, following the prompts. This report is then presented to the stakeholders.

This section described the functionality of the prototype, using a fictional scenario to demonstrate the prototype to the reader. The author described every part of the process that the prototype is involved in, including onboarding new users, setting up data collection policies, checking for suspicious behaviour, investigating an incident and reporting on the incident. The next section concludes this chapter.

11.3 Conclusion

This chapter presented the prototype that was built from the model and architecture outlined in the previous section. The author "demonstrated" the prototype through the use of a fictional scenario, showing where the prototype interacts with the various roles present in an organisation.

In the next section of this research, the author critically evaluates the model, architecture and prototype that has been presented in this research, using a variety of criteria.

PART V

Evaluation and summary

The final part of this research performs a critical evaluation of the research presented, revisiting requirements and analysing the benefits and shortcomings. The author identifies related work and research contributions and concludes the research.

CHAPTER 12

Evaluation according to requirements

12.1 Introduction

In chapter 10 the author defined some requirements that a solution needs to meet in order to solve the problem that has been stated. This chapter analyses how the proposed solution compares to the requirements stated in chapter 10, applying the design science evaluation process. This section is divided up in the same way as chapter 10 - first the business and market requirements, then the functional, non-functional and UI requirements respectively.

12.2 Business and market requirements

The two business requirements stated in section 10.2 is to **reduce the costs of digital forensic investigations** and to **have more complete data** for investigations. The solution succeeds in both these goals, as it is no longer necessary to manually retrieve evidence from users' devices, reducing costs. Additionally, the data collection policies can be configured to collect all the data that is required.

12.3 Functional requirements

This section evaluates the solution based on the functional requirements that are presented in section 10.3 and summarised in table 10.1, following the same structure as section 10.3.

12.3.1 Mobile functional requirements

The mobile functional requirements defines the functionality of the mobile components of the solution.

The mobile device should be associated with a user. The solution registers a device to a user, linking the device to the user. The solution does not allow for the link between a user and a device to be destroyed.

The mobile device should collect data about the user's actions. The mobile client runs in the background monitoring the user's actions. If it detects suspicious activity, it collects and uploads the data.

The administrators should control the device's data collection actions. The solution allows system administrators to define the policies that control the data collection. System administrators can also select which policies to link to a user.

The device should upload the collected data to the investigators in a forensically sound manner. The solution makes use of authentication token, signatures using keypairs and checksums to verify both the source and the integrity of the data as it is being sent between components.

The solution meets all the functional requirements for the mobile application. Next, the author analyses the functional requirements for the administration console.

12.3.2 Administration functional requirements

The administrative functional requirements define the functionality of the administrative sections of the solution.

Administrators should be able to create data collection policies. The solution allows the administrators to create new policies.

Administrators should be able to create new users on the system. The solution allows the administrators to create new users and assign roles to the new users.

Administrators should be able to change the policies assigned to a user. The solution allows the administrators to associate policies with a user, as well as changing the policies that have been assigned to the user.

Administrators should be able to view devices associated with a user. When an administrator accesses the user details page, the solution displays a list of devices that have been linked to the user.

Administrators should be able to mark devices as inactive. On the user details page, the solution allows an administrator to deactivate a device that is linked to the user. This deactivation will not destroy the link between the user and the device, but the device will stop collecting data after the authentication token has expired.

The solution meets all the requirements from an administrative perspective. Next, the investigation functional requirements are evaluated.

12.3.3 Investigation functional requirements

The investigative functional requirements dictate the functionality of the investigation sections of the solution.

The digital forensic investigator should be able to start an investigation. The solution allows an investigator to start an investigation, which creates an entry in the database for the investigation.

The digital forensic investigator should be able to add users to an investigation. Once an investigation has been opened, the solution allows the investigator to search for and add users to the investigation.

The digital forensic investigator should be able to view the data related to the users under investigation. Once a user has been added to the investigation, the investigator can view all data entries and audit logs related to the user. The investigator can also add the data and audit entries to the active investigation.

The digital forensic investigator should be able to mark an investigation as complete. The solution allows the investigator to mark a solution as complete. After being marked as complete, the investigation can no longer be modified.

The digital forensic investigator should be able to generate a report from a completed investigation. Once the investigation has been completed, the investigator can view a report of the investigation.

Mobile		**Administration**		**Investigation**	
Associate with user	✓	Create collection policies	✓	Start investigation	✓
Collect data	✓	Create new users	✓	Add users	✓
Controlled by administrators	✓	Change assigned policies	✓	View data	✓
Forensically sound	✓	View devices	✓	Complete investigation	✓
		Deactivate devices	✓	Generate report	✓

Table 12.1: Functional requirements [1]

Table 12.1 lists the functional requirements as summarised in table 10.1 and shows which of the requirements have been met. Next, the non-functional requirements is examined.

[1]Checkmarks indicate that the requirement has been met.

12.4 Non-functional requirements

The non-functional requirements are defined in section 10.4 and is divided into performance and operating requirements. This section follows the same structure.

12.4.1 Performance requirements

Performance requirements place bounds on the performance of the solution.

Mobile data collection should run in the background. After prompting the user for their credentials, the mobile client runs purely in the background. The only exception to this is if the mobile client requires the user's credentials again.

Mobile data collection should not consume excessive resources. The mobile client runs jobs at scheduled times. The prototype makes use of the framework's `JobScheduler` which defers and batches jobs to preserve the device's resources [9].

The server should always be available. Deploying the server application to multiple physical servers will increase availability. Additionally, the server was implemented in such a way that it is not necessary to take it offline for maintenance.

The server should complete operations in a reasonable time. The server does not execute any unnecessary operations and terminates unsuccessful operations as soon as possible. This reduces the load on the server and ensures that calls to the server return as quickly as possible.

The server should ensure the identity of the mobile device. The server makes use of the user's authentication token to verify that the device belongs to the user. Additionally, all the requests from the mobile client are signed using a private key, which the server verifies using the corresponding public key.

The administration console should be available during business hours. Deploying the administration console to multiple physical servers will increase availability. Additionally, maintenance can be performed without shutting it down.

The administration console should be access controlled. In order to access any functionality on the console, the administrators and digital forensic investigators are required to log in.

Access levels should be verified before a user is allowed access to any sensitive functionality. The solution checks the access levels of the logged in user before allowing them to access the functionality on the console. For example, an administrator will not be allowed to access investigative functionality, and an investigator will not have access to administrative functions.

The solution meets all the performance requirements, with the caveat that some requirements will only be fully met if sufficient hardware is available. Next, the operating requirements are evaluated.

12.4.2 Operating requirements

The operating requirements place bounds on the operation of the solution.

Mobile data collection should not interrupt the user. The mobile client runs in the background entirely, except when the user is required to enter their credentials. The user is only prompted for their credentials when the authentication token expires.

The server should be accessible from anywhere. The server is designed and implemented to be accessible from the internet. If the server is deployed inside a corporate network, the administrators need to verify that it can be accessed from outside the network.

The administration console should be accessible inside the organisation's network. Once the solution has been deployed in an organisation's network, the administrators will be able to access the console.

Performance		**Operating**	
Run in background	✓	Don't interrupt user	✓
Resource efficient	✓	Server accessible from anywhere	✓*
Server availability	✓*	Console available in network	✓*
Server performance	✓*		
Mobile identity	✓		
Console availability	✓*		
Access control	✓		
Verify access levels	✓		

Table 12.2: Non-functional requirements [2]

Table 12.2 reproduces the non-functional requirements table 10.1 and shows which of the requirements have been met. Any requirements that depends on external dependencies to be fully met is marked with an asterisk. The next section evaluates the UI requirements.

[2]Checkmarks indicate that the requirement has been met and asterisks indicate that some external input is required for the requirement to be met in full.

12.5 UI requirements

The UI requirements for this solution is defined in section 10.5. UI requirements define the behaviour of the solution's user interface. This section evaluates how the solution meets the UI requirements.

The mobile client should have a login screen. The mobile client has a login screen that is displayed when the mobile client requires the user's credentials.

The administration console should have a login screen. The administration console shows a login screen to the administrators and digital forensic investigators before they are allowed access to the functionality.

The console should have a screen for viewing and editing policies and users. The solution allows the administrators to add and edit users and policies. Additionally, it allows the administrators to associate and remove policies from users.

The console should have a screen for managing investigations. The solution allows the digital forensic investigator to start, close and conduct investigations.

UI	
Client login screen	✓
Console screen	✓
View and edit users	✓
View and edit policies	✓
Manage investigations	✓

Table 12.3: UI requirements [3]

Table 12.3 reproduces table 10.3, which shows the UI requirements. The checkmarks indicate which of the requirements have been met. The next section analyses how well the solution defends against common threats to MDM systems.

12.6 Conclusion

This chapter evaluated the proposed solution according to the requirements that has been defined in chapter 10. In the next chapter, the author evaluates the solution critically based on additional criteria.

[3] Checkmarks indicate that the requirement has been met.

CHAPTER 13

Overall Critical Evaluation

13.1 Introduction

In this research, the author presented a model, architecture and prototype created to solve the problem as stated, namely that MDM systems are purely preventative and that it is expensive and complicated to retrieve evidence from personal devices after an incident occurs.

In this chapter, the work that was done is critically evaluated. Specifically, the author evaluates how well the solution defends against the threats outlined in chapter 4.5. After that, the benefits and shortcomings of the solution are analysed and finally, the author addresses the privacy concerns that arise from a solution designed to collect user data.

13.2 Defence against threats

Chapter 4.5 outlined some common threats against MDM systems, which attack the confidentiality, integrity and availability of the system. This section briefly defines each threat and evaluates how well the presented solution holds up against these threats.

Spoofing. A spoofing attack is when a malicious third party attempts to masquerade as a legitimate user by replaying interactions [86]. The solution makes use of unique identifiers to distinguish data entries, making spoofing attacks ineffective, since the system will reject any duplicate entries.

Tampering. Tampering is when a malicious third party modifies the interactions between components as they are happening [57]. The solution protects against these types of attacks by signing all requests going to the server. Since any interceptors do not possess the private keys, the signatures will not match and the request will be rejected.

Repudiation. Repudiation is when a malicious entity attempt to hide their actions on a system by manipulating the data stored by the system [101]. The solution logs all activity that happens on the system for audit purposes. The users cannot change these audit logs, and the data is spread over two storage mechanism, namely a relational database and a document store. To fully mitigate this threat the locations where the data is stored has to be access controlled.

Information disclosure. Information disclosure happens when an unauthorised third party gains access to sensitive information [102]. To mitigate this threat, the relational database and document store need to be access appropriately controlled. This is out of scope for this research, since it is in the purview of database security specialists.

Denial of service. A denial of service attack happens when a malicious entity attempts to prevent the system from functioning as intended by flooding the system with data or commands [62]. The server is stateless and as such more instances of the server can be added to the server pool to handle a more significant load. However, to entirely mitigate this threat the organisation will have to consult with an expert in denial-of-service mitigation, as the server is not the only vulnerable point.

Elevation of privilege. An elevation of privilege attack is when an attacker manipulates the system in such a way that they gain access to functionality and data that they are not allowed to access [37]. The solution checks the access levels of the user before allowing them to use any functionality, which mitigates this threat.

Malware. It is possible for a user to either intentionally or unintentionally install malicious software on their device, which may compromise the system. The server and administrator console run on the organisation's hardware, making it less susceptible to malware. The organisation can use the MDM part of the system to block known malicious applications, but new malware is released daily [1], which means that there is not a 100% guarantee that there won't be malware on the device.

Spoofing	✓
Tampering	✓
Repudiation	✓*
Information disclosure	*
Denial of service	✓*
Elevation of privilege	✓
Malware	✓̸
Users	✓̸*

Table 13.1: Common threats against MDM systems. [1]

Users. Users may compromise a system by sharing credentials or by attempting to bypass the system. To prevent users from sharing credentials, the employee

[1]Checkmarks indicate that the requirement has been met, a crossed checkmark that the requirement has been partially met and asterisks indicate that external input is needed for the requirement to be met in full.

is required to log into the device for the first time in the presence of the system administrator. After the device is registered to the user, another user's credentials will not work on the device. Since the mobile client runs in the background, the user will also not be able to bypass the system. The solution has no mechanism to prevent administrators and investigators from sharing credentials. The administrators and investigators are a relatively small number of users, they should be trained in how to use the system securely.

In summary, the solution defends against most of the common threats against MDM systems, as shown in table 13.1, with checkmarks indicating the threats that are mitigated. Threats marked with a star requires some outside dependencies to mitigate fully. Next, the author analyses the benefits and shortcomings of the solution.

13.3 Benefits and shortcomings

The solution presented by the author to add digital forensic readiness to an MDM solution has a number of benefits and shortcomings. This section analyses those benefits and shortcomings, starting with the benefits.

13.3.1 Benefits

There are a number of benefits to using this solution, namely

- Reduced costs,
- faster investigations,
- more complete data and
- enabling the standard investigative process.

The prototype **reduces the costs of a digital forensic investigation** by automating the evidence collection process. Investigators don't have to extract the data from the subject's device manually, eliminating the need for expensive mobile device forensic software and tools.

Additionally, this **speeds up the investigation**, as the investigators no longer have to go through the process of obtaining physical access to the device under investigation, which may include obtaining warrants and retrieving the device from its owner.

The solution also collects **more complete data**, since it collects the data when it detects the action, utilising the concept of digital forensic readiness, allowing it to gather transient data which may no longer be on the device when an investigation is started.

Finally, the solution **enables the investigative process** by allowing the investigator to examine and collate the data that has been gathered according to the international standard ISO 27043. The investigator is also able to generate a report to present to the stakeholders of the investigation.

As mentioned, there are a number of benefits to the solution as presented. However, there are also some shortcomings, which is explored in the next section.

13.3.2 Shortcomings

While the solution reduces costs and speeds up investigations, it also has some shortcomings, namely

- Requires cooperation from employees,
- resource consumption,
- framework limitations,
- lack of granularity in configuration and
- excessive or irrelevant data.

First, the solution requires **the employee to install** the application on their phone. If an employee refuses to do so, the investigators will have to obtain a warrant and analyse the device manually should an incident occur [2].

Another shortcoming is the **resources consumed by the mobile application**. While the application was designed to consume minimal resources, it still needs to run in the background and communicate with the server. This may negatively impact the battery life and data consumption of employees' devices, especially if they make use of low-end devices with insufficient resources.

The mobile application is also **limited by the framework and operating system**. Since the operating system on the mobile device controls access to all resources and data, as well as sandboxing applications, the application can only detect and collect data on actions if the mobile operating system allows it to do so. Additionally, different operating systems and different versions of the same operating system have different rules about what an application is allowed to do.

On the administrative side, the **configuration options are not very granular**. The policy configuration does not allow for wildcards, and the policies have to be applied manually to every user that is created in the system, which may lead to the administrators being overburdened in a larger organisation.

[2]It may be possible for the employer to force the employee to install the application, but it is a legal grey area - especially if the device belongs to the employee.

Finally, the solution may gather **excessive or irrelevant data**. Especially in cases where a service may be used for both legitimate and unauthorised activities, the prototype will gather the data for all the activity on the service. This complicates the job of the investigator, as they have to distinguish between legitimate and unauthorised activities manually. It may also place limitations concerning time on investigations.

In addition to the shortcomings covered above, there are also some privacy concerns related to the amount and category of data that is being collected. These privacy concerns are presented separately in the next section since it is a big drawback of such a solution.

13.4 Privacy concerns

There are some privacy concerns around the usage of such a digital forensic readiness system on mobile devices. The solution gathers a large set of data about employees' activities, some of which may be private activities that has no relevance to the organisation.

Specifically, many countries have laws that govern how individuals' data may be collected and stored by organisations. For example, the Protection of Personal Information (POPI) act in South Africa and the General Data Protection Regulation (GDPR) in the European Union have strict guides as to how data may be collected, stored and destroyed.

Before any data can be collected, the employee needs to consent to their data being collected, processed and stored by their employer. This can be achieved by requiring the employee to sign an agreement to the effect as a condition of employment.

In some jurisdictions, the employee must be given access to the data that has been collected about them and information about the processing of the data, should they request it. In such instances, the employee should be allowed to view the data without the data being made available to third parties.

Some data regulations also specify the "right to be forgotten," which allows individuals to request the erasure of personal data that has been collected about them. In such a case, the organisation needs to consult with their legal department about complying without destroying potential evidence.

As with all monitoring software, this solution has some privacy concerns. These can be mitigated by having written agreements, policies and legal counsel that is familiar with regulation.

During this research some literature was discovered that related to the research presented. This related work is presented in the next section.

13.5 Related work

This section presents literature that relates to the presented research.

Farjamfar et al. [29] reviews some digital forensic process models as relating to mobile devices, presenting some of the challenges around mobile device forensics and covering the various states that a mobile device may be in. The paper then presents and compares a number of digital forensic process models that have been proposed in other literature.

Ahmed and Dharaskar [8] explores the state of the art of mobile forensics. Even though this paper was published in 2008, it still provides value, discussing the difference between "standard" digital forensics and mobile forensics, how data retrieved from mobile phones may be used as evidence and some future trends in the mobile device space.

Grover [36] presents a model for automated data collection from an Android mobile device to facilitate a digital forensic investigation. This paper focuses on collecting data that is not collected by other tools and outlines anti-forensic measures that may be deployed against such a model. The data collection aspects of the paper align closely with this research.

There are also some other papers exploring how mobile applications collect data from mobile devices. For example, Buchka and Firsh [15] analysed a malware title *Skygofree*, analysing how it collects data from its victims. Similar techniques are used to collect data to be utilised for digital forensic readiness purposes, also in this solution.

This section presented literature related to research that has been completed. The next section outlines the main contributions that this research made to the state of the art.

13.6 Research contribution

The problem this research attempted to address is the fact that MDM systems are purely preventative and the author proposed a solution, adding digital forensic readiness to the MDM system. The author proposed a model for such a system and showed that it is practical by implementing a prototype.

13.7 Conclusion

This chapter evaluated the solution that was presented in various ways. First by evaluating the solution's adherence to the specified requirements, second by exploring how it enables the digital forensic process, third by presenting the benefits and shortcomings of the solution and finally by presenting privacy concerns.

The next chapter concludes this research by re-examining the original problem statement, summarising the work done and outlining potential future work.

CHAPTER 14

Conclusion

14.1 Introduction

This research presented a model for the addition of digital forensic readiness to an existing Mobile Device Management solution, as well as an evaluation of the proposed solution. This chapter concludes the research.

First, the author summarises the research that has been completed. After that, the author restates and presents the problem statement as stated initially in chapter 1. Finally, some possible future work is outlined, and the research is concluded with some final remarks.

14.2 Summary

In chapter 1 the author introduced a problem, namely that MDM systems are preventative, which means that if an incident occurs, the investigators cannot retrieve information from the device easily or cheaply. In some cases, it may not be possible at all.

To solve this problem, the author proposed to add a digital forensic readiness component to the MDM system, allowing for evidence collection remotely while an incident is occurring. This component reduces the costs and duration of investigations since evidence has already been collected.

Chapter 2 introduced the concept of design science and presented how design science would be applied to this research.

In chapters 3 to 5 the author presented the background to their research.

In chapter3 the author defined the concept of "Bring Your Own Device (BYOD)", and outlined the advantages and disadvantages of such a policy.

Chapter4 covered Mobile Device Management (MDM) systems, including defining the concepts and presenting the architecture and deployment of such a system, as well as enumerating the threats that MDM systems are vulnerable to.

The final chapter in the background section, chapter 5 introduced digital forensics and digital forensic readiness. The author defined both of these terms, presented the processes as defined in ISO 27043 and analysed the benefits and drawbacks of having digital forensic readiness in place.

Chapters 6 and 7 presented a high-level model and design science methods to solve the problem, presenting the various components of the system and how data would flow through the system. These chapters also outlined how the model ensures data integrity over an untrusted network.

Chapter 8 expanded the model and methods, showing what the architecture of such a solution might look like. This chapter presented the architecture of each component in more detail, including how the component is subdivided and showing program flows through each component. The author also presented how the storage of data is structured.

Chapter 9 evaluated the proposed model and architecture according to ISO 27043, the standard for Incident investigation principles and processes. This chapter examined how the solution interacted with and enables the standard.

Chapter 10 defined the requirements that a successful solution to the problem will adhere to, looking at various categories to define these requirements.

In chapter 11 the author described the prototype that was implemented, presenting some implementation details and going through the prototype step-by-step using a fictional scenario to demonstrate the functionality to the reader.

Chapter 12 critically evaluated the solution at the hand of the requirements defined in chapter 10.

Finally, the author critically evaluated the model, architecture and prototype in chapter 13 by means of the threats presented in chapter 4.5. The author analysed the benefits and shortcomings of the solution, addressed the privacy concerns, highlighted some related work and presented the research contributions.

14.3 Revisiting the Problem statement

The problem statement, as presented in section 1.2 is that "most MDM systems do not include a digital forensic readiness component, leaving investigators with little to no device-related historical data when an incident does occur."

The rising popularity of smartphones, the adoption of BYOD policies and the purely preventative nature of existing MDM solutions combined to form a situation where almost no mobile forensic evidence could be obtained without having physical accesses to the device - a costly procedure.

The research presented a solution to this problem, namely adding a digital forensic readiness aspect to an MDM solution. The author described a model for such a solution, refined it in an architecture and finally presented a prototype that implements this solution.

The proposed solution solves the problem as stated in chapter 1 by detecting suspicious activity as defined by the system administrators and transmitting the collected data to the server in a forensically sound manner. Finally, the solution enables the investigation of incidents by allowing investigators to examine the collected data.

14.4 Future work

During the research, several other opportunities for research came up, which were out of scope for this research. This section presents opportunities for future work.

14.4.1 Policy definitions

The model that has been presented in this research does not present the definition of policies in great detail, and the prototype utilises a simple implementation of the policy management.

There is room for defining exactly how policies in this model are defined, going into more detail on which values are required and stored. An additional factor that can be addressed is wildcard values, for example, monitoring for all subdomains at a specific URL.

Finally, the policy may be extended to define which (seemingly unrelated) pieces of information are gathered when a policy is triggered. For example, the user's location may be collected when a specific website is accessed.

14.4.2 Legal compliance

All countries have different legal requirements surrounding the collection and storage of data related to individuals (personally identifiable information). For example, South Africa has the Protection Of Personal Information (POPI) act and the European Union requires compliance with the General Data Protection Regulation (GDPR).

Another avenue for research is to evaluate the proposed solution against the various legal frameworks, assessing whether or not the solution is compliant and proposing suggestions to achieve compliance.

14.4.3 Mobile platforms

The prototype presented in this research makes use of the Android framework to prove that the model is practical. Another possibility for future work is to extend the prototype to other mobile platforms, like iOS and Windows Phone.

14.4.4 Expand data collection

Currently, the prototype collects data about applications being installed and removed from the device, DNS lookups and connections made to external resources. It also detects when passwords are changed and any security events that the operating system deems important. Many additional data points can be collected from the device, like GPS, network connections and various other data points. Future work can include expanding the data points that the prototype is capable of collecting.

14.4.5 Automated data analysis and incident detection

The solution presented in this research collects data about the actions that the user takes on their device and this data is stored in a document store that is controlled by the organisation. Currently, the solution requires that an investigator manually examine the data entries to find suspicious activity. The solution could be extended to apply data mining and artificial intelligence techniques to the data to automatically analyse the data and detect anomalies.

14.4.6 Protection against anti-forensics

The solution as presented in this research has some protection against threats, as described in section 13.2. However, the solution has not been hardened against advanced anti-forensic techniques. Anti-forensics are tools and techniques used to hinder digital forensic investigations [31]. In the future, the solution may be analysed from an anti-forensic perspective, and some suggestions may be made to improve the resilience to anti-forensic attacks.

14.4.7 Modularization of collection capabilities

Currently, the prototype mobile application is a monolithic application containing all the data collection capabilities that have been implemented. If an organisation requires that a new form of data be collected, the application has to be updated on all the employees' devices. Future work can focus on modularising these capabilities, adding the ability to "hot-load" new data collection capabilities into the mobile client.

Next, the author concludes this research with some final remarks.

14.5 Final conclusion

At the start of this research the author presented a problem, namely that MDM systems are purely preventative and presented the background of the research. After the background, the author presented a solution to the problem, describing a high-level model and architecture for such a system, as well as implementing a prototype to judge the practicality of such a system. Finally, the author critically evaluated the proposed solution.

As the processing power of mobile devices grow, preferences will shift even more strongly from traditional computers and laptops to mobile devices. The line between what is a "computer" and what is a "phone" will become more blurred, and as such, it is imperative to come up with security, digital forensic and digital forensic readiness frameworks for such devices.

Adding a MDM system with a Digital Forensic Readiness component as described in this research will give organisations the power to become completely mobile, while still maintaining regulatory compliance and industry best practises. The solution presented in this research can be a potent tool in an organisations journey to digital transformation.

During this research, several papers were produced that relates to the completed research:

1. Elsabé Ros and HS Venter. 'n Hoëvlak model vir die byvoeging van forensiese gereedheid tot mobiele toestelbestuur. Studentesimposium in die Natuurwetenskappe, 2017
2. Elsabé Ros and HS Venter. A high-level model for providing forensic readiness to mobile device management. Accepted at ECCWS18, withdrawn due to funding issues, 2018
3. Elsabé Ros and HS Venter. Digital forensic readiness in mobile device management systems. TBC, 2019

PART VI

Appendices

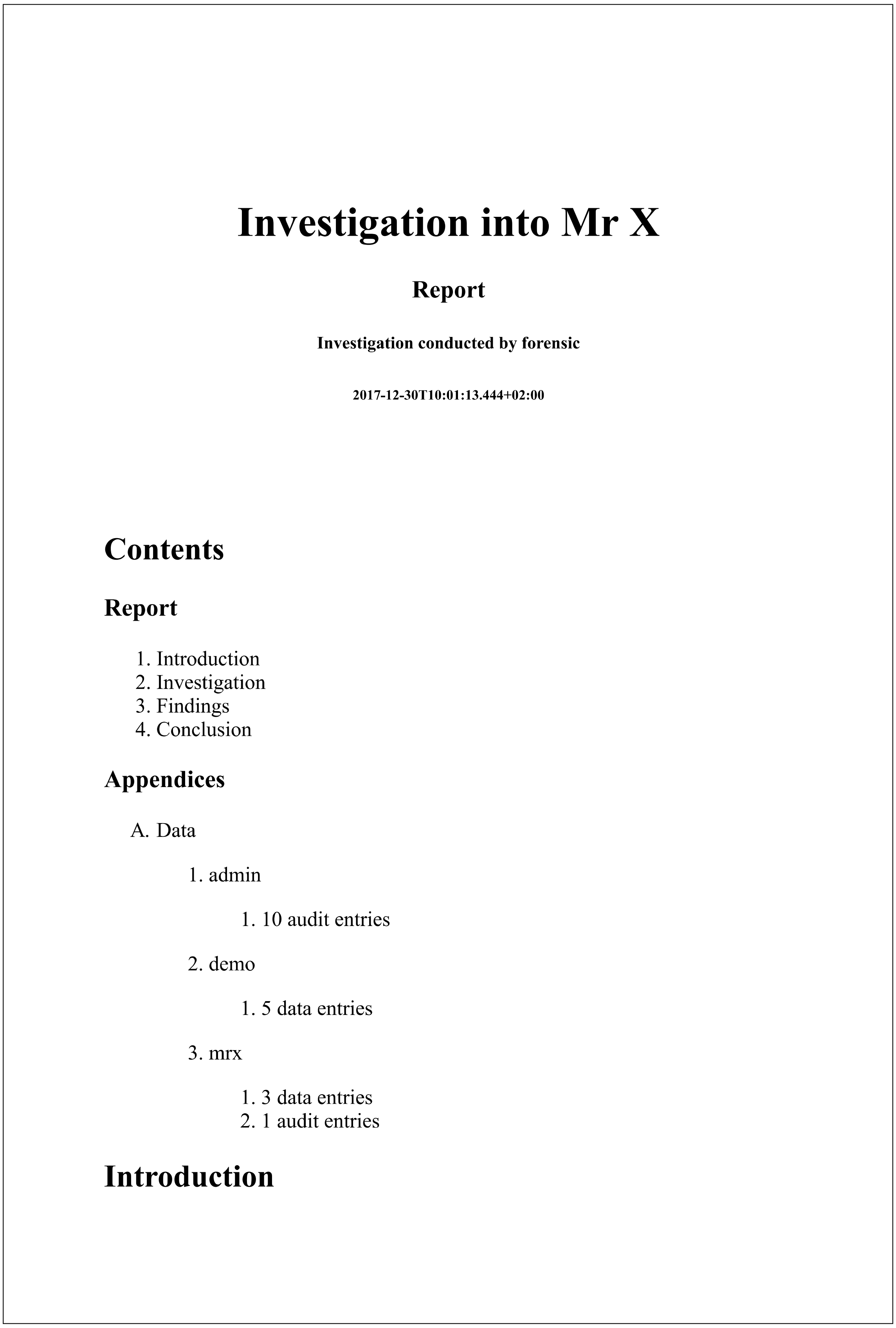
APPENDIX A

Generated report

This appendix contains a full example of the report that is generated by the prototype that the author implemented. This report has been pulled directly from the prototype and converted to PDF, **as is**. It is the responsibility of the investigator to complete the report, following the prompts that the solution generated.

The report contains the following details:

- The name of the investigation (*Investigation into Mr X*)
- The username of the investigator (*forensic*)
- When the forensic investigation was started (*2017-12-30T10:01:13.444+02:00*)
- A table of contents
- Four sections for the investigator to fill out, with prompts
- An appendix containing all the data that the investigator selected as relevant.



Investigation into Mr X

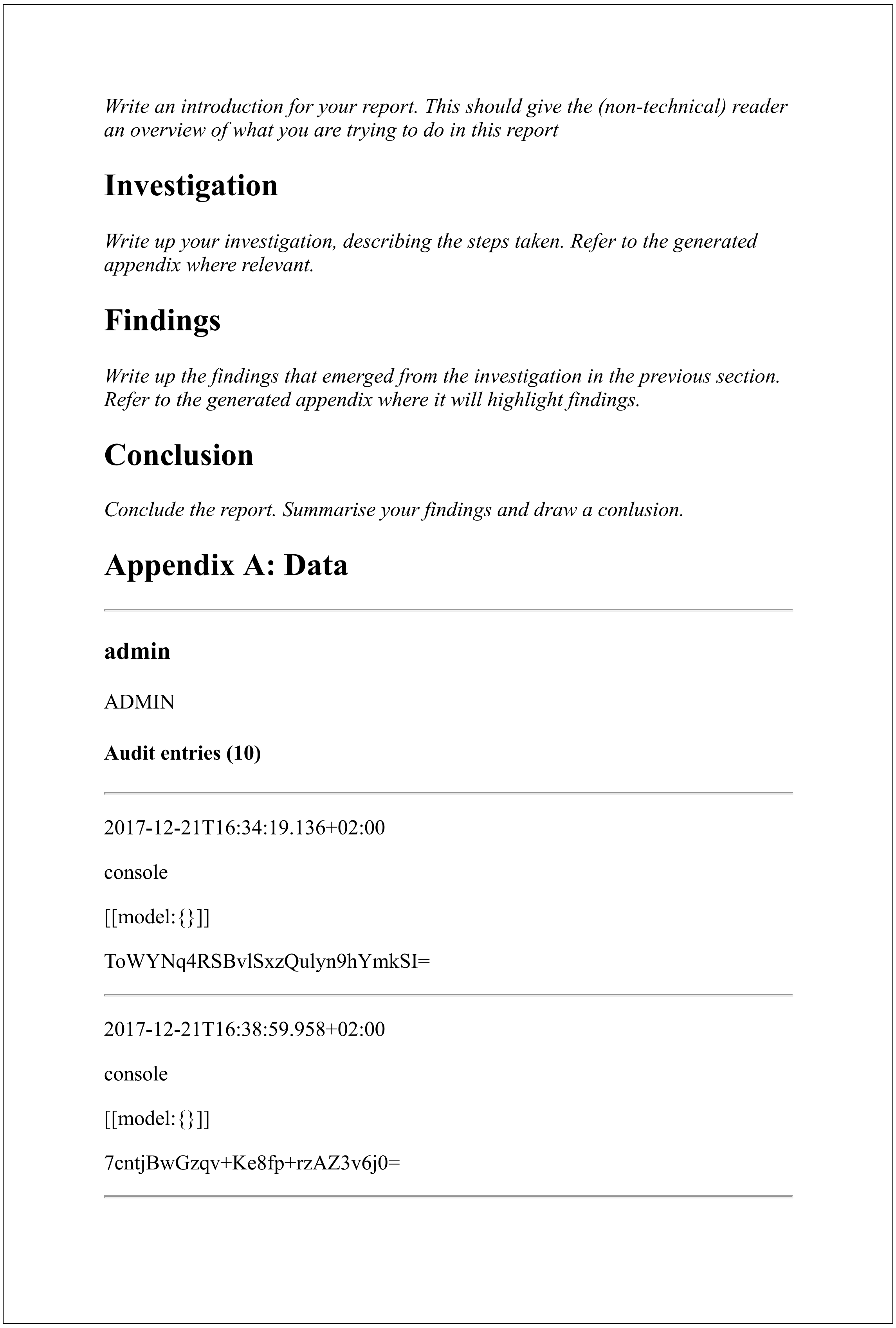
Report

Investigation conducted by forensic

2017-12-30T10:01:13.444+02:00

Contents

Introduction



Write an introduction for your report. This should give the (non-technical) reader an overview of what you are trying to do in this report

Investigation

Write up your investigation, describing the steps taken. Refer to the generated appendix where relevant.

Findings

Write up the findings that emerged from the investigation in the previous section. Refer to the generated appendix where it will highlight findings.

Conclusion

Conclude the report. Summarise your findings and draw a conlusion.

Appendix A: Data

---

admin

ADMIN

Audit entries (10)

2017-12-21T16:34:19.136+02:00

console

[[model:{}]]

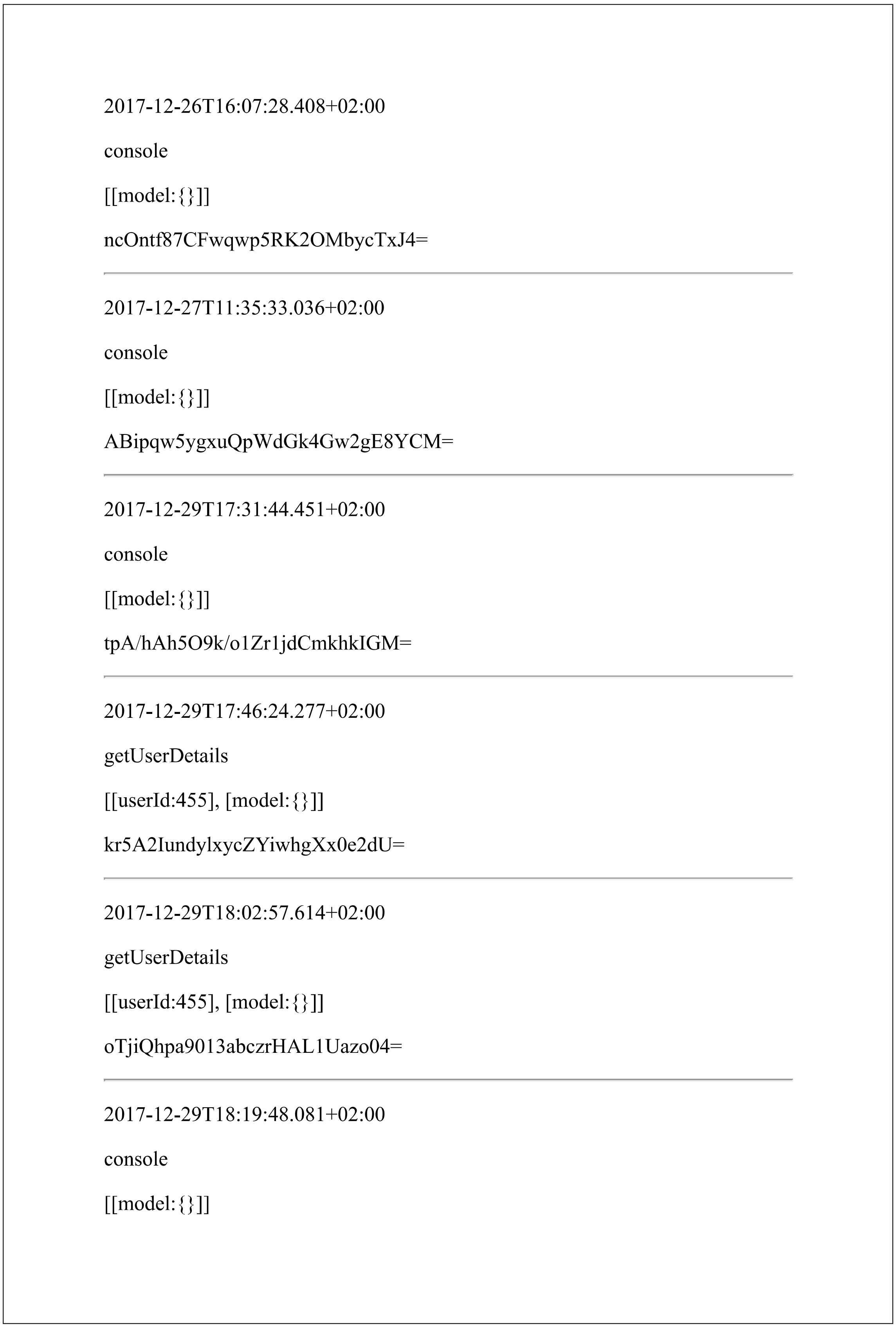
ToWYNq4RSBvlSxzQulyn9hYmkSI=

2017-12-21T16:38:59.958+02:00

console

[[model:{}]]

7cntjBwGzqv+Ke8fp+rzAZ3v6j0=



2017-12-26T16:07:28.408+02:00

console

[[model:{}]]

ncOntf87CFwqwp5RK2OMbycTxJ4=

2017-12-27T11:35:33.036+02:00

console

[[model:{}]]

ABipqw5ygxuQpWdGk4Gw2gE8YCM=

2017-12-29T17:31:44.451+02:00

console

[[model:{}]]

tpA/hAh5O9k/o1Zr1jdCmkhkIGM=

2017-12-29T17:46:24.277+02:00

getUserDetails

[[userId:455], [model:{}]]

kr5A2IundylxycZYiwhgXx0e2dU=

2017-12-29T18:02:57.614+02:00

getUserDetails

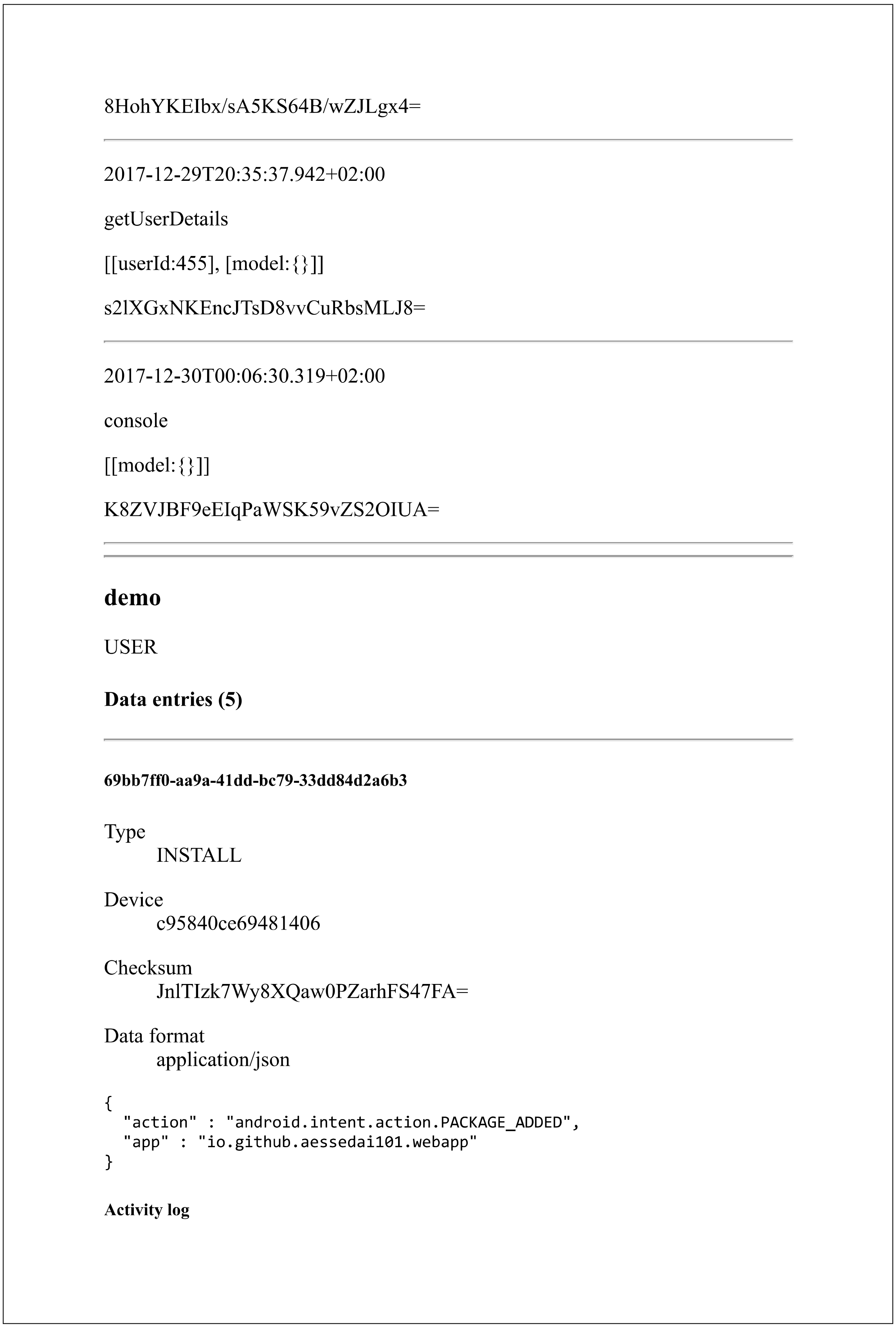
[[userId:455], [model:{}]]

oTjiQhpa9013abczrHAL1Uazo04=

2017-12-29T18:19:48.081+02:00

console

[[model:{}]]



8HohYKEIbx/sA5KS64B/wZJLgx4=

2017-12-29T20:35:37.942+02:00

getUserDetails

[[userId:455], [model:{}]]

s2lXGxNKEncJTsD8vvCuRbsMLJ8=

2017-12-30T00:06:30.319+02:00

console

[[model:{}]]

K8ZVJBF9eEIqPaWSK59vZS2OIUA=

---

demo

USER

Data entries (5)

69bb7ff0-aa9a-41dd-bc79-33dd84d2a6b3

Type
    INSTALL

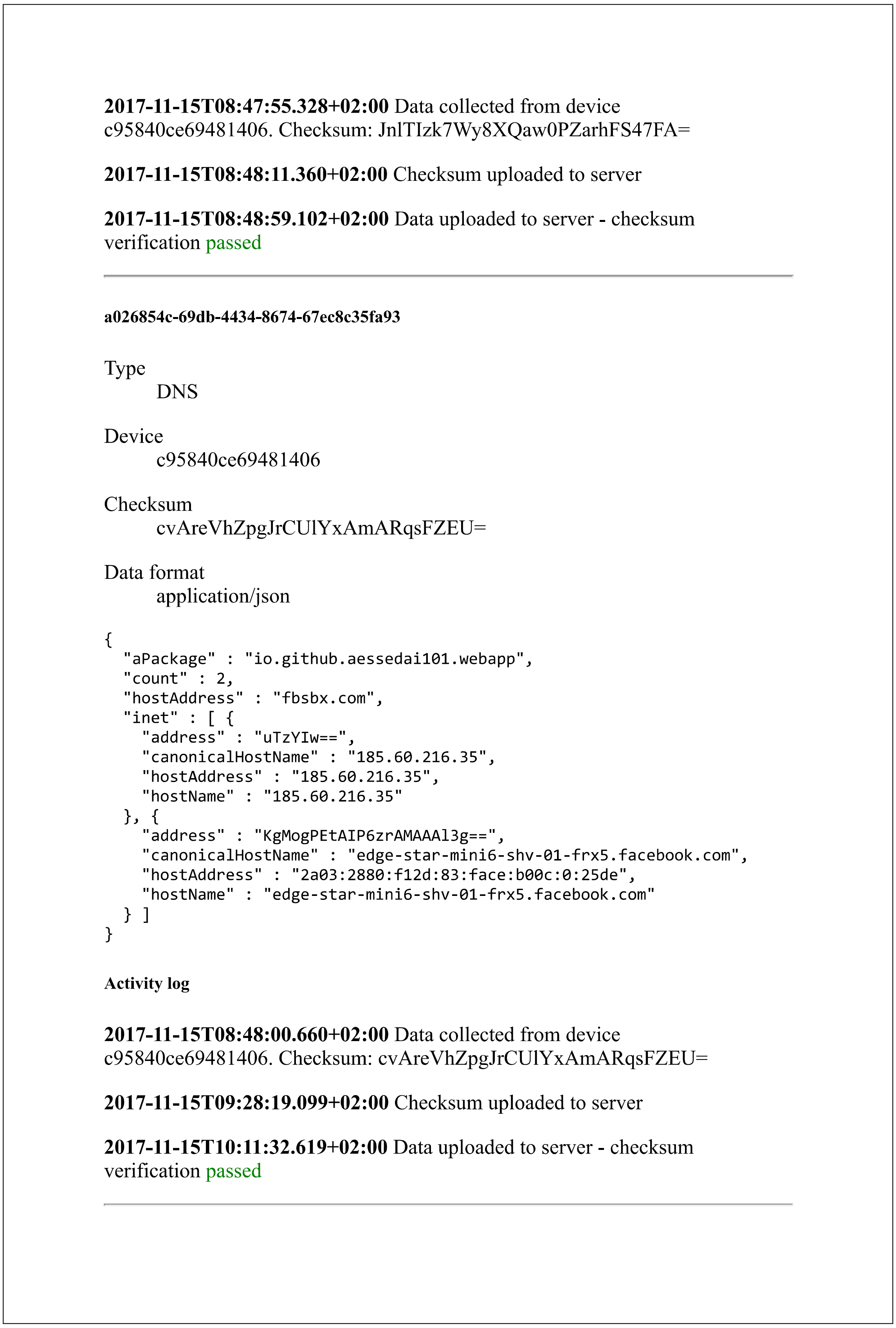
Device
    c95840ce69481406

Checksum
    JnlTIzk7Wy8XQaw0PZarhFS47FA=

Data format
    application/json

```
{
  "action" : "android.intent.action.PACKAGE_ADDED",
  "app" : "io.github.aessedai101.webapp"
}
```

Activity log



2017-11-15T08:47:55.328+02:00 Data collected from device c95840ce69481406. Checksum: JnlTIzk7Wy8XQaw0PZarhFS47FA=

2017-11-15T08:48:11.360+02:00 Checksum uploaded to server

2017-11-15T08:48:59.102+02:00 Data uploaded to server - checksum verification passed

---

a026854c-69db-4434-8674-67ec8c35fa93

Type
    DNS

Device
    c95840ce69481406

Checksum
    cvAreVhZpgJrCUlYxAmARqsFZEU=

Data format
    application/json

```
{
  "aPackage" : "io.github.aessedai101.webapp",
  "count" : 2,
  "hostAddress" : "fbsbx.com",
  "inet" : [ {
    "address" : "uTzYIw==",
    "canonicalHostName" : "185.60.216.35",
    "hostAddress" : "185.60.216.35",
    "hostName" : "185.60.216.35"
  }, {
    "address" : "KgMogPEtAIP6zrAMAAAl3g==",
    "canonicalHostName" : "edge-star-mini6-shv-01-frx5.facebook.com",
    "hostAddress" : "2a03:2880:f12d:83:face:b00c:0:25de",
    "hostName" : "edge-star-mini6-shv-01-frx5.facebook.com"
  } ]
}
```

Activity log

2017-11-15T08:48:00.660+02:00 Data collected from device c95840ce69481406. Checksum: cvAreVhZpgJrCUlYxAmARqsFZEU=

2017-11-15T09:28:19.099+02:00 Checksum uploaded to server

2017-11-15T10:11:32.619+02:00 Data uploaded to server - checksum verification passed

---

2e174b6a-9e1f-4514-abe2-0bb5cefd0f68

Type
    DNS

Device
    c95840ce69481406

Checksum
    r8x4o1N0RqB/2IpzDib0Uy8TfwU=

Data format
    application/json

```
{
  "aPackage" : "io.github.aessedai101.webapp",
  "count" : 2,
  "hostAddress" : "mobile.twitter.com",
  "inet" : [ {
    "address" : "xxCcaw==",
    "canonicalHostName" : "199.16.156.107",
    "hostAddress" : "199.16.156.107",
    "hostName" : "199.16.156.107"
  }, {
    "address" : "xxCcKw==",
    "canonicalHostName" : "199.16.156.43",
    "hostAddress" : "199.16.156.43",
    "hostName" : "199.16.156.43"
  } ]
}
```

Activity log

2017-11-15T08:48:07.431+02:00 Data collected from device c95840ce69481406. Checksum: r8x4o1N0RqB/2IpzDib0Uy8TfwU=

2017-11-15T09:28:59.542+02:00 Checksum uploaded to server

2017-11-15T10:11:33.402+02:00 Data uploaded to server - checksum verification passed

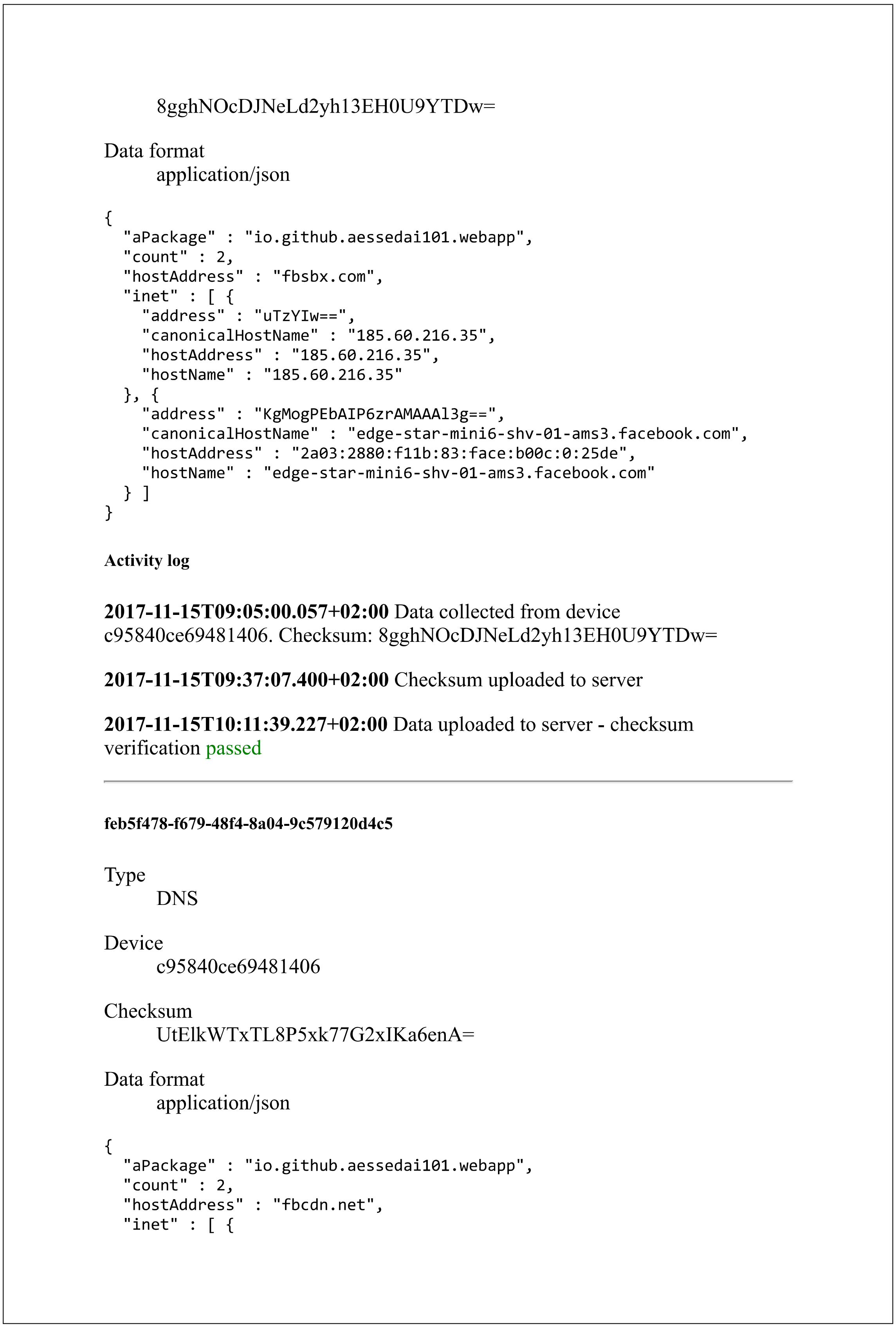
---

2afbc7cf-ddc0-4e9b-9ace-c0675db12008

Type
    DNS

Device
    c95840ce69481406

Checksum



8gghNOcDJNeLd2yh13EH0U9YTDw=

Data format
    application/json

```
{
  "aPackage" : "io.github.aessedai101.webapp",
  "count" : 2,
  "hostAddress" : "fbsbx.com",
  "inet" : [ {
    "address" : "uTzYIw==",
    "canonicalHostName" : "185.60.216.35",
    "hostAddress" : "185.60.216.35",
    "hostName" : "185.60.216.35"
  }, {
    "address" : "KgMogPEbAIP6zrAMAAAl3g==",
    "canonicalHostName" : "edge-star-mini6-shv-01-ams3.facebook.com",
    "hostAddress" : "2a03:2880:f11b:83:face:b00c:0:25de",
    "hostName" : "edge-star-mini6-shv-01-ams3.facebook.com"
  } ]
}
```

Activity log

2017-11-15T09:05:00.057+02:00 Data collected from device c95840ce69481406. Checksum: 8gghNOcDJNeLd2yh13EH0U9YTDw=

2017-11-15T09:37:07.400+02:00 Checksum uploaded to server

2017-11-15T10:11:39.227+02:00 Data uploaded to server - checksum verification passed

---

feb5f478-f679-48f4-8a04-9c579120d4c5

Type
    DNS

Device
    c95840ce69481406

Checksum
    UtElkWTxTL8P5xk77G2xIKa6enA=

Data format
    application/json

```
{
  "aPackage" : "io.github.aessedai101.webapp",
  "count" : 2,
  "hostAddress" : "fbcdn.net",
  "inet" : [ {
    "address" : "uTzYIw==",
    "canonicalHostName" : "185.60.216.35",
    "hostAddress" : "185.60.216.35",
    "hostName" : "185.60.216.35"
  }, {
    "address" : "KgMogPEtAIP6zrAMAAAl3g==",
    "canonicalHostName" : "edge-star-mini6-shv-01-frx5.facebook.com",
    "hostAddress" : "2a03:2880:f12d:83:face:b00c:0:25de",
    "hostName" : "edge-star-mini6-shv-01-frx5.facebook.com"
  } ]
}
```

Activity log

2017-11-15T09:10:59.969+02:00 Data collected from device c95840ce69481406. Checksum: UtElkWTxTL8P5xk77G2xIKa6enA=

2017-11-15T09:39:48.682+02:00 Checksum uploaded to server

2017-11-15T10:11:34.906+02:00 Data uploaded to server - checksum verification passed

---

mrx

USER

Data entries (3)

4b5aff1c-d6e3-4f2c-a18a-f9251c163b32

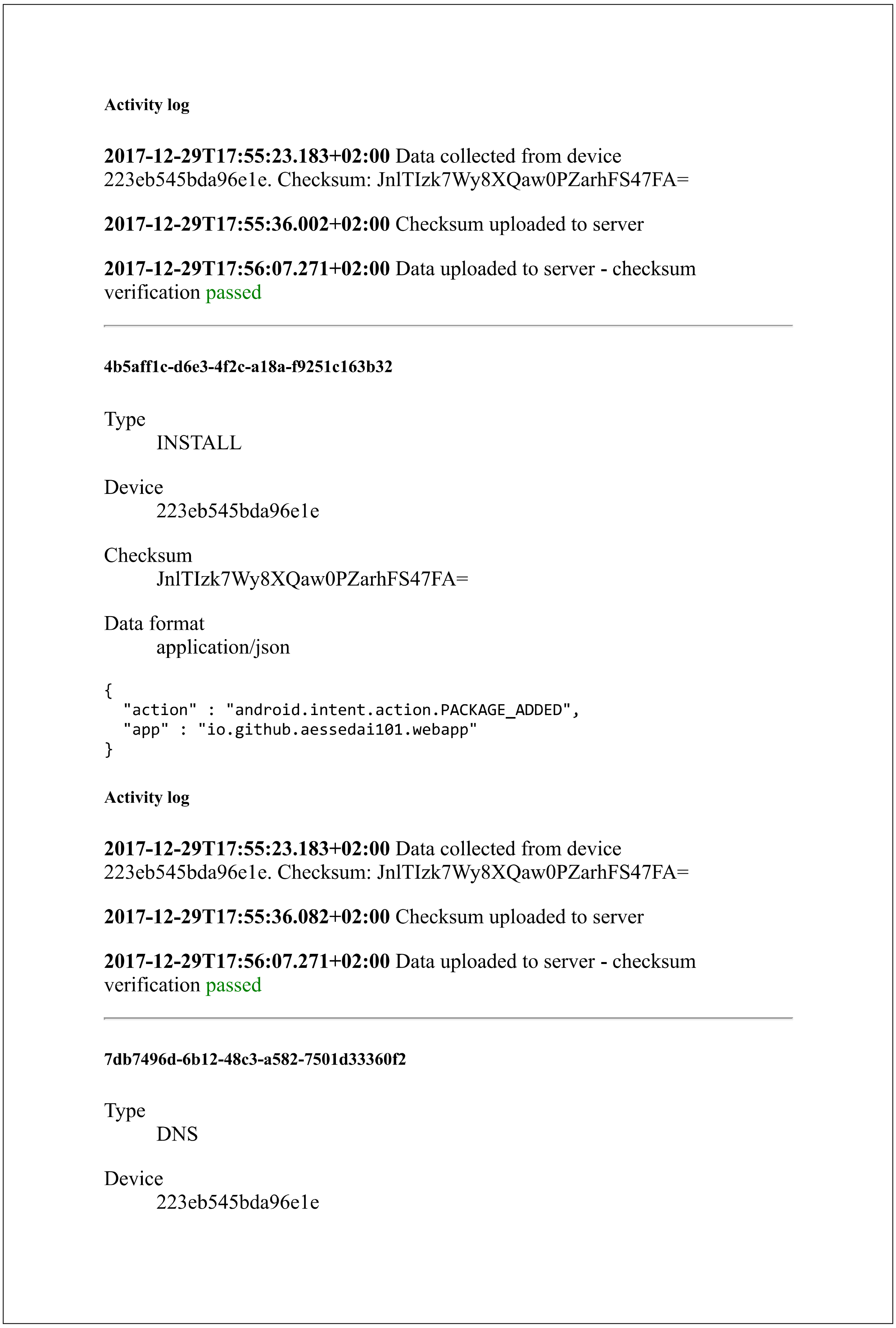
Type
    INSTALL

Device
    223eb545bda96e1e

Checksum
    JnlTIzk7Wy8XQaw0PZarhFS47FA=

Data format
    application/json

```
{
  "action" : "android.intent.action.PACKAGE_ADDED",
  "app" : "io.github.aessedai101.webapp"
}
```



Activity log

2017-12-29T17:55:23.183+02:00 Data collected from device 223eb545bda96e1e. Checksum: JnlTIzk7Wy8XQaw0PZarhFS47FA=

2017-12-29T17:55:36.002+02:00 Checksum uploaded to server

2017-12-29T17:56:07.271+02:00 Data uploaded to server - checksum verification passed

---

4b5aff1c-d6e3-4f2c-a18a-f9251c163b32

Type
: INSTALL

Device
: 223eb545bda96e1e

Checksum
: JnlTIzk7Wy8XQaw0PZarhFS47FA=

Data format
: application/json

```
{
  "action" : "android.intent.action.PACKAGE_ADDED",
  "app" : "io.github.aessedai101.webapp"
}
```

Activity log

2017-12-29T17:55:23.183+02:00 Data collected from device 223eb545bda96e1e. Checksum: JnlTIzk7Wy8XQaw0PZarhFS47FA=

2017-12-29T17:55:36.082+02:00 Checksum uploaded to server

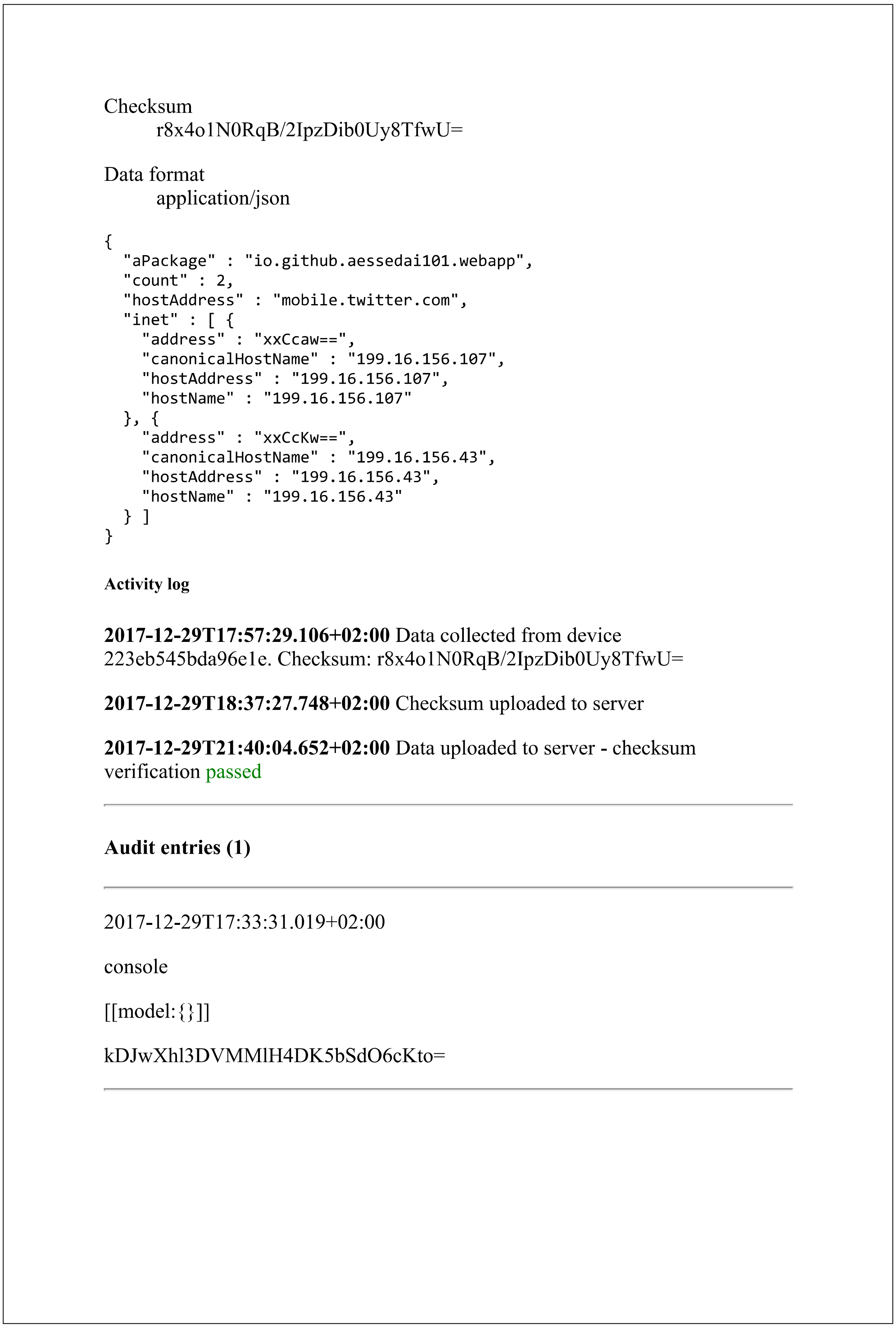
2017-12-29T17:56:07.271+02:00 Data uploaded to server - checksum verification passed

---

7db7496d-6b12-48c3-a582-7501d33360f2

Type
: DNS

Device
: 223eb545bda96e1e



Checksum
    r8x4o1N0RqB/2IpzDib0Uy8TfwU=

Data format
    application/json

```
{
  "aPackage" : "io.github.aessedai101.webapp",
  "count" : 2,
  "hostAddress" : "mobile.twitter.com",
  "inet" : [ {
    "address" : "xxCcaw==",
    "canonicalHostName" : "199.16.156.107",
    "hostAddress" : "199.16.156.107",
    "hostName" : "199.16.156.107"
  }, {
    "address" : "xxCcKw==",
    "canonicalHostName" : "199.16.156.43",
    "hostAddress" : "199.16.156.43",
    "hostName" : "199.16.156.43"
  } ]
}
```

Activity log

2017-12-29T17:57:29.106+02:00 Data collected from device 223eb545bda96e1e. Checksum: r8x4o1N0RqB/2IpzDib0Uy8TfwU=

2017-12-29T18:37:27.748+02:00 Checksum uploaded to server

2017-12-29T21:40:04.652+02:00 Data uploaded to server - checksum verification passed

---

Audit entries (1)

2017-12-29T17:33:31.019+02:00

console

[[model:{}]]

kDJwXhl3DVMMlH4DK5bSdO6cKto=

APPENDIX B

Implementation details

B.1 Introduction

As mentioned in the research, there are five components in the solution: the server, the mobile client, the console, the database and the data store. This section gives a quick overview of some of the interesting implementation details of the server, mobile client and the console. The database and the data store is not analysed separately, as their structures have been covered in details as part of the architecture definition. The author used Java for the implementation, with the Spring framework for the server and console and the Android framework for the mobile client. The server will be presented first.

B.2 Server

The server is the component that receives and services requests from the mobile client. This server performs six main functions, namely:

- login,
- registering a device,
- generating an authentication token,
- loading data collection policies,
- receiving checksums and
- receiving data.

Each of these functions is mapped to a RESTful endpoint [1], shown in table B.1. This table lists the function, the endpoint associated with it, the expected input and the given output. The full request and response objects are listed in appendix C, sections C.3.1 and C.3.2.

The source code that the server uses to verify the authentication token, verify the signature and decode the public key is listed in appendix C, sections C.3.3, C.3.4 and C.3.5 respectively.

[1]REST stands for REpresentational State Transfer. If the reader is interested in learning more I suggest Costello's article "Building Web Services the REST Way" [25] as a starting point.

Function	**Endpoint**	**Input**	**Output**
login	`POST /status`	username password device id auth token	status
registering device	`POST /device`	username password device id public key	success/failure
generate authentication token	`POST /device/deviceId`	username password signature	auth token expiry date
loading policies	`GET /device/deviceId/policy`	none	policy types policy values
receive checksums	`POST /device/deviceId/checksum`	policy type auth token data id checksum signature collect date	success/failure
receive data	`POST /device/deviceId/data`	auth token signature data entries	success/failure

Table B.1: Server endpoints

The server serves as an interface to the database and data store for the mobile client. The mobile client collects the data from the mobile devices and uploads it to the server. The implementation of the mobile client is presented next.

B.3 Mobile client

The mobile client was implemented using the Android framework, targeting Android 7.0 (API Level 24 - Nougat). The mobile client was forked from the `android-testdpc` sample project [33] available from `https://github.com/googlesamples/android-testdpc` and modified to suit the model's needs.

In addition to the core DFR-MDM functionality, the author added some utilities, specifically acquiring a device id (source code in section C.2.1), saving the downloaded policy details (section C.2.2) and a helper method to schedule the tasks (section C.2.3).

This section covered some of the implementation details of the mobile client. The next section presents the implementation details of the console, which is used to manage users, devices and do digital forensic investigations.

B.4 Console

As mentioned before, the console was implemented in Java using the Spring framework. The author chose to make use of a web interface, as it is responsive and can be accessed from many devices.

The console is a reasonably straightforward web application that checks authorisation and allows users to perform the actions that are described in the previous two chapters, namely viewing and editing users and policies, marking devices as inactive and performing digital forensic investigations.

The console also has automated auditing, that logs a database entry every time a user performs an action on the console. The source code for this can be found in appendix C, section C.4.1.

This appendix presented some of the implementation details of the prototype for the interested reader.

APPENDIX C

Source code listings

C.1 Introduction

This appendix contains the source code listings that are referenced in this research. The listings are listed according to the component it is contained in: mobile client, server and administration console.

C.2 Mobile client

The mobile client prototype was implemented in the Android framework using Java and requires Android 7.0 (API level 24) or above. This section contains the mobile client code snippets that are referenced.

C.2.1 Get device id

This code snippet gets a unique identifier for the device it is running on. It attempts to acquire the identifier from the system by requesting the `SECURE_ID`, the `deviceId` and the `IMEI`. If all of those mechanisms fail, the client generates an identifier and stores it securely.

```
public static String getDeviceIdentifier(Context context) {
    String androidId = Settings.Secure.getString(context.
       ↪ getContentResolver(),
    Settings.Secure.ANDROID_ID);
    if (androidId != null && !androidId.isEmpty() &&
    !"9774d56d682e549c".equals(androidId)){
        return androidId;
    }

    String deviceId = ((TelephonyManager)
    context.getSystemService(Context.TELEPHONY_SERVICE)).getDeviceId()
       ↪ ;
    if (deviceId != null && !deviceId.isEmpty()) {
        return deviceId;
    }

    if (android.os.Build.VERSION.SDK_INT >= 26) {
        deviceId = ((TelephonyManager)
        context.getSystemService(Context.TELEPHONY_SERVICE)).getImei()
           ↪ ;
        if (deviceId != null && !deviceId.isEmpty()) {
            return deviceId;
        }
    }

    final SharedPreferences prefs = context.getSharedPreferences(
        ↪ PREFS_FILE, 0);
    final String id = prefs.getString(PREFS_DEVICE_ID, null);
    if (id != null && !id.isEmpty()) {
        return id;
    }

    UUID uuid = UUID.randomUUID();
    String uuidString = uuid.toString();
    prefs.edit().putString(PREFS_DEVICE_ID, uuidString).commit();
    return uuidString;
}
```

C.2.2 Policy helper

This class is used to download and store the data collection policies. Since the policy download process can be started as part of multiple processes, the author isolated the functionality in a helper class. This class also takes care of checking the timestamp when the policy was last updated to prevent unnecessary network traffic.

```
public class PolicyHelper {
    private static final String TAG = PolicyHelper.class.getSimpleName
        ↪ ();
    private Application application;

    public PolicyHelper(Application application) {
        this.application = application;
        Log.d(TAG, "policies␣=␣" +
        RealmHelper.getRealm(application).where(Policy.class).findAll
            ↪ ());
    }

    public void loadPolicy() throws KeyUtil.KeyException,
        CryptoUtil.CryptoException, ParseException {
        Log.d(TAG, "loadPolicy()␣called");
        Date parsedDate = PreferenceHelper.getPolicyLoadDate(
            ↪ application);
        Log.d(TAG, "loadPolicy:␣parsedDate␣=␣" + parsedDate);

        Calendar limit = Calendar.getInstance();
        limit.add(Calendar.HOUR, -3);

        if (parsedDate == null || parsedDate.before(limit.getTime()))
            ↪ {
            Log.d(TAG, "loadPolicy:␣reloading␣policy");
            String authToken = PreferenceHelper.getAuthToken(
                ↪ application);
            Log.d(TAG, "loadPolicy:␣authToken␣=␣" + authToken);
            String signature = CryptoUtil.sign(KeyUtil.getPrivateKey()
                ↪ ,
            authToken.getBytes());
            Log.d(TAG, "loadPolicy:␣signature␣=␣" + signature);

            ServiceUtil.getService().getDevicePolicy(DeviceUtil.
                ↪ getDeviceIdentifier(application),
             authToken, signature)
            .subscribe(new DevicePolicyObserver(this));
        } else {
            Log.d(TAG, "loadPolicy:␣not␣loading␣policy␣-␣deadline␣hasn
                ↪ 't
␣␣␣␣␣␣␣␣␣␣␣␣expired␣yet");
        }
    }

    private void save(PolicyResponse policyResponse) {
        Log.d(TAG, "save()␣called␣with:␣policyResponse␣=␣[" +
            ↪ policyResponse +
        "]");

        Realm realm = RealmHelper.getRealm(application);

        realm.beginTransaction();
        RealmResults<Policy> policyList = realm.where(Policy.class).
            ↪ findAll();
        policyList.deleteAllFromRealm();
        realm.commitTransaction();

        PreferenceHelper.savePolicyLoadDate(application, new Date());

        for (Map.Entry<PolicyType, List<String>> entry :
        policyResponse.getPolicies().entrySet()) {
            realm.beginTransaction();
            RealmList<PolicyValue> valueList = new RealmList<>();
            for (String s : entry.getValue()) {
                PolicyValue v = realm.createObject(PolicyValue.class);
                v.setValue(s);
                valueList.add(v);
            }
            Policy policy = realm.createObject(Policy.class);
            policy.policyType(entry.getKey());
            policy.setValues(valueList);
            realm.commitTransaction();
        }
    }

    public static final class DevicePolicyObserver implements
    Observer<PolicyResponse> {
        private PolicyHelper policyHelper;

        public DevicePolicyObserver(PolicyHelper policyHelper) {
            Log.d(TAG, "DevicePolicyObserver()␣called␣with:␣
                ↪ policyHelper␣=␣[" +
            policyHelper + "]");
            this.policyHelper = policyHelper;
        }

        @Override
        public void onSubscribe(Disposable d) {
            Log.d(TAG, "onSubscribe()␣called␣with:␣d␣=␣[" + d + "]");
        }

        @Override
        public void onNext(PolicyResponse policyResponse) {
            Log.d(TAG, "onNext()␣called␣with:␣policyResponse␣=␣[" +
            policyResponse +
            "]");
            policyHelper.save(policyResponse);
        }

        @Override
        public void onError(Throwable e) {
            Log.e(TAG, "onError:␣", e);
        }

        @Override
        public void onComplete() {
            Log.d(TAG, "onComplete()␣called");
        }
    }
}
```

C.2.3 Schedule tasks

This helper method is used to schedule a job to start all the tasks that has to run on a schedule.

```
public static void schedule(Context context) {
    Log.d(TAG, "schedule()␣called␣with:␣context␣=␣[" + context + "]");
    ComponentName serviceComponent = new ComponentName(context,
    ForensicService.class);
    JobInfo.Builder builder = new JobInfo.Builder(0, serviceComponent)
       ↪ ;
    builder.setPersisted(true);
    builder.setPeriodic(TimeUnit.MILLISECONDS.convert(5, TimeUnit.
       ↪ HOURS));

    JobScheduler jobScheduler = context.getSystemService(JobScheduler.
       ↪ class);
    jobScheduler.schedule(builder.build());
}
```

C.3 Server

The server was implemented in Java, making use of the Spring framework. The server exposes RESTful endpoints that accept and produce JSON payloads. This section contains the source code for the server component.

C.3.1 Requests

This section contains all the request payloads as Java classes. The Spring framework takes care of converting these Java data transfer objects into payloads.

C.3.1.1 Request status

```
public class StatusRequest {
    private String username;
    private String password;
    private String deviceId;
    private String authToken;
}
```

C.3.1.2 Device registration request

```
public class DeviceRegistrationRequest {
    private String username;
    private String password;
    private String deviceId;
    private String publicKey;
}
```

C.3.1.3 Token generation request

```
public class TokenGenerationRequest {
    private String username;
    private String password;
    private String signature;
}
```

C.3.1.4 Checksum upload request

```
public class ChecksumRequest {
    private String type;
    private String authToken;
    private String dataId;
    private String checksum;
    private String signature;
    private Date collectionDate;
}
```

C.3.1.5 Data upload request

```
public class DataUploadRequest {
    private String authToken;
    private String signature;
    private List<DataUpload> data;
}

public static final class DataUpload {
    private String dataId;
    private String checksum;
    private String signature;
    private String dataType;
    private String data;
}
```

C.3.2 Responses

Again, the response payloads are represented as Java classes, as the framework takes care of the conversion to JSON payloads.

C.3.2.1 Status response

```
public enum Status {
    INCORRECT_CREDENTIALS,
    DEVICE_NOT_FOUND,
    DEVICE_REGISTERED_SOMEONE_ELSE,
    DEVICE_MARKED_INACTIVE,
    AUTH_TOKEN_NOT_CREATED,
    AUTH_TOKEN_EXPIRED,
    AUTH_TOKEN_VALID
}
```

C.3.2.2 Policy response

```
public class PolicyResponse {
    private Map<PolicyType, List<String>> policies;
}

public enum PolicyType {
    DNS,
    CONNECT,
    SECURITY,
    UNKNOWN,
    INSTALL,
    PASSWORD
}
```

C.3.2.3 Token generation response

```
public class TokenGenerationResponse {
    private String authToken;
    private Date authTokenExpiry;
}
```

C.3.2.4 Generic response

The generic response is sent in instances where the server does not need to send specific data to a mobile client, but just an acknowledgement and an indication of success or failure.

```
public class GenericResponse {
    private boolean result;
    private String message;
}
```

C.3.3 Verify authentication token

Before the server performs any critical functions, it verifies the caller's authentication token. This code snippet shows how the token is retrieved from the database and verified against the token that was sent by the mobile client.

```
private void verifyAuthToken(Device device, String authToken) throws
ServiceErrorException {
    AuthToken storedToken = authTokenRepository.findByDeviceAndUser(
       ↪ device,
    device.getUser());
    if (storedToken == null) {
        throw new ServiceErrorException("Invalid␣token");
    }
    if (!storedToken.getToken().equals(authToken)) {
        throw new ServiceErrorException("Invalid␣token");
    }
    if (storedToken.getExpiryDate().before(new Date())) {
        throw new ServiceErrorException("Expired␣token");
    }
}
```

C.3.4 Verify signature

Verifying the signature on a request is another way to verify the identity of the mobile client that is calling the server. This code snippet shows how the signature is verified. The public key comes from the database, where it is stored when the device is first registered.

```
private void verifySignature(PublicKey publicKey, byte[] data, byte[]
givenSignature) throws ServiceErrorException {
    try {
        BufferedInputStream dataIn = new BufferedInputStream(new
        ByteArrayInputStream(data));

        Signature signature = Signature.getInstance("SHA256WithRSA");
        signature.initVerify(publicKey);

        byte[] buffer = new byte[2048];
        int len;
        while (dataIn.available() != 0) {
            len = dataIn.read(buffer);
            signature.update(buffer, 0, len);
        }

        dataIn.close();

        boolean valid = signature.verify(givenSignature);
        if (!valid) {
            throw new ServiceErrorException("Signatures␣did␣not␣match!
               ↪ ");
        }
    } catch (NoSuchAlgorithmException | InvalidKeyException |
       ↪ IOException |
    SignatureException e) {
        throw new ServiceErrorException(e);
    }
}
```

C.3.5 Decode public key

The format in which the public key is stored in the database is not directly compatible with the Java classes that use it. This code snippet shows how to change the public key encoded in the database into the Java `PublicKey` class.

```
private PublicKey decodeKey(String encodedKey) throws
   ↪ ServiceErrorException {
    try {
        byte[] publicKeyBytes = Base64Utils.decodeFromString(
           ↪ encodedKey);

        return KeyFactory.getInstance("RSA").generatePublic(new
        X509EncodedKeySpec(publicKeyBytes));
    } catch (InvalidKeySpecException | NoSuchAlgorithmException e) {
        throw new ServiceErrorException(e);
    }
}
```

C.4 Console

The final component of the three is the console. This component allows administrators to manage users and policies and forensic investigators to conduct investigations and generate reports.

C.4.1 Logging audit events

All actions taken on the console are automatically logged into the database for audit purposes. To achieve this, the author used Aspect Oriented Programming [1] to apply the logic in the `logAuditEvent` method to all public methods in the `AdminController` and `ForensicController`.

```
@Aspect
@Component
public class AdminAuditAspect {
    private static final Logger log =
    LoggerFactory.getLogger(AdminAuditAspect.class);

    private final AuditEventRepository auditEventRepository;
    private final AuditChecksumRepository auditChecksumRepository;

    @Autowired
```

[1] AOP increases the maintainability of a project by extracting cross-cutting concerns out of the core business logic. This allows the developer to apply shared logic across many places without duplicating code. If the join point expressions are written correctly, the developer of new methods may not even have to make any changes to the aspect - it will automatically be invoked.

```
    public AdminAuditAspect(AuditEventRepository auditEventRepository,
    AuditChecksumRepository auditChecksumRepository) {
        this.auditEventRepository = auditEventRepository;
        this.auditChecksumRepository = auditChecksumRepository;
    }

    @Before("execution(*
␣␣␣␣io.github.aessedai101.masterspoc.controller.AdminController.*(..))
   ↪ ")
    public void logAuditAdmin(JoinPoint joinPoint) {
        log.debug("logAuditAdmin()␣called␣with␣" + "joinPoint␣=␣[" +
            ↪ joinPoint
        + "]");
        logAuditEvent(joinPoint);
    }

    @Before("execution(*
␣␣␣␣io.github.aessedai101.masterspoc.controller.ForensicController
   ↪ .*(..))")
    public void logAuditForensic(JoinPoint joinPoint) {
        log.debug("logAuditForensic()␣called␣with␣" + "joinPoint␣=␣["
            ↪ +
        joinPoint + "]");
        logAuditEvent(joinPoint);
    }

    private void logAuditEvent(JoinPoint joinPoint) {
        Authentication auth =
        SecurityContextHolder.getContext().getAuthentication();
        String name = auth.getName(); //get logged in username

        Signature signature = joinPoint.getSignature();
        String signatureName = signature.getName();

        Object[] arguments = joinPoint.getArgs();
        String[] argLabels = new String[arguments.length];
        LabeledArgument[] labeledArguments = new
        LabeledArgument[arguments.length];

        if (signature instanceof CodeSignature) {
            argLabels = ((CodeSignature) signature).getParameterNames
                ↪ ();
        } else {
         Arrays.fill(argLabels, "");
        }
        for (int i = 0; i < arguments.length; i++) {
            labeledArguments[i] = new LabeledArgument(argLabels[i],
            arguments[i]);
        }

        Date timestamp = new Date();
        AuditEvent event = auditEventRepository.save(new AuditEvent(
            ↪ name,
        timestamp, signatureName, labeledArguments));
        auditChecksumRepository.save(new AuditChecksum(event.getId(),
        Hash.calculateHash(event.toString())));
    }

    public static final class LabeledArgument {
        private String label;
        private Object value;

        public LabeledArgument(String label, Object value) {
            this.label = label;
            this.value = value;
        }

        public String getLabel() {
            return label;
        }

        public void setLabel(String label) {
            this.label = label;
        }

        public Object getValue() {
            return value;
        }

        public void setValue(Object value) {
            this.value = value;
        }

        @Override
        public String toString() {
            return "[" + label + ":" + value + ']';
        }
    }
}
```

Bibliography

[1] Av test: Malware. `https://www.av-test.org/en/statistics/malware/`. Accessed: 5 October 2018.

[2] Gartner survey shows that mobile device adoption in the workplace is not yet mature. `http://www.gartner.com/newsroom/id/3528217`. Accessed: 9 January 2017.

[3] The Advantages and Disadvantages of BYOD. `http://www.optimussourcing.com/learninghintsandtips/the-advantages-and-disadvantages-of-byod`, 2013. Accessed: 10 January 2017.

[4] Device Management Architecture. Technical Report OMA-AD-DM-V2_0-20160209-A, Open Mobile Alliance, February 2016.

[5] Device Management Requirements. Technical Report OMA-RD-DM-V2_0-20160209-A, Open Mobile Alliance, February 2016.

[6] OMA Device Management Protocol. Technical Report OMA-TS-DM_Protocol-V2_0-20160209-A, Open Mobile Alliance, February 2016.

[7] Sisira Adikari, Craig McDonald, and John Campbell. Little design up-front: A design science approach to integrating usability into agile requirements engineering. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5610 LNCS(PART 1):549–558, 2009.

[8] Rizwan Ahmed and Rajiv V. Dharaskar. Mobile Forensics: an Overview, Tools, Future trends and Challenges from Law Enforcement perspective. *6th International Conference on E-Governance, ICEG, Emerging Technologies in E-Government, M-Government*, pages 312–323, 2008.

[9] Android. Jobscheduler. `https://developer.android.com/reference/android/app/job/JobScheduler.html`. Accessed: 3 January 2018.

[10] Nicoló Andronio, Stefano Zanero, and Federico Maggi. HelDroid: Dissecting and Detecting Mobile Ransomware. In Herbert Bos, Fabian Monrose, and Gregory Blanc, editors, *Research in Attacks, Intrusions, and Defenses*, pages 382–404, Cham, 2015. Springer International Publishing.

[11] Marzie Astani, Kathy Ready, and Mussie Tessema. Byod issues and strategies in organizations. *Issues in Information Systems*, 14(2), 2013.

[12] David Barske, Adrie Stander, and Jason Jordaan. A digital forensic readiness framework for south african sme's. In *Information Security for South Africa (ISSA), 2010*, pages 1–6. IEEE, 2010.

[13] Venansius Baryamureeba and Florence Tushabe. The Enhanced Digital Investigation Process Model. In *Digital Forensic Research Workshop*. 2004.

[14] Martin Bichler. Design science in information systems research. *Wirtschaftsinformatik*, 48(2):133–135, 2006.

[15] Nikita Buchka and Alexey Firsh. Skygofree: Following in the footsteps of hackingteam. `https://securelist.com/skygofree-following-in-the-footsteps-of-hackingteam/83603/`. Accessed: 16 January 2018.

[16] Ngoc Duong Bui, Alla Grigorievna Kravets, Tuan Anh Nguyen, and Le Thanh Tung Nguyen. Tracking events in mobile device management system. *IISA 2015 - 6th International Conference on Information, Intelligence, Systems and Applications*, 2016.

[17] Brian Carrier. Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of digital evidence*, 1(4):1–12, 2003.

[18] Brian Carrier and Eugene H Spafford. An event-based digital forensic investigation framework. In *Digital forensic research workshop*, pages 11–13, 2004.

[19] Brian Carrier, Eugene H Spafford, et al. Getting physical with the digital investigation process. *International Journal of digital evidence*, 2(2):1–20, 2003.

[20] Eoghan Casey. What does "forensically sound" really mean? *Digital Investigation*, 4(2):49–50, 2007.

[21] J Morris Chang, Pao-Chung Ho, and Teng-Chang Chang. Securing byod. *IT Professional*, 16(5):9–11, 2014.

[22] Lawrence Chung and Julio Cesar Sampaio do Prado Leite. On Non-Functional Requirements in Software Engineering. *Conceptual modeling: Foundations and applications*, pages 363–379, 2009.

[23] Fred Cohen. *Digital forensic evidence examination.* 5 edition, 2012.

[24] M. I. Cohen. PyFlag - An advanced network forensic framework. *Digital Investigation*, 5(SUPPL.):112–120, 2008.

[25] Roger L Costello. Building web services the REST way. `http://www.xfront.com/REST-Web-Services.html`. Accessed: 27 December 2017.

[26] Georg Disterer and Carsten Kleiner. BYOD Bring Your Own Device. *Procedia Technology*, 9:43–53, 2013.

[27] Mohamed Elyas, Sean B Maynard, Atif Ahmad, and Andrew Lonie. Towards a systemic framework for digital forensic readiness. *Journal of Computer Information Systems*, 54(3):97–105, 2014.

[28] Meisam Eslahi, Maryam Var Naseri, H Hashim, NM Tahir, and Ezril Hisham Mat Saad. Byod: Current state and security challenges. In *Computer Applications and Industrial Electronics (ISCAIE), 2014 IEEE Symposium on*, pages 189–192. IEEE, 2014.

[29] A Farjamfar, M T Abdullah, R Mahmod, and N I Udzir. A review on mobile device's digital forensic process models. *Research Journal of Applied Sciences, Engineering and Technology*, 8(3):358–366, 2014.

[30] Jon Friedman and Daniel V Hoffman. Protecting data on mobile devices: A taxonomy of security threats to mobile computing and review of applicable defenses. *Information Knowledge Systems Management*, 7(1, 2):159–180, 2008.

[31] Simson Garfinkel. Anti-forensics: Techniques, detection and countermeasures. In *2nd International Conference on i-Warfare and Security*, volume 20087, pages 77–84, 2007.

[32] Arnab Ghosh, Prashant Kumar Gajar, and Shashikant Rai. Bring your own device (byod): Security risks and mitigating strategies. *Journal of Global Research in Computer Science*, 4(4):62–70, 2013.

[33] Google. Test Device Policy Control (Test DPC) App. `https://github.com/googlesamples/android-testdpc`, 2017.

[34] C P Grobler and C P Louwrens. Digital Forensic Readiness as a Component of Information Security Best Practice. In *IFIP International Information Security Conference*, pages 13–24, 2007.

[35] C.P. Grobler, C.P. Louwrens, and S.H. von Solms. A Framework to Guide the Implementation of Proactive Digital Forensics in Organisations. In *2010 International Conference on Availability, Reliability and Security*, pages 677–682. IEEE, feb 2010.

[36] Justin Grover. Android forensics: Automated data collection and reporting from a mobile device. In *Digital Investigation*, volume 10, 2013.

[37] Spyros T Halkidis, Alexander Chatzigeorgiou, and George Stephanides. A qualitative evaluation of security patterns. In *International Conference on Information and Communications Security*, pages 132–144. Springer, 2004.

[38] Alan Hevner, Salvatore March, Jinsoo Park, and Sudha Ram. Design Science Research in Information Systems. *MIS quarterly*, 28(1):75–105, 2004.

[39] Alan R Hevner. A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems © Scandinavian Journal of Information Systems*, 19(192):87–92, 2007.

[40] ISO 27043. INTERNATIONAL STANDARD ISO / IEC 27043: Information technology — Security techniques — Incident investigation principles and processes. 2015.

[41] Myeongju Ji, Sungryong Kim, Yongjin Park, and Jeong Hyun Yi. Mobile Device Management System with Portable Devices. pages 3–4, 2015.

[42] Audun Jøsang and Simon Pope. User centric identity management. In *AusCERT Asia Pacific Information Technology Security Conference*, page 77. Citeseer, 2005.

[43] Michael Kohn, Jan H P Eloff, and Martin S Olivier. Framework for a Digital Forensic Investigation. *Proceedings of the ISSA 2006 from Insight to Foresight Conference*, 2006.

[44] Kevin P Kopp. Electronic communications in the workplace: E-mail monitoring and the right of privacy. *Seton Hall Const. LJ*, 8:861, 1997.

[45] Alla G. Kravets, Ngoc Duong Bui, and Mohammed Al-Ashval. Mobile Security Solution for Enterprise Network. In *Communications in Computer and Information Science*, volume 466 CCIS, pages 371–382. 2014.

[46] Arun Kumar. Bring your own device (BYOD) Advantages and Disadvantages. `http://www.thewindowsclub.com/bring-your-own-device-byod`, 2014. Accessed: 10 January 2017.

[47] Paul Leach, Michael Mealling, and Rich Salz. A universally unique identifier (uuid) urn namespace (rfc 4122). Technical report, 2005.

[48] Benedikt Lebek, Kenan Degirmenci, and Michael H Breitner. Investigating the influence of security, privacy, and legal concerns on employees' intention to use byod mobile devices. 2013.

[49] Adrian Leung. A mobile device management framework for secure service delivery. *Information Security Technical Report*, 13(3):118–126, 2008.

[50] User: levi (https://stackoverflow.com/users/766548/levi). Why do access tokens expire? StackOverflow. URL:https://stackoverflow.com/questions/7030694/why-do-access-tokens-expire (version: 2016-05-28).

[51] Bob Lightsey. Systems engineering fundamentals. Technical report, DEFENSE ACQUISITION UNIV FT BELVOIR VA, 2001.

[52] Leslie Liu, Randy Moulic, and Dennis Shea. Cloud Service Portal for Mobile Device Management. *2010 IEEE 7th International Conference on E-Business Engineering*, pages 474–478, 2010.

[53] March and Storey. Design Science in the Information Systems Discipline: An Introduction to the Special Issue on Design Science Research. *MIS Quarterly*, 32(4):725, 2008.

[54] Salvatore T. March and Gerald F. Smith. Design and natural science research on information technology. *Decision Support Systems*, 15(4):251–266, dec 1995.

[55] Ben Martini and Kim Kwang Raymond Choo. An integrated conceptual digital forensic framework for cloud computing. *Digital Investigation*, 9(2):71–80, 2012.

[56] Rodney McKemmish. When is digital evidence forensically sound? *IFIP International Federation for Information Processing*, 285:3–15, 2008.

[57] Catherine Meadows. Detecting attacks on mobile agents. In *Foundations for Secure Mobile Code Workshop*, pages 64–65. Citeseer, 1997.

[58] George Mohay. Technical challenges and directions for digital forensics. In *Systematic Approaches to Digital Forensic Engineering, 2005. First International Workshop on*, pages 155–161. IEEE, 2005.

[59] Bill Morrow. Byod security challenges: control and protect your most sensitive data. *Network Security*, 2012(12):5–8, 2012.

[60] Antonis Mouhtaropoulos, Chang Tsun Li, and Marthie Grobler. Digital forensic readiness: Are we there yet?, 2014.

[61] Emilio Raymond Mumba and Hein S Venter. Mobile forensics using the harmonised digital forensic investigation process. In *Information Security for South Africa (ISSA), 2014*, pages 1–10. IEEE, 2014.

[62] Roger M Needham. Denial of service. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 151–153. ACM, 1993.

[63] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. *Proceedings of the conference on The future of Software engineering - ICSE '00*, 1:35–46, 2000.

[64] Stacey Omeleze and Hein S Venter. Testing the harmonised digital forensic investigation process model-using an android mobile phone. In *Information Security for South Africa, 2013*, pages 1–8. IEEE, 2013.

[65] Kevin Ortbach, Tobias Brockmann, and Stefan Stieglitz. Drivers for the Adoption of Mobile Device Management in Organizations. *Proceedings of the 22nd European Conference on Information Systems (ECIS)*, 2014.

[66] Gary Palmer. A road map for digital forensic research. In *First Digital Forensic Research Workshop, Utica, New York*, pages 27–30, 2001.

[67] Dhirendra Pandey, U. Suman, and A.K. Ramani. An Effective Requirement Engineering Process Model for Software Development and Requirements Management. *2010 International Conference on Advances in Recent Technologies in Communication and Computing*, pages 287–291, 2010.

[68] G. Pangalos, C. Ilioudis, and I. Pagkalos. The importance of Corporate Forensic Readiness in the information security framework. *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, pages 12–16, 2010.

[69] Karen P Patten and Mark A Harris. The need to address mobile device security in the higher education it curriculum. *Journal of Information Systems Education*, 24(1):41, 2013.

[70] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3):45–77, 2007.

[71] Charles Pfleeger and Shari Lawrence Pfleeger. *Security in Computing*. 2007.

[72] Markus Pierer. *Mobile Device Management*. Springer Fachmedien Wiesbaden, Wiesbaden, 2016.

[73] User: poncho (https://crypto.stackexchange.com/users/452/poncho). how does https key get shared? Crypto Stack Exchange. URL:https://crypto.stackexchange.com/a/9091/8507 (version: 2013-07-07).

[74] Jan Pries-Heje, Richard Baskerville, and John R Venable. Strategies for design science research evaluation. In *ECIS*, pages 255–266, 2008.

[75] Balasubramaniam Ramesh, Lan Cao, and Richard Baskerville. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5):449–480, 2010.

[76] K. Reddy and H. S. Venter. The architecture of a digital forensic readiness management system. *Computers and Security*, 32:73–89, 2013.

[77] Mark Reith, Clint Carr, and Gregg Gunsch. An Examination of Digital Forensic Models. *International Journal of Digital Evidence*, 1(3):1–12, 2002.

[78] Keunwoo Rhee, Woongryul Jeon, and Dongho Won. Security Requirements of a Mobile Device Management System. *International Journal of Security and its Applications*, 6(2):353–358, 2012.

[79] Keunwoo Rhee, Hawon Kim, and Hac Yun Na. Security test methodology for an agent of a mobile device management system. *International Journal of Security and Its Applications*, 6(2):137–142, 2012.

[80] Keunwoo Rhee, Dongho Won, Sang Woon Jang, Sooyoung Chae, and Sangwoo Park. Threat modeling of a mobile device management system for secure smart work. *Electronic Commerce Research*, 13(3):243–256, 2013.

[81] Robert Rowlingson. A Ten Step Process for Forensic Readiness. *International Journal of Digital Evidence*, 2(3), 2004.

[82] Gruia-Catalin Roman. A taxonomy of current issues in requirements engineering. *Computer*, 18(4):14–23, apr 1985.

[83] Elsabé Ros and HS Venter. 'n Hoëvlak model vir die byvoeging van forensiese gereedheid tot mobiele toestelbestuur. Studentesimposium in die Natuurwetenskappe, 2017.

[84] Elsabé Ros and HS Venter. A high-level model for providing forensic readiness to mobile device management. Accepted at ECCWS18, withdrawn due to funding issues, 2018.

[85] Elsabé Ros and HS Venter. Digital forensic readiness in mobile device management systems. TBC, 2019.

[86] Stephanie AC Schuckers. Spoofing and anti-spoofing measures. *Information Security technical report*, 7(4):56–62, 2002.

[87] David Schuetz. The ios mdm protocol. *Intrepidus Group, Inc*, 29, 2011.

[88] Michael Shrivathsan. Types of software requirements. `http://rmblog.accompa.com/2012/04/types-of-software-requirements/`. Accessed: 22 December 2017.

[89] Niharika Singh. Byod genie is out of the bottle–"devil or angel". *Journal of Business Management & Social Sciences Research*, 1(3):1–12, 2012.

[90] Alan D Smith and Robert A Faley. E-mail workplace privacy issues in an information-and knowledge-based environment. *Southern Business Review*, 27(1):8, 2001.

[91] Murugiah Souppaya and Karen Scarfone. Guidelines for managing the security of mobile devices in the enterprise. *NIST special publication*, 800:124, 2013.

[92] Paul Steiner. Going beyond mobile device management. *Computer Fraud & Security*, 2014(4):19–20, 2014.

[93] John Tan. Forensic Readiness Assessment. *Cambridge, MA:@ Stake*, pages 1–23, 2001.

[94] Philip M Trenwith and H.S. Venter. Digital forensic readiness in the cloud. In *2013 Information Security for South Africa*, number January, pages 1–5. IEEE, aug 2013.

[95] Aleksandar Valjarevic and Hein S Venter. A comprehensive and harmonized digital forensic investigation process model. *Journal of forensic sciences*, 60(6):1467–1483, 2015.

[96] J Venable. The Role of Theory and Theorising in Design Science Research. *Proceedings of DESRIST*, pages 24–35, 2006.

[97] K Vijay and Communication Technology. Introduction to Design Science Research in Information and Communication Technology. In *Design Science Research Methods and Patterns*, pages 7–30. 2007.

[98] Yong Wang, Jinpeng Wei, and Karthik Vangury. Bring your own device security issues and challenges. In *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, pages 80–85. IEEE, 2014.

[99] Myria Watkins Allen, Stephanie J Coopman, Joy L Hart, and Kasey L Walker. Workplace surveillance and managing privacy boundaries. *Management Communication Quarterly*, 21(2):172–200, 2007.

[100] Rodrigo Werlinger, Kasia Muldner, Kirstie Hawkey, and Konstantin Beznosov. Preparation, detection, and analysis: the diagnostic work of it security incident response. *Information Management & Computer Security*, 18(1):26–42, 2010.

[101] Bing Wu, Jianmin Chen, Jie Wu, and Mihaela Cardei. A survey of attacks and countermeasures in mobile ad hoc networks. In *Wireless network security*, pages 103–135. Springer, 2007.

[102] J Christopher Zimmer, Riza Ergun Arsal, Mohammad Al-Marzouq, and Varun Grover. Investigating online information disclosure: Effects of information relevance, trust and risk. *Information & management*, 47(2):115–123, 2010.

[103] Dirk Zimmermann and Lennart Grötzbach. A Requirement Engineering Approach to User Centered Design. *Proceedings of the 12th international Conference on Human-computer interaction: Interaction Design and Usability*, pages 360 – 369, 2007.