



CHALMERS
UNIVERSITY OF TECHNOLOGY

Security Analysis of Machine Monitoring Sensor Communication

A threat modeling process implementation and evaluation

Master's thesis in Computer System and Networks

MARTIN LJUNGDAHL
MICHAEL NORDSTRÖM

MASTER'S THESIS 2016

Security Analysis of Machine Monitoring Sensor Communication

A threat modeling process implementation and evaluation

MARTIN LJUNGDAHL
MICHAEL NORDSTRÖM



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

The Authors grants to Chalmers University of Technology and University of Gothenburg right to publish the Work electronically and to in a non commercial purpose make it accessible on the Internet. The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Security Analysis of Machine Monitoring Sensor Communication
A threat modeling process implementation and evaluation

MARTIN LJUNGDAHL
MICHAEL NORDSTRÖM

© MARTIN LJUNGDAHL, 2016.
© MICHAEL NORDSTRÖM, 2016.

Supervisor: Riccardo Scandariato, Chalmers Univeristy
Supervisor: Viktor Lindström, Cybercom Sweden AB
Examiner: Erland Jonsson, Chalmers University

Master's Thesis 2016
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in L^AT_EX
Gothenburg, Sweden 2016

Security Analysis of Machine Monitoring Sensor Communication

A threat modeling process implementation and evaluation

MARTIN LJUNGDAHL

MICHAEL NORDSTRÖM

Department of Computer and Engineering

Chalmers University of Technology

Abstract

The number of small devices that are connected to the Internet is increasing rapidly and the system that controls them are becoming more and more complex. Using these devices in products and system has the potential to lower costs, increase performance and provide new functionality. A substantial amount of these devices are used in "smart homes" or to monitor and control critical electro-mechanical systems. When developing such system often functionality and performance is prioritized in comparison to security and many systems have computer security and network security concerns.

To help the developers create secure systems it exist a practice named Threat Modeling in which you work with the system through different stages to find its vulnerabilities. There exist several threat models that are aimed for specific systems of a certain type.

It exists limited research about threat models aimed for system consisting of small devices connected to the Internet. In this project a threat modeling process will be conducted and applied on a smartphone/IoT system developed by one of Cybercom's customer. In addition, the threat modeling process will be evaluated for correctness and applicability when applying it to a smartphone/IoT system and how the process might be improved. Platform specific threat libraries created by accredited sources will be used to for both validation and improvements. Penetration testing will be carried out with a subset of the threats generated by the threat modeling process and from the threat libraries in order to validate the applicability of the threats.

Keywords: Threat model process, OWASP Mobile, IoT, STRIDE, DFD

Acknowledgements

We want to extend our thanks to our supervisors; Viktor Lindström at Cybercom and Riccardo Scandariato at Chalmers University of Technology for their guidance along the way. We would also like to thank Marcus Tannerfalk and Henrik Lundqvist at Cybercom for the opportunity to carry out this thesis at Cybercom. Last but not least we want to thank Erland Jonsson, at Chalmers University of Technology, for being our examiner.

The Authors, Göteborg 31/5/16

Preface

This thesis has been conducted at Cybercom Sweden AB. Cybercom is an IT consulting company that assists leading companies and organizations to benefit from the opportunities of the connected world. The company's areas of expertise span the entire ecosystem of communications services. Cybercom's domestic market is the Nordic region, and in addition the company offers global delivery capacity for local and international business. The project has been conducted on a system developed in collaboration between Cybercom and one of their customers.

Glossary

- Threat - Anything that can exploit a vulnerability, intentionally or accidentally, and obtain, damage, or destroy an asset.
- Vulnerability - Weaknesses or gaps in a security program that can be exploited by threats to gain unauthorized access to an asset.
- Risk - The potential for loss, damage or destruction of an asset as a result of a threat exploiting a vulnerability.

Contents

List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 Research question	2
1.2 Description of work	2
1.2.1 Threat Modeling	2
1.2.2 Penetration Testing	2
1.3 Contribution	2
1.4 System Overview	3
1.5 Limitations	4
1.5.1 Android	4
1.5.2 Sensor Penetration Testing	4
1.5.3 Risk Assessment	5
1.5.4 Revised Copy	5
1.6 Thesis Outline	5
2 Background and Related Work	7
2.1 Threat Modeling	7
2.1.1 Assets	7
2.1.2 Entry Points	7
2.1.3 Trust Levels	8
2.1.4 Extended CIA Triad	8
2.1.5 External Dependencies	9
2.1.6 Data Flow Diagram	9
2.1.7 STRIDE	10
2.1.8 Attack Trees	13
2.1.9 Risk Assessment	14
2.1.10 Threat Modeling Process	15
2.2 OWASP	17
2.2.1 OWASP Mobile Security Project	18
2.3 Microsoft Threat Modeling Tool	19
2.4 Android Basics	19
2.4.1 Android Architecture	19
2.4.2 Android Storage Options	20

2.4.3	Interprocess Communication	21
2.4.4	Rooted Android Device	22
2.5	Android Devices	22
2.6	Penetration Testing	22
2.7	Related Work	23
2.7.1	Mobile Application	23
2.7.2	Threat Modeling	23
2.7.3	Penetration Testing	24
3	Threat Modeling	25
3.1	Modeling	25
3.1.1	Defined Assets	25
3.1.2	Defined External Dependencies	25
3.1.3	Defined Trust Levels	26
3.1.4	Defined Entry Points	26
3.2	Defined Data Flow Diagram	27
3.2.1	Level-1 Data Flow Diagram	28
3.3	Resulting Threat Model	34
3.4	Evaluation of The Threat Modeling Process	34
4	Penetration testing	37
4.1	Insecure Authentication and Session Management	37
4.2	Cross-Site Scripting (XSS)	38
4.3	Insecure Data	40
4.4	Insufficient Cryptography	40
4.5	SQL Injection	41
4.6	Validation	41
5	Discussion and Conclusion	43
5.1	Using STRIDE	43
5.2	Future Work	43
5.2.1	Communication Protocols for Sensors	43
5.3	Conclusion	44
	Bibliography	45
	A Appendix	I
	B Appendix	XV

List of Figures

1.1	Basic graphical overview of the system	4
2.1	Visual representation of the basic elements used to create data flow diagrams	10
2.2	Simple data flow diagram of an example system	13
2.3	Example of an attack tree listing the threats against a physical safe .	14
2.4	Generic process for threat modeling	17
2.5	Iterative process of threat modeling that should be conducted throughout development of the system	17
3.1	Conceptual level-0 data flow diagram of the system	28
3.2	Level-1 data flow diagram of the system	29
4.1	Screenshot of the first page in the web-interface with the fields for input highlighted	38
4.2	Screenshot of the subpage Group/Create new group in the web-interface with the field for input highlighted	39
4.3	Screenshot of the results of HTML code insertion	39
4.4	Screenshot on part of the session cookie	39

List of Tables

2.1	Example of different trust levels	8
2.2	Description of the different elements used to create data flow diagrams	10
2.3	Description of the STRIDE mnemonic	11
2.4	Microsoft's version of STRIDE-per-Element	12
2.5	STRIDE-per-Interaction: Threat applicability per interaction for each element	13
2.6	Description of different common mitigation strategies	16
3.1	Description of the defined assets	26
3.2	Description of the defined trust levels	27
3.3	Description of the defined entry points	30
3.4	Summarization of all the processes and a subset of both the data stores and interactors. Each entry describes their respective purpose and how they interact with the rest of the system.	31

1

Introduction

The number of internet connected devices is becoming increasingly prevalent and have become an essential tool in the modern world. Ranging from "smart homes" to sensors that monitor and control electro-mechanical systems. Including computers and network capabilities in products and systems gives the potential to lower cost, increase performance and provide new functionality. As an example, in a "smart home" today everything from the lights to the curtains and to the dishwasher can be connected to the internet and controlled remotely.

A substantial amount of these devices are used to monitor and control electromechanical systems. The spread of the domain of these devices have risen computer security and network security concerns. Researchers have shown, in for example [1], that there exist multiple security problems and vulnerabilities in these internet connected device systems, also referred as Internet of Things (IoT). Similar problems can with high probability be found in other areas, where internet connected devices have been rapidly introduced and where security has not been considered a primary factor in the design.

To aid developers develop secure systems already from the beginning, threat modeling is often used. Threat modeling is a structured approach used to analyze the security by systematically identify and rate all the threats that are applicable to the developed systems. The core concept of the threat modeling process is typically divided into three different stages. First, create an architectural overview of the system in order gain an extensible amount of knowledge in how the system operates. Second, utilize one or several different threat generation methodologies to generate and rank all applicable threats to the developed system based on the architectural overview. Lastly, determine the countermeasures and mitigation techniques that will be used to address the applicable threats.

When analyzing the architectural overview it is important to know which platform the system will run on. This is because it might exist platform specific threats that will not show up among general system threats. For new systems it might not exist a threat model that covers all its potential threats and the developers must look for known threats against that kind of system.

1.1 Research question

The research question of this thesis is how a threat model could be applied to a smartphone/IoT system, since there exists limited research about threat models for such systems. Further, it addresses how well a traditional threat modeling process works and how it may be improved.

1.2 Description of work

Below is found a short description of the work split into two major components; Threat Modeling and Penetration Testing.

1.2.1 Threat Modeling

In this project a threat modeling will be conducted and applied to an existing smartphone/IoT system which is developed by one of Cybercoms customers. The system will be analyzed in all the different stages of the general threat modeling process: decompose, identify and validate. The decomposition of the system will be done by manually testing the system, reading the manual and having discussions with the developers. In order to identify all the applicable threats to the system, common threat generation techniques will be used. These results will then be compared and complemented with lists of platform specific threats created by accredited sources to be able to compare what the threat modeling process missed and also to create a more complete threat model. In addition, the threat modeling process will be evaluated for possible limitations and how it could be improved when applied to a smartphone/IoT system.

1.2.2 Penetration Testing

In order to validate a subset of the threats found in the threat modeling process, some penetration testing will be performed. The penetration testing will be carried out by using existing tools recommended from accredited sources in conjunction with performing some static code analysis. The penetration testing on the system will not be exhaustive as only a subset of the different threats generated from the threat modeling will be assessed.

1.3 Contribution

Since the area of threat modeling for smartphone/IoT systems is relatively new, the scope of the work started from the beginning of a threat modeling process and ended with a contribution to the on-going research on threat modeling for mobile platforms. The contribution consists of how a working threat modeling process for a smartphone/IoT systems could be improved by utilizing complementary threat libraries from well accredited sources.

An investigation and evaluation of a methodological approach of an enhanced threat

modeling process aimed towards smartphone/IoT system with focus on the Android™ platform will be presented. The model could serve as a foundation for future research aimed towards threat model for other platforms or a generic threat model for all mobile platforms.

1.4 System Overview

The purpose of this system is to allow employees working in the industry to collect and analyze the status of different industrial machines. The system utilizes different types of small sensors to gather measurements such as temperature readings which may be sent to expert analyst for further analysis. The goal is to make the maintenance procedure more efficient. The system consists of a central mobile device running either the Android or Apple operating system having a special developed system application installed. The application is communicating to a backend server through either the device Wi-Fi or cellular connection. The system can, through the mobile device, connect to several pre-authorized sensors that can provide the application with measured data, via USB, Bluetooth or Wi-Fi.

The backend server hosts a web interface where the system administrators can upload and assign work orders. A work order is a set of instructions that a user should perform when conducting maintenance. The work orders may include taking pictures for documentation, retrieving sensor data or answering questions in text. Once a user has fulfilled his work orders he can upload a report containing all the answers to the backend server which automatically compiles the results and forwards it for analysis.

The users have personal user credentials that they can use to login in the application on the mobile device. When a user logs in to the application it checks the server for new work orders assigned to that user, which can also be done manually within the application. As long as the work order have been downloaded on the mobile device the work of fulfilling it can be conducted offline. However, it is necessary to go online to upload and synchronize the reports to the backend server. Figure 1.1 illustrates the system overview.

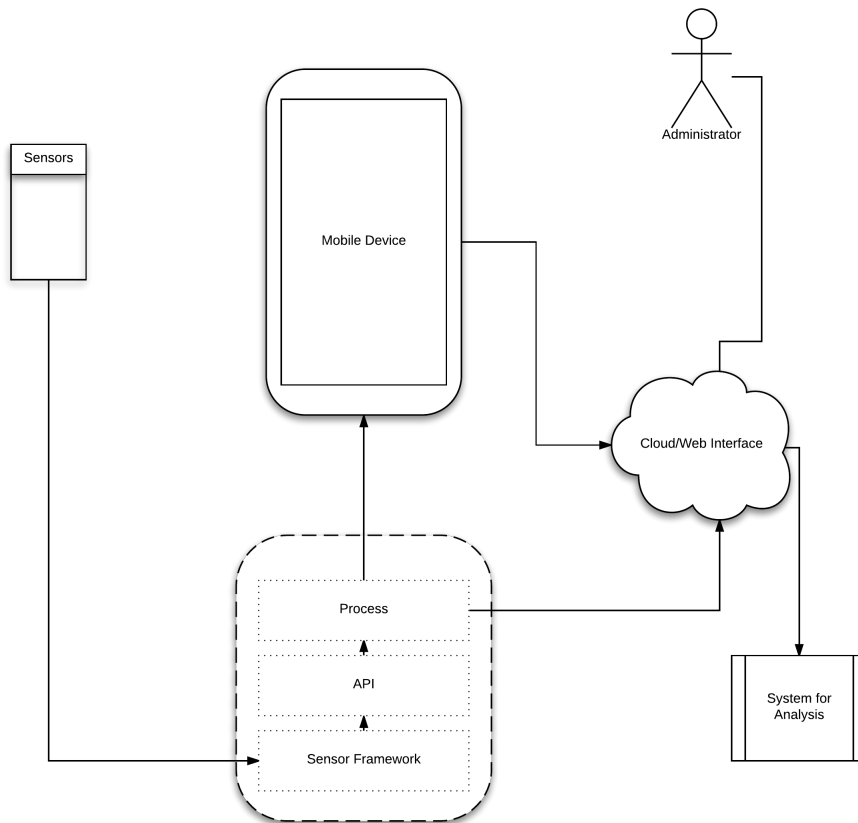


Figure 1.1: Basic graphical overview of the system

1.5 Limitations

Below are the limitations made while conducting this thesis.

1.5.1 Android

The mobile application developed for this system is available for both Android and iOS™. Both applications are identical in functionality and features. This thesis will however only cover the Android applications for several reasons. First, Android has an easily accessible file system and it is fairly easy to acquire root privileges since it is based on a Linux system. Second, compared to iOS, Android is an open platform with more freedom for the programmers but also weaker against malicious attacks [2]. Therefore, it is more likely to be targeted by an attack. Lastly, since Android has a larger market share [3] and is more widely used on the commercial market it is even more likely to be targeted by malicious behavior.

1.5.2 Sensor Penetration Testing

The interaction between the mobile application and the sensors will all be part of the security analysis and threat modeling but will not be part of the penetration

testing. This is because the configuration of the system application is to only establish a connection to known sensors and close the communication channel whenever the desired data have been received. In addition, the system implements numerous different types of communication channels when communicating with a sensor. Therefore, conducting penetration testing on all the different channels are considered to be out of scope for this thesis.

1.5.3 Risk Assessment

Risk assessment is an important part of any threat modeling process but in this report only the theory behind it will be explained. This work will later be done by the system owners since in this work it is not possible for us to classify the impact that the threats and vulnerabilities could impose.

1.5.4 Revised Copy

This copy of the report has been revised by Cybercom's customer, the system owner who wishes to be anonymous, where part of the text and some of the figures have been removed or changed due to confidential information.

1.6 Thesis Outline

The outline of the report is structured as follows; Chapter 2 consists of the background that is relevant for this project and the related work used. Chapter 3 describes the methodology and results of the threat modeling process. Then Chapter 4 discusses and presents the results of the penetration testing. Lastly, in Chapter 5 the results in general and their importance for future work are discussed. Finally the thesis is concluded.

2

Background and Related Work

To get a better understanding of the thesis results, information related to the background and terminology of threat modeling, OWASP, Microsoft Threat Modeling Tool, Android and penetration testing is presented. The chapter is finalized with a section of related work.

2.1 Threat Modeling

Threat modeling is an approach used for analyzing the security of a computer system or a software application. The process of threat modeling is a structured approach used in order to identify, quantify and address all possible threats applicable to the computer system or the software application [4]. First, this section will cover basic concepts regarding threat modeling. Second, descriptions of the threat models and threat generation related to this project will be given. Lastly, a description of the general procedure of a threat modeling process will be described.

2.1.1 Assets

An asset is defined as a system/application resource which are inherently important to the system/application design. In relation to threat modeling it is impossible to have a threat without a corresponding asset, because assets are essentially threat targets. An asset can be viewed as either a concrete or an abstract resource which needs to be protected against being misused by an adversary [5]. A concrete asset could be a specific process or data collection while an abstract asset could be data consistency or organizational reputation [4]. However, a distinction regarding what an asset is needs to be defined. An asset is a resource that needs to be protected and not the mean used to protect that resource. Hence, a password is not necessarily considered to be an asset as the purpose of the password is to protect other sensitive data, but in some cases it might be considered an asset.

2.1.2 Entry Points

Entry points define the interfaces that an adversary can utilize to interact with the system/application. Entry points can be divided into both external and internal entry points [6]. An external entry point is used to provide access to external users or components (e.g. open port for remote access). Whereas, internal entry points may be used for intercommunication between different internal components in the

system/application, but have no interaction with external users (e.g. database access for authentication). Even though internal entry points may not be as extensively exposed as the external entry points, they still need to be well documented in the case of an adversary that manages to bypass the first layer of defense and therefore gets direct access to the internal entry points.

2.1.3 Trust Levels

Trust levels are assigned to the different entry points to define the level of privileges an external user or component needs in order to access and interact with the system [7]. Trust levels are categorized according to the privileges or access rights needed in order for an external user or component to access and interact at each entry point, as well as the requirements needed to interact with each asset. Table 2.1 shows an example of different trust levels which then is used to cross-reference to different entry points and assets.

Trust Levels	
Name	Description
Anonymous User	User who has not provided valid user credentials
User with valid credentials	User that has provided valid credentials
Administrator	Application or system administrator

Table 2.1: Example of different trust levels

2.1.4 Extended CIA Triad

CIA in this case is a security model which stands for Confidentially, Integrity and Availability. It is developed to help individuals when they look into important aspects of IT security. A security measure is often developed to protect one, or sometimes more, parts of the CIA triad [8]. It also exists an extended model of the CIA triad since the original one is not considered sufficient and further security measure needs to be included [9].

Below is the description of the extended model:

- **Confidentiality:** Confidentiality means protecting the information from unauthorized access and parties. This means that the information intended to be kept secret, stays secret. Often this involves separating information into collections based on who should have access (Authorization) and how sensitive that information is. One example of confidentiality on an individual system is the UNIX file system permissions.
- **Integrity:** To preserve integrity the information must be protected from unauthorized modifications or deletions and also allow for the authorized changes to be logged (Authenticity) or reversed if it was a bad change that harmed the system. Cryptography and hashmaps are normally used to ensure integrity. An important part of integrity is **Non-repudiation** which implies that one party of a transaction can never deny its involvement in that transaction.

- **Availability:** To be secure, the system, access channels and authentication mechanisms must be available at all times when needed. Denied access to information is today one of the most common attacks, e.g. DDoS. A distributed partly off-site system is a good practice to preserve availability.
- **Authentication:** Authentication is the process of verifying a claim of identity and how that is proven. In computer system this is usually done by comparing the provided credentials to those in an authentication server.
- **Authorization:** Authorization process is the process of giving someone permission and access to do something that they are authorized to do. This is often seen in the form of your privileges in a computer system.

2.1.5 External Dependencies

External dependencies are components that the system/application is interacting with but will neither be included in the threat model as they are components that the developers do not have any, or limited, control over. An example could be a web server that interacts with a remote database not controlled by the server administrator to retrieve or send data. It is important to model this communication to analyze how the web server handles the incoming or outgoing data, but not necessarily to model the database as the administrators do not have explicit control over the database. Another type of external dependency is that a specific system/application may be designed to run in a specified environment where certain requirements need to be fulfilled (e.g. communication only over a private network) [4]. Therefore, external dependencies are very important as they provide critical information about the components that are being modeled and is also extensively used when creating data flow diagrams which is covered in Section 2.1.6.

2.1.6 Data Flow Diagram

Data flow diagrams is a very common modeling tool when it comes to threat modeling [10]. The data flow diagrams allow a system designer or reviewer to gain a better understanding of the system/application by providing a visual representation of how the system/application processes data. The main purpose of data flow diagrams is to show how data moves through the different system/application components and how the data is affected during transit [4]. Data flow diagrams are structured in a hierarchically way in order to allow the system/application to be decomposed into one or several subsystems, each of them covering one specific part which allows for a more detailed overview. The hierarchically structure is most commonly implemented by creating multiple data flow diagrams. Each diagram contains increased detail on certain parts of the system/application. These diagrams are commonly labeled with the word "*level*" paired together with a numeric number to indicate the level of detail in the diagram. Typically a level-0 data flow diagram is referred as a conceptual diagram that gives an overview of the complete system but with very limited information. In order to represent the different components of the system/application, data flow diagrams utilizes basic building blocks referred to as

2. Background and Related Work

elements. Table 2.2 gives an overview of the basic elements used to create data flow diagrams and Figure 2.1 gives a visual representation of how the elements may look [4][10].

Data Flow Diagram Elements			
Type	Appearance	Description	Examples
Process	Circle, rounded rectangle or concentric circles	All running code	Software written in C, Python, Java etc
Data Flow	Arrow	Communication or data flow between different elements (processes and data stores or external interactors)	HTTP, UDP, IPsec etc
Data Store	Two parallel lines	Medium that stores data	Files, databases, Shared memory etc
External Interactor	Rectangle	Human interactors, processes/databases with no control over	External DNS-servers, databases or remote users etc
Trust Boundaries	Dotted lines	Represent the change of privilege levels as the data flows between different elements	Change in trust levels etc

Table 2.2: Description of the different elements used to create data flow diagrams



Figure 2.1: Visual representation of the basic elements used to create data flow diagrams

2.1.7 STRIDE

STRIDE is an acronym that stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. The STRIDE approach to threat modeling was invented by Loren Kohnfelder and Praerit Garg in 1999 for use in Microsoft software development [10]. The STRIDE mnemonic was developed

with the purpose of helping developers to identify and understand the different computer security threats and potential vulnerabilities through categorization. The STRIDE model provides examples of typical vulnerabilities that may be exploited by different attacks in order realize a threat [11]. Table 2.3 provides a description of the mnemonic.

The STRIDE Acronym			
Threat	CIA Violation	Definition	Targets
Spoofing	Authentication	Spoofing is where an entity successfully manages to pretend to be someone/something other than itself	Processes, external interactors, users etc
Tampering	Integrity	Modifying data without valid permission	Data stores, Data flows, processes etc
Repudiation	Non-repudiation	Unsuccessfully associating certain malicious actions or changes to a unique individual	Processes
Information Disclosure	Confidentiality	Information leaked to unauthorized parties	Processes, Data stores, Data flows
Denial of Service	Availability	Unable to provide intended functionality to intended users	Processes, Data stores, Data flows
Elevation of privileges	Authorization	Allow an entity to perform certain actions that require higher privileges beyond those initially granted	Processes

Table 2.3: Description of the STRIDE mnemonic

According to A.Shostack [10], there exists two prominent variants of STRIDE; STRIDE-per-Element and STRIDE-per-Interaction which can be used to further enhance the efficiency and the level of detail of the threat generation.

STRIDE-per-Element

STRIDE-per-Element puts emphasis on observing which typical kind of threats are more prevalent to the different elements of the data flow diagram. By focusing on a specific set of threats applicable to each element it makes the process of finding the threats easier. Table 2.4 shows how Microsoft applies the STRIDE-per-Element variant. For example, when analyzing a system/application using this chart the focus will be on *Spoofing* and *Repudiation* related threats when considering an external interactor, while the other threats will not be considered in detail for that specific element. This allows for an efficient process in generating applicable threats. Note, Table 2.4 will not be a good representation of every system and should instead be tailored to fit the specific system/application that is being modeled. However, assigning more types of threats to each of the elements results in that the analysis will be applying original STRIDE for each element. In terms of comprehensiveness, this is favorable but the purpose of STRIDE-per-Element is lost.

Element	S	T	R	I	D	E
External Interactor	X		X			
Process	X	X	X	X	X	X
Data Flow		X		X	X	
Data Store		X	X	X	X	

Table 2.4: Microsoft's version of STRIDE-per-Element

STRIDE-per-Interaction

STRIDE-per-Interaction puts emphasis on all the interactions between different elements in the data flow diagram in order to enumerate possible threats. Tuples consisting of origin, destination and interaction are taken into consideration when enumerating possible threats for each interaction. This variant of STRIDE was developed by Larry Osterman and Douglas MacIver, both working for Microsoft at that time. The purpose of this variant is to reduce the amount of information the developers needs to consider when conducting threat modeling. However, this approach yields as many threats as STRIDE-per-Element and as stated by A.Shostack in [10] the threats may be more easily understandable. In contrast to STRIDE-per-Element the use of tables is essential when working with STRIDE-per-Interaction due to the increased complexity and amount of information that needs to be documented. Commonly these tables consist of:

- **Number:** Number used for referencing to each line
- **Element:** The element of the data flow diagram which is being considered
- **Interaction:** What kind of interaction the element have, e.g. outgoing data flow to a data store

- **STRIDE mnemonic:** What parts of the STRIDE mnemonic affect this particular interaction

Figure 2.2 shows a simple example of a data flow diagram of a system and Table 2.5 shows the corresponding table used for the STRIDE-per-Interaction process. Once each interaction have been defined for each element the table is further improved. Instead of only showing what threats that is applicable to the system it instead gives a description of possible threats in detail. When comparing Table 2.4 to Table 2.5 it shows that STRIDE-per-Interaction gives a higher level of detail and is more structured during the threat generation process in comparison to STRIDE-per-Element but instead have increased complexity.

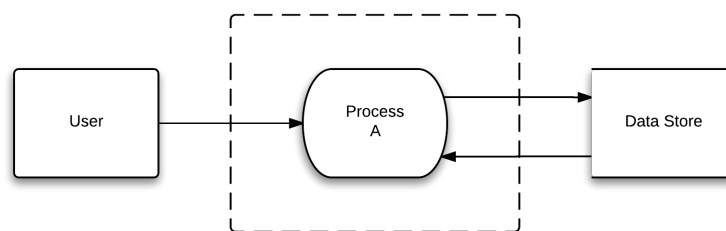


Figure 2.2: Simple data flow diagram of an example system

Number	Element	Interaction	S	T	R	I	D	E
1	Process A	Inbound data flow from User	X			X	X	
2		Inbound data flow from Data Store	X	X			X	X
3		Outbound data flow to Data Store	X			X		
4	Data Store	Outbound data flow to Process A			X	X	X	
5		Inbound data flow from Process A		X	X	X	X	
6	User	Outbound data flow to Process A	X		X	X		

Table 2.5: STRIDE-per-Interaction: Threat applicability per interaction for each element

2.1.8 Attack Trees

An attack tree is a simple but methodical overview model of the security of a system. It has a tree structure where the root is always the goal of the attacker and the children are different ways of achieving that goal. You either need one child or more combined to achieve the parent node, this is often noted with “AND” and “OR”

in the graphical overview where “AND” is written out otherwise “OR” is assumed. Attack trees can be used to list attacks against any system, as the Figure 2.3 by Bruce Schneier for a safe shows, but is today increasingly used in computer systems.

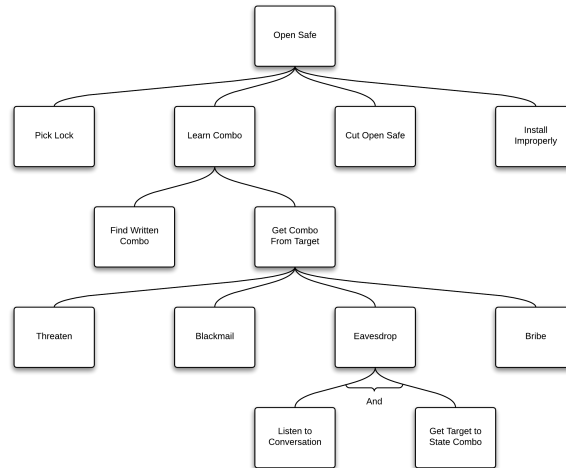


Figure 2.3: Example of an attack tree listing the threats against a physical safe

In addition, an attack tree can also be used to rate the different ways to achieve the goal. By performing such analysis it is possible to determine which way is the most probable to use in order to realize the root goal. Bruce Schneier gives a few examples in [12] where he rates each children with possible and impossible to see which ways that makes the goal achievable. In another example he calculates the price of each step and can then easily see which way is the cheapest in order to achieve the root goal. This also gives an overview of where to apply the mitigation strategies.

2.1.9 Risk Assessment

Risks is assessed by first identifying all threats and vulnerabilities. Then determine the impact of a specific vulnerability and the likelihood of it occurring [13]. Then repeat the process for each known vulnerability applicable to the system/application. There exist two main categories of risk assessment, namely, quantitative and qualitative risk assessment.

Quantitative Risk Assessment

Quantitative risk assessment is the most exhaustive, costly and time consuming approach of performing a risk assessment. It uses methodologies commonly used by financial institutions and insurance companies when computing the risk factor. However, the most prominent benefit is identification of your greatest risk based on the financial impact of the organization. In short, quantitative risk assessment assigns an economical cost associated with each vulnerability and threat realization.

Qualitative Risk Assessment

In Qualitative risk assessment, the assets are defined and reviewed for known vulnerabilities and then measured against relative scales to determine the probability of a threat realization. Similarly to quantitative risk assessment, the definition of impact and likelihood is essential and most often a very complex task. However, instead of assigning an economical value to impact and likelihood, qualitative risk assessment uses a predefined non-numerical value. Typically values used are *Low*, *Moderate* and *High*.

2.1.10 Threat Modeling Process

Figure 2.4 represents how threat modeling can be decomposed into three general steps, namely, *Decompose the application*, *Determine and rank threats* and *Determine countermeasures and mitigations* [4].

1. **Decompose The Application:** The first step in the threat modeling process involves decomposing the application into its basic building blocks and functionalities in order to define the *Assets*, *Entry Points*, *Trust Levels* and *External Dependencies*. Lastly, data flow diagrams are created to express the data flow and trust boundaries within the application.
2. **Determine and Rank Threats:** The second step in the threat modeling process is to determine all possible threats applicable to the application and then rank the threats in order of prioritization. Threat categorization models such as STRIDE is commonly used in conjunction with the data flow diagram created in step one to generate all possible threats. Second, threat trees may be created to give further detail about the threats and corresponding vulnerabilities. However, in certain scenarios the creation of attack trees is not applicable due to the level of complexity of the system and thus is left out. Lastly, each threat is assigned an appropriate risk factor which can be estimated using either a qualitative or quantitative risk assessment model depending on the organizational needs.
3. **Determine Countermeasures and Mitigations:** Third step in the threat modeling process involves deciding upon appropriate countermeasures and mitigation strategies to use based on the prioritization of threats conducted in step two. Table 2.6 gives a description of common risk mitigation strategies [10] [13]. Decision of what strategy to use depends on the impact of exploitation and the likelihood of occurrence of the threat as well as the economic impact of adopting a certain mitigation strategy. It is common to only mitigate the vulnerabilities for which the cost to avoid, transfer or address the problem is less than the potential business impact derived by the exploitation of the vulnerability.

Risk Mitigation Strategies	
Mitigation	Description
Avoiding Risks	Avoiding a potential risk entirely by changing the requirements or system design. Assess whether a potential risk is greater than the potential reward
Addressing Risks	By adding appropriate countermeasure to prevent threat realization. E.g. adding cryptography over an insecure data flow
Accepting Risks	Threat is acceptable under the organization boundaries. The organization accepts the risks and takes full responsibility after a threat realization. Accepting risks are common when the impact is very minor or the likelihood of occurring is minuscule
Transferring Risks	Transferring risks is the process of allowing another party to accept the risks on your behalf
Ignoring Risks	Ignore the threat and risk entirely. Basically hoping for the best. Used to be a traditional approach before new regulations
Warn About Risks	Send out warnings about risks and threats to the intended users

Table 2.6: Description of different common mitigation strategies

Threat modeling is designed to aid the developers to find possible threats in the early stages of the development. However, this process should not be performed only once but instead be considered an iterative process throughout the complete development life cycle of the system/application (see Figure 2.5) [6]. In addition, the documentation created by the threat modeling process allows both internal and external security reviewers to get a greater understanding of the architectural design which allows for better prioritization of which parts to review in detail.

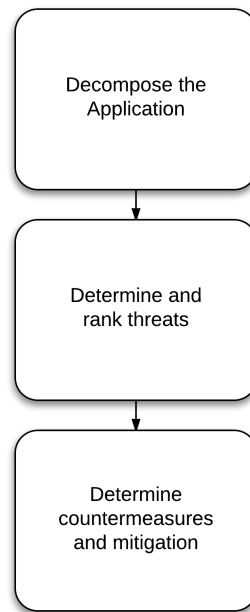


Figure 2.4: Generic process for threat modeling

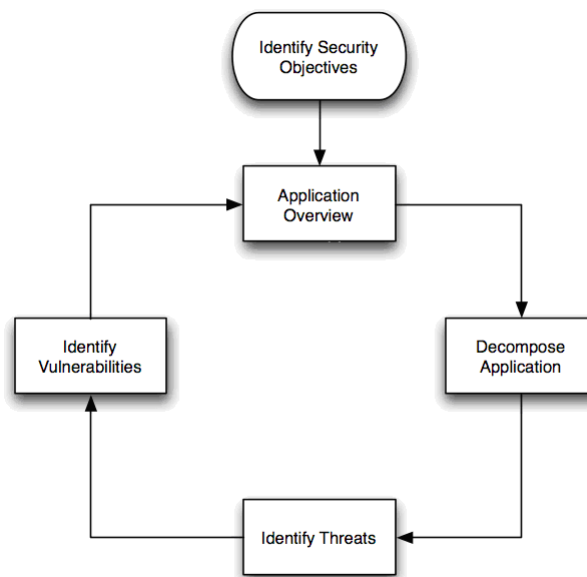


Figure 2.5: Iterative process of threat modeling that should be conducted throughout development of the system

2.2 OWASP

The Open Web Application Security Project, OWASP, is a multinational non-profit organization whose focus is to improve the security in software. Their main focus is to make software security visible so that anyone can be up-to-date with the latest

flaws. Since 2003 they have published "Top 10 Security Risks" lists which in the beginning consisted of attacks, vulnerabilities and countermeasures. In 2007 the main focus was on vulnerabilities and since 2010 also security risks [14]. A new version is released about every three years and the list for 2016 have not yet been released so the OWASP Top Ten references in this report will be based on the 2013 report. OWASP has received a lot of attention and is among other recommended by IBM [15] and CISSP[16].

2.2.1 OWASP Mobile Security Project

One branch of the OWASP project is their Mobile Security Project which is a centralized resource intended for developers and security experts to use both during development and in security assessments of mobile applications [17]. The focus of the project is to gather and maintain information, as well as classify mobile security risks and provide developmental controls in order to reduce the consequences and probability of exploitation of threats.

Similarly to the OWASP Top Ten security risk for regular web applications. The OWASP mobile security project have created a list that focuses only on the top mobile risks.

The list for 2015 contains the following threat categories:

1. **Improper Platform Usage:** This category covers non intended use of certain platform specific features or failure to follow the platform security policy. It might include misuse of the Android explicit and implicit intents, platform permissions and misuse of the different storage options or other security control that is part of the mobile operating system.
2. **Insecure Data:** This category covers insecure data storage and unintended data leakage. This is very apparent on the Android platform since it is an open platform based on Linux.
3. **Insecure Communication:** This category covers all aspects of insecurely transferring data from A to B. It includes mobile to mobile communications, application to backend communications or mobile to anything else communications. That includes all communications technologies and protocols that a mobile device might use: TCP/IP, WiFi, Bluetooth, NFC/RFID, GSM, 3G, SMS, etc.
4. **Insecure Authentication:** This category covers all notions of authenticating the users and bad session management. This also includes authentication over insecure channels.
5. **Insufficient Cryptography:** This category covers the implementation of insufficient cryptography. Meaning when cryptography was attempted but for some reason was not done correctly.
6. **Insecure Authorization:** This category covers any failures in authorization, for example authorization decisions in the client side or forced browsing. A

common threat is to search for unlinked contents on a website

7. **Client Code Quality Issues:** This category covers code-level implementation problems in the mobile client, which is distinct from server-side coding mistakes.
8. **Code Tampering:** This category covers code tempering such as binary patching and local resource modification. This means that when the application is delivered to the mobile device the attack can modify the code, change memory contents and replace or modify the system API.
9. **Reverse Engineering:** This category covers the reverse engineering of the final core binary to determine its source code, libraries, algorithms and assets.
10. **Extraneous Functionality:** This category covers the fact that developers sometimes include hidden backdoor functionality or other internal security controls meant for development. These were never intended to be released into the final product.

In addition, the mobile security project have created a checklist containing information about the top relevant mobile threats including platform specific ones. This checklist is aimed to be used as a foundation of any mobile application development [18]. The checklist will serve as the complementary information used in conjunction with the threat modeling process described in Section 2.1.10.

2.3 Microsoft Threat Modeling Tool

Microsoft Threat Modeling Tool is a tool used in the Microsoft Security Development Lifecycle (SDL) which helps the developers to create an overview and analyze their systems for vulnerability and threats. The tool supports the feature to allow the developers to graphically create data flow diagrams by dragging elements from a stencil and then connect how they interact. The tool can then analyze the model for threats, using STRIDE-per-interaction, and return a report where all parts of the model is analyzed and potential threats and vulnerabilities is listed for each interaction [19].

2.4 Android Basics

This section will cover some fundamental concepts about the Android operating system in regards to security which will be used and discussed throughout this report.

2.4.1 Android Architecture

The Android operating system is built upon the Linux kernel and thereby inherits its core principles of the UNIX process isolation, least designated privileges and user permissions [20]. Android applications are commonly written in the Java programming language and, together with other resource files, compiled into an *Android*

Package File (APK file) which is used in order to install the application on a device running Android [21]. During compilation of the APK file, all the files containing code is further compiled into *Dalvik Executable Files* (dex files) which is executed during runtime by either the *Dalvik Virtual Machine* (VM) or the *Android Runtime* (ART). The Dalvik VM is the predecessor to the ART and is used in all Android versions up to and including version 4.4 while the ART is included in version 4.4 and will be the standard runtime environment in newer versions [22] [23].

In addition, Android utilizes a technique referred to as the *Application Sandbox* for its process isolation where each application gets assigned a dedicated part of the memory and file system. This restricts any access and results in isolation of application data and code execution to any other user interaction and application installed on the device [24].

2.4.2 Android Storage Options

Android provides numerous options when it comes to storing persistent data related to the applications. Depending on the requirements and specific needs, such as privacy or space requirement, each option will be either more or less suitable [25].

The following options are available:

- **Shared Preferences:** The shared preferences options allows developers to store and retrieve persistent key-value pairs of primitive data types, e.g. integers, strings, booleans. This information will be stored across user sessions even if the application have been terminated in between. Shared preferences will in addition not be publicly available but will be protected and contained within the application sandbox and only be accessible to the application storing the data.
- **Internal Storage:** Similarly to shared preferences, the internal storage option only allows access within the application sandbox and thus this option can be considered private to the application. However, it is not restricted to primitive data types and therefore allows storing of any file types, e.g. pictures, text files, cache files.
- **External Storage:** The external storage will store the application data publicly available to any user or application that wants to read this data. Therefore, external storage is not suitable for storing sensitive data which needs to be protected from both other applications as well as other user interactions, such as accessing the data via computer interactions or file browsing applications.
- **SQL Database:** Android provides full support of SQL databases which is ideal for storing structured data, such as contact and user information. The SQL database is stored within the application sandbox and thus is not publicly available to any other user or application interaction.

2.4.3 Interprocess Communication

Android provides several options for handling interprocess communication (IPC) in order to allow the developers to share data and information between the processes within an application or share data to an external application [24]. These IPC mechanisms allows the developers to both verify the identity of the destination process as well as apply certain security policies on the IPC. In addition, it is possible to define whether an IPC should be accessible outside of the *Application Sandbox* or be contained within it.

The following common and suggested options available are:

- **Intents:** An intent is a messaging object which is used to initiate certain actions within an application [26]. Even though intents may be used in several ways for communication between specific application components there exists three fundamental use cases. Namely, deliver a broadcast, start an activity (an activity represents a single screen in an application) and start a service (a service is a set of actions to be performed in the background without user interaction):
 - **Deliver broadcast:** By using intents applications are able to broadcast information, such as alerting of an incoming SMS message or a change in the network state, which other applications then can use and perform specific actions, e.g. show the content of the incoming SMS message.
 - **Start an activity:** Using intents is a common way of changing in between different activities within an application. Whenever a new activity needs to be started (changing screen in an application) an intent is passed as an argument and may contain extra information that will be required in the new activity.
 - **Start a service:** Similar to starting activities, intents are also commonly used in order to start services. The services will run in the background without a user interface and complete certain computations or tasks that may be required by another activity, e.g. downloading a file.

There exist two types of intents with different characteristics and use cases:

- **Explicit Intent:** Explicit intents are used when the destination component is specified, e.g. starting an activity within the application. This results in that the communication will not be accessible by any component other than the destination target.
- **Implicit Intent:** Implicit intents in contrast to explicit intents do not specify the target destination component, but instead relies on a feature called *intent-filters*. This type of intent is commonly used to perform specific actions that have not been implemented within the application. But instead outsources this functionality to another application. One typical use case is to send out an implicit intent to find any application with a matching intent-filter that, for example, implements a camera functionality.

- **Binders:** Using binders is the preferred way of handling remote procedure calls (RPC) in Android applications. They provide a well-defined interface that implements mutual authentication if required. Binders are typically used to invoke certain actions of different components, instead of just sharing data between them.

2.4.4 Rooted Android Device

Rooting an Android device refers to the process of modifying the operating system to allow the user and applications to gain elevated root privileges. This level of privileges is similar to the Linux super user and allows both the user and application to perform operations which would be prohibited. This results in full control of the operating system [20]. In addition, by rooting a device, it allows the user to both uninstall pre-installed system application and install applications that requires root privileges, such as file browsing applications that can read and write private data protected by the application sandbox.

2.5 Android Devices

During this project two different Android devices were used, one Samsung Galaxy Tab Active (SM-T360) and one LG Nexus 5, which was rooted. The Samsung device has Android 5.1.1 installed and the LG Nexus 5 Android 5.0.1. Both devices have the same version of the application installed (2.5.1).

2.6 Penetration Testing

In addition to the OWASP Top Ten mobile risks described in Section 2.2.1 the following two risk categories from the regular OWASP Top Ten security risks is included and described below, as they are part of the penetration testing of the backend server.

- **Cross-site Scripting (XSS):** XSS against web applications uses known vulnerabilities in the application itself, their servers, or the plug-in systems where they insert scripts to activate these vulnerabilities. The two most common types are reflected and persistent scripts. Reflected script uses vulnerabilities in the web client, often in its HTTP query parameters where server-side scripts immediately parse them and display the result. Persistent scripts is when the script is saved by the server and can thus be executed on other user sessions.
- **SQL Injection:** SQL injection is a code injection technique used by malicious users to inject SQL queries in data-driven applications, commonly in web application input fields. A successful SQL injection can return sensitive data from the database, modify the database or execute administration operations. In OWASP Top Ten security risks for web applications, injection attacks are ranked the number one threat to web applications.

The system has been checked with several OWASP recommended tools [27], including but not limited to:

- **Burp Suite Proxy** [28]: Burp is a proxy that can intercept and modify all HTTP and HTTPS traffic that goes in or out of the system.
- **Hydra** [29]: Hydra is a brute-force authentication tool supporting several protocols.
- **MITM Proxy** [30]: Man-in-the-middle proxy that can intercept and change HTTP and HTTPS messages and generate SSL certificates.
- **SQL Map** [31]: A tool that is used for detecting and exploiting SQL injection flaws, resulting in compromise of database servers.
- **The Browser Exploitation Framework** [32]: BeEF is a penetration tool that looks for vulnerabilities through the web browser and can launch directed command modules in the browser context.
- **Cross Site "Scripter"** [33]: XSSer is an automatic tool for detecting and exploiting Cross Side Scripting vulnerabilities

2.7 Related Work

In this section the related work for this project is presented. Because the project consists of several parts this section has been divided into three subsections.

2.7.1 Mobile Application

Security in mobile applications has received focus in the latest year and much academic research has been done on the subject. A.Kesäniemi talks in [34] about if we have enough protection in mobile applications, what data needs protection and the importance of attack trees. The security company DataTheorem presents in [35] threats towards the different mobile platforms, Android, iOS and Windows Mobile™ and threats in their core programming languages.

L.Dua et al. in [36] a lot of confidential and sensitive information is stored in mobile devices. The authors present that in 2013 98% of the mobile malware were targeted and at Android and that there exist increasing amounts of greyware, mobile applications that do not do any harm but instead collect information about you for example marketing purposes.

2.7.2 Threat Modeling

There exists a lot of research in the area of threat modeling and the research is still highly active. D.Dhillon conclude in [37] that data flow diagrams do not capture all the details to efficiently preform a threat model. To perform STRIDE a significant amount of security knowledge is needed. For these reasons the authors have developed a threat library with common weaknesses that can be identified early in

the design phase. R.Scandariato et.al is on the same track in [38], that traditional threat models need to be complemented in order to achieve their full potential. By presenting an evaluation of their privacy threat analysis methodology LINDDUN which is inspired by STRIDE. Two e-health systems and one smart grid system at which LINDDUN was applied to have shown high success rate and even beat the author's expectation.

J.Clark et al. [39] takes another approach in his paper about threat modeling for sensor networks and Internet of Things and lists different threat types, sorted in subcategories, and also how different threats to sensor networks might possess a threat towards, for example, confidentiality, integrity, authentication and availability. Instead of elaborating on STRIDE this paper elaborates on threats towards IoT systems in general.

2.7.3 Penetration Testing

N. Anutes et al. in [40] writes about the importance of carrying out penetration testing to ensure the security of your system and the constant threat of attacks against web services. The author's evaluates different testing tools while focusing on SQL injection, a very substantial threat for web applications.

A. Austin presents in [41] several ways to carry out manual or automatic penetration testing how many vulnerabilities each of them finds. The authors test the same system in different ways and find that, for example, Cross Site Scripting systematic manual penetration testing finds more vulnerabilities, but have a higher false positive rate, while the automatic tools finds less vulnerabilities but have a perfect score.

R.Scandariato et.al [42] continues in this discussion by comparing static and dynamic penetration testing and see which has the more favorable cost-to-benefit ratio. The authors find that the static tests produce more true positives in the limited amount of time the test was concluded in. It should also be mentioned that the persons carrying out the testing in this paper were returning master students at an American University.

3

Threat Modeling

This chapter contains the methodology and results of the threat modeling process implementation, including the data flow diagram and the evaluation of this process.

3.1 Modeling

The base concept of doing a qualitative threat analysis is having a thorough understanding of the system and a good data flow diagram.

As mentioned in Section 2.1.10 the first step to help understand the system is to decompose the system into its basic components, assets, entry points, trust levels and external dependencies. Several meetings were held early in the project with the systems developers to discuss the functionality of the system and how the data flows between processes. Due to lack of technical documentation the developers instead performed live demos as well as providing a copy of the API specification and a user manual. Based on this material all the assets, trust levels, entry points and external dependencies were defined and later used to define the data flow diagram.

3.1.1 Defined Assets

After discussions with the system developers all the assets for the system was defined and is summarized in Table 3.1. Even though the ticket can be seen as a password, and thereby a mean of protection, it was decided to be defined as an asset because of how it is implemented. If compromised, it would result in exposure of the company and sensor data.

3.1.2 Defined External Dependencies

The system interacts with two type's external dependencies, namely the sensors and the system for data analysis. The interactions between these dependencies and the system will be included in the threat model. However, the model will not include the architectural and logical design of these dependencies as there exist numerous types of sensors. The system for data analysis is not developed or maintained by the developers of this system.

Assets	
Name	Description
Sensor Data	Data collected from the sensors containing information and status on different machines
Company Data	Information regarding the company. E.g. Organization name, employees information, information about company inventory and user information
Backend Servers	Servers containing information about all companies and user information
Session Ticket	Used as authentication to the backend servers for the API requests when retrieving both company and sensor data

Table 3.1: Description of the defined assets

3.1.3 Defined Trust Levels

The system only allows a user with valid login credentials to use and interact with the system. However, a distinction needs to be made as depending on the entry points the users will have different privileges even though using the same account. A user with assigned administration privileges only have elevated privileges when accessing the web interface. But when logging in via the mobile application, there is no difference in privileges between a regular system user and a user with administrator rights. Table 3.2 summarizes the different trust levels defined for the system.

3.1.4 Defined Entry Points

The system consists of both internal and external entry points. The external entry points correspond to all the direct interactions with the users of both the mobile application and the web interface. In addition, another external entry point is the system interaction with the external sensors. The internal entry points corresponds to the interprocess communication types within the mobile application described in Section 2.4.3. Table 3.3 summarizes the defined entry points for the system.

Trust Levels		
Number	Name	Description
1	User without administrator privileges	Regular system user that only have access to the mobile application. This type of user cannot access and login into the web interface
2	User with administrator privileges	User with administrator privileges can login into the web interface to access additional system functionality. No additional privileges in the mobile application
3	User without valid credentials	User without valid credentials which is not allowed to utilize the system functionality
4	Backend Server Administrator	Administrator with local access to the back-end servers for configuration. E.g. via SSH

Table 3.2: Description of the defined trust levels

3.2 Defined Data Flow Diagram

As mentioned in Section 2.1.6 it is very common to create data flow diagrams when conducting a threat analysis to get a visual representation of the system. In collaboration with the system developers a first conceptual level-0 data flow diagram was created (Figure 3.1) based on the information described in Section 3.1.

In order for the threat model to include detailed information about possible threats a higher level data flow diagram was created that provides a greater overview and increased level of detail. It was decided that only one higher level diagram was sufficient in order to perform an adequate analysis, which is shown in Figure 3.2 and is described in detail in Section 3.2.1.

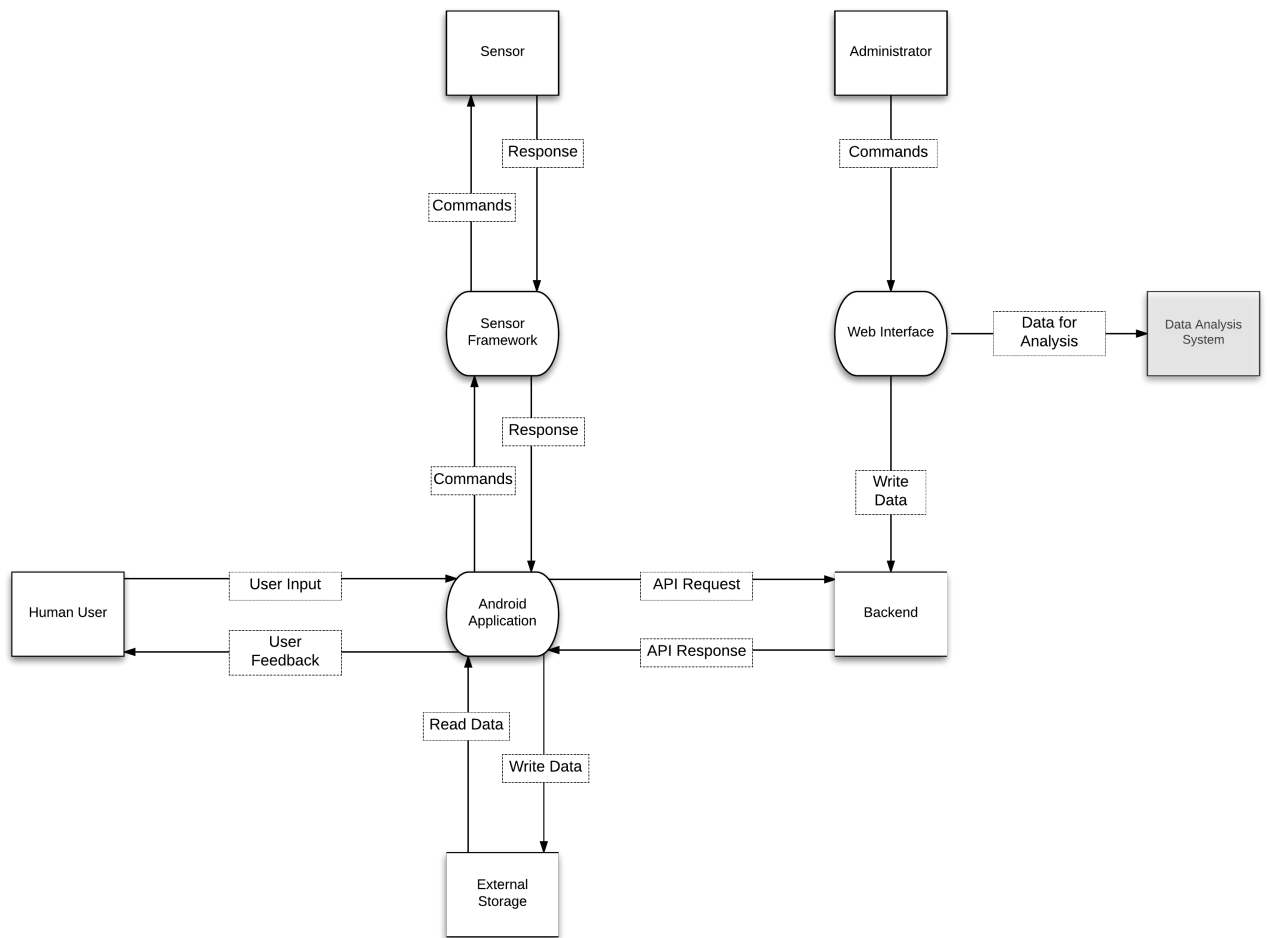


Figure 3.1: Conceptual level-0 data flow diagram of the system

3.2.1 Level-1 Data Flow Diagram

As the data flow diagram is the main building block when conducting a threat modeling process it is essential that the diagram includes all the necessary details and having these details documented as thoroughly as possible. Table 3.4 lists all of the processes and a subset of both the data stores and external interactors. It also gives an overview of their respective functionality and how each process interacts with the rest of the system components. All the data stores, except the backend, was left out of the table as they only have one outgoing interaction with the destination being the *Data Storage Process*. Their functionality is only to store data for later use whenever a process request the data.

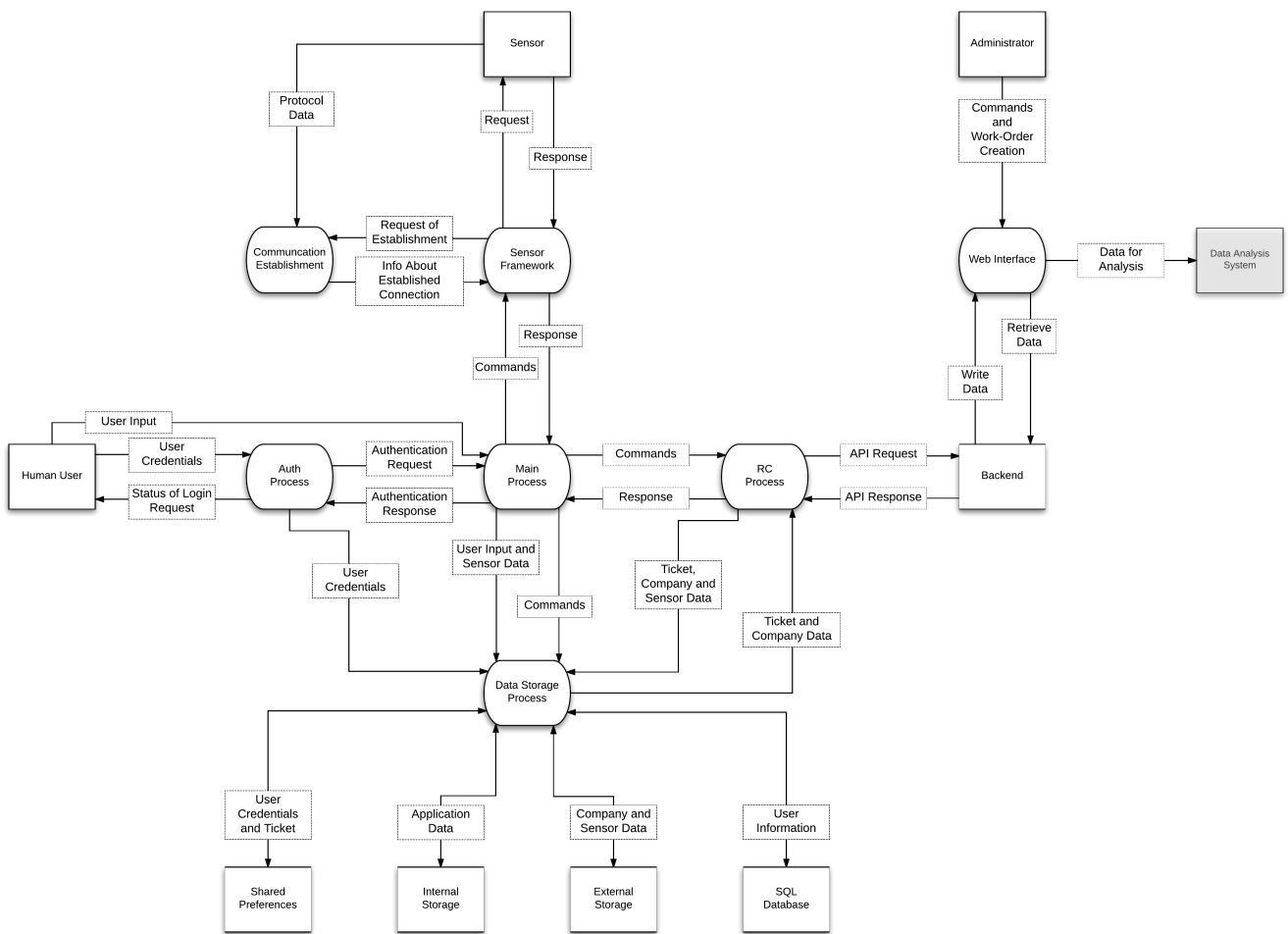


Figure 3.2: Level-1 data flow diagram of the system

Entry Points		
Name	Description	Trust Levels
Application login activity	Login fragment used to authenticate legitimate users	1) 2) 3)
Application main activity	The core functionality of the application where users can input data and answer the forms related to different work orders	1) 2)
Sensor pairing	The mobile application connects to an external sensor for measurements	1) 2)
Backend synchronization	The application sends API requests in order to synchronize data from the backend servers. E.g. new work orders, forms, change of API version etc	1) 2)
Web interface login page	Login page for the web interface used to define forms, work orders, users and user groups etc	1) 2) 3)
Backend server configuration	Remote or physical access to the servers used for configurations	4)
External Storage	Publicly available data for the device user used by the application	1) 2) 3)
Implicit intent for camera application	Implicit intent used to start a camera application to use in order to answer certain work orders	1) 2)
Broadcast receiver	One broadcast receiver listening on change in network states	1) 2) 3)

Table 3.3: Description of the defined entry points

Table 3.4: Summarization of all the processes and a subset of both the data stores and interactors. Each entry describes their respective purpose and how they interact with the rest of the system.

Level-1 Data Flow Diagram Description				
Process/ Data store/ Interactor	Description	Outgoing Interactions	Target Process/ Data Store	Description
Main Process	The process which is connecting all the components together. Its main purpose is to propagate the requests received from the other processes	Commands	RC Process	Sends what type of API request RC process should perform. E.g. validate credentials or synchronize data
			Data Storage Process	Sends what data RC process will need for synchronization
			Sensor Framework	Sends what type of sensor to request specified data from. E.g. sensor measuring temperature data
		Authentication Response	Auth Process	Propagate the response received from RC process based on previous authentication request
RC Process	Process handling all the API requests to the backend servers. E.g. credentials validation, synchronization of company and sensor data	API Request	Backend	Sending appropriate API request
		Response	Main Process	Response containing information about previous requests. E.g. credential validation
		Ticket, company & sensor data	Data Storage Process	Data received from backend sent for storage
Sensor Framework	Process handling all the sending and receiving data from the sensors	Request	Sensor	Requesting the data that was specified by the main process
		Request of Establishment	Comm. Establishment	Request sent in order to establish a connection to the specified sensor as specified by the main process

Continuation of Level-1 Data Flow Diagram Description

Process/ Data store/ Interactor	Description	Outgoing Interactions	Target	Description
		Response	Main Process	Sensor data sent as response based on previous request
Auth Process	Process handling the authentication requests of user requesting login to the application	Status of login request	Human User	User gets a response with the status of the login request
		Authentication Request	Main Process	Request sent for authentication to main process for propagation to RC Process
		User Credentials	Data Storage Process	User credentials sent for storage
Data Storage Process	Process handling all the reading and writing data to the different types of data stores available to the application	Ticket, Company & Sensor Data	RC Process	Data sent for synchronization to the backend
		User Credentials & Ticket	Shared Preferences	User credentials and ticket is sent for local storage on the mobile device
		Application Data	Internal Storage	Storing data used during runtime of the application. E.g. logging files
		Company & Sensor Data	External Storage	Storing sensor data and user input used to complete work orders
		User & work order Information	SQL Database	Storing user and work order information. What user answered which form and also list the answers
Web Interface	Process running the web interface where the administrator can configure the user privileges and create and modify work orders	Write Data	Backend	Writing to backend storage with updated information on user privileges and work orders
		Data for analysis	Data Analysis System	Data received from the sensors may be sent for further analysis conducted by experts. E.g. temperature data

Continuation of Level-1 Data Flow Diagram Description

Process/ Data store/ Interactor	Description	Outgoing Interactions	Target	Description
Comm. Esthb.	Process handling all the establishing of connections between the sensor and the application. E.g. completing the pairing process when using Bluetooth or setting up the USB communication	Protocol Data	Sensor	Protocol data used to establish a connection. E.g. session key generated after Bluetooth pairing is completed
		Information about connection	Sensor Framework	Sending information about established connection with a sensor to sensor framework in order for it to send and receive data from the sensor
Human User	User interacting with the mobile application	User input to answer the forms	Main Process	User provide input in order to complete the work orders
		User Credentials	Auth Process	User provide credentials for login
Sensor	External sensor used to gather specific measurement data. E.g. temperature data	Protocol Data	Comm. Esthb.	Protocol data used to establish a connection. E.g. session key generated after Bluetooth pairing is completed
		Response	Sensor Framework	Sending measured data after specified request from sensor framework
Backend	Backend servers which store all information regarding work orders and user privileges. Provides the API framework	Retrieve Data	Web Interface	User and work order information sent to the web interface
		API Response	RC Process	Sending data that was specified by the RC process in a previous API request. E.g. authentication validation, user ticket
End of Table				

3.3 Resulting Threat Model

With the completion of the data flow diagram the next step was to conduct the actual threat generation process in order to find the applicable threats to the system. The threat generation process was divided into two parts. First, the data flow diagram was modeled in the Microsoft Threat Modeling Tool which later also was used to generate a comprehensive list of numerous possible threats. As the Microsoft Threat Modeling Tool applies the STRIDE-per-interaction methodology when generating threats, and does not take into consideration what platform the application is built on, it resulted in that numerous threats were considered to not be applicable. The resulting threat model generated by the Microsoft Threat Modeling Tool is presented in Appendix A. As can be seen from the report only a subset of the interactions is included and that the majority of threats all corresponds to interactions between components which interact over a trust boundary. In addition, a relation can be seen between the types of threats generated and the OWASP Top Ten mobile risks described in Section 2.2.1, where the majority of the threats directly corresponds to one or numerous of the different risk categories. For example, the interactions between the different storage options and the data storage process all have threats related to information disclosure and elevation of privileges which relates to *Insecure Data*, *Improper Platform Usage* and *Insufficient Cryptography*.

Second, the information acquired during the first step of the threat modeling process was used when conducting the threat generation based on the OWASP threat list, which is presented in Appendix B. The threat generation process was conducted in collaboration with the system developers in order to gain as much insight as possible. The threat list comprises of numerous platform specific threats and their applicability to the system. As can be seen from the list, there exist threats which are considered to not be applicable to the system. For example, threat number eleven, *Attacker Can Bypass Second Level Authentication*, is one threat in the list, which is not applicable as there currently does not exist any implementation of second level authentication. However, for the majority of the threats it was possible to determine their compliance with the system.

3.4 Evaluation of The Threat Modeling Process

As stated by Adam Shostack, the process of threat modeling is like learning to play the violin [10]. Threat modeling should be considered a skill which requires a lot of practice and learning to accumulate and gather enough experience in order to be able to conduct quality threat modeling. When enough experience is acquired the threat modeling can be applied to increasingly complex systems with good results. However, by following a methodological approach when conducting a threat modeling process it aids with both improved structures as well as finding applicable threats which may not have been discovered otherwise.

By evaluating the methodological approach, used throughout this project, when ap-

plying the threat modeling process, described in Section 2.1.10, to our type of system it was made apparent that the process has certain limitations. During the first step, *Decomposing the Application*, when the assets, entry points, trust levels and external dependencies was defined, it showed that the process of determining these components for a mobile application in comparison to a traditional web application was very similar. Also the same approach could be used with high accuracy. However, certain caution had to be taken when defining the entry points to the system as they directly corresponds to what platform the application is built upon. Depending on the underlying architecture there exist differences in how a user can interact with the system and supply it with data. In the case of an Android application, it is possible for that application to rely on data received from another application. This is possible via the use of sending an implicit intent and allowing another application with the corresponding intent-filter to perform some action where the results of the action is later sent in response.

For the second step, where the data flow diagram was created and then would form the foundation for the threat generation process using STRIDE. This showed that the approach of defining a data flow diagram and use it to model a mobile application deemed to be a successful approach. As can be seen in Appendix A and described in Section 3.3 the threat generation process successfully generated numerous different threats that were considered applicable to the system and thus needed further investigation. In addition, several of the threats also had a direct relation to the OWASP Top Ten mobile risks which further strengthens the argument that this approach is applicable and usable when modeling mobile applications. However, as the resulting threat model included numerous threats, which was considered to not be applicable to our system further analysis was performed to determine the cause and reason as to why this was the case. It showed that the reoccurring factor that generated these non applicable threats was due to inadequate specification of the different types of data flows as well as storage options that was present in the system. All data flows in the system was modeled as generic data flows and only explicitly expressed what type of information these flows consisted of but did not take into consideration of what communication type being used. Thereby, resulting in numerous threats related to, for example, tampering and spoofing was generated and due to the underlying communication type was considered non applicable. Therefore, by including specification about the different communication types being used, such as the Android IPC, the efficiency and accuracy of the threat generation process could be increased as less resources would be required to evaluate the non applicable threats.

Even though the process of conducting a threat modeling process on our type of system was deemed successful and resulted in several applicable threats it is not an exhaustive list consisting of all possible threats. By comparing the threats produced by following the threat modeling process and the threats listed in Appendix B, which was determined by applying the checklist given from OWASP it is clear that there exist differences in the two approaches. Both approaches results in numerous applicable threats and even though there exists threats that is mutual to

both approaches, it exists threats that are unique to one or the other. For example, the threat number 15, *Debug is set to true*, from Appendix B is a platform specific threat to Android specifically and there exists no corresponding threat generated from the threat modeling process. Therefore, by utilizing both approaches when conducting threat modeling of a system, it will likely generate in a more efficient process as well as better coverage.

Thus, we propose that the methodological approach for threat modeling aimed towards mobile systems should consist of both utilizing a threat modeling process that uses STRIDE in the threat generation, as well as using platform specific threat libraries designed by accredited sources, such as the ones from OWASP.

4

Penetration testing

In this chapter the top results of the penetration testing will be presented. The results were then used to validate the implementation of the threat model.

4.1 Insecure Authentication and Session Management

The username to access the system is normally a cooperate email, which at many companies uses the following policy: 'firstname.lastname@company.com'. This means that the usernames can easily be guessed if one knows who is a privileged user or if it is known that the company uses this system and then for example publish their email addresses on their website.

The passwords have a weak password structure policy as it only needs five characters and they can for example be all lowercase letters. Both Microsoft [43] and the Online Trust Alliance [44] recommend to have at least eight characters with a mix of lowercase and uppercase letters, numbers and special characters. The tool Hydra was used in conjunction with a list containing the one million most common passwords [45] in order to launch a brute-force attack against the system. Hydra was configured to seize execution upon the first valid match. Resulting in that a user with administrator privileges used a password comprised of five lower case letters.

When logging in the user receives a session-ticket which later is attached to the messages to authenticate the user towards the backend servers. But the session ticket does not have an expiration time, after testing this for almost two months it has been confirmed by the system developers. Neither does it time out when the user actively logs out. By using MITM proxy to catch messages it was possible to impersonate another user with higher privileges. This was done by changing the session-ticket to the session ticket of another user in the HTTP header before sending the messages. The system responded by sending back work orders, collections and media belonging to the user which was impersonated.

It does also seem that the system does not check the session ticket when sending links for media (e.g. images, videos, recordings), all media was still downloaded on

the Android device when the ticket was changed or removed completely. The direct media URLs can be accessed by anyone who has the URL.

4.2 Cross-Site Scripting (XSS)

Shown in Figure 4.1 is the first page that is shown after a valid login for a user with administrator privileges. The highlighting shows the three fields of user input available, all potentially vulnerable to reflected or persistent cross cite scripting. But neither the recommended scripts from OWASP[46], The Browser Exploitation Framework[32] or XSSer[33]] gave any results on this page. None of the field executed any of the scripts entered.

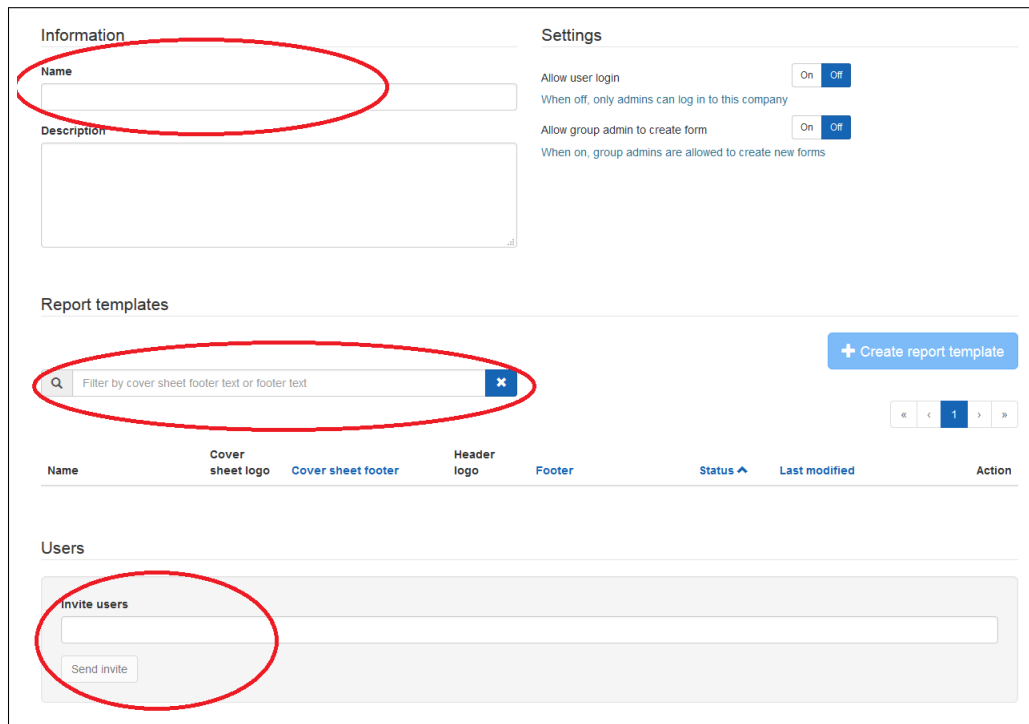
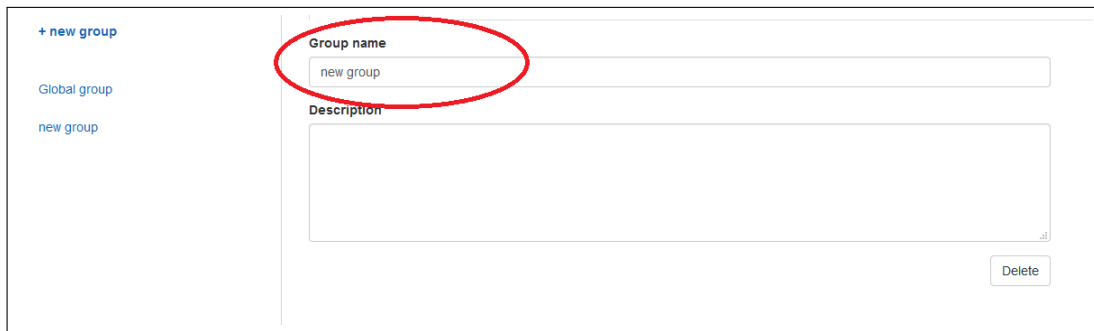


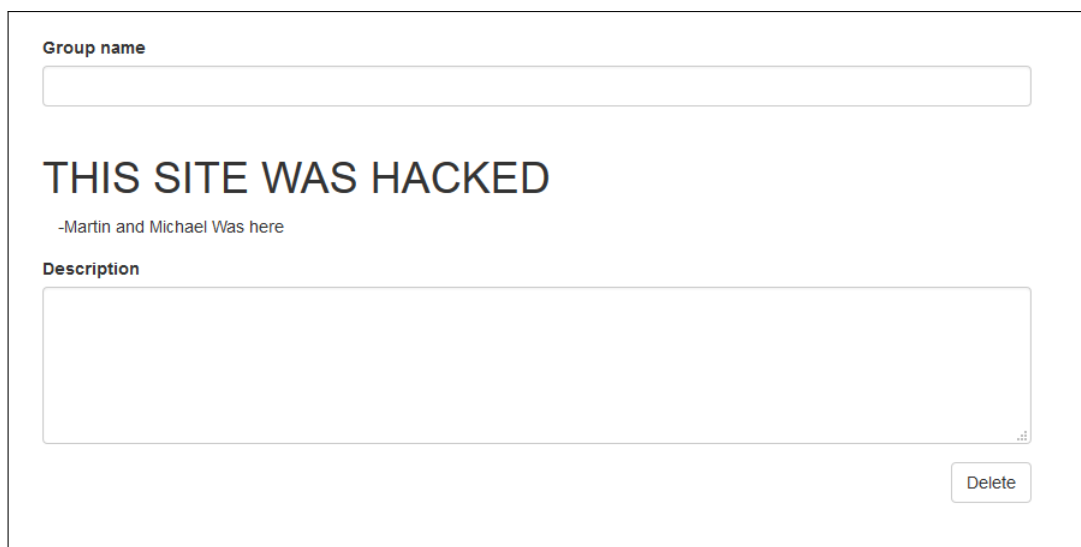
Figure 4.1: Screenshot of the first page in the web-interface with the fields for input highlighted

Then the same attacks were tried against the 'Group Name' input field on the next page, as shown in Figure 4.2. It was found that it was vulnerable against some persistent XSS scripts. As shown in Figure 4.3 it was possible to insert HTML code between 'Group Name' and 'Description' which was saved on the server site on that group. A script inserted also returned part of the session cookie, as shown in Figure 4.4. The cookie returned was not complete due to the maximum size of the alert box that used to present it in.



The screenshot shows a web interface for creating a new group. On the left, there is a sidebar with a '+ new group' link and a list of existing groups: 'Global group' and 'new group'. The main content area has a 'Group name' label above an input field containing 'new group', which is circled in red. Below it is a 'Description' label above a large text area. A 'Delete' button is located at the bottom right of the form.

Figure 4.2: Screenshot of the subpage Group/Create new group in the web-interface with the field for input highlighted



The screenshot shows the result of an HTML injection attack. The page content is replaced with the text 'THIS SITE WAS HACKED' and '-Martin and Michael Was here'. The original form elements, including the 'Group name' input field, the 'Description' text area, and the 'Delete' button, are still visible but partially obscured by the injected content.

Figure 4.3: Screenshot of the results of HTML code insertion



```
__utma=214234370.696912345.1460361929.1462799343.1462863162.19;
__utmz=214234370.1460361929.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=
(none); __utmc=214234370;
__utma=228309407.1913116711.1462348498.1462348498.1462348498.1;
__utmc=228309407; __utmz=228309407.1462348498.1.1.utmcsr=(direct)|utmccn=
(direct)|utmcmd=(none); PLAY_LANG=en
```

An 'Ok' button is located at the bottom right of the screenshot.

Figure 4.4: Screenshot on part of the session cookie

The system seems to be using black listing as a protection mechanism on the 'Group' page since `<script>` worked while `%3c script %3e` did not work, allowing XSS variants to succeed. (`%3c` and `%3e` is the hexadecimal representation of `<` and `>`).

4.3 Insecure Data

Analysis of the data that is associated and created by the application was performed in order to check whether the application stores any sensitive data exposed on the device. This test was carried out by manually using different applications to read the content on the local storage of the device. The first test checked the data storage folder according to the Android storage policy, `storage/emulated/0/Android/data/'application name'/'`, with the use of a file explorer. The results showed that numerous different types of media, collections, work orders and xml files containing information about which user that was assigned which work order and who assigned it, all in clear text.

The second test checked the internal storage by utilizing a file explorer on the rooted Android device. Android storage policy for internal storage is `/data/data/'application name'` and that folder contained data stored in shared preferences which included xml files with company ID, user ID, encrypted username and encrypted session ticket. By putting the Android device into debugging mode and used Android Debug Bridge (Adb), which is a versatile command line tool for Android, the database files from internal storage could be extracted. One database contained information about the user ID, username, company ID name and the email address for the current logged in user, all in clear text. This means that the encryption of these user credentials in shared preferences did not leave this data protected.

When a user signed of/log out, it was shown that only the shared preference content is changed. All media is left on the device and if a new user logs in, their user preferences are just appended in the database.

4.4 Insufficient Cryptography

The process of testing whether the application was susceptible to the insufficient cryptography risk was conducted in two steps. First, reverse engineering of the application was conducted where the APK file was decomposed in order to gain access to the dex file. Reverse engineering was conducted to get access to the source code of the application. This was performed by using the tool *APKtool* [47] which is a tool designed for decoding any binary APK file in order to gain access to its contents. Once the dex files were retrieved, it was further decomposed by the tool *dex2jar* [48] which is a tool designed to recreate the class files containing the original java code.

Second, static code analysis was conducted on the extracted class-files in order to determine the implementation of the cryptographic scheme. It showed that the implementation used in the application uses a weak cryptographic scheme, namely the *Data Encryption Standard* (DES). DES is widely considered to be an insecure cryptographic scheme as it has been proven that it is possible to break via brute-force due to its short key length [49][50]. In addition to using a weak scheme, the application also improperly handles storing of the cryptographic keys. Once generated, the keys are split into two equal parts and then appended to each side of the cipher which is then stored in the shared preferences. Therefore, even though the extra step of implementing another layer of security by using cryptography was taken, it showed that it does not increase the security of the application.

4.5 SQL Injection

By using Burp Suite Proxy it was possible to catch a HTTP header for both a valid and an invalid login attempt towards the web and the Android application interface. The headers could then be used by SQLMap to automatically scan the login page for any SQL injection vulnerabilities. But a full SQLMap scan could not find any vulnerabilities in the login interface for either the web interface or the Android application.

4.6 Validation

In this chapter it was shown that vulnerabilities in several areas could be found where the threat model reported potential threats. It is also found that several defense mechanisms are implemented but not all of them are implemented correctly. The penetration testing has not been exhaustive and it has been performed by two master students with limited prior practical experience. But it was mainly carried out to validate that the threats that the threat modeling process found was actual threats and not false positives.

5

Discussion and Conclusion

This chapter contains a small discussion about working with STRIDE when conducting threat modeling. Then future work is discussed before the thesis is concluded in the conclusion.

5.1 Using STRIDE

While doing this thesis we have realized that D.Dhillion in [37] has a point when he states out that to use STRIDE efficiently you need, at least, some attack knowledge. Otherwise it will be very difficult and time consuming trying to analyze all parts against Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege. We can neither assume that all software developers have security knowledge or that all development teams contains at least one security expert. Instead we need well aimed threat models that can help them find potential threats already from the start since that gives them a better chance to fix them appropriately. It is also necessary to sponsor non-profit charitable organizations like OWASP so that they can continue their work of improving the security in software by making threat top lists and updated articles that are free for all.

5.2 Future Work

This report has shown that traditional threat models might not cover all threats on smartphone/IoT systems. New, or updates to old, threat models needed to be developed to help developers develop more secure systems already from the start. With the wide spread of 'smart homes' and monitoring of electro mechanical systems this becomes an important issue that needs more future research. Many software developers are ignorant or inexperienced when it comes to security so the tools available must be up to date and work with different platforms to making it easy to secure new systems.

5.2.1 Communication Protocols for Sensors

As mentioned in Section 1.5.2 the different communication channels were not covered in the penetration testing and validation. However, when considering other systems in the future it may be essential to include validation and penetration testing of these components. Especially with the inclusion of low energy devices that have limited computational power and relies on battery power. For these kind of systems,

communication technologies that allow for low energy consumption is essential, such as Bluetooth Low Energy. But the developers must be aware which protocols they use and how they implement them. As an example it does exist severe threats against Bluetooth, one is Bluebugging that exploit some permission in the Bluetooth backbone and make it possible for the attacker to use most of the phones features. Another attack is Bluesnarfing which is a kind of 'brute pair' and the attacker can then read out most of the data saved on the paired device.

Therefore, for future work on other smartphone/IoT systems this must be taken into account and analyzed. The choice of communication protocol must be thoroughly analyzed, maybe protocols like WiFi-direct or LoRa have less vulnerabilities and are easier to monitor.

5.3 Conclusion

Hopefully the thesis goals has been made clear and proven useful; that the threat modeling process successfully have been implemented on a smartphone/IoT system using complementary information from OWASP. The report made from the data flow diagram was good, but it needs to be complemented with the OWASP Top list, since it does not yet have the option of choosing which platform that is being analyzed.

Bibliography

- [1] M.Asplund and S.Nadjm-Tehrani. “Attitudes and perceptions of IoT security in critical societal services”. In: *IEEE Access* PP.99 (2016). ISSN: 2169-3536.
- [2] D.M.Ahmad and P.Javed. “Security Comparison of Android and IOS and Implementation of User Approved Security (UAS) for Android”. In: *Indian Journal of Science and Technology* 9.14 (2016).
- [3] IDC. *Smartphone OS Market Share*. URL: <http://www.idc.com/prodserv/smartphone-os-market-share> (visited on 05/10/2016).
- [4] OWASP. *Application Threat Modeling*. URL: https://www.owasp.org/index.php/Application_Threat_Modeling (visited on 02/26/2016).
- [5] B.Larcom P.Saitta and M.Eddington. *Trike v.1 Methodology Document [Draft]*. July 2005.
- [6] Microsoft. *Threat Modeling*. URL: <https://msdn.microsoft.com/en-us/library/ff648644.aspx> (visited on 03/01/2016).
- [7] SANS. *Threat Modeling: A Process To Ensure Application Security*. URL: <https://www.sans.org/reading-room/whitepapers/securecode/threat-modeling-process-ensure-application-security-1646> (visited on 03/01/2016).
- [8] K.Fenrich. “Securing Your Control System”. In: *Power Engineering* 112.2 (2008), pp. 44-51.
- [9] Bel G Raggad. *Information security management: Concepts and practice*. CRC Press, 2010.
- [10] A.Shostack. *Threat modeling : designing for security*. John Wiley and Sons, 2014. ISBN: 9781118809990.
- [11] L.Kohnfelder and G.Praerit. “The threats to our products”. In: *Microsoft Interface, Microsoft Corporation* (1999).
- [12] Bruce Schneier. *Attack Trees - Modeling security threats*. Dec. 1999. URL: https://www.schneier.com/cryptography/archives/1999/12/attack_trees.html.
- [13] SANS. *An Introduction to Information System Risk Management*. URL: <https://www.sans.org/reading-room/whitepapers/auditing/introduction-information-system-risk-management-1204> (visited on 03/07/2016).
- [14] C. Warren Axelrod. *Engineering Safe and Secure Software Systems (Artech House Information Security and Privacy)*. Artech House, 2012. ISBN: 1608074722.
- [15] IBM Developer Works. *OWASP top 10 vulnerabilities*. URL: <https://www.ibm.com/developerworks/library/se-owasptop10/> (visited on 05/07/2016).
- [16] S.Harris. *CISSP Certification All-in-One Exam Guide, Fourth Edition (Cissp All-In-One Exam Guide)*. McGraw-Hill Osborne Media, 2007. ISBN: 0071497870.

- [17] OWASP. *OWASP Mobile Security Project*. URL: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project (visited on 05/15/2016).
- [18] OWASP. *OWASP Mobile Checklist Final 2016*. URL: <https://drive.google.com/file/d/0Bx0Pagp1jPHWYmg3Y3BfLVhMcmc> (visited on 05/15/2016).
- [19] Microsoft. *SDL Threat Modeling Tool*. URL: <https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx> (visited on 05/13/2016).
- [20] J.J.Drake et al. *Android Hacker's Handbook*. Wiley, 2014. ISBN: 111860864X.
- [21] Android Developers. *Application Fundamentals*. URL: <https://developer.android.com/guide/components/fundamentals.html> (visited on 05/13/2016).
- [22] Android Developers. *Verifying App Behavior on the Android Runtime (ART)*. URL: <https://developer.android.com/guide/practices/verifying-apps-art.html> (visited on 05/14/2016).
- [23] Android Open Source Project. *ART and Dalvik*. URL: <http://source.android.com/devices/tech/dalvik/index.html> (visited on 05/14/2016).
- [24] Android Developers. *Security Tips*. URL: <https://developer.android.com/training/articles/security-tips.html> (visited on 05/14/2016).
- [25] Android Developers. *Storage Options*. URL: <https://developer.android.com/guide/topics/data/data-storage.html#netw> (visited on 05/14/2016).
- [26] Android Developers. *Intents and Intent Filters*. URL: <https://developer.android.com/guide/components/intents-filters.html> (visited on 05/14/2016).
- [27] OWASP. *Testing Tools - OWASP*. URL: https://www.owasp.org/index.php/Appendix_A:_Testing_Tools (visited on 05/14/2016).
- [28] Portswigger. *Burp Suite*. URL: <https://portswigger.net/Burp/> (visited on 05/14/2016).
- [29] THC. *Hydra*. URL: www.thc.org/thc-hydra (visited on 05/14/2016).
- [30] mimiproxy. *MITM-Proxy*. URL: <https://mitmproxy.org/> (visited on 05/14/2016).
- [31] sqlmap. *sqlmap: Automatic SQL injection and database takeover tool*. URL: <http://sqlmap.org/> (visited on 05/14/2016).
- [32] BeEF. *The Browser Exploitation Framework*. URL: <http://beefproject.com/> (visited on 05/07/2016).
- [33] XSSer. *Cross Site "Scripter"*. URL: <https://xsser.03c8.net/> (visited on 05/07/2016).
- [34] A.Kesäniemi. *Mobile Application Threat Analysis*. The Owasp foundation, 2012. (Visited on 05/15/2016).
- [35] DataTheorem. "Threat Model for Mobile Applications Security and Privacy". In: (2013).
- [36] L.Dua and D.Bansal. "Review on Mobile Threats and Detection Techniques". In: *International Journal of Distributed and Parallel systems* 5 (2014), pp. 21–29.
- [37] D.Dhillon. "Developer Driven Threat Modeling: Lessons Learned in the Trenches". In: *IEEE Security Privacy* (2011).
- [38] R.Scandariato K.Wuyts and W.Joosen. "Empirical evaluation of a privacy-focused threat modeling methodology". In: *Journal of Systems and Software* 96 (2014), pp. 122–138.

- [39] J.A.Clark et al. “Threat modeling for mobile ad hoc and sensor networks”. In: (2007), pp. 25–27.
- [40] N.Antunes and M.Vieira. “Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services”. In: (2009), pp. 301–306.
- [41] A.Austin and L.Williams. “One Technique is Not Enough: A Comparison of Vulnerability Discovery Techniques”. In: (2011), pp. 97–106.
- [42] J.Walden R.Scandariato and W.Joosen. “Static analysis versus penetration testing: A controlled experiment”. In: (2013), pp. 451–460.
- [43] Microsoft Technet Magazine. *Best Practices for Enforcing Password Policies*. URL: <https://technet.microsoft.com/en-us/magazine/ff741764.aspx> (visited on 05/07/2016).
- [44] Online Trust Alliance. *Security & Privacy Best Practices*. URL: <https://otalliance.org/resources/security-privacy-best-practices> (visited on 05/07/2016).
- [45] *danielmiessler/SecLists*. URL: <https://github.com/danielmiessler/SecLists>.
- [46] OWASP. *XSS Filter Evasion Cheat Sheet*. URL: https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet (visited on 05/07/2016).
- [47] R.Wiśniewski. *Apktool - A tool for reverse engineering Android apk files*. URL: <http://ibotpeaches.github.io/Apktool/> (visited on 05/18/2016).
- [48] *dex2jar*. URL: <https://github.com/pxb1988/dex2jar> (visited on 05/18/2016).
- [49] B.Schneier. *The Legacy of DES*. URL: https://www.schneier.com/blog/archives/2004/10/the_legacy_of_d.html (visited on 05/18/2016).
- [50] OWASP. *Guide to Cryptography*. URL: https://www.owasp.org/index.php/Guide_to_Cryptography (visited on 05/18/2016).

A

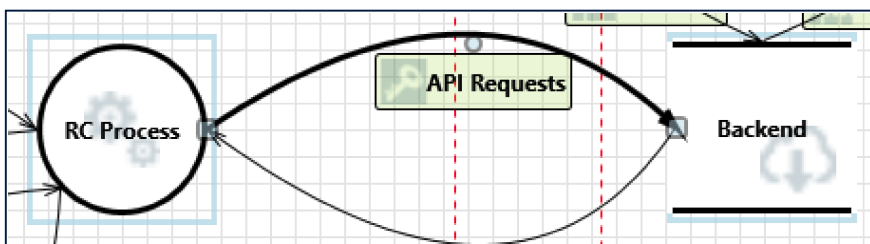
Appendix

Threat Modeling Report

Threat Model Summary:

Not Started	0
Needs Investigation	32
Total	32

Interaction: API Requests



1. Data Store Inaccessible [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent prevents access to a data store on the other side of the trust boundary.

Justification: If traffic is intercepted there seems to be no timeout for syncing data

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

2. Data Flow API Requests Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: If traffic is intercepted there seems to be no timeout for syncing data

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

3. Weak Credential Storage [State: Needs Investigation] [Priority: High]

Category: Information Disclosure

Description: Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored

Justification: The ticket is encrypted before storage but is decrypted before transmission. Ticket is presumably stored in plaintext in the backend. Encrypted ticket and encryption technique is possible to

extract with physical access to the mobile device.

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

4. The Backend Data Store Could Be Corrupted [State: Needs Investigation] [Priority: High]

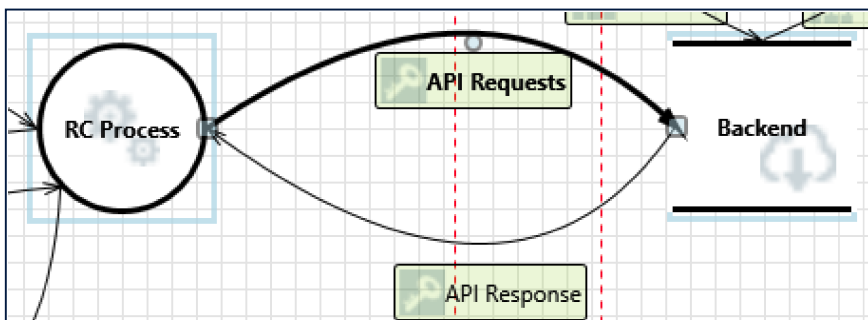
Category: Tampering

Description: Data flowing across API Requests may be tampered with by an attacker. This may lead to corruption of Backend. Ensure the integrity of the data flow to the data store.

Justification: With physical access to a mobile device and a user ticket it is possible to tamper with the data to gain access to unprivileged data.

Short Description: Tampering is the act of altering the bits. Tampering with a process involves changing bits in the running process. Similarly, Tampering with a data flow involves changing bits on the wire or between two running processes.

Interaction: API Response



5. Spoofing of Source Data Store Backend [State: Needs Investigation] [Priority: High]

Category: Spoofing

Description: Backend may be spoofed by an attacker and this may lead to incorrect data delivered to RC Process. Consider using a standard authentication mechanism to identify the source data store.

Justification: The Backend needs to receive either user credentials or the user ticket in order to send and retrieve data. If spoofing would be possible then the spoofed backend needs to know this information anyway.

Short Description: Spoofing is when a process or entity is something other than its claimed identity. Examples

Description: include substituting a process, a file, website or a network address.

6. Weak Access Control for a Resource [State: Needs Investigation] [Priority: High]

Category: Information Disclosure

Description: Improper data protection of Backend can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: Possible to retrieve data by using another user ticket.

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

Description:

7. Data Flow API Response Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: If traffic is intercepted there seems to be no timeout for syncing data

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

8. Data Store Inaccessible [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

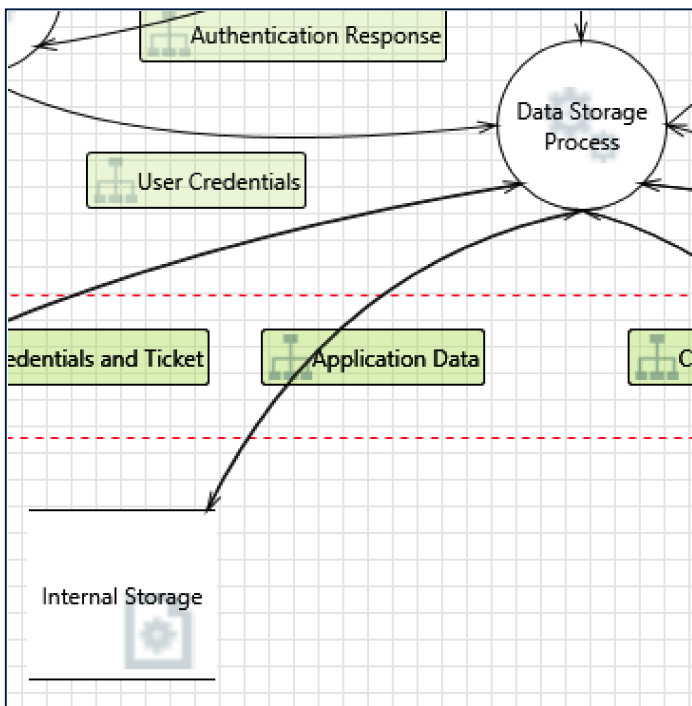
Description: An external agent prevents access to a data store on the other side of the trust boundary.

Justification: If traffic is intercepted there seems to be no timeout for syncing data

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

Description: requests or perform up to spec.

Interaction: Application Data



9. Elevation by Changing the Execution Flow in Data Storage Process [State: Needs Investigation] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into Data Storage Process in order to change the flow of program execution within Data Storage Process to the attacker's choosing.

Justification: Possible to change information used for authentication that affects the backend authentication and authorization

Short Description: A user subject gains increased capability or privilege by taking advantage of an implementation

Description: bug.

10. Weak Access Control for a Resource [State: Needs Investigation] [Priority: High]

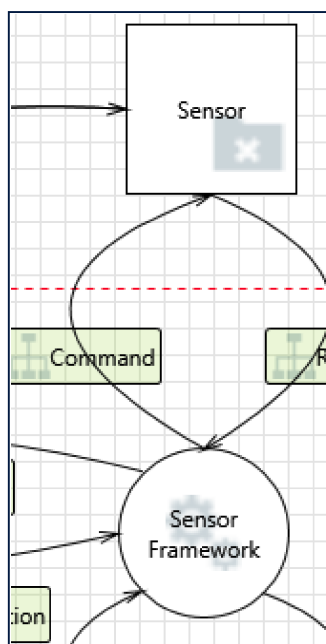
Category: Information Disclosure

Description: Improper data protection of Internal Storage can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: With physical access to the device information can be read. Protection mechanism such as crypto needs to be implemented.

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

Interaction: Command



11. Spoofing of the Sensor External Destination Entity [State: Needs Investigation] [Priority: High]

Category: Spoofing

Description: Sensor may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of Sensor. Consider using a standard authentication mechanism to identify the external entity.

Justification: Sensor Framework only communicates with known sensors. However, if enough information is known it might be possible to impersonate a sensor

Short Description: Spoofing is when a process or entity is something other than its claimed identity. Examples

Description: include substituting a process, a file, website or a network address.

12. Data Flow Command Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

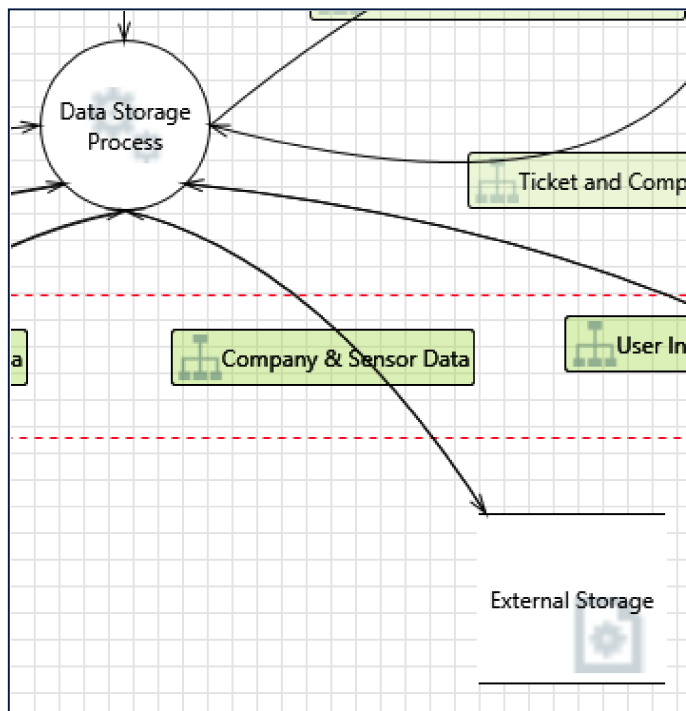
Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: Depending on the communication medium used. An adversary may be able to intercept traffic and cause a DoS

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming

Description: requests or perform up to spec.

Interaction: Company & Sensor Data



13. Elevation by Changing the Execution Flow in Data Storage Process [State: Needs Investigation] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into Data Storage Process in order to change the flow of program execution within Data Storage Process to the attacker's choosing.

Justification: With physical access to a device an adversary may alter information and modify work orders but will not however affect the system execution

Short Description: A user subject gains increased capability or privilege by taking advantage of an implementation bug.

14. Spoofing of Source Data Store External Storage [State: Needs Investigation] [Priority: High]

Category: Spoofing

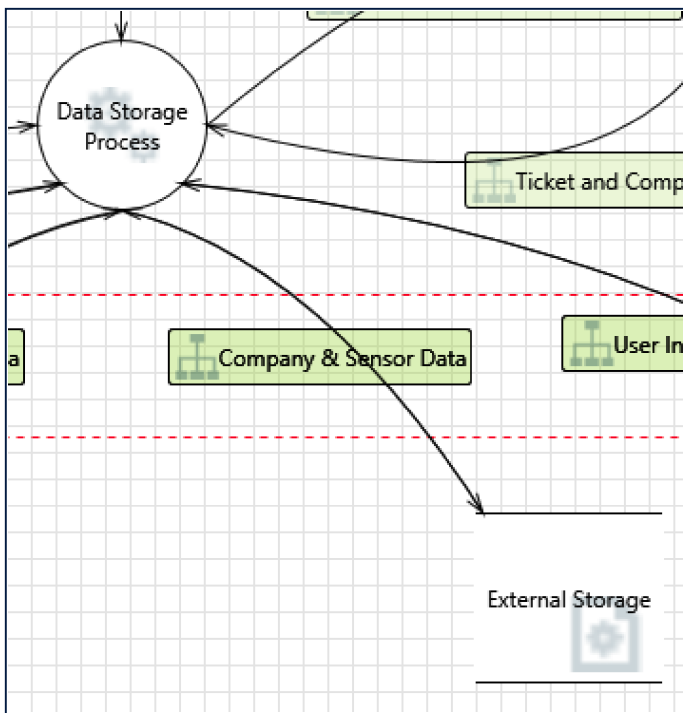
Description: External Storage may be spoofed by an attacker and this may lead to incorrect data delivered to Data Storage Process. Consider using a standard authentication mechanism to identify the source data store.

Justification: An adversary might be able to insert data into the external storage which will be read at later stage

Short Description: Spoofing is when a process or entity is something other than its claimed identity. Examples

Description: include substituting a process, a file, website or a network address.

Interaction: Company & Sensor Data



15. Data Store Inaccessible [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent prevents access to a data store on the other side of the trust boundary.

Justification: With physical access to a device it might be possible to unmount the external storage

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

16. Data Flow Generic Data Flow Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: With physical access to a device it might be possible to unmount the external storage

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

17. Spoofing of Destination Data Store External Storage [State: Needs Investigation] [Priority: High]

Category: Spoofing

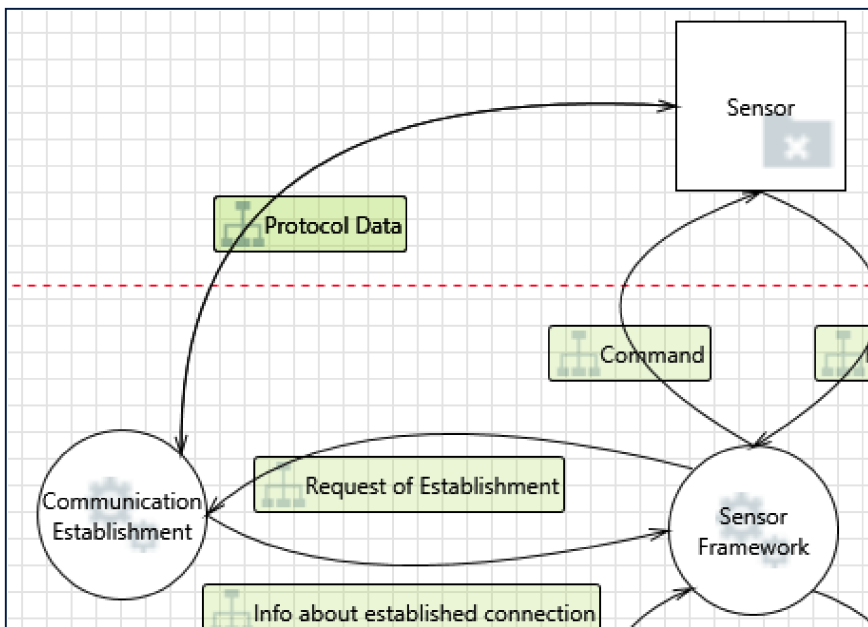
Description: External Storage may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of External Storage. Consider using a standard authentication mechanism to identify the destination data store.

Justification: Storage is located locally on the device. However the SDcard might be removable with physical access depending on the device

Short Description: Spoofing is when a process or entity is something other than its claimed identity. Examples

Description: include substituting a process, a file, website or a network address.

Interaction: Protocol Data



18. Data Flow Generic Data Flow Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: Depending on medium an adversary might be able to spoof a sensor without knowing the exact protocol making the establishment not succeed resulting in a DoS

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

19. Data Flow Sniffing [State: Needs Investigation] [Priority: High]

Category: Information Disclosure

Description: Data flowing across Protocol Data may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations. Consider encrypting the data flow.

Justification: Depending on communication medium an adversary might be able to gain information about the current session which can be used to listen or tamper with future communication

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

20. Spoofing the Sensor External Entity [State: Needs Investigation] [Priority: High]

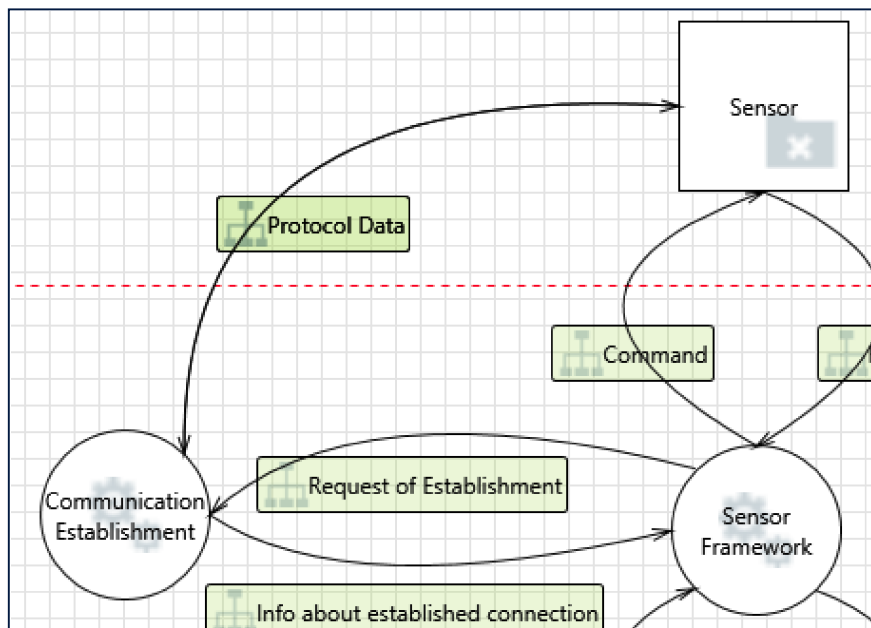
Category: Spoofing

Description: Sensor may be spoofed by an attacker and this may lead to unauthorized access to Communication Establishment. Consider using a standard authentication mechanism to identify the external entity.

Justification: Sensor Framework only communicates with known sensors. However, if enough information is known it might be possible to impersonate a sensor

Short Description: Spoofing is when a process or entity is something other than its claimed identity. Examples include substituting a process, a file, website or a network address.

Interaction: Protocol Data



21. Data Flow Generic Data Flow Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: <no mitigation provided>

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

22. Spoofing of the Sensor External Destination Entity [State: Needs Investigation] [Priority: High]

Category: Spoofing

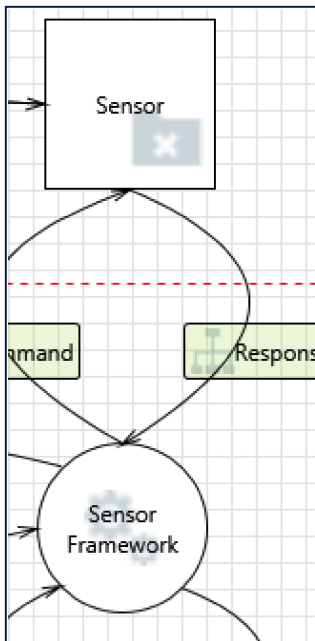
Description: Sensor may be spoofed by an attacker and this may lead to data being sent to the attacker's target instead of Sensor. Consider using a standard authentication mechanism to identify the external entity.

Justification: <no mitigation provided>

Short Description: Spoofing is when a process or entity is something other than its claimed identity. Examples

Description: include substituting a process, a file, website or a network address.

Interaction: Response



23. Spoofing the Sensor External Entity [State: Needs Investigation] [Priority: High]

Category: Spoofing

Description: Sensor may be spoofed by an attacker and this may lead to unauthorized access to Sensor Framework. Consider using a standard authentication mechanism to identify the external entity.

Justification: Sensor Framework only communicates with known sensors. However, if enough information is known it might be possible to impersonate a sensor

Short Description: Spoofing is when a process or entity is something other than its claimed identity. Examples

Description: include substituting a process, a file, website or a network address.

24. Data Flow Response Is Potentially Interrupted [State: Needs Investigation] [Priority: High]

Category: Denial Of Service

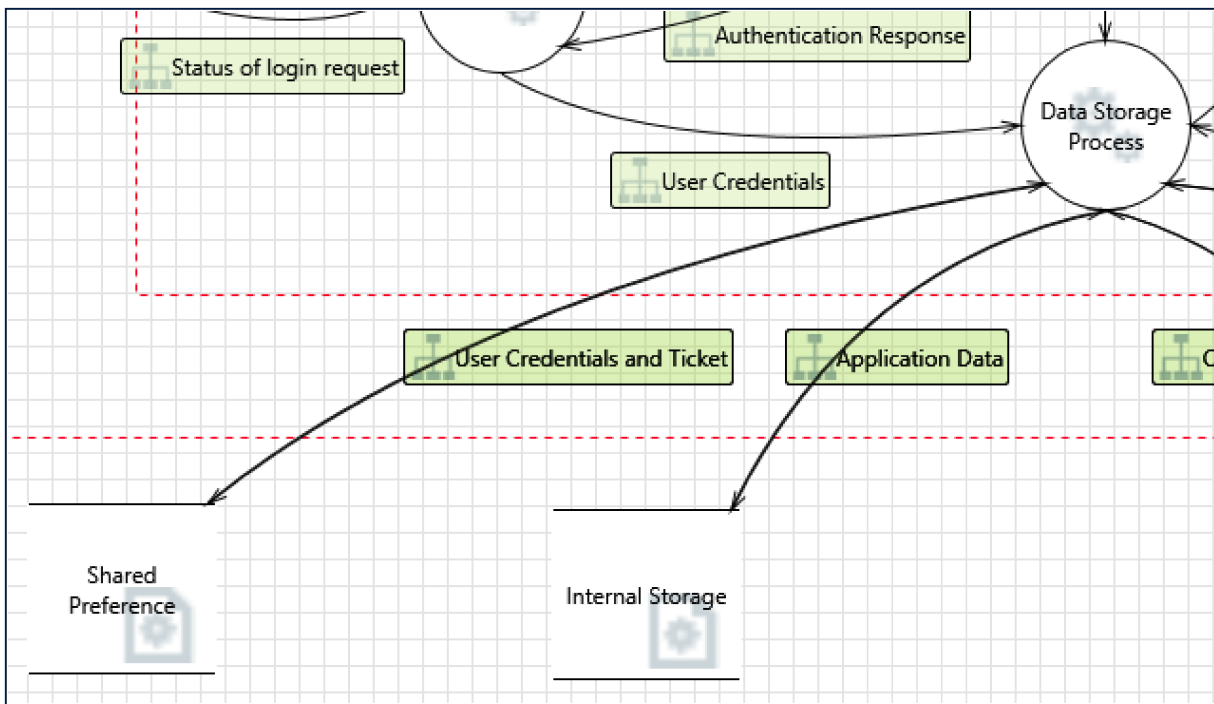
Description: An external agent interrupts data flowing across a trust boundary in either direction.

Justification: Depending on the communication medium used. An adversary may be able to intercept traffic and cause a DoS

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming

Description: requests or perform up to spec.

Interaction: User Credentials and Ticket



25. Weak Credential Storage [State: Needs Investigation] [Priority: High]

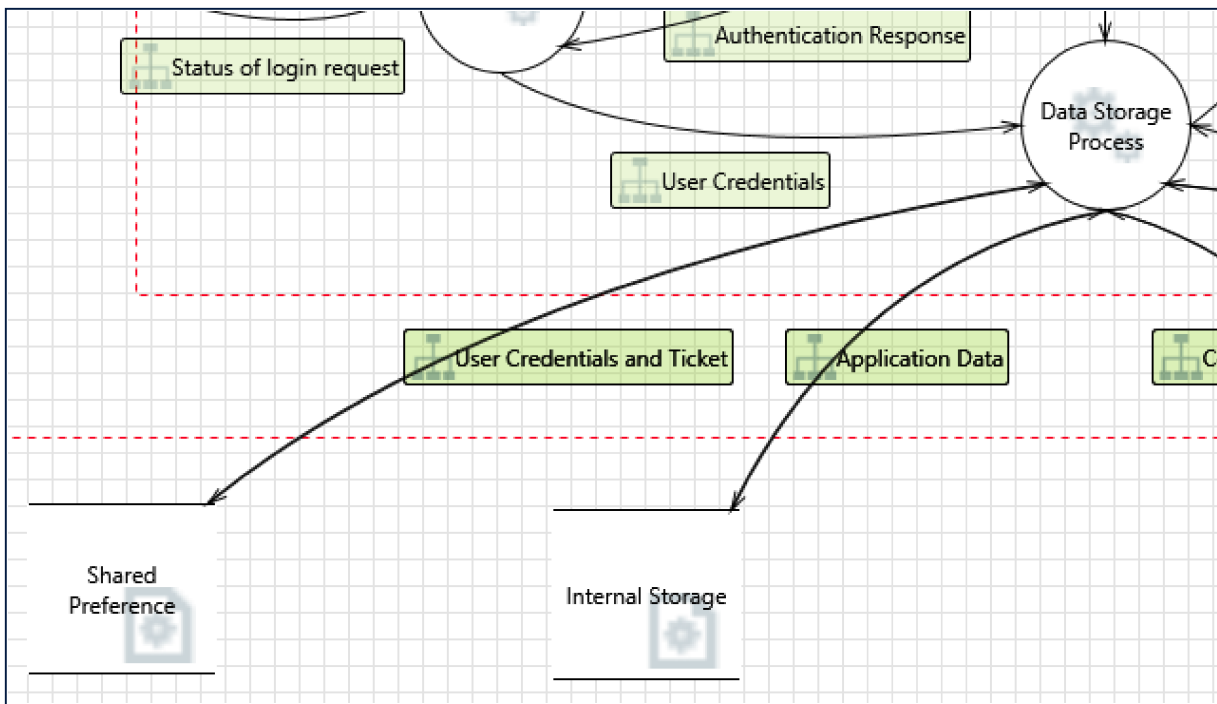
Category: Information Disclosure

Description: Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored

Justification: Credentials are stored encrypted but the encryption key can be retrieved from the stored cipher

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

Interaction: User Credentials and Ticket



26. Elevation by Changing the Execution Flow in Data Storage Process [State: Needs Investigation] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into Data Storage Process in order to change the flow of program execution within Data Storage Process to the attacker's choosing.

Justification: If adversary have access to the local file system and root access the file can be tampered with and change for example ticket or user credentials

Short Description: A user subject gains increased capability or privilege by taking advantage of an implementation bug.

27. Weak Access Control for a Resource [State: Needs Investigation] [Priority: High]

Category: Information Disclosure

Description: Improper data protection of Shared Preference can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: Accessible if adversary have root access on the device.

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

28. Spoofing of Source Data Store Shared Preference [State: Needs Investigation] [Priority: High]

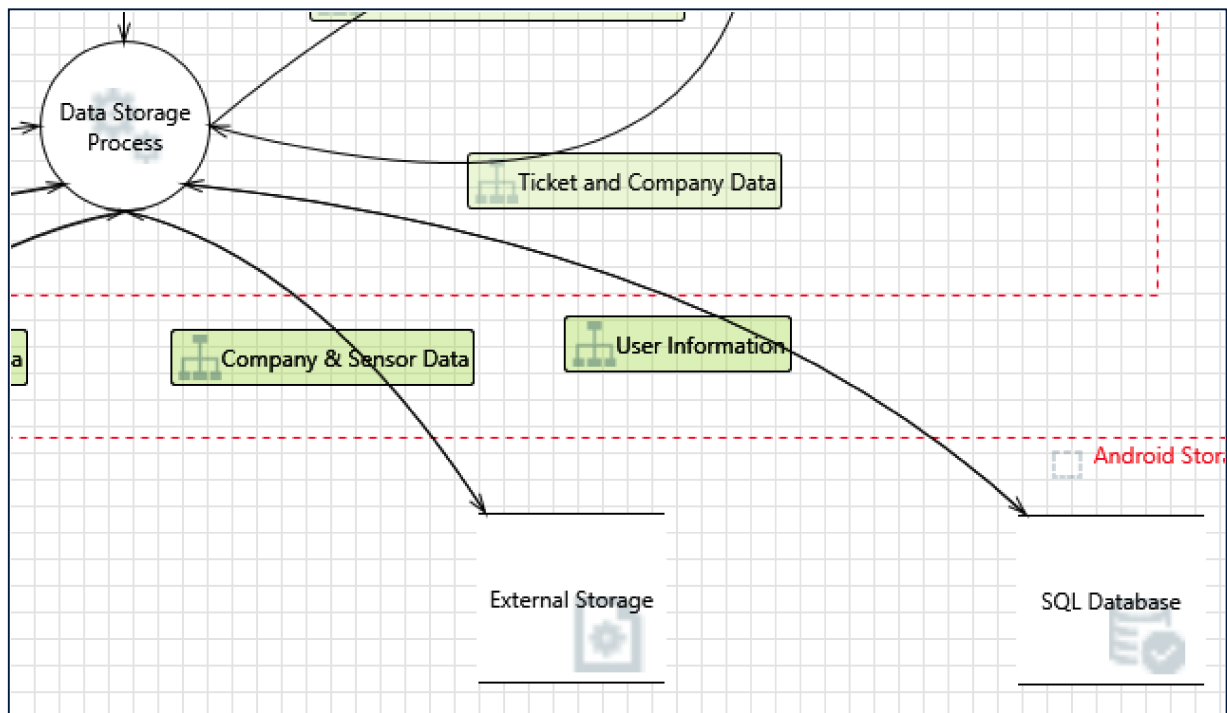
Category: Spoofing

Description: Shared Preference may be spoofed by an attacker and this may lead to incorrect data delivered to Data Storage Process. Consider using a standard authentication mechanism to identify the source data store.

Justification: File can not directly be spoofed but instead be tampered with if adversary have access to the local file system and root access

Short Description: Spoofing is when a process or entity is something other than its claimed identity. Examples include substituting a process, a file, website or a network address.

Interaction: User Information



29. Elevation by Changing the Execution Flow in Data Storage Process [State: Needs Investigation] [Priority: High]

Category: Elevation Of Privilege

Description: An attacker may pass data into Data Storage Process in order to change the flow of program execution within Data Storage Process to the attacker's choosing.

Justification: With physical access to a device an adversary may alter information and modify work orders but will not however affect the system execution

Short Description: A user subject gains increased capability or privilege by taking advantage of an implementation bug.

30. Weak Access Control for a Resource [State: Needs Investigation] [Priority: High]

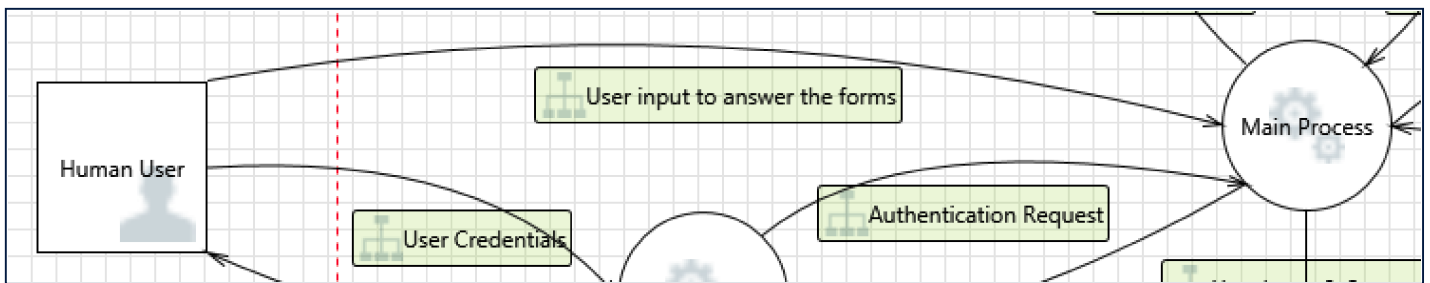
Category: Information Disclosure

Description: Improper data protection of SQL Database can allow an attacker to read information not intended for disclosure. Review authorization settings.

Justification: With physical access to a device an adversary will be able to extract the database file.

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

Interaction: User input to answer the forms



31. Potential Process Crash or Stop for Main Process [State: Needs Investigation] [Priority: High]

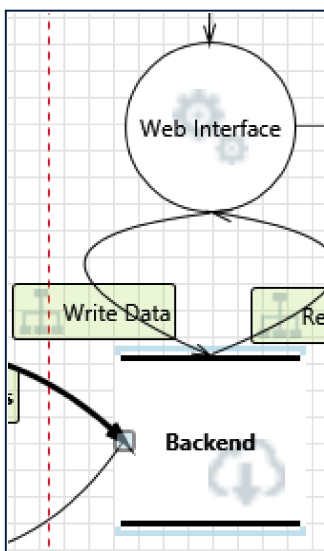
Category: Denial Of Service

Description: Main Process crashes, halts, stops or runs slowly; in all cases violating an availability metric.

Justification: An adversary may input data into the system that causes it to crash. However that only affects the current device and login session.

Short Description: Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

Interaction: Write Data



32. Weak Credential Storage [State: Needs Investigation] [Priority: High]

Category: Information Disclosure

Description: Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored

Justification: As passwords are sent in cleartext under SSL tunnel it is likely that passwords are stored in cleartext

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

B

Appendix

Nr	Vulnerability Name	Compliant: Yes/No/		Classification
		Platform	Not Applicable/-	
1	Application is Vulnerable to Reverse Engineering Attack/Lack of Code	ALL	Yes	Static Checks
2	Account Lockout not Implemented	ALL	Yes	Dynamic Checks
3	Application is Vulnerable to XSS	ALL	No	Dynamic + Static Checks
4	Authentication bypassed	ALL	Yes	Dynamic Checks
5	Hard coded sensitive information in Application Code (including Crypt)	ALL	No	Static Checks
6	Malicious File Upload	ALL	No	Dynamic Checks
7	Session Fixation	ALL	No	Dynamic Checks
9	Privilege Escalation	ALL	Yes	Dynamic Checks
10	SQL Injection	ALL	No	Dynamic + Static Checks
11	Attacker can bypass Second Level Authentication	ALL	Not Applicable	Dynamic Checks
12	Application is vulnerable to LDAP Injection	ALL	Not Applicable	Dynamic Checks
13	Application is vulnerable to OS Command Injection	ALL	No	Dynamic Checks
15	Debug is set to TRUE	Android	No	Static Checks
16	Application makes use of Weak Cryptography	ALL	Yes	Static Checks
17	Cleartext information under SSL Tunnel	ALL	Yes	Dynamic Checks
18	Client Side Validation can be bypassed	ALL	Yes	Dynamic Checks
19	Invalid SSL Certificate	ALL	No	Static Checks
20	Sensitive Information is sent as Clear Text over network/Lack of Data	ALL	No	Dynamic Checks
21	CAPTCHA is not implemented on Public Pages/Login Pages	ALL	Yes	Dynamic Checks
22	Improper or NO implementation of Change Password Page	ALL	No	Dynamic Checks
23	Application does not have Logout Functionality	ALL	No	Dynamic Checks
24	Sensitive information in Application Log Files	ALL	Yes	Dynamic Checks
25	Sensitive information sent as a querystring parameter	ALL	No	Dynamic Checks
26	URL Modification	ALL	Not Applicable	Dynamic Checks
27	Sensitive information in Memory Dump	ALL	No	Dynamic Checks
28	Weak Password Policy	ALL	Yes	Dynamic Checks
29	Autocomplete is not set to OFF	ALL	No	Static Checks
30	Application is accessible on Rooted Device	ALL	Yes	Static Checks
31	Back-and-Refresh attack	ALL	Not Applicable	Dynamic Checks
32	Directory Browsing	ALL	Not Applicable	Dynamic + Static Checks
33	Usage of Persistent Cookies	ALL	Yes	Dynamic Checks

34	Open URL Redirects are possible	ALL	No	Dynamic Checks
35	Improper exception Handling: In code	ALL	-	Static Checks
36	Insecure Application Permissions	ALL	No	Static Checks
37	Application build contains Obsolete Files	ALL	-	Static Checks
38	Certificate Chain is not Validated	ALL	yes	Dynamic Checks
39	Last Login information is not displayed	ALL	No	Dynamic Checks
40	Private IP Disclosure	ALL	No	Static Checks
41	UI Impersonation through RMS file modification [1]	JAVA	-	Dynamic Checks
42	UI Impersonation through JAR file modification	Android	-	Dynamic Checks
43	Operation on a resource after expiration or release	All	Not Applicable	Dynamic Checks
44	No Certificate Pinning	All	Yes	Dynamic Checks
45	Cached Cookies or information not cleaned after application removal	All	No	Dynamic Checks
46	Clipboard is not disabled	All	No	Dynamic Checks
48	Android Backup Vulnerability	Android	No	Static Checks
49	Unencrypted Credentials in Databases (sqlite db)	ALL	Yes	Dynamic Checks
50	Store sensitive information outside App Sandbox (on SDCard)	ALL	Yes	Dynamic Checks
51	Allow Global File Permission on App Data	Android	No	Dynamic Checks
52	Store Encryption Key Locally/Store Sensitive Data in ClearText	All	Yes	Dynamic Checks
53	Bypass Certificate Pinning	All	Not Applicable	Dynamic Checks
54	Third-party Data Transit on Unencrypted Channel	All	No	Dynamic Checks
55	Failure to Implement Trusted Issuers	Android	No	Static Checks
56	Allow All Hostname Verifier	Android	-	Static Checks
57	Ignore SSL Certificate Error	All	-	Static Checks
58	Weak Custom Hostname Verifier	Android	-	Static Checks
59	App/Web Caches Sensitive Data Leak	All	Yes	Dynamic Checks
60	Leaking Content Provider	Android	No	Dynamic Checks
61	Redundancy Permission Granted	Android	No	Static Checks
62	Use Spoof-able Values for Authenticating User (IMEI, UDID)	ALL	No	Dynamic Checks
63	Use of Insecure and/or Deprecated Algorithms	All	-	Static Checks
64	Local File Inclusion (might be through XSS Vulnerability)	ALL	Not Applicable	Dynamic + Static Checks
65	Activity Hijacking	Android	Yes	Static Checks
66	Service Hijacking	Android	No	Static Checks
67	Broadcast Thief	Android	No	Static Checks

68	Malicious Broadcast Injection	Android	No	Static Checks
69	Malicious Activity/Service Launch	Android	No	Static Checks
70	Using Device Identifier as Session	All	No	Dynamic Checks
72	Lack of Check-sum Controls/Altered Detection	Android	No	Dynamic Checks
73	Insecure permissions on Unix domain sockets	Android	-	Static Checks
74	Insecure use of network sockets	Android	-	Static Checks