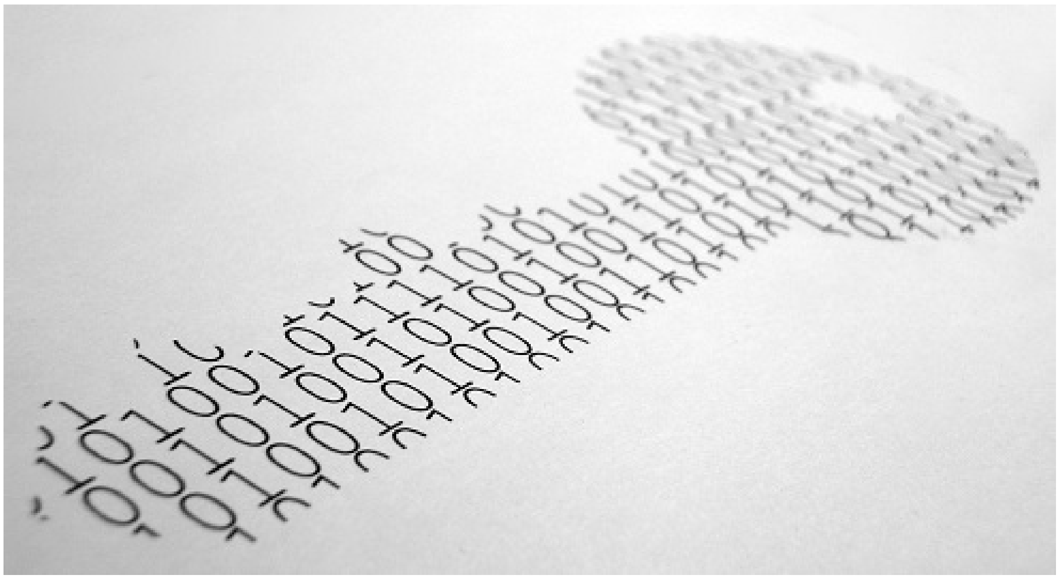
CHALMERS



Secure Continuous Deployment of Military Software

Johannes Blomquist

Johan Härdmark

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2015

Secure Continuous Deployment of Military Software

JOHANNES BLOMQUIST
JOHAN HÄRDMARK

Examiner: DAVID SANDS
Supervisor: TOMAS OLOVSSON

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Department of Computer Science and Engineering
Gothenburg, Sweden. November 2015.

Abstract

Continuous deployment of military software over the Internet could introduce many positive features, like shorter development times, better customer fitted products and on-demand functionality upgrades. It however also introduces a number of new threats and vulnerabilities that needs to be thoroughly investigated and assessed. This Master Thesis proposes a methodology for how to design and evaluate a system with security in mind that allows the designer to build the system from scratch by adding components to mitigate the discovered vulnerabilities. The methodology is based on Microsoft's thread model, with methods like STRIDE and DREAD. Different security protocols and concepts are examined in order to evaluate whether and how *continuous deployment* could be applied in the military industry. The proposed methodology is not military specific and could therefore be used by anyone that wants to build a secure system, although the specific vulnerabilities and their implications presented in this report are with a military setting in mind.

As a big Swedish defence contractor, some of the information Saab handles is of sensitive nature with respect to the Swedish national security. By Swedish law any information classified as *defence secrets* have to be handled according to *Swedish Defence Material Administrations* (FMV) policies. Parts of these policies are by themselves classified as *defence secrets* and are therefore not allowed to be present in this report. As of this, the solutions proposed will apply to information that does not contain *defence secrets*, but could in other aspects be sensitive, and are based on public protocols, algorithms and products.

Acknowledgements

We would like to thank Saab Electronic Defence Systems in Gothenburg, Rafael Aramburo and Anders Martinsson for giving us the opportunity to perform this thesis. We would also like to thank Sven Nilsson and Fredrik Magnusson for support and feedback during this project. Finally we would like to thank or supervisor at Chalmers University of Technology, Tomas Olovsson, for support and guidance during this thesis.

Johannes Blomquist and Johan Härdmark, Gothenburg 5/11/15

List of Terminology

Below follows a list of useful words used in the report. The list is retrieved from RFC 2828, Internet Security Glossary, May 2000 [1].

Adversary (threat agent) An entity that attacks, or is a threat to, a system.

Attack An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

Authenticate Verify (i.e., establish the truth of) an identity claimed by or for a system entity.

BIOS Basic Input/Output System, is the lowest-level software in the computer; it acts as an interface between the hardware and the operating system.

Brute Force A cryptanalysis technique or other kind of attack method involving an exhaustive procedure that tries all possibilities, one-by-one.

Certificate General English usage: A document that attests to the truth of something or the ownership of something.

Countermeasure An action, device, procedure, or technique that reduces a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken.

Cryptography The mathematical science that deals with transforming data to render its meaning unintelligible (i.e., to hide its semantic content), prevent its undetected alteration, or prevent its unauthorized use. If the transformation is reversible, cryptography also deals with restoring encrypted data to intelligible form.

Digital Signature A value computed with a cryptographic algorithm and appended to a data object in such a way that any recipient of the data can use the signature to verify the data's origin and integrity.

Dictionary Attack An attack that uses a brute-force technique of successively trying all the words in some large, exhaustive list.

Man-in-the-middle A form of active wiretapping attack in which the attacker intercepts and selectively modifies communicated data in order to masquerade as one or more of the entities involved in a communication association.

RFID Radio-frequency identification, is the wireless use of electromagnetic fields to transfer data, for the purposes of automatically identifying and tracking tags attached to objects.

Risk An expectation of loss expressed as the probability that a particular threat will exploit a particular vulnerability with a particular harmful result.

Security Policy A set of rules and practices that specify or regulate how a system or organization provides security services to protect sensitive and critical system resources.

System Resource (Asset) Data contained in an information system; or a service provided by a system; or a system capability, such as processing power or communication bandwidth; or an item of system equipment (i.e., a system component—hardware, firmware, software, or documentation); or a facility that houses system operations and equipment.

Threat A potential for violation of security, which exists when there is a circumstance, capability, action, or event, that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

Vulnerability A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy.

Contents

1

Introduction

Saab is a big Swedish defence contractor, developing a wide range of solutions for both civil and military use [2]. The products span from fighter aircrafts to submarines, naval warships, radar installations, surveillance and security systems, camouflage, missiles, and torpedoes. Saab have a long history of delivering a variety of military systems, primarily to the *Swedish Armed Forces*. These deliveries have traditionally involved physical products with accompanied software. In time, the development of Saab's products have shifted focus, where the software has become a vital part of the products, a trend that is expected to sustain. Alongside this, the methods for product development have progressed and are now implemented in an agile methodology, Scrum [3], introducing concepts as *continuous integration* and *deployment*.

In the strive to make product development more efficient, Saab have invested, and prospered, in the concept of *continuous integration*, cutting the development time considerably. In *continuous integration*, the developers merge and compile all developed software with a shared mainline, running several automated tests every day. If the software does not compile or a flaw is discovered, a developer has to attend to the issue right away, all in an effort to keep bugs away. Saab continues to develop new functionality and advanced features even after the system has been delivered, functionality that is offered as different upgrades to the customer. Previously, the software has been hardware-specific, meaning that a software upgrade could result in replacement of the underlying hardware as well. However, in the new development, the entire software should run on virtual machines, making it hardware-independent, allowing upgrades without the need to always replace the hardware. Eventually, the hardware needs to be replaced because it might lack required components or performance. Saab would like to allow a customer to acquire a sort of subscription, to get the latest versions of the software as soon as they have been properly tested and validated, so-called *continuous deployment*. Saab would also like to offer the possibility to add new software features to their systems on demand. That means that a customer should have the opportunity to buy extra functionality if the system is placed in a different scenario than first intended, maybe only for a limited time. A similarity to what Volvo Polestar [4] offers, where Volvo owners can buy new software to their cars in order to get more power out of the engine as well as better fuel efficiency, without changing any hardware. There is however an issue with *continuous deployment* in a military setting. Military systems have high demands on verification and validation of the software, this since it is crucial that the system is functioning and operational in the sudden event of war and that the systems is free of spyware that could reveal troop movements or strategies. Whenever an upgrade or patch is installed, the entire system needs to be reinstalled and reverified. This is a very time and resource consuming process and is therefore not realistic to push out updates every other week.

Another element of *continuous deployment* is customer feedback (logs as well as user input) during testing and operation. This in an effort to work closer to the customer, in order to make the systems more adapted to the operator that uses the system on a daily basis. Today, all the software upgrades and patches are delivered in person. This, since they may contain sensitive information classified as *defence secrets* or as *company confidential*, resulting in unnecessary high costs and long transportation times. To obtain feedback from the customer and identify areas of improvement, Saab demonstrates different functions and interfaces during the development process and also monitors how the operators use the system. *Continuous deployment* could be achieved in the development process by establishing a *customer-lab* environment at the customers site where Saab can push out updates fast and the costumer leave feedback. In such a set-up, the demands on the software verification may not be as high.

Even though a transition to a digital system could significantly reduce the cost and delivery times, both for software and feedback, it could at the same time expose sensitive information to undesired entities, e.g. competing companies, military intelligence agencies, organized crime and terrorist organisations. It furthermore exposes both Saab's internal network and the end-system to the Internet, which introduces several critical threats. As soon as the information leaves the comfort of the internal network (given that the internal network is secure) it can, with the right resources, be recorded, modified, delayed, duplicated or deleted on the way to the destination. Imagine what could happen if someone was able to insert a bug in a patch. A patch that for example is intended for a whole country's radar defence system and the bug makes the whole system crash. Another possibility is that competing companies want to get hold of Saab's proprietary solutions in order to gain a competitive advantage. The military industry could be a very lucrative business where the US defence budget alone is a staggering $585 billion [5], indicating that there are likely some heavy competitors with tremendous resources and motivation.

McAfee reports that, even with a conservative estimate, the annual cost to the global economy from cybercrime and cyberespionage is as high as $375 billion [6]. They also point out that social engineering and vulnerability exploitation are the two most common exploitation techniques. In vulnerability exploitation, the adversary takes advantage of an implementation or programming failure to gain access to a resource or an asset. The fact that many systems today allow older protocols and standards, e.g. old version of

Secure Socket Layer (SSL) and broken cryptographic algorithms, makes such exploits no surprise. In social engineering the adversary tricks, threatens or bribes a legitimate user to grant him similar access. The fact that insiders are such a common exploitation technique, emphasis the need to secure the system from external as well as internal threats.

1.1 Problem Definition

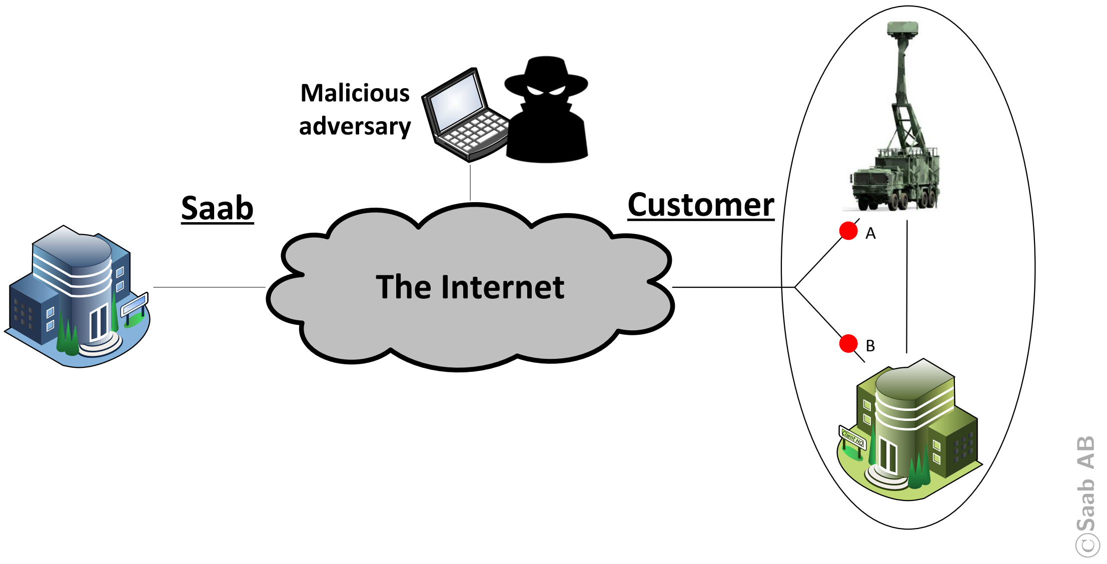


Figure 1.1: An overview of a simple delivery system where the software is transmitted over the Internet either directly to the end-system (point A) or to the customer headquarters (point B). A malicious adversary might however be able to intercept the information.

To achieve *continuous deployment*, physical delivery of information seems to be an outdated and inefficient solution. A transition to a digital delivery system is certainly desired. Considering the information such a system would handle, security is likely the most critical part of the system design. One very important matter is to decide where Saab's responsibility end. Depending on how the system should be used and what kind of equipment it requires, the *point of delivery* could be either at the end-system itself or at the customer's headquarters, where they later are responsible for further transportation to the end-system. Regardless of which solution, verifying that the software developed at Saab is in fact the same one that is installed at the end-system is essential in such a delivery system. Figure 1.1 illustrates such a delivery system, where a customer has some kind of headquarters and a Giraffe AMB radar unit [7]. The customer should be able to receive different patches and upgrades from Saab. The delivery system needs to be designed in such a manor that the threats identified by our threat model in chapter

3 are handled in a satisfying way. Today, there exists no direct link between Saab and the end-system, yielding in some relevant questions;

- Where is the *point of delivery*?
- How is user feedback and user data transmitted back?
- Where does Saab's responsibility end?
- Can Saab guarantee that the installed software is the same as the intended one?

One big issue is that it is hard to know what the adversary's capabilities are, he might have discovered a flaw in an otherwise robust protocol or he might have far superior resources then initially imagined. There is also the possibility that the malicious adversary is not only present on the public network, like previously mentioned, using insiders to obtain sensitive information is one of the top exploitation techniques. Not only may he have the possibility to gain knowledge of the systems properties, he might even be authorized to administrate it.

1.2 Purpose

The purpose of this master thesis is to suggest a methodology for designing a secure system that could deliver military software over the Internet, to evaluate if and how Saab could implement *continuous deployment* into their product delivery and discuss which new possibilities it introduces.

1.3 Contribution

We propose a methodology when designing a secure system for delivery of sensitive data over an uncontrolled medium, like the Internet. The process proposed is based on research concepts in the area of system design. Different solutions to identified threats are proposed and solutions to mitigate or eliminate these are provided. We would like to emphasise that the methods, vulnerabilities and solutions we present are with respect to this specific system and might not be a universal solution. Although, we would like to believe that this thesis is a good inspiration when designing secure systems.

1.4 Limitations

This thesis does not provide a complete system, but rather a methodology that can be used when designing a secure system. Even if the report proposes different solutions to threats identified, the solutions are of a high level nature, and contains several vulnerabilities that need to be investigated further and mitigated before the system can be operational.

1.5 Outline

The rest of the thesis is outlined as follows:

Chapter 2 Explains some background regarding classifications, computer security concepts, protocols and intrusion detection systems.

Chapter 3 Contains our complete proposed methodology for how to design a secure system. Including the systems assets, a threat model and a method for risk assessment.

Chapter 4 Presents and evaluates different solutions that can mitigate or prevent the vulnerabilities.

Chapter 5 Discusses the proposed solutions with respect to different areas of usage and conditions.

Chapter 6 Summarizes the report with our concluding remarks as well as our thoughts on future work.

2

Classifications and Security Concepts

This section gives a brief introduction to different information classifications as well as background to relevant security concepts that are used in the thesis. A reader with knowledge of the common security concepts may go directly to the next chapter.

2.1 Military and Company Classifications

All information handled at Saab has a classification according to *defence secrets*, *company confidentiality* and *export control*. The classification is a statement of how sensitive the information is for Saab and national security. The classification of information is used to determine how the information should be handled as well as who is allowed to handle it.

2.1.1 Defence Secrecy

Information that could endanger Swedish national security is classified as *defence secrets*. This information has strict rules of how it should be handled, specified in Industrisäkerhetsskyddmanualen (ISM) [8] by the *Swedish Defence Material Administration*(FMV) [9]. The different information security levels are **TOP SECRET**, **SECRET**, **CONFIDENTIAL** and **RESTRICTED**, otherwise the information is **UNCLASSIFIED**. Regulations then specify how information in each information level should be handled, which equipment, personnel that are authorized as well as what security measures that must be enforced. Details about some of these regulations and equipment are classified as *defence secrets* themselves and can therefore not be mentioned in this report.

Since Saab also have other customers than the *Swedish Armed Forces*, other countries' *defence secrets* might be involved. All types of foreign *defence secrets* are applied with the same policies as when dealing with Swedish *defence secrets.* This is always with permission and involvement of FMV. FMV is furthermore the Swedish authority that acts as a guarantor for that Saab handles this information in a correct way.

2.1.2 Company Confidentiality

Saab also classify all information with respect to *company confidentiality.* That is information that could harm Saab or its customers, typically financially if proprietary solutions leaks to competitors. Information could be classified as open or in one of three levels of company confidentiality that further specify how sensitive the information is.

2.1.3 Export Control

Information that could be used for military purposes have a classification of *export control.* Export control means that the information is only allowed to be shared with countries that have been approved to read this information. Otherwise, it is not free to transport and share the information outside Sweden. When *export controlled* information is handled, everything needs to be logged; what is exported, from where, to where, who is the sender and who is the recipient.

2.2 Confidentiality

Confidentiality is the concept of keeping the information secret from everyone that is not authorized to read it. In computer security, this is often achieved by encrypting the data to be sent [10].

There are two types of ciphers that are used to encrypt data, symmetric and asymmetric cryptography. In symmetric cryptography, both the sender, Alice, and receiver, Bob, have the same key. This key has to be negotiated between them in some secure way, which is one of the drawbacks with symmetric cryptography. In order to securely transmit the shared key, a secure channel is needed. This could for example be that Alice delivers the key in person to Bob or through another, already established, encrypted channel. Another way to agree on a shared key is to use a key negotiation protocol, like Diffie-Hellman. Here Alice and Bob can agree on a shared key over a public channel without any prior knowledge of each other. This can be done because of the *discrete logarithm problem* [11], which is a well-known mathematical problem for which no efficient general methods of calculating is known, atleast not with conventional computers.

In asymmetric, or public key, cryptography there are two keys, called a key pair. One is for encrypting the information, the other is for decrypting. In this scheme Alice generates a key pair, keeps one of the keys to herself, the private key, and makes the other key public. Bob, or anyone else for that matter, can encrypt information with Alice's public key and transmit it to Alice. She will then be the only one who can decrypt the message, with her private key. One issue that public key cryptography does not address on its own is: how can Bob know that the public key belongs to Alice? This is typically solved with a so-called Public Key Infrastructure (PKI). In a PKI, a trusted authority manages, verifies and revokes public keys, making it possible for Bob to acquire and trust that the public key belongs to Alice.

2.3 Integrity

Integrity is the concept of keeping the data intact during the transfer. This means that an adversary should not be able to intercept and modify a message from Alice to Bob, without Bob's knowledge [10]. One way to achieve this is with a method called signing. In signing, Alice first calculates the hash of the message, the message digest. Next, the message digest is encrypted using asymmetric cryptography, the same scheme is used as described in the previous section, only that it is reversed. This means that Alice encrypts the digest with her private key. The now encrypted digest is transmitted along with the message to Bob. Bob also calculates the hash of the message and compares it to the digest he received from Alice, decrypted with Alice's public key. If the digests match, Bob can trust that Alice is the author of the message and that the message has not been modified. This means that anyone with access to Alice's public key can verify that she is the true author of the message.

2.4 Availability

The concept of availability is to ensure that users will have access to a resource [10]. The more critical a resource is, the higher is the level of availability required. If the delivery system is used for updates that contains different mitigations to publicly known weaknesses, a delay of such updates could result in severe consequences. However, if the system is only used for less critical updates, such as cosmetic updates, the consequences might not be as severe and the availability requirements might thereby be lower.

One of the biggest threats to availability is if someone deliberately want to deny users access to a resource. A common way to achieve this is by launching a so-called Denial of Service (DoS) attack, flooding the target network with traffic. The routers and firewalls could then be overwhelmed when their buffers are filled with malicious packets, rendering the resource unavailable to the legitimate users. However, one host is often not enough to generate the required amount of traffic, therefore the attacker utilizes more hosts in a so-called Distributed Denial of Service (DDoS) attack. These other hosts often belong to companies or regular users, infected with a virus or trojan, letting the attacker generate traffic to his target of choice.

2.5 Authenticity

The concept of authenticity is to be able to positively identify a person, device or other entity in a computer system, often in order to grant access to some resource. This process is typically done by verifying some identity with a pre-shared secret, e.g. a username and a password. O'Gorman [12] defines two different types of authentication.

First, in machine-by-machine authentication (*machine authentication*) one machine, or system, authenticates itself to another machine, often using well-established protocols such as SSH or TLS. The process may include verifiable certificates as well as long keys to achieve strong security. This however, only authenticates the machine, not the user, which is typically desired, especially in a systems with sensitive information.

Second, user-by-machine authentication (*user authentication*) is when the user authenticates himself to the machine, traditionally with a memorable password, thus completing the authentication chain. The issue however is that its hard to achieve really strong user authentication. Humans have a tendency to choose really short and simple passwords, because they are easy to remember. But, they could on the other hand be relatively easy to obtain by an adversary. Furthermore, users tend to use the same password for multiple logins, e.g. computer login, social media, subscription sites, etc. []. This means that if an adversary can obtain your password from a site with low security, he might also be able to access other resources, potentially with even more critical information.

2.5.1 Password Authentication

System administrators have acknowledged the issue with weak passwords and many have enforced stronger password policies. These policies can for example be that the passwords are required to be of a specified length, contain upper- and lower case letters, digits and symbols, and be changed after a specified number of days. These stronger passwords are more resilient against brute force and dictionary attacks, but they are also more difficult for the user to remember, which is also the drawback of the policy. When the password policy is too strict, users find ways to circumvent the policy. For example, the user writes down the password on his phone or on a note next to the computer, or the user utilizes a password similar to those he used before, only that he adds a few numbers after it and then simply just increment that number every time he is prompted to change the password.

2.5.2 Biometric Signatures

Biometrics include human features like fingerprints, voice, face and retina [13]. A biometric signature is easy to use in the sense that it does not require the user to remember a particular passphrase or require him to carry something with him. On the other hand, a system based on biometric signatures must be able to differentiate one persons signature from another with high certainty. If several people have the same biometric characteristic, for example length, the authentication method cannot be considered secure. In the same manor a biometric feature cannot be misplaced, it cannot be changed if stolen. That is, if someone manages to obtain the signature of your biometric, he might be able to reuse it to access other resources.

2.5.3 Security Token

A security token, or simply token, is a device that is mobile and the user can carry with him. A token enables the use of protocols similar to machine authentication, typically a challenge-response protocol. Here the authenticator sends a random bit sequence, called a challenge, to the user that is to be authenticated. The sequence is then fed to the token which then encrypts the sequence with a pre-shared secret key, that may be of an extensive length, far longer than a human would feel comfortable to remember. The encrypted data, called a response, is then transmitted back to the authenticator, who compares the received sequence with its own response of the challenge encrypted with the pre-shared key. If the responses match, the user is considered authenticated, if not, he is rejected.

2.6 Authorization

Authorization, or access control, states which user has access to which resources [10]. In high security systems it is very important that a user only has access to exactly the resources that is needed. Access can be restricted in different ways, mechanisms can be applied to keep unauthorized users completely out of the system while letting authorized users through. Access policies can be implemented to grant users access to certain resources based on identity, role our security clearance.

2.7 Accountability

Accountability is the concept that the actions of users in a system are logged and can be traced back, that users could be held accountable for their actions [10]. Accountability provides protection against non-repudiation, where a user claims he did not perform an action, when he in fact did. If the system fulfils the accountability concept, then there is an easy way to determine what each user has done. Accountability does not provide authentication, it must therefore rely on the authentication concept already used in a system. If authentication cannot be guaranteed, the accountability property fails.

2.8 Network Protocols

There are different kinds of protocols for transfer of data over the Internet. Different protocols provides different features, such as encryption, authentication and integrity. In this report we will discuss TLS, SSH and IPSec.

2.8.1 TLS

The *Transport Layer Security* (TLS) protocol, is one of the common security transportation protocols. The protocol provides confidentiality, integrity and authentication [10].

For authentication, TLS uses certificates. The certificates are issued and signed, with public key cryptography, by a Certificate Authority, CA [10]. Public keys of trusted CAs are built into the users web browsers so that certificates issued by those CAs can be verified. These certificates can also be used in a certificate chain, where the signatures can be followed in a certificate chain until a trusted CA is found. Then the certificate is considered valid and the browser trusts the server. If a CA further down the chain starts issuing certificates to false servers, the other CAs should revoke their certificate for that CA, deeming it untrustworthy.

For confidentiality, the communication is encrypted with a symmetric encryption algorithm using a shared session key. Integrity is provided by a one-way cryptographic hash function, also using a shared session key. The session keys are negotiated in the handshake that are exchanged during the initialization of the communication. The handshake is used for establishing the shared session keys and negotiating on different settings used, such as which encryption algorithm, compression algorithm and *Message Authentication Code* (MAC) algorithm. Both the server and client have lists of the protocols and algorithms they support and prefer, called the cipher suite. The client then sends its cipher suite to the server, who chooses the first preferred set-up they both support.

2.8.2 IPSec

IPSec is an end to end security scheme operating at the Internet layer of the OSI model [14]. It encrypts and authenticates each IP packet in a communication session. The protocol establishes mutual authentication and negotiate on shared sessions keys. IPSec can be used for securing communication between both hosts and networks, i.e a host to host, network to network or network to host communication. It supports network level peer authentication, data origin authentication, data integrity, data confidentiality and replay protection.

IPSec supports two different modes of implementation, tunnel mode and transport mode. Tunnel mode is used for creating VPN for all types of communications. In the tunnel mode the entire IP packet is encrypted and authenticated before it is repacked in a new IP packet and header. In transport mode, only the payload of the IP packet are encrypted and authenticated, the rest of the packet are unchanged.

2.8.3 SSH

SSH (Secure SHell) is a protocol that was developed for initializing text based shell sessions on a remote host securely. This means that a user can connect to a host securely via an insecure network. Often SSH is used for running command lines on a remote computer, but it can also be used to create a secure tunnel to transmit data over a insecure network [15]. Often SSH servers works with a PKI system, where the servers have public keys that are distributed to the users so they can authenticate the server, called host keys. Often the user is authenticating himself with username and password, but a PKI system is also supported. The communication is encrypted and the different settings are negotiated in similar ways as in TLS.

2.9 Intrusion Detection

Today, most computer systems need to communicate with other systems, information needs to be exchanged, especially in the corporate world. This is typically done by interconnecting all resources, including computers, servers and printers, within the organisation in a local network. Often, being connected in a local network is not enough. information needs to be exchanged with other organisations around the world. The local network is therefore connected to a external network, i.e. the Internet. The information handled inside the local networks is often sensitive (for example business strategy, research and development, and product details) and must, for competitive reasons, be kept secret. Without any protection mechanisms, any user connected to the Internet can access all assets on the organizations local network. There are several techniques to prevent this kind of unauthorized access, such as firewalls and *Intrusion Detection and Prevention Systems* (IDPS).

2.9.1 Firewalls

One of the most common protection devices in networks are firewalls. A firewall is a filtering device that is placed between two networks, one trusted zone and one untrusted, typically between a local network and the Internet, or between networks with different information classification [10]. The firewall inspects the passing traffic, comparing the packets' headers with a given set of rules. If the packet does not match any rule, it is silently dropped, or discarded with an error message to the source. These firewalls typically check source and destination port numbers and IP addresses, if the port number matches the protocol and if the IP address belongs to the ”inside” or the ”outside”. The rules could be added so that some protocols, ports and addresses are forbidden to pass through the network.

More advanced firewalls have additional capabilities, to not only screen the packets, but also to monitor the state of the connections, a so-called stateful firewall. These firewalls record all connections passing through and can therefore determine whether a packet is a new connection, belongs to an existing connection, or not part of any connection. For example, the three-way handshake in TCP consists of three packets. The firewall will only let the third packet through, if it has seen the first two, even though the packet itself is valid.

Even more advanced firewalls are also available. These firewalls are referred to as application layer firewalls. They have an even deeper understanding of the protocols and applications than the stateful firewall. The application layer firewall may detect if someone is trying to bypass the firewall by abusing a protocol or trying to reroute the traffic to a port not normally used by that particular protocol. These firewalls can be tailored for certain protocols, for example the web application firewall which specifically monitors HTTP/HTTPS traffic for abnormal behaviour, and could be placed in a hardware device or at the different hosts in the network.

2.9.2 IDPS

Other common protection mechanisms are the *Intrusion Detection and Prevention Systems.* An IDPS refers to two different subsystems. An *Intrusion Detection System* (IDS) is a device or an application that unlike the typical firewall, inspects the whole package, i.e. both packet header and payload [10]. An IDS has two general detection modes, signature-based detection and anomaly-based detection. In signature-based detection the IDS has access to a database with known attack patterns, signatures. In anomaly-based detection, the system is first put through a training phase, where it is fed with traffic it should consider to be normal. In operation it will then react to abnormal traffic. When the IDS detects an attack, the system simply sends an alert and logs the event. It is then up to the network administrator to review the event and take some kind of action.

An *Intrusion prevention system* (IPS) contains all capabilities that an IDS does and typically also operates in the anomaly-based or the signature-based mode. Although, when an attack is detected an IPS has the ability to prevent the ongoing attack, for example by closing the connection, logging out the user or even shut down the entire network. Making it a mixture of a firewall and a IDS, even though it should be noted that the gap between a firewall and an IPS has grown smaller. Where firewalls are getting more and more features that the IPS have, especially the application firewalls, which inspect more than just the header of the packet. The difference is that the firewall operates policy based, while the IPS is pattern based.

3

Threat Model

When developing a secure system, identifying all the threats is a crucial part of the process. It is almost impossible to tell when all threats against a system have been discovered. In order to identify as many as possible, the threat modelling needs to be performed in a thorough and systematic way. By using a structured method to find threats, it is less likely to miss any serious ones. Myagmar et al. [16] concludes that:

> *Despite the best efforts of security researchers, it is impossible to guarantee 100% security. However, we can work toward 100% risk acceptance.*

Risk acceptance is when a threat or vulnerability in a system is known and accepted, but no further preventive actions are taken. To know when to accept a threat or not, different methods can be used in order to help the designer evaluate each threat in a systematic and fair way. Maygmar et al. also argues that for each threat identified there is a choice of;

> **Accept the risk** - The risk is so low and so costly to mitigate that is it worth accepting.
> **Transfer the risk** - Transfer the risk to somebody else via insurance, warnings ect.
> **Remove the risk** - Remove the system component or feature associated with the risk if the feature is not worth the risk.
> **Mitigate the risk** - Reduce the risk via countermeasures.

When designing a system, security should be integrated in the design from the beginning. To add security to an already existing system with no previous security is hard to do. The vulnerabilities still discovered in the connected car [17] is a deterrent example of a system that suddenly became exposed to the Internet without adequate security.

Microsoft have developed a methodology [18], containing several steps and methods for designing and evaluating systems with security in mind. This Microsoft methodology is however, like most threat models we have encountered, more applicable when evaluating an existing system or the system already has a basic design. As one of the steps is to decompose the application, where the designer, for example, have to reflect on which keys are used for encryption and which encryption algorithms to use, the methodology suggests that these decisions already have been made. There is no step where the designer can evaluate different solutions and decide which techniques suits this particular application best.

Therefore, we have modified the process to better suit our purpose, which is designing a secure system from scratch. We have kept the basic contribution of each building block from Microsoft's methodology, but have somewhat rearranged the order and in some case the usage. This allows the designer to, instead of decomposing the application to find its components, build the system by choosing the different components.

The process has five steps and are as follows:

1. Identify Assets
2. Architecture Overview
3. Identify Vulnerabilities
4. Vulnerability Mitigation
5. Risk Assessment and Acceptance

These steps are basically the same as in the original Microsoft process only that their *application decomposition step* has been replaced with a step that allows different solutions to the identified vulnerabilities to be surveyed, *step 4*. Our approach also introduce an iterative process where the steps are revised during the design as additional information is added and more knowledge about the system is acquired, see figure 3.1. For example, when deciding on an encryption method, the key management introduces new threats and vulnerabilities that needs to be further analysed and mitigated.

When performing this threat analysis, our potential adversary is believed to have great computational power, resources, personnel and motivation. As some of the information handled by the system could be of great value for some parties, the motivation for an attack could be extensive. Potential parties with interest of attacking the system could be military intelligence agencies, competing companies, organized crime and terrorist organisations. However, as mentioned in section 1, insiders are one of the two most common exploits in computer security, adding the insider to the list of potential adversaries.

3.1 Identity Assets

The first step is to identify the assets of the system, that is the parts of the system containing sensitive information that could be of interest to an attacker. The assets of our system in this initial iteration are:

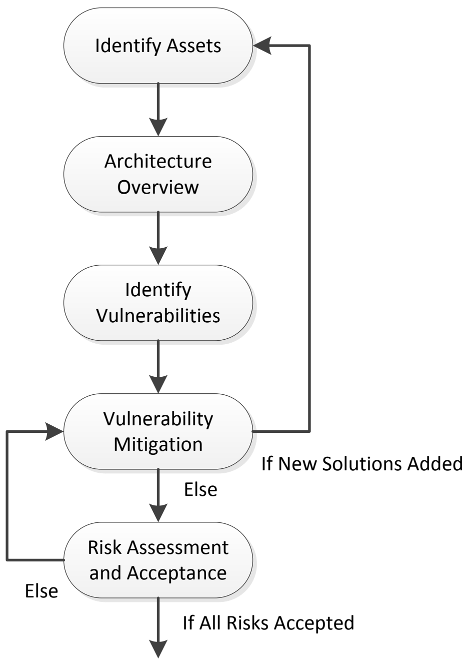


Figure 3.1: Flowchart of our proposed process

- Users credentials - The users of the system, both at Saab and the customer side, have access to sensitive information. If an attacker has access to valid user credentials, the sensitive information may be accessible.
- Saab's internal network - The internal network contains sensitive information about products, strategies, financials and customers. If an attacker gets access to Saab's internal network, the sensitive information may be accessible.
- Network equipment - The network equipment, such as servers and routers, could also be of interest to an attacker. This since the servers potentially contains information of interest. A vulnerable router could direct passing information or requests to the attacker.
- Data - The software sent to the customer and the feedback transmitted back from the customer or end-system. This information could be of a sensitive nature since it contains product-related information as well as information about how it is operated.
- The end system - The end-system contains sensitive information and could be a vital part of the customers defence capabilities, acquiring information or access to such a system could have devastating effects.

3.2 Architecture Overview

An architecture overview allows the designer to get an easy overview of the system in order to understand information flows, trust boundaries and points of weakness. The initial system architecture can be seen in figure 3.2, where Saab connects to the customer or end-system over the Internet. The information is transmitted from Saab, to the *point of delivery* and feedback is transmitted back.

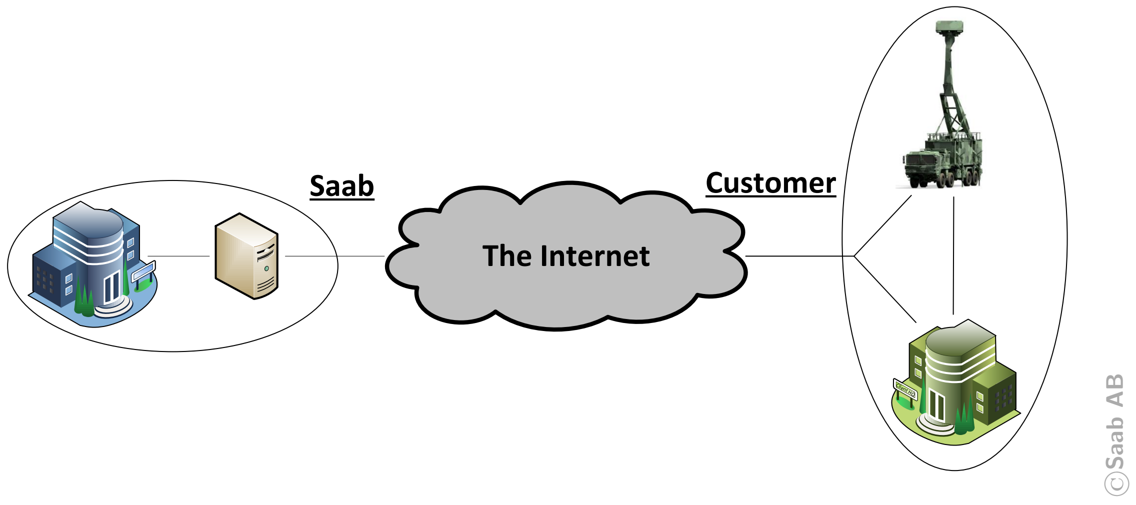


Figure 3.2: Initial architecture overview of the delivery system.

3.3 Identify Vulnerabilities

STRIDE [19] is a method developed by Microsoft used to, in a systematic way, identify vulnerabilities a system could experience, in order to mitigate them before they are exposed. There are six different categories and together their first letters forms the name STRIDE. The categories are defined as follows:

- Spoofing Identity - Pretending to be someone you are not. For example someone pretending to be an authorized user with stolen credentials, or a malicious computer pretending to be a friendly server.
- Tampering with data - Malicious modifications of data. An unauthorized change to the data in any way. Either as cipher or plain text.
- Repudiation - Repudiation is the concept that each user of a system should not be able to deny any action they have made. This means that another party should be able to prove that an illegal action has been performed even if the user performed it denies it.

- Information Disclosure - Unauthorized users getting access to information they are not supposed to have access to. This could be either a legit user of the system or a adversary with malicious intent.
- Denial of Service - When service to valid users are denied. This could involve making a server unavailable or denying access for users to the system.
- Elevation of Privileges - When a user gets more access then intended to. The user could become a part of the trusted system itself, and thereby have the rights to modify the entire system.

STRIDE	Vulnerability
Spoofing Identity	• Pretend to be a customer • Pretend to be a Saab user • Pretend to be a system administrator • Pretend to be network equipment (server, router, etc.)
Tampering with Data	• Modify data during development • Modify data on the sever • Modify data in transit • Modify data after delivery
Repudiation	• Saab denies an action has taken place • Customer denies an action has taken place
Information Disclosure	• Acquiring information from data in transit • Acquiring information from data on server • Acquiring information before it is uploaded to server • Acquiring information after it is downloaded from server
Denial of Service	• Saab cannot access its server • Customer cannot access Saab's server
Elevation of Privilege	• Saab user gets more access than supposed to • Customer gets more access than supposed to • External party gets more access than supposed to

Table 3.1: Table of vulnerabilities categorized with STRIDE

Table 3.1 presents our initial threat model categorized by STRIDE, where different vulnerabilities are shown. These vulnerabilities lay the foundation of the system. For each solution to the identified vulnerabilities, STRIDE will need to be applied again, since all solutions that are proposed may yield in new vulnerabilities that needs to be identified.

3.3.1 Spoofing Identity

Pretend to be the Customer/Saab user/system administrator - If an attacker is able to fool the system into thinking it is dealing with a legitimate user, the attacker could get access to files and information he is not authorized to. This could be a result of poorly implemented access control or if valid login credentials have been obtained. Spoofing different type of users could result in consequences of various severity. This since a Saab user is likely to have more privileges in the system, for example, Saab users may be involved in several projects and could have the clearance to upload new files to the server. To spoof a system administrator is an even more serious event since system administrators have the highest privileges in the system and probably have access to all files on the server.

Pretend to be network equipment - The attacker could also spoof routers, servers or computers. This could, depending on the target, fool legitimate users to upload the feedback to the adversary's server or have them download malicious software, for example by spoofing a *Domain Name System* (DNS) server.

3.3.2 Tampering with Data

Unauthorized modifications of the data could potentially occur during any step from development to delivery. The modification could render the data corrupted or could insert bugs and backdoors.

Modify data during development - The modification could be performed by either an insider or an outsider. If a developer inserts malicious code, it could be very hard to detect, as the system could be complex and the insider could be individually responsible for a certain part. Even an outsider could manage to get access to the information, either feedback or code under development, for example if the developers are using an insecure network.

Modify data on the server - Since the server is exposed to the Internet, there is always a possibility that it is accessible for an adversary or other users that might be able to modify stored data. It could also be altered by authorized user, by mistake or intentionally.

Modify data in transit - As the Internet is an open network as well as an insecure communication medium, the information could be exposed to modification.

Modify data after delivery - Depending on how the receiver stores and manages the data, an attack could be more or less advanced. As one might not expect an attack to occur inside the comfort of the own organization, it is therefore possible that an attack is undetected and malicious software is installed or faulty feedback is processed.

3.3.3 Repudiation

Saab/customer denies an action taking place - In the event that something has gone wrong in the system, neither Saab nor the customer should be able to deny an action that they in fact have performed. For example that Saab denies ever uploading a file or the customer denies downloading it.

3.3.4 Information Disclosure

The information transmitted with this system could be of varying sensitive nature. A small update to a system, in order to eliminate a bug, may not contain any sensitive data at all, and the update alone may not leak any information about the system or customer. A larger update however, that adds new functionality to a system, might contain more sensitive data and provide knowledge about the system and the customer. The feedback could also contain sensitive information such as system logs.

Acquiring information from data in transit - When the data is transmitted over the Internet, an adversary could eavesdrop on the conversation, acquiring knowledge about different products and customers. The protocols used for transmitting the data could also leak information, or have some weakness that might allow an adversary to gain access to sensitive information.

Acquiring information from data on server - By getting access to the information on the server, a potential adversary could acquire a huge amount of information about which system that are used by different costumers, how they are built and operated. Knowledge about sever details, which *operating system* (OS) and services running, could give an attacker the possibility to exploit a weakness and thereby getting access to the server.

Acquiring information before it is uploaded to server - An adversary could acquire knowledge about Saab's development processes. Different solutions and ideas could leak to rivalling companies, decreasing Saab's competitiveness.

Acquiring information after it is downloaded from server - The level of security at the customer is something Saab do not control or have much knowledge about, making it difficult to determine whether the information is kept secure or not. Although, one might assume that military organisations have a high level of security.

3.3.5 Denial of Service

There are several ways an attacker can perform a denial of service attack, all have the goal of denying the service to the legitimate users. There are two different types of attacks, hard and soft. Hard attacks are physical attacks against the hardware, such as cutting the power or destroying the sever. Natural disasters such as flooding or earthquakes are also included in this category. Soft attacks are attacks against the software in the different parts of the system, such as firewalls, databases, routers and servers.

Saab/customer cannot access the server - The attacker could prevent Saab from uploading software to the server or deny the customers access to it. As the delivery process is usually not time critical, the threat is of low priority, although if the software must be delivered urgently, a denial of service attack could be devastating. Nonetheless, it is not desirable in a system and measures should be taken to prevent it.

3.3.6 Elevation of Privileges

Poorly implemented access control could, by mistake or if attacked, may grant someone access to restricted resources. It is particularly serious if a customer or especially an external party gets elevated privileges. This could mean that a user could view, create or delete software intended for another customer or getting access to the Saab's internal network or even the end-system.

Saab user gets more access than supposed to - Saab user gets access to information they do not have clearance for.

Customer gets more access than supposed to - Customer could get access to other customers software and the internal network.

External party gets more access than supposed to - An external party, that should not have any access to the system, somehow gets access to the system. The privileges has thereby been elevated from nothing to something.

3.4 Vulnerability Mitigation

This is the step that we have introduced that lets the designer provide solutions that eliminates or mitigates the previously identified threats. The different mitigation techniques are presented in the next chapter. The idea with this step is to find one or several solutions to the vulnerabilities and thereafter compare the different advantages and disadvantages in order to decide which is more suitable and should be included in the next iteration. In some cases, it might be difficult to choose between two different solutions that provides similar features, as possible advantages or disadvantages might only be discovered later in the iteration. Therefore it is possible to keep both solutions in the process and make a decision later, when more details about the system is defined.

3.5 Risk Assessment and Acceptance

To be able to evaluate each threat or vulnerability in a fair way, the method must be performed equally for each threat. The purpose with threat evaluation is to be able to accept or mitigate each threat in order to achieve 100% risk awareness. DREAD [20] is a risk assessment model also developoed by Microsoft in order to evaluate threats by giving points in six different categories. Each category has a scale of 0 to 10 to grade the attack, by taking the average score of each threat, they can be prioritized. Similar to STRIDE, the name is from the different categories the threats are evaluated by.

- Damage Potential - How bad would the damage be from a successful attack.
  0 - Nothing
  10 - Complete system or data destruction

- Reproducibility - How easy is it to reproduce the attack.
  0 - Very hard or impossible
  10 - Just a web browser and the address bar is sufficient, without authentication.

- Exploitability - What is needed to exploit the attack.
  0 - Advanced programming and networking knowledge, with custom or advanced attack tools
  10 - A web browser

- Affected Users - How many users are affected.
  0 - None
  10 - All users

- Discoverability - How easy is it to discover the attack.
  0 - Very hard to impossible; requires source code or administrative access
  10 - Details of faults like this are already in the public domain and can be discovered using a search engine.

As this thesis does not evaluate any threats with this model, an example is provided, table 3.2, in order to give an idea of how to use it. The threat to be evaluated is if the encrypted information transported with TLS is vulnerable to an attack where the attacker wants to access the information, i.e break the encryption. In order for the evaluation to be possible, some assumptions have been made. The algorithm for encryption is AES-256, the key negotiation protocol is 1024 bits Diffie-Hellman where the prime and the generator (described in section 4.4) are chosen carefully and replaced regularly.

DREAD	Score	Comment
Damage Potential	8	The information sent could contain sensitive data.
Reproducibility	1	As presented in section 4.4 with state-level capabilities, it would take about 30 days to break the encryption.
Exploitability	2	State-level capabilities.
Affected Users	10	All users are affected.
Discoverability	10	The attack is well-known among security experts
Average score	6.2	

Table 3.2: DREAD performed on the threat to decrypt information transported with TLS.

The evaluation of the example threat results in an average score of 6,2 in DREAD, which is rather high. It should however be noted that state-level capabilities are required and even they must put in a tremendous amount of resources to pull it off. From this point, there is a choice of either accept, transfer, remove or mitigate the risk.

Trying to mitigate the risk means a new iteration with a new solution, meaning that either the algorithms and key lengths are exchanged or that TLS altogether is replaced by a more secure protocol. But even so, many of the DREAD parameters would stay the same and the result may still not reach a desired level. If the risk cannot be accepted and no option to transfer the risk exist, the risk will have to be removed.

4

Vulnerability Mitigation and Elimination

In order to mitigate or eliminate the threats identified in Chapter 3, different solutions are proposed. The solutions are based on well known practices and modern research in the field of computer and network security. Some vulnerabilities may have several possible solutions, each with its own advantages and disadvantages.

4.1 Countering Identity Spoofing

Some very serious threats to a system are different spoofing attacks. If an attacker is able to fool the system into thinking it deals with an authorized user, it might let him perform lots of different actions that could harm all the legitimate users in the system.

4.1.1 User Authentication

One common way to mitigate identity spoofing, Saab user, customer and system administrator, is through user authentication. The stronger the authentication, the harder it is for an attacker to spoof the system. Some times a simple password is not strong enough and a security token, such as a RFID tag, could be misplaced and misused by an adversary. The concept of two-factor authentication strengthens the authentication by requiring two factors, e.g. both the token and the password. Two-factor authentication is widely adopted in the industry and is for example employed in bank applications and is the minimum requirement when dealing with *company confidential* information at Saab. An example of such a scheme is the electronic identification application *BankID* [21], which is used to verify identities online, for example to a banks web portal. The application is downloaded to a Smartphone, the token, and is protected with a *Personal Identification Number* (PIN), the password.

For systems with even more sensitive information an even stronger authentication scheme may be opted for. In three-factor authentication, the previous mentioned scheme is strengthened with another layer of security, the BankID application could for example be strengthened with a biometric signature, something you are. Meaning that the scheme contains a password, a security token as well as a biometric signature. As mentioned in section 2, a password can be changed, a security token can be replaced, but a biometric signature cannot. It is therefore vital to store the biometric signature in a secure way, so that it is not leaked to an malicious adversary. The problem is that the biometric capture devices are remotely located. Most three-factor authentication schemes lets a smart card perform the biometric validation and thereby preserves the privacy of the user. But on the other hand, forces the servers to trust the smart cards to properly handle the task [22]. However, if an attacker manages to skip the validation process in the smart card, how can the remote server know? Ideally, the remote server should be the one responsible for the biometric validation, but due to its sensitive nature, users may not be willing to reveal biometrics to the servers.

Fan and Lin [22] proposes a solution where the remote server actually conducts the validation, but without being able to extract the biometric characteristics. In their solution, in short, the user registers to a service and combines his biometric signature with a random string, so that the server does not get the signature in clear text. In this scheme, the server will not store any sensitive data about the client but it will still be responsible for the validation check.

4.1.2 Machine Authentication

It is furthermore important that the machines in the system gets properly verified, for example servers and terminals. The client needs to know that it is talking to the correct server. A common method to initiate such a connection is via TLS, where certificates are used to verify the servers identity. In another method, used in SSH, the server can be authenticated with a so-called server fingerprint. A fingerprint is a generated bit sequence, based on the servers public key. If the key changes, so does the fingerprint. This means that for an imposter to replicate another server, he must use the same public key as the server, which means that he must know the servers private key as well in order to decrypt incoming messages and requests.

The three-factor authentication scheme by Fan and Lin could also be used for this purpose as it authenticates the user as well as the server.

4.2 Countering Data Tampering

Data tampering, or unlawful modification of data, is a highly unwanted scenario in a military system. Since Internet is the transportation medium, the data is naturally susceptible to tampering.

4.2.1 Transmission Integrity

In order to detect unauthorized modification of the data during the transmission, the concept of singing is applied. However, it does not protect against the modification itself, as the signature only can indicate that a modification has not occurred. There are different protocols and algorithms to choose from when implementing signing, each with different attributes and features. In most secure transmission protocols, like TLS, SSH and IPSec, integrity validation is a part of the protocol and does not need to be implemented additionally. There are however some attacks that can be carried out against the signing scheme. The *Birthday Attack* [23] is an attack against the hash function used when signing the data. Since there might be collisions in the hash table, other messages might result in the same signature. Although the chance of finding such a message is very low if a good hash function is used.

Another attack that can be carried out is a so-called *Data Diddeling* attack [24]. Here, the data is replaced before it is signed and the signature will therefore be considered valid. This requires the attacker to be able to replace or modify the information earlier in the delivery chain, before it transmitted over the Internet.

4.2.2 End-To-End Integrity

Data Diddeling attacks or the fact that the data can even be replaced after the transmission requires something more than just validating the transmission, since it is crucial that the correct software is running on the end-systems. A way to achieve a sort of *End-To-End* integrity is, according to Haldar et al. [25], to apply the concepts of *Trusted Computing* and *Remote Attestation.*

Trusted Computing uses both hardware and software that verifies the integrity of a system, assuring it has not been tampered with. To achieve this, a device known as a *Trusted Module*(TM) is inserted in the system. The TM contains the private key of a asymmetric key pair as well as a signed hash of the BIOS. At boot-up the TM recomputes the hash of the BIOS and compares it to its stored value, if they match the BIOS has not been tampered with and the TM can pass along the control to it. The BIOS performs a similar check before control is passed to the boot-loader and then the operating system and applications. Thus achieving a *certificate chain*, originating from the TM and verifying each layer of the system up to the applications. If all checks out, the system has not been tampered with since the last reboot.

Remote Attestation is further the concept of authenticating the software to remote parties. The entire *certificate chain* is transmitted to the remote party, for example Saab or if the responsibility has been transferred to the customer, which can verify each step of the chain. This, to verify that the intended software is running on the end-system and thereby be able to make certain assumptions about the behaviour of the software. It furthermore lets the remote party pinpoint in which step of the *certificate chain* a modification has been applied. With these concepts, the software currently running on the end-system can be verified before transmitting any new patches or upgrades and the end-system can on its own, verify that the software has not been tampered with since the update.

One problem that Haldar et al. has identified with this *End-To-End* integrity scheme is that patches and upgrades could be hard to handle. Since the end-systems in our case is quite complex, containing several subsystems that all can run different versions of software and that patches or upgrades can be applied to only small parts of the system, possibly in an arbitrary order, the possible configurations could be countless. Since all these configurations might be acceptable configurations, the remote party needs to store all the different hashes of them so that they all can be verified when performing the *Remote Attestation* which could result in exponential possibilities for any given end-system.

4.3 Enforcing Non-Repudiation

It is important to be able to trace back events in a system in case of a dispute. The customer or Saab should not be able to deny having participated in a communication, so-called non-repudiation. The receiver wants a non-repudiation of origin evidence of the information and the transmitter wants a non-repudiation of receipt evidence in return, a so-called fair exchange. It is furthermore important not to only have non-repudiation between the organizations, but also inside the organizations. Who did what, when and at which computer. It is important that sufficient information is gathered and logged in each step, so that if the data has been modified, replaced or deleted it should be possible to trace it back to when it happened, if it where an insider in either organization, hijacked when transported over the public network or just lost due to a network error. According to Hole et al. [26] a *Trusted Third Party* (TTP) is essential to provide fair non-repudiation. A TTP is an entity that collects, validates, timestamps, signs and stores relevant information that both organizations can trust. The TTP will, in case of a dispute, weigh in the evidence and take a decision in favour of one of the parties without ambiguity.

An example of an application without a TTP is the Norwegian *BankID*, where non-repudiation is desired between the customer and the Norwegian banks. *BankID* is owned by the Norwegian banks, giving them a large amount of control. Hole et al. illustrates this problem with the following example. Assume that a *BankID* customer and his bank have both digitally signed a specific document, for example a document on the bank's Internet application. If it later arises a conflict of some sort between the two, it is up to the customer to show that the bank also have signed the document.

4.4 Enforcing Information Disclosure

The sensitive nature of the information in the system puts high requirements on the information disclosure property. Such information needs to be protected in every part of the system, during as well as before and after the transmission.

4.4.1 The transmission

As the information is transmitted over an insecure network, the Internet, there are several ways an adversary could intercept the data. It is therefore important that any potential data that could be captured is incomprehensible for an unauthorized user, known as the concept of confidentiality. A common way to achieve this is to through encryption. There are several different techniques and protocols available for transporting data in a secure way over the Internet. They can be divided into two general groups, hardware and software encryption. Software encryption consists of protocols like TLS, SSH and IPSec that do not require any specific hardware. Hardware encryption on the other hand, as the name suggest, requires a specific *Hardware Security Module* (HSM) for both encryption and decryption. Both types of techniques can support additional functionality besides encryption, such as signing and authentication. The encryption is in both cases often based on symmetric key encryption, this since it is a lot faster than asymmetric encryption at encrypting big chunks of data.

A HSM is a stand-alone device that can be placed on the network link between a private network and an insecure one. All the traffic that flows through the HSM will be encrypted and thereby tunnelled over the insecure network to the other side where another HSM can decrypt the data, a so-called end-to-end encryption. HSMs are typically used when encrypting highly sensitive information. The cryptographic algorithm within a HSM can vary from a standardized one, like the *Advanced Encrpytion Standard* (AES), to a custom made, secret algorithm used by the military. For example, to transmit information over the Internet, that according to Swedish *defence secrecy* is classified as **RESTRICTED** or above, requires the usage of a HSM with specific algorithms. This since a HSM better can, compared to a software application, keep the cryptographic keys secret, even in the event that the device is stolen or otherwise obtained by a malicious entity, for example the for example the SecuriVPN ISA [27] which is a commercial version of a HSM used by the *Swedish Armed Forces*. One downside with a HSM is that due to fact that special hardware is required, the solution becomes less mobile. The HSM needs to be carried out in the field together with the end-system. There is also a cost and delivery time associated with setting up this solution for each customer.

Another issue is the key management. In a HSM, a master secret is used to derive keys for each session. This so that even if the master secret is obtained by an adversary, it should not be possible to decrypt previous sessions, or future ones, known as forward secrecy. However, with the master secret, the adversary could be able to impersonate the server, which requires the master secret to be replaced on a regular basis. The master secrets need to be generated and then distributed securely to all HSM in the network.

The other type of encryption technique, software encryption, could also be used as an end-to-end encryption solution. TLS is one of the most common protocols and it supports a variety of ciphers, for example AES. TLS is used to encrypt traffic between a server and a client and does not need any key distributions, since the keys are mutually agreed on between the parties in each session, known as the key negotiation. A very popular key exchange protocol, also used by several other protocols like SSH and IPSec, is *Diffie-Hellman* (DH) [28]. Since new keys are generated in every session, TLS with DH supports forward secrecy. However, it has recently been discovered that the key negotiation protocol has some in particular severe flaws, known as the *Logjam* attack by Adrian et al. [29]. In short, the server selects the DH parameters, a prime (p) and a generator (g), which are known and transmitted in clear text. The server and the client then chooses a random, private number respectively. With the known p and g, both parties will perform some calculations and then exchange some public values. After another round of calculations, they will both agree on a mutual shared secret, the session key. Any adversary that monitors the communication between the participants will not be able to extract the shared secret, or atleast that is the idea. The adversary want to extract one of the private numbers, so that the session key is no longer secret. The best current technique is to compute the discrete logarithm of the public values exchanged between the participants. This however, is a very difficult and time consuming process that cannot be done to break connections for an arbitrary session in real-time. The problem is that it has become common practise to use standardized or widely shared DH parameters, allowing a powerful enemy to perform a large precomputation for a specific prime (p). When the precomputation is complete, arbitrary discrete logarithms related to p can quickly be calculated. All servers that uses that same p is then eligible for attack and the connections can be broken in real-time, meaning that beside being able to extract the encrypted information, information can be modified, deleted or added in a man-in-the-middle attack.

Diffie-Hellman supports several bit-lengths for the prime, p, where the lowest, 512-bits, is known as *export grade DH*. It is an obsolete version that was used during in the 1990s on which the above mentioned attack even could be carried out by adversary with limited resources. The precomputations for this step was measured to seven days per prime and the two most common primes alone account for 92% of all Alexa Top 1 Million domains [30] that support *export grade DH*. According to the authors, a surprising amount of servers still support this obsolete version of DH. Modern browsers do not support it and it is therefore very seldom used. However, if an attacker manages to execute a man-in-the-middle attack, the approved cipher suite may be downgraded to allow *export grade DH*. Under normal circumstances, the user would notice the altered message when the session keys has been agreed on and they both send their own view of the previous exchange in an encrypted format. For the attack to go undetected, the adversary must break the session key and alter the servers message so that it looks like a stronger version of DH was used. All this in real-time and before the connection times-out. The authors measured the median time it took them to calculate the individual logs to 90 seconds, with single calculations as quick as 38 seconds.

It is however not only 512-bits DH that is suspectable to *Logjam*, both the 768- and, the widely used, 1024-bit DH are affected. However, the calculations for these primes are a lot more resource demanding and is not feasible by the typical adversary. Adrian et al. concludes that a 768-bit DH prime is within reach with computing power available to an academic organization. Furthermore, that it might be possible for a state-level actor to actually breach 1024-bit DH. They experimentally estimate that the precomputation necessary in *Logjam* could be as high as 30 days per prime. Meaning that the motivation needs to be really high to start performing computations of that magnitude.

4.4.2 The server

Although a secure transmission is essential in a system like this, the security in the other parts of the system cannot be neglected. The adversary is likely to attack the system where the security is low or other vulnerabilities exist. This means that the data should always be encrypted, regardless if it is stored on a server, computer, disc or hard drive, especially if the device is connected to the Internet. This applies both to the storage at Saab and how it is stored and handled by the customer.

4.5 Countering Denial of Service

As previously mentioned, *Denial of Service* (DoS), can be achieved through both hard and soft attacks and there are different defence mechanisms for both types.

4.5.1 Soft Denial of Service

There are numerous ways to mitigate DoS attacks. Zargar et al. [31] describes different defence mechanisms for distributed flooding attacks. They classify the mechanisms into different categories, depending on where in the system they operate. Some mechanisms can be applied near the source of the attack, some near the victim, some on the network interconnecting attacker and victim, and some are hybrid solutions. Since a DDoS flooding attack wastes a lot of network resources, the attack is advantageously stopped as close to the source as possible. The problem is that a DDoS attack is hard to detect near the sources, this since each source could be spread out in different parts of the Internet, making detection only possible at the victim. This is where the hybrid mechanisms can be applied. They enable, for example, that detection can be made near the victim and the response actions is distributed to nodes near the attacker, decreasing the stress on the intermediate network.

Some mechanisms that could be placed close to the victim, i.e. the server, are IP Traceback mechanisms [32] as well as Packet marking and filtering mechanisms [33, 34]. In IP Traceback, the packets are traced back to the true source and not to the spoofed source address. Routers on the packets path normally add information to the packets they handle, hence making a traceback to the true source possible. This method however, requires support from the routers, if they do not support traceback, this method is unusable. In Packet marking and filtering mechanisms, the marking is used to create dynamic filters for bad traffic rather then tracing it. Some filters that can be applied are History-based IP filtering [33] and Hop-count filtering[34]. The History-based filtering the victim keeps an database with IP addresses of previous frequent users, making it possible to grant only them access to the resource during a DoS attack. Zargar et al. however points out that a large-scale DDoS attack, simulating normal traffic, will defeat such mechanism. Hop-count filtering works in a similar way, only that in addition to the source IP address, the number of hops between the source and destination is recorded. During an attack, the router can check incoming packets' IP addresses and hop-counts and compare them to the previously stored information. It might however be possible for the attacker to spoof an IP address with the same hop-count as a legitimate packet and thereby circumvent the mechanism. Also, it is possible that a legitimate packet gets blocked if it takes a different route then previously recorded.

The other mentioned classes of mechanisms requires efforts of other parties, for example Saab's Internet Service Provider, ISP. They might offer Network-based and support Hybrid mechanisms, for example *Route-based packet filtering* [35] and *Aggregate-based Congestion Control* (ACC) [36]. The upside with ACC is that it never blocks legitimate users since they respect the rate limit whereas an attacker will not, since it renders the attack harmless.

Denial of Service attacks are tricky situations to handle. Many of the solutions proposed have great downsides, especially if the attacker is clever and resourceful. This since by spoofing specific addresses from roughly the same location, he can actually trick the DoS defence mechanisms to effectively deny a legitimate user access to the resource.

4.5.2 Hard Denial of Service

A completely different type of Denial of Service threats are physical threats, connected to the location where the system is installed. This includes threats of manipulating or destroying data on the server directly on site. An attacker could take out the power to the server or cut its connection to the Internet. Other possible threats are natural disasters with earthquakes and storms which could yield in power outages and floodings. All this suggests that the system should be placed in a secure location, with access restricted to only those who need it. The building should furthermore be guarded and able to endurance severe wind and water. Depending on how high availability is rated, it could also be necessary to have backup power supply and an alternative Internet connection.

4.6 Countering Elevation of Privilege

In order achieve elevation of privileges, the attacker must first gain some kind of access to the system by utilizing some of its attack vectors. The access points can be both physical and virtual and the attacker may have some or no legit access.

4.6.1 Network Defence Mechanisms

To protect against the physical access points, such as USB, Ethernet and FireWire, the same measures as for physical DoS attacks, where the system needs to be installed in a secure location where only authorized personnel has access, can be applied. This applies to both the sending and receiving end of the delivery system.

The largest attack vector however, is probably the fact that the delivery system is connected to the Internet, making it vulnerable to different long-distance attacks. One common way to mitigate network-based attacks is to employ different network defence mechanisms such as stateful and application firewalls, and *Intrusion Detection and Prevention Systems* (IDPS). These devices monitors the connections in order to detect malformed or malicious packets and either discards them or sends an alert upon detection. It could therefore be required to have some sort of incident response team that can take immediate actions to limit or avoid damage if an intrusion is detected.

A unwanted scenario is if the attacker manages to gain access to Saab's internal network through the delivery system. It is therefore important to separate the server from the internal network. This is typically done by placing the devices that should be accessible from the outside in a delimited network [10]. This network is separated from the internal network with different defence mechanisms such as internal firewalls, which provides two-way protection. It protects the internal network from attacks launched from the delimited network and vice versa. Another way to protect the internal network is to apply unidirectional gateways (or data diodes) between the delimited and internal networks. Since the delivery system should handle two-way communication, software sent to the customer and feedback received, it could be required to have two servers, one for sending information and one for receiving. The use of data diodes makes it possible to separate the machines that handle ingoing and outgoing information. Figure 4.1 illustrates how the system could look like with some of these network defence mechanisms in place.

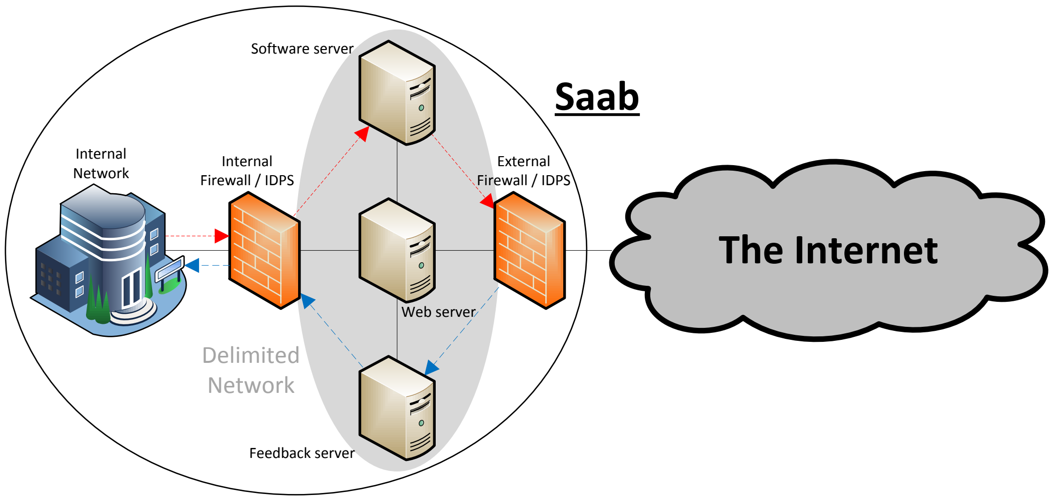


Figure 4.1: Saab's side of the system with different defence mechanisms such as firewalls, IDPSs, data diodes and a delimited network. The red, small-dashed, line illustrates the path of the transmitted software, whereas the blue, big-dashed, line illustrates the received feedback.

In the event than an attack has occurred, it is furthermore important to thoroughly investigate the breach so that future attacks can be prevented. This requires extensive logging, which is a complicated task. This, since careful consideration is required when deciding what to log. If to much information is logged, it might be difficult for the investigator to find the specific information that is needed to resolve the incident. To little information, on the other hand, could make it impossible to understand how the attack was conducted.

Another way to reduce the attack vectors is to apply methods for safe configuration of the devices used in the system, for example routers, servers, firewalls and IDPSs. The *National Institute of Standards and Technology* (NIST) [37] proposes that a security configuration checklist (also known as a harderning guide) should be used. This checklist contains a series of instructions for how to configure a device to a certain security level. The checklists can be provided by the vendor of the devices, but it could also be other organizations such as consortia, academia and government agencies. The checklist may include automatic configuration files, recommended security settings, methods of how to install and configure a device, policy documents and administrative practises.

4.6.2 Access Control

Even with the previously mentioned defence measures, methods to separate information of different users is required. One user with legitimate access or an attacker that has circumvented the defence measures should not be able to read information intended for other users. This can be achieved by using one or several access control policies. An access control policy dictates what types of access are permitted, by whom and under what circumstances. The policies are normally grouped in three different categories [10].

First, *Discretionary Access Control* (DAC), which controls access to a resource with the identity of the requester and a set of access rights (read, write, delete, execute, etc.). All users and their access rights to each file or resource in the system are stored in an access matrix that is checked every time a request is made. The owner of a file or resource in DAC often have special privileges and may grant other entities access to that same resource.

Second, in *Role-Based Access Control* (RBAC), the access control is based on roles rather than the identities, typically as a job function or title within an organization or system. Each role is assigned a set of access rights and they are thus the same for every user with the same role, although a user can have several roles and thereby other privileges. One good practise is to use the concept of least privilege, which means that each role should have the minimum amount of access rights needed for that role and that users then are assigned to the role only to perform what the role requires. RBAC can also be implemented with access matrices, one for the roles each user have and one for the access rights of each role.

Third, in *Mandatory Access Control* (MAC), access is based on comparing security labels with security clearance. The higher the security label, the more sensitive the data is. MAC is related to military information security and the need to restrict access to those with clearance only. In these military security models, for example the widely known Bell-Lapadula model [38], only a user with **TOP SECRET** clearance can read information that is **TOP SECRET**. This however, means that everybody with the same security clearance can access the same information.

5 Discussion

Implementing *continuous deployment* could introduce new possibilities to Saab in how they develop and deliver software. There are however some obstacles we have identified, that are outside the scope of this thesis, which needs to be overcome before *continuous deployment* during the operational phase is realistic. First, the fact that a end-system today could contain several hundred software components, that all could be individually patched, implies that the possible number of configurations could be endless, increasing the management complexity and thereby also the cost. Second, the installation process is, as mentioned in chapter 1, both time and resource demanding, also impacting the costs if done at a too high frequency. Finally, since the entire system needs to be reinstalled to apply a patch, the entire software needs to be transmitted each time. The probability that such a transmission would include *defence secrets* is thereby very high and means that the public protocols and solutions the delivery system is based on are not applicable.

However, until those obstacles have been resolved, *continuous deployment* could be implemented in a *customer-lab* set-up, as this may not have the same requirements on installation and management as in an operational setting. This would allow Saab as well as the customer to evaluate *continuous deployment* in order to decide whether or not it is suitable for an operational phase.

One of the toughest challenges with constructing a secure system is to know when a sufficient level of security has been achieved. One factor lies in the difficulty to determine what capabilities and motivation the adversary has, another is the inability to foresee all possible scenarios and thereby apply all required countermeasures. It is therefore appropriate to apply a well developed and systematic method. The process proposed by Microsoft meets those criteria. The modifications we applied to the process allows the designer to start with a very general view of the system, describing which features the system should have and look at which security measures that are required, and then iteratively build the system. The threat model STRIDE works well when designing a system like this, where the different categories provides a good foundation for identifying vulnerabilities. However, we found the risk assessment model, DREAD, hard to apply to the system at an early stage, as the assessment requires many, at this time, unknown factors when evaluating the threats. As DREAD do take the cost of mitigating a threat in consideration, we would recommend that in addition to DREAD, evaluate the cost of implementing a solution versus the risk of losing reputation and financial consequences. This report is a first iteration of this process, proposing high level solutions to the vulnerabilities identified, further iterations of the process is necessary in order to design a complete secure system.

The fact that the *point of delivery* is not yet decided could make different solutions more or less beneficial, i.e. one solution might be better suited if the *point of delivery* is at the customer headquarters whilst another if it is at the end-system. It is important to emphasize that even if the *point of delivery* is at the customer's headquarters, the software needs to be transported to the end-system in an, atleast, equally secure manor, as the delivery system is as secure as its weakest part.

This raises questions about where Saab's responsibility end, is it at the *point of delivery*, the end-system, or during the entire lifetime of the end-system. If the *point of delivery* is at the customers headquarters, and they are responsible for the transport to the end-system. Are Saab still responsible that the software installed is the correct one. This is a important issue to consider when deciding on the *point of delivery.*

In the following sections we discuss the solutions provided to the threats identified.

Authenticity - The concept of authenticity is used to mitigate the spoofing vulnerabilities identified. Where both the user and machine, i.e. the server, should be authenticated. The user authentication should follow Saab's directions to be atleast a two-factor scheme. Beside Saab's directions, two-factor authentication is also standard in high security systems, making it a viable option.

A robust three-factor scheme could further strengthen the security in the system, but is on the other hand not that widely used and requires more advanced equipment, e.g. a fingerprint scanner. The most interesting feature with the three-factor authentication proposed by Fan and Lin is that the biometric signature is never revealed to the system administers. One might assume that military personnel of a foreign nation are unwilling to give up such sensitive information, thereby making the solution particularly interesting in a military setting. The question is if the higher security motivates the increased cost such an upgrade would bring. One other aspect of this issue is that the solution must be user friendly, if to complicated to use, the users might find other ways to ease the process, at the cost of security.

Integrity - Knowing that the data delivered and installed on an end-system is the intended one is a fundamental property when dealing with sensitive information. The transmission of data, in both directions, needs to be signed so that the receiver can validate its integrity. But the fact that the software can be replaced both prior and after the transmission or even that malicious software can be installed directly on the end-system, makes *End-To-End* integrity with *Trusted Computing* and *Remote Attestation* a vital part of such a delivery system. The issue with patches and upgrades that Haldar et al. identified is a serious issue that puts high demands on the version handling. Maybe the set of configurations that should be accepted must be limited to a finite number, that some patches must be applied in a certain order or not at all. Another positive aspect with a more narrow version handling is that it could at the same time facilitate support and error handling. If the number of possible configurations is low, the possible sources of error is smaller.

However, if an insider modifies data during development, the above mentioned mechanisms are ineffective, as the data is under development. Protection against insiders with malicious intent is a difficult problem to solve, and much of the responsibility lies with the co-workers and manages to notice. However, there are several methods that ease the detection of a malicious code. Automated testing, review and analysing are example of such methods and would hopefully detect malicious code. To further ease the identification of an insider, security logs, strong authentication and version control systems could be used.

Accountability - All users of the system benefit from a fulfilment of non-repudiation. If something happens, there is a way to find out what has happened, without having to question the honesty of the other party. Normally a TTP is a desired entity in order to achieve a fair judgement in case of a dispute. Although, when dealing with military information, there are some issues when selecting a TTP. Both parties needs to find a mutual TTP which they feel confident sharing the information with, in some cases a TTP might even introduce more threats and problems than they solve. Although there are solutions where the system owner, in this case Saab, is responsible for the non-repudiation system, like *BankID* in Norway. We believe that even if the customer trust the owner, a situation where that trust is tested, should if possible be avoided, as it could result in reluctance to further cooperation. However, in this case, as a TTP is presumably difficult to find, it might be the only practical solution.

As mentioned in section 2, the *Swedish Defence Material Administration* (FMV) acts as a guarantor for Saab's management of other countries' *defence secrets*. FMV may therefore be a suitable TTP when Saab deals with foreign authorities, although as a government authority, FMV may not be able to provide this service and furthermore, they might not be considered unambiguous by the foreign authority.

We would like to point out that depending on the purpose of the system, *continuous deployment* during the development phase or during the operational phase, the requirements for the non-repudiation system may vary. If used for continuous deployment during development, the consequences may not be as severe if the wrong software are delivered, lowering the need for non-repudiation. Although, if used for the entire life cycle of a end-system, where a faulty update could be disastrous, non-repudiation is of importance as it is a reassurance for both parties if something goes wrong.

Confidentiality - The two types of encryption techniques, hardware and software, have different advantages over the other in the areas of key management, cost, security and mobility. A *hardware security module* (HSM) requires distribution of both hardware and cryptographic keys, requiring additional resources. A software based solution, such as TLS, would not require the same kind of resources, as no specific hardware is required and key management is a part of the protocol. It however, requires other kinds of resources like configure, update and monitor servers and the protocols used. Also, a careless implementation of the key negotiation is vulnerable to attacks such as Logjam, where the attacker is able to hijack the communication in real time.

As the two different techniques could provide different levels of security, information of various classifications could be allowed depending on the solution. There might be an alternative where both hardware and software encryption are applied depending on the information transmitted. Information classified as Swedish *defence secrets* needs to be transmitted with HSMs approved by FMV. A possible solution is to use public HSM for information not classified as Swedish *defence secrets* and when dealing with classified information switch the HSMs against the, by FMV, approved ones.

Like previously mentioned, an attacker is likely to attack the system where security is low, meaning that it is equally necessary to maintain confidentiality throughout the entire delivery chain, from development to installation. The software should preferably always be stored encrypted or in secure facilities and only be decrypted during installation.

Availability - In order to achieve a reliable delivery system, availability needs to be ensured. Even if the system may not have any time-critical requirements, the consequences of failing to deliver may be minor, compared to the damage to Saab's reputation as a security company. Therefore, it is important to ensure that the system is resistant to different kinds of denial of service attacks. The history-based filtering mechanism suits our system well, as there is a small number of users, making it possible to control the allowed IP-addresses. Investigation of witch prevention techniques the ISP support, such as *Route-based packet filtering* and ACC, could be done in order to further prevent an attack.

Authorization - The need for access control cannot be stressed enough. Users both inside and outside of the organisation should only have access to resources they have a need for and clearance for. By having strict access policies for the employees, a potential insider may not have the same opportunities to damage the products and in extension Saab's reputation. The different policies could be used on different users, some customers might require the military policy, MAC, for their own personnel, while for system administrators, RBAC might be the best suited policy. However, the policy suited for most situations are perhaps DAC, although it might not be the easiest to administrate. The best solution is most likely a combination of all mentioned policies, depending on the customers needs and the information handled.

The security logs of the system should be administrated by an incident response team. If an attack is successful, the delimited network, IDPS and firewalls in cooperation with the incident response team, should minimize the information acquired and damage done. If the *point of delivery* is at the end-system, a similar protection system needs to be applied in order to prevent an attack and minimize the potential damage.

We would like to emphasize that there are not a single solution that secures the system, it is the cooperation of all components, with different features, that provides the security. It should also be noted that there exists several administrative regulations that also need to be fulfilled regarding security policies of both Saab and its customers. For example that the whole delivery system needs to be approved by *Swedish Military Intelligence and Security Service* (MUST) if it handles information classified as *defence secrecy.*

6

Conclusion and Future Work

During the thesis we have examined the possibility to implement a *continuous deployment* system in a military setting. We have presented a methodology for finding vulnerabilities when designing a secure system. We have furthermore presented different ways to mitigate or eliminate the discovered vulnerabilities.

6.1 Conclusion

A *continuous deployment* system could have many benefits. It is however not as straight forward in a military setting as in many other industries. The information handled in these organisations could be extremely sensitive and the consequences could be very severe if systems designed to provide safety fails. Apart from *company confidentiality*, information could also be classified as *export control* and Swedish, or other countries, *defence secrets* which requires that strict laws and regulations must be acknowledged. The obstacles of installation, verification and management of software in military products implies that a *continuous deployment* system is unrealistic to apply during the operational phase of such a product today. It could however be very suitable during the development phase, in a so-called *virtual lab* concept, as rapid testing and feedback from the intended customer could improve the product experience as well as cutting down development times.

Regardless of how the system could be used, the issue of constructing a truly secure system is the same. The chain of security, built to protect the system, is only as strong as its weakest link. With Saab being a defence and security company, it is of utmost importance to use reliable systems and secure ways of communication. If the system is proved flawed, and sensitive information is leaked, the financial cost of that specific loss might be a lot smaller than the potential damage to the brand and thus the future development of the company. However, we believe that the technology for a secure *continuous deployment* system exists today and the limitations are in the current processes and regulations regarding patching and installation. The perhaps most important aspect to remember when dealing with security is the need to constantly revise and update the system. What is considered secure today might not be so tomorrow.

6.2 Future Work

Continuous deployment is a hot topic and is incorporated in more and more products around us as rapid updates are increasingly important. Although there are several challenges to overcome before *continuous deployment* could be applied in an operational phase in the military industry, it might even be an inevitable evolution. We believe that the first step to take is to decide the *point of delivery*, since different solutions might be more or less suitable. The natural next step would be to continue to apply our proposed methodology, i.e. investigate whether our proposed solutions introduce new assets, alters the architectural overview, introduce new threats and vulnerabilities and finally propose new solutions to counteract them. When the designer believes that the system is complete and all threats have been identified, it is time to assess the remaining risks and decide what to further do with those risks; accept, transfer, remove or mitigate.

The obstacles identified in chapter 5 needs to be addressed and it is necessary to consider whether the increased functionality of a digital delivery system outweighs the risk and cost. Finally, all administrative security policies and routines, specified in various laws and regulations obviously needs to be fulfilled as well if the system is going to be deployed.

Bibliography

[1] R. Shirey, Rfc 2828: Internet security glossary, The Internet Society.

[2] Saab group, Saab's solutions, `http://www.saab.com` (2015).

[3] Scrum.org, Scrum, `https://www.scrum.org/` (2015).

[4] Polestar, Volvo polestar, `http://www.polestar.com/` (2015).

[5] C. Harress, House passes $585 billion defense budget for 2015, `http://www.ibtimes.com/house-passes-585-billion-defense-budget-2015-1734920` (2015).

[6] McAfee, Net losses: Estimating the global cost of cybercrime - economic impact of cybercrime ii, Tech. rep., Center for Strategic and International Studies (June 2014).

[7] Saab group, Giraffe amb, `http://saab.com/land/ground-based-air-defence/ground-based-surveillance/giraffe-amb/` (2015).

[8] Swedish Defence Material Administration (FMV), Industrisäkerhetsskydssmanual.

[9] Swedish Defence Material Administration, FMV, `http://www.fmv.se/en/` (2015).

[10] W. Stallings, L. Brown, Computer security, Principles and Practice.

[11] K. S. McCurley, The discrete logarithm problem, in: Proc. of Symp. in Applied Math, Vol. 42, 1990, pp. 49–74.

[12] L. O'Gorman, Comparing passwords, tokens, and biometrics for user authentication, Proceedings of the IEEE 91 (12) (2003) 2021–2040.

[13] V. M. Jr., Z. Riha, Toward reliable user authentication through biometrics, IEEE Security & Privacy 1 (3) (2003) 45–49.

[14] N. Doraswamy, D. Harkins, IPSec: the new security standard for the Internet, intranets, and virtual private networks, Prentice Hall Professional, 2003.

[15] T. Ylonen, C. Lonvick, The secure shell (ssh) protocol architecture.

[16] S. Myagmar, A. J. Lee, W. Yurcik, Threat modeling as a basis for security requirements, in: Symposium on requirements engineering for information security (SREIS), Vol. 2005, 2005, pp. 1–8.

[17] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, et al., Comprehensive experimental analyses of automotive attack surfaces., in: USENIX Security Symposium, San Francisco, 2011.

[18] J. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla, A. Murukan, Improving web application security: threats and countermeasures, Microsoft Redmond, WA, 2003.

[19] Microsoft, The STRIDE Threat Model, `https://msdn.microsoft.com/library/ms954176.aspx` (2015).

[20] Microsoft, Threat model, `https://msdn.microsoft.com/en-us/library/ff648644.aspx` (2015).

[21] Finansiell ID-Teknik BID AB, BankID e-legitimation, `https://www.bankid.com/` (2015).

[22] C.-I. Fan, Y.-H. Lin, Provably secure remote truly three-factor authentication scheme with privacy protection on biometrics, Information Forensics and Security, IEEE Transactions on 4 (4) (2009) 933–945.

[23] M. Bellare, T. Kohno, Hash function balance and its impact on birthday attacks, in: Advances in Cryptology-Eurocrypt 2004, Springer, 2004, pp. 401–418.

[24] C. Paquet, Implementing Cisco IOS Network Security (IINS):(CCNA Security exam 640-553)(Authorized Self-Study Guide), Cisco Press, 2009.

[25] V. Haldar, D. Chandra, M. Franz, Semantic remote attestation: a virtual machine directed approach to trusted computing, in: USENIX Virtual Machine Research and Technology Symposium, Vol. 2004, 2004.

[26] K. J. Hole, T. Tjostheim, V. Moen, L.-H. Netland, Y. Espelid, A. N. Klingsheim, Next generation internet banking in norway.

[27] Advenica, SecuriVPN ISA, `https://advenica.com/sv/vpn/securivpn` (2015).

[28] E. Rescorla, Diffie-hellman key agreement method.

[29] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, et al., Imperfect forward secrecy: How diffie-hellman fails in practice.

[30] Alexa, Alexa Top 1 Million domains, `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip` (2015).

[31] S. T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks, Communications Surveys & Tutorials, IEEE 15 (4) (2013) 2046–2069.

[32] A. John, T. Sivakumar, Ddos: Survey of traceback methods, International Journal of Recent Trends in Engineering 1 (2) (2009) 241–245.

[33] T. Peng, C. Leckie, K. Ramamohanarao, Protection from distributed denial of service attacks using history-based ip filtering, in: Communications, 2003. ICC'03. IEEE International Conference on, Vol. 1, IEEE, 2003, pp. 482–486.

[34] H. Wang, C. Jin, K. G. Shin, Defense against spoofed ip traffic using hop-count filtering, IEEE/ACM Transactions on Networking (ToN) 15 (1) (2007) 40–53.

[35] K. Park, H. Lee, On the effectiveness of probabilistic packet marking for ip traceback under denial of service attack, in: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol. 1, IEEE, 2001, pp. 338–347.

[36] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, S. Shenker, Controlling high bandwidth aggregates in the network, ACM SIGCOMM Computer Communication Review 32 (3) (2002) 62–73.

[37] National Institute of Standards and Technology, Security Configuration Checklists Program, `http://csrc.nist.gov/groups/SNS/checklists/` (2014).

[38] D. E. Bell, L. J. LaPadula, Secure computer systems: Mathematical foundations, Tech. rep., DTIC Document (1973).