DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2020

Ethical hacking: Threat modeling and penetration testing a remote terminal unit

SAM HAMRA

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Ethical hacking: Threat modeling and penetration testing a remote terminal unit

SAM HAMRA

Date: November 26, 2020
Supervisor: Mathias Ekstedt
Examiner: Pontus Johnson

Abstract

Remote terminal units are microprocessor controlled electronic devices that acts as an interface between control systems and objects in the real world. They are used in a range of highly critical infrastructures, and thus their security is of high priority. This thesis will present a security analysis and testing of a remote terminal unit. A threat model was created to identify threats to the system and a few key threats were selected for further penetration testing. The testing lead to the identification of a denial of service vulnerability as well as code injection vulnerability in the SD card storage of the remote terminal unit. The conclusions is that the system is rather robust from a remote attackers perspective although more vulnerabilities arise as an attacker gains physical access to the device.

Sammanfattning

Fjärrkontrollsterminaler är elektroniska enheter som agerar som ett gränssnitt mellan kontrollsystem och objekt i den riktiga världen. De används i många kritiska infrastrukturer och därför är deras säkerhet högt prioriterad. I denna rapport presenteras säkerhetsanalys och testning av en fjärrkontrollsterminal. En hotmodell skapades för att identifiera hot mot systemet och ett antal hot valdes för vidare penetrationstestning. Testandet visade på svagheter mot denial of service attacker mot TCP/Ethernet gränssnittet samt kodinjicering mot SD-kortet som sitter i fjärrkontrollsterminalen. Slutsatsen är att systemet är relativt säkert mot fjärrattacker medan svagheterna mot attacker som kräver en fysisk tillgång till fjärrkontrollsterminalen är fler.

Contents

Chapter 1

Introduction

Many critical infrastructures require a degree of automation to function optimally. To allow for autonomous operation, remote terminal units are deployed as a means to control and monitor electronic field devices from a central control system. As more and more devices gets connected to the infrastructure, it becomes harder to protect the system as a whole. Because remote terminal are used in a wide range of critical applications like nuclear power plants[1], electrical substations and water treatment plants[2], security is of great importance.

But how does one tell if a device or system is secure? One of the ways to audit the security of a computer system is to use ethical hackers. Ethical hacking is a broad term that includes all types of cyber attacks, but perhaps none are more widespread than penetration testing. The goal is to identify vulnerabilities so that they can be fixed before criminals can exploit them[3]. The term white hat hacker is sometimes used interchangeably with ethical hacker, and this is to differentiate it with so called black hat hackers that violate computer security for personal gain or maliciousness.

The purpose of this project is to audit the security of a Siemens SICAM CMIC. As a means of analyzing the system's security a threat model was created to better understand the attack surfaces, so that threats could be identified and assessed. A slightly modified version of the threat modeling methods presented in [4] was used to find attack surfaces and threats. A few key threats was selected for further penetration testing.

1.1 Ethical hacking

One approach of determining the level of security in an organization or system is by using ethical hackers to test the security of the system. Some sources[5] use the term penetration testing as a synonym to ethical hacking, whilst other[6] use ethical hacking as a umbrella term of methods and techniques used by the ethical hackers. One argument for using ethical hackers is that just like one would not submit a thesis without having it proofread, a system needs to be tested for it to be considered secure. One of the major ways the white hat hacker differs from a black hat hacker is the way they use the vulnerabilities they find. The white hat hacker has a responsibility, not only ethically but sometimes also lawfully to responsibly disclose vulnerabilities found[7]. Responsible disclosure often involves contacting the vendor in question to make a coordinated disclosure of the findings. In many cases the vendor will assess the vulnerabilities found and if needed create a Common Vulnerabilities and Exposures(CVE) report that packages information about a vulnerability or exploit. CVE is a list as well as a format for its entries, and one famous database that stores these CVE entries is the U.S. National Vulnerability Database(NVD) that was launched by NIST in 2005[8].

1.2 Remote terminal units

Remote terminal units(RTU) are microprocessor controlled electronic devices that acts as an interface between objects in the real world, and an automation system[9]. They operate by transmitting data to and from the automation system in order to provide control and supervision. RTU's can connect to a host of different real world objects such as electrical distribution substations, gas distribution substations, hydropower plants, pipelines, railway power supplies, as well as in building protection or for alarm signaling[10]. Besides being able to communicate with an automation system or a supervisory control and data acquisition system(SCADA) the RTU can also change the physical states of connected devices to manage the flow of electricity, water, gas etc. RTU's are able to be customized with application-specific control logic[9]. This logic is programmed into firmware or software, and it enables the RTU to autonomously control devices based on changing conditions, without having to be actively controlled from a host system[9].

Because the RTU needs to be able to operate in environmentally challenging locations, they are able to communicate not only over a wire, but often

using fiber optic cables, cellular and sattelite communications [9]. Most modern RTU's supports many different communication protocols like Ethernet, Modbus RTU, IEC 80870-5-101 and DNP[10].

1.3 Security in remote terminal units

Because remote terminal units provides an interface between physical objects and SCADA systems, RTU security is deeply connected to SCADA security. If breached, the RTU can compromise the security of both SCADA and physical objects. One of the most general threats to the RTU and SCADA system is that of an unauthorized entity gaining access to the system. With full control of the system, attackers can cause some serious harm, potentially causing environmental damage or even worse, loss of human life[11].

Threats to RTUs can come from many potential threat agents, such as hostile nation states, spies, malicious hackers, ethical hackers and the disgruntled employee. One example of a real world scenario that is very similar to a situation that could arise from a RTU being compromised is the Stuxnet attack. Stuxnet is a computer worm that was discovered in 2010 when it was reported that it had caused one-fifth[12] of Iran's nuclear centrifuges to start spinning faster than originally programmed, tearing them to pieces. The Stuxnet worm targets PLCs, but it is not domain specific and could be modified to attack SCADA systems. Because the RTU and SCADA systems are not isolated in terms of connectivity, having multiple points of contact, and multiple ways of communicating over each medium, this becomes one of the most serious vulnerabilities faced by SCADA systems today[11]. Other common vulnerabilities lie in the operating system code that runs on the hardware, shared libraries or other common code used by RTU applications. Because RTUs can communicate over Ethernet, many vendors have started to ship RTU's with FTP servers, web servers and other services that are potentially vulnerable to attacks[11].

1.4 Scope and goal

This report will focus on threat modeling and penetration testing the remote terminal unit SICAM CMIC, with regards to its SD card and communications over TCP and HTTPS, thus excluding RS-232/454 specific attacks as well as NTP attacks. I/O modules and I/O devices will not be analyzed, as well as any other attacks on the physical layer of communication and physical enti-

ties i.e.unscrewing the case, fiddling with hardware. The SICAM CMIC will be tested "as is", meaning no updates will be made to firmware. The current firmware on the RTU is CPC-80 v10, SWEB00 v02.03. The version of firmware CP-8000 is unknown, except that it is a version released before V14. The question that this thesis will aim to answer is:
Is the SICAM CMIC a secure device in regards to its web server and storage of information on the SD card?

1.5 Previous work

The SICAM CMIC is developed by a large company that takes security very seriously, and it is safe to assume that thorough testing has already been performed. To go along with this a CVE[13] was released from a third-party testing company that found a vulnerability concerning a denial of service attack on the web server. More specifically it was a so called Billion laughs attack, or an XML Bomb that rendered the web server unable to respond to requests, requiring a manual restart of the device. As for similar research, a study was performed by Christensen and Dannberg in 2019[14]. In that study, the authors used threat modeling to help perform penetration testing on IoT devices.

Chapter 2

Method

In this chapter the methods that are in the center of this thesis will be introduced and explained. A general threat modeling and penetration testing method will be presented. The methods will be slightly modified as to better fit the needs of a ethical hacker.

2.1 Threat modeling

One technique of gathering and structuring information about the security of a system is using a threat model. A threat model is basically a representation of all available information that can jeopardize the security of a system[15]. What differs the threat model from any other model of a system, is that the threat model tries to model the system from a security perspective. The general purpose behind using a threat model is to find security bugs as well as help developers understand the security requirements of the system. Although threat modelling is preferably carried out during the development phase of a system, it is also useful after the fact.

By really thinking about and enumerating all the possible threats to the system, it becomes much easier to create sound security requirements. Some threats might not be relevant to your business, whilst other threats might have been overlooked. There may be a set of threats that are not worth the cost of mitigating, threat modeling helps when trying to understand all of this better to create a more secure system[4]. Adam Shostack[4] has created a four-step framework that begins with the four questions:

1. What are you building?
2. What can go wrong?
3. What should you do about those things that can go wrong?
4. Did you do a decent job of analysis?

2.1.1 Model system

Creating a model of a system is usually done by drawing diagrams. By dividing the system into its constituent parts, a diagram provides a great way to visualize this. By also adding so called trust boundaries, that is the borders between two parts that are controlled by different entities, for example user accounts, network interfaces, different physical computers, virtual machines, organizational boundaries, the task of finding threats can be made even easier. To be able to create an accurate model of the system, initial information gathering and scanning is necessary.

2.1.2 Identify threats

There are a multitude of ways of going through the process of identifying threats. A useful starting point is thinking about it from the perspective of the attacker. Who is the attacker? And what resources do the attacker possess? If one knows who to protect their system from, it becomes much easier to make good decisions on what security features the system needs. Now that more is known about the attacker, it is time to answer the question: what can go wrong? One of the most traditional ways of enumerating threats is by brainstorming, this obviously sounds very simplistic, but having a room full of experts brainstorming can be very effective, given their experience on the subject. Although brainstorming often has the downside of being unstructured, there are ways of going about it as to make it more structured. For example by focusing on the assets, or focusing on the things that the attackers want or things that is valuable to protect. One very popular way of identifying and enumerating threats, is using the STRIDE mnemonic to group and classify threats. The upside of grouping threats this way, is that by protecting against each group, it ensures that desirable properties of the system holds. These properties include Authenticity, Integrity, Non-repudiability, Confidentiality, Availability and Authorization.

Spoofing	Pretending to be something or someone you're not	Authenticity
Tampering	Modifying something you're not supposed to modify	Integrity
Repudiation	Claiming you didn't perform a certain action	Non-repudiability
Information Disclosure	Exposing information to unauthorized individuals	Confidentiality
Denial of Service	Attacks that aim to deny a system service, either by crashing it, making it slow or by other means unresponsive	Availability
Elevation of Privilege	A program or user is able to perform actions they are not supposed to be able to	Authorization

Table 2.1: STRIDE

2.1.3 Assess threats

In Adam Shostack four-step framework, which is more catered towards security as a design principle, the third step is addressing the threats that have been identified. From an attacker's perspective, this is not as relevant, therefore the use of the framework will be slightly modified to better aid us in our task as ethical hackers. Instead, penetration testing methods will be used to assess the identified threats.

2.1.4 Validate threats

After penetration testing, our work needs to be validated for completeness, this will be done in a later discussion section.

2.2 Penetration testing

There are generally three different models that can be utilized by the ethical hacker, the white box model, black box model and the grey box model[16][17]. They differ from each other in that the information accessible to the attacker varies from zero information, to having an insider providing valuable insight on the system's inner workings. Using the black box model, the ethical hacker can be viewed as a "distrusted outsider"[17] with little or no prior knowledge about the system's security policies. The ethical hacker must gather knowledge about the system from the outside, much like a real attacker would. In this thesis the focus will be on the black box model of ethical hacking, since the penetration tests are basically starting out from pitch black, with no guidance, information or assistance from any insiders.

The black box method can generally be divided into phases[16][17], however they will have to be slightly modified to fit the needs of an ethical hacker. We don't really have to clear our tracks, because we are not worried about getting caught, therefor getting rid of evidence and clearing traces of the attack will be outside the scope of this report. Threat modeling techniques explained previously will also be weaved in with the penetration testing methodology.

2.2.1 Information gathering/Reconnaissance phase

During this phase, information is collected from any public sources, be it websites, brochures, manuals or any other publicly available source of information about the system. Another popular source of this type of information is

WHOIS databases that can be queried for information about public domains. Job advertisements can also be used to gain more insight, by looking for the required competencies in job advertisements, a hacker can potentially pick out which programming languages and systems that are being used in the targeted system[16]. In this project the SICAM CMIC's manual will be the main source of information regarding the system.

2.2.2 Scanning phase

There are multiple types of scans that can be of value to a penetration tester, each with a specific goal. First out is port and operating system scanning, which works by sending network packets to a target host. By examining its responses, information regarding open ports, running services and operating systems can be extracted. Port scans can help attackers and network administrators map out their network to gain a better overview over what is running on the system. Application scanners targets a specific service or application, for example a web server and scans the application for potentially dangerous files and Common Gateway Interface(CGI) vulnerabilities[18].

There are many scanning tools and software that attackers can use, but probably none is more popular than the network mapper Nmap. Nmap is one of the main tools that will be used for Operating System scans, port scans and vulnerability scans.

Nmap is a free and open source utility for network discovery and security auditing[19]. It works by sending and receiving raw IP packets to determine available hosts on the network, what services and operating systems those hosts are running. It was originally designed to rapidly scan large networks for hosts with open ports, but it works just as fine against single hosts. As time has passed, Nmap has developed many more features, like discovering what type of packet filters/firewalls are in use. It also has functionality for running attack scripts targeting specific ports on a target host.

Operating system scan

A Nmap operating system scan works by using TCP/IP stack fingerprinting[20]. Because every developer of an operating system implements the TCP/IP stack differently, the same TCP/IP conversation with two different operating systems could yield different results[21]. Some of the flags that operating systems differ in are initial TTL, window size, maximum segment size and initial packet size. This information is then combined to create a fingerprint that is unique to each operating system[22].

A Nmap operating system scan specifically works by sending a series of packets to a target host, with contents specifically crafted to force the operating system into answering with a packet that is either consistent with responses from another operating system, or unique in regards to other operating systems[21]. To make this work, Nmap stores a database with known responses from the most common operating systems, to be able to compare the target responses to responses from other operating systems.

Port scan

Seemingly all systems that are connected to a LAN or the Internet run services that listen to some ports. Most default operating system installations have numerous ports open to offer greater flexibility, however this can also become a liability[23].

Port scanning is one of the most popular techniques used to discover which services are running on the target system. There are many different port scanning techniques available, but they all essentially work by sending a network request to the target host, and by examining the response, conclusions about the specific ports availability can be drawn.

The most common scan, and the default for Nmap is the TCP SYN scan, where an attacker or scanner sends a TCP packet with its SYN flag set to every port of the target system[24].This is sometimes referred to as a half-open scan, since it never completes the connection, it just opens one and listens in on the response. It is a very fast way of scanning a target and also unobtrusive due to the fact that it never completes any connections.

Other popular scans are the UDP Scan that simply sends mostly empty UDP packets to each port in range. Another useful scan is the Internet Protocol scan which iterates through protocol numbers of the Internet Protocol header. By examining the responses from the packets sent, Nmap classifies each port as being in a specific state. The most relevant states are:

- **open:** The application is actively accepting TCP connections and UDP datagrams on this port. This is what attackers and penetration testers are looking for, but also something that administrators actively try to protect without limiting accessibility[25].

- **closed:** A closed port is still receiving and responding to network packets, but there is no application listening on it. Because these ports are reachable they still show that the host is up on the provided ip address. If a port is classified as being closed, this means that packet filtering is

not blocking nmap from accessing that port, something that attackers might want to take note of in case of an application running on that port in the future.

- **filtered:** If a port is classified as being filtered, this means that Nmap is unable to determine whether the port is open because of firewall filters that prevent probes from reaching the target port. It should be added that Nmap sometimes is unable to exactly determine which one of these states a port is in, because it perhaps is open, but is configured to not respond, in these cases nmap will be kind enough to report that the port is open|filtered[24].

Vulnerability scan

Vulnerability scanning is the act of the scanning a target host or network for vulnerabilities. One approach is called banner grabbing, where an attacker extracts information out of banners that many operating systems and applications send to users as a greeting. These banners often contain system information, like versions and services being run. HTTP is a great example of this where a lot of information is exposed in the HTTP header, like which version of HTTP is being used, which web server that is running and more. Any information gathered from banners are usually looked up in some database of known vulnerabilities and exploits. Some scanners even go as far as actually attempting to exploit weaknesses, and if poorly configured even to the extent that services crash[26].

Some vulnerability scanners are general, in that they are not limited to a specific application or protocol, but rather scans the target for open ports and running applications, and then in turn scans those applications for vulnerabilities. Some tools only focus on specific applications, like web servers, and there are many different tools that does this, some of whom will be used in this project.

One known issue with vulnerability scanners is the fact that they sometimes give false negatives and false positives, i.e. tells us there is a vulnerability when there in fact is none, or tells us everything is fine when a vulnerability actually exists. For banner grabbing, some banners might be outdated in that versions have been updated without the correct information being inserted into the banner. Vulnerability scanners may then flag for vulnerabilities that no longer exist in the target host. Because vulnerability scanners can only scan for known vulnerabilities, if any previously unknown vulnerability is present in a system, scanners are guaranteed to give false-negatives. Because

of the prevalence of false-positives, penetration testers must spend a considerable amount of time to verify each reported vulnerability by trying to exploit them[27]. False positives can generally be split into two categories, technical false positives and contextual false positives[26].

Technical false positives can occur because of bugs present in the scanning scripts, but also for when servers behave in ways that scanning programmers did not expect[26]. For example a web server becoming unavailable during a denial of service test, may make the scanner believe the server is actually vulnerable to that specific attack, where in reality the server became unavailable because of routine maintenance. Another very common reason for technical false positives is packet filters, firewalls, proxies and Intrusion Detection and Prevention Systems.

Contextual false positives are the situations where the scanner flags for a vulnerability, but the vulnerability is not significant, thus it is a false positive in the context in which the scanner is being ran. Some examples of this is where an actual vulnerability exists, but network administrators have chosen to "live with it" or implemented a workaround or another form of mitigation[26]. Another scenario that falls under this category is when a service is running a version with a known vulnerability, but the attacker requires valid credentials to exploit the vulnerability.

2.2.3 Gaining access

After successfully gathering information about the targeted system and having performed scans to map out potential vulnerabilities. It is now time to perform the actual attack, that is gaining access to the system. Sometimes this can be as easy as accessing a private URL[28], but it could also be done by attacking password authentication[16]. Another way of gaining entry to a system is by using an exploit. "In the context of software, an exploit can be defined as a piece of code that takes advantage of a vulnerability to cause unintended behavior in the software."[28]

Exploits are usually classified as either being locally or remotely available, with remotely available exploits obviously being more useful[28].

2.3 Combining methods of threat modeling with penetration testing

The methods described in chapter 2 this far will be slightly modified and combined to create a methodology that will fit the task of an ethical hacker. To create an accurate model of the system, information gathering and scanning will be performed prior to creating a threat model. By using the threat model, threats will be identified and enumerated using STRIDE. Furthermore a few key threats will be selected for further penetration testing.

To summarize, the method will consist of the following steps:

1. Information gathering
2. Scanning and enumerating
3. System modeling
4. Threat identification
5. Penetration testing selected threats
6. Threat validation

Chapter 3

Information gathering and threat modeling results

In this chapter, the results from the initial information gathering and scanning phases as well as a system model and threat enumeration will be presented.

3.1 System description

The SICAM CMIC is a small and compact remote terminal unit that was released in late 2013. According to the developers, it is suitable for electrical distribution substations, gas distribution substations, hydropower plants, pipelines, railway power supplies for example[10]. It will usually be mounted inside a key-locked case, and the SICAM CMIC provides no physical perimeter security mechanisms on its own.

The SICAM CMIC has an onboard monitor and function keys for local monitoring, although no controlling functions are available from the onboard monitor. It can handle digital inputs and outputs, as well as support for coupling I/O modules that can handle analog inputs. The SICAM CMIC also has two Ethernet-LAN TCP/IP 10/100BASE-TX interfaces, one RS-485 and one RS-232 interface. It supports a multitude of protocols such as IEC 60870-5-101/103/104, Modbus RTU, DNP3.0, DNP(i), NTP / SNP and vendor specific protocols on request[10].

The SICAM CMIC uses an SD card as its non-volatile memory and stores firmware, application, diagnostic data and its configuration parameters on the card.

Configuration of the system is solely done via the SD card and its appurtenant engineering software. There are two options for configuring the SD card, firstly, the desktop engineering software, which is a trademarked software suite and is under a strict license-to-use, and there are a number of hoops to jump in order to get hold of this software, as it requires manual inspection by the regional sales unit. There also exists a possibility to install web server firmware onto the SD card, and when loaded into the SICAM CMIC, it runs the web server which an engineer then can connect to via Ethernet and HTTPS. However use of these two methods are exclusive, meaning that once one of them have been used for parameterization, the other can only be used for reading data[29]. The SICAM CMIC has some security features like a integrated crypto chip for secure storage of passwords and more. It also conforms to the German Association of Energy and Water Industries(BDEW) Whitepaper Requirements for Secure Control and Telecommunication Systems [29]. The RTU firmware continuously gets patched with new security features.

The current state of the SICAM CMIC has firmware versions CPC80 v 10 and SWEB00 02.03 installed. All testing will be done on these versions of the firmwares, to best simulate a real attack.

3.2 Scanning the system

Initial scans of the system was done using Nmap. Standard port and operating system scans were performed and their results are presented below.

3.2.1 Operating system scan

After performing operating system scans with Nmap, the tool was unable to determine which operating system that was running on the SICAM CMIC. The actual TCP/IP fingerprint was printed and a snapshot of this is shown below. A scan with the guessing option enabled was also made and the results from that scan is also presented below.

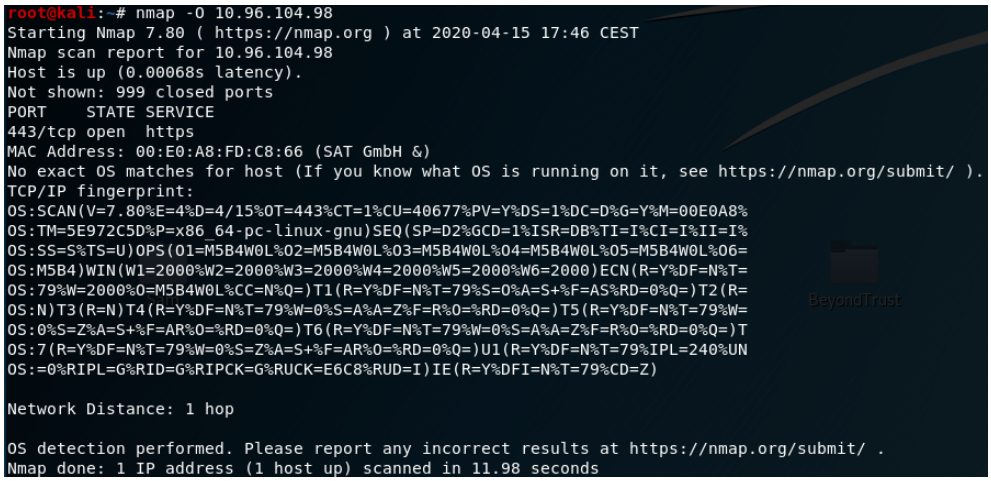

Figure 3.1: Nmap operating system scan

As shown in figure 3.1, Nmap was unable to find a exact OS match. Therefore a scan with the -osscan-guess will be made. With the -osscan-guess flag, Nmap is forced to lists the closest matches, even if a full match has not been found.

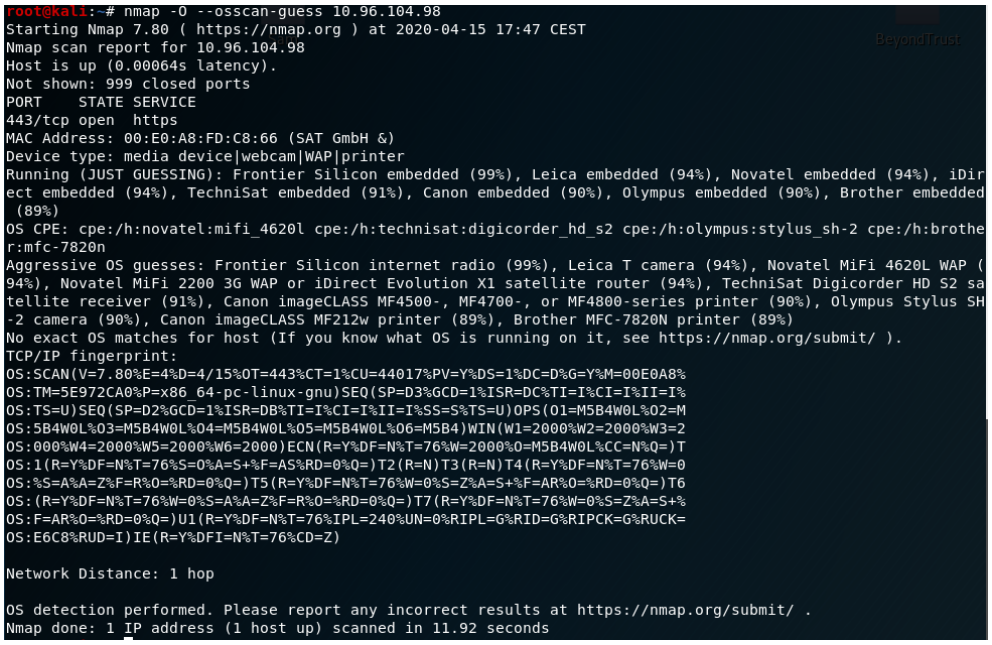

Figure 3.2: Nmap operating system scan with the guessing option enabled

The best guesses as can be seen in figure 3.2 were Frontier Silicon embedded with 99% confidence, followed by Novatel, Idirect, TechniSat and Canons embedded operating systems.

This is rather unlikely, the SICAM CMIC is produced by a large company, so they probably would not reuse an operating system built for another embedded device. It is more likely that the SICAM CMIC and these other embedded operating systems listed share a common "parent" operating system that it has extended. Even if it was accurate, very little information is out on vulnerabilities and exploits for these operating system, so going forward, this report will discard this result as useless and or inaccurate.

3.2.2 Port scan

By using Nmap again, probes are sent to the target to try to determine which ports and services are running. Some of the most basic scans described in section 2.2.2 are launched, e.g. the TCP SYN, TCP Connect, UDP and Internet Protocol scans. Some less popular techniques like the TCP Window and TCP Maimon scans are also performed. All scans were run with default settings, with no port range specified.

The TCP SYN scan came back to show that TCP port 443 is open. 443 is the default port for HTTPS[30] which is consistent with information gathered

```
root@kali:~# nmap -sS 10.96.104.98
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-15 17:22 CEST
Nmap scan report for 10.96.104.98
Host is up (0.0071s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE
443/tcp open  https
MAC Address: 00:E0:A8:FD:C8:66 (SAT GmbH &)

Nmap done: 1 IP address (1 host up) scanned in 0.56 seconds
```

Figure 3.3: Nmap TCP SYN scan

from the SICAM CMIC's display which shows that web server firmware is installed.

```
root@kali:~# nmap -sU 10.96.104.98
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-15 17:23 CEST
Nmap scan report for 10.96.104.98
Host is up (0.00064s latency).
Not shown: 999 closed ports
PORT    STATE         SERVICE
123/udp open|filtered ntp
MAC Address: 00:E0:A8:FD:C8:66 (SAT GmbH &)

Nmap done: 1 IP address (1 host up) scanned in 289.15 seconds
```

Figure 3.4: Nmap UDP scan

After running the UDP scan, Nmap flagged UDP port 123 as being open, the port that the Network Time Protocol(NTP) uses by default[30].

The port scans show that the HTTPS and NTP ports are open, which seems reasonable and valid when pairing this information with information gathered from the SICAM CMIC user manual. The user manual mentions using HTTPS for communicating with its web server and NTP for communication with time servers[29].

3.3 Threat modeling

The information gathered from the scans and SICAM CMIC user manual was used to create a model of the system. Threats were enumerated using STRIDE, and a few key threats selected for further penetration testing.

3.3.1 System model

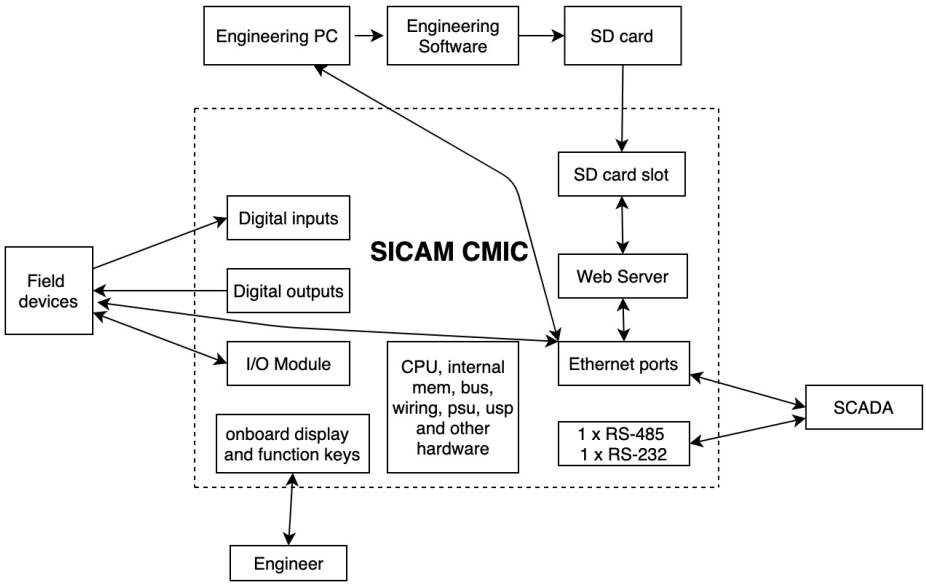

Figure 3.5: SICAM CMIC System model

The diagram shows the SICAM CMIC and its components inside the trust boundary. The ways of entering the system is going through the same path a field device would take, that is using digital inputs and outputs, alternatively communicating with a I/O Module, for example an analog converter. Other ways of crossing the trust boundary of the SICAM CMIC is using the RS-485/232/Ethernet interfaces, either by using the engineering software or web server. The SD-card and the onboard function keys and displays are also pathways into the system.

3.3.2 Identifying threats

As the major pathways to enter the system has been identified, these can be regarded as the general attack surfaces and it makes sense to focus on each one when trying to identify threats to the system as a whole. From a system designers perspective, these would be the points of entry that one would want to protect. These paths are the SD card, the RS-232/454/Ethernet interfaces, the digital I/O and potential I/O coupling modules and the onboard function keys and display.

As the initial port scans showed that the HTTPS and NTP protocols are active on the SICAM CMIC, this gives us one attack surface that is worth looking further into. The web server that runs on the HTTPS server is open and running, but requires a username+password combination to log in. This can only be stated for the specific SICAM CMIC that was tested in this scenario as the system design allows for HTTPS/NTP, but by no means enforces its use.

Because of the limitations of this thesis, only the SD card and communication over the Ethernet TCP/IP interface will be considered, with focus on the web server in particular. The onboard function keys has also been manually tested, without finding any glaring weaknesses or vulnerabilities other than displaying information about the RTU. Some of the information displayed was actually useful like the MAC and IP address as well as the installed firmware and versions of that firmware. So the information disclosed through the onboard display could potentially aid the attacker in the same way it aids the engineers.

By first grouping threats by targets, and then using STRIDE to aid in brainstorming threats, multiple threats were enumerated and a set of threats were selected for further testing and analysis.

Ethernet TCP/IP interface

The Ethernet TCP/IP interfaces can be used to both communicate up, i.e with a control system or downstream, to field devices or routers. Ethernet operates on the physical layer of the OSI model, thus many higher layer protocol may be used for actual data communication. Many modern RTU's ship with FTP, NTP and HTTP servers, even though some of these protocols are considered insecure. So when thinking about threats to the Ethernet TCP/IP interface, it must also be considered that other application layer protocols are involved in the actual communication. In the case of the SICAM CMIC the initial port scans has already confirmed that both NTP and HTTPS protocols are activated over the Ethernet TCP/IP interface.

The most obvious threat to the Ethernet communication channel is that of an attacker that gains access to the RTU and issues control commands to connected devices[11]. Because Ethernet and TCP by default offers no encryption or authentication, they would be considered insecure, thus the communication that goes through this channel must be considered unsafe without protection from higher level protocols, something that is also considered as one of the main threats to remote terminal units[11]. This was the case for a group of RTU's used in several european countries where a vulnerability that allowed a

rogue node on the network to send unauthorized commands and take control of the industrial process[31].

A previous experimental study[32] describes a scenario of an insider attack where an attacker connects to a switch that is located between the RTU and a control system. By connecting to a port that has been configured to act as a mirror, all communications between the RTU and the control system will be copied to the mirror port, where the attacker is waiting. By sniffing on the captured packets, the attacker could gain knowledge on how to forge control messages as to be able send spoofed control messages. The study also mentions an easy way to bypass switch authentication by just disconnecting the plug from the switch and connecting it to the attackers laptop instead.

As for all devices connected to a network, or offering services to multiple users, denial of service must be considered as a potential threat. This threat has been confirmed in several RTU's as reported by CERT.LV[31] when they found a vulnerability that lead to a denial of service condition. Even the SICAM CMIC has had vulnerabilities surfaced regarding denial of service, specifically a so called Billion laughs attack that required a manual restart to restore operations[13].

When thinking about the application layer protocols that could be in use, the focus will be on the HTTP(S) protocol and the web server that the SICAM CMIC ships with. One obvious threat is authentication and most web applications employ password authentication. During an assessment of the network security of power substations[11], some of the vulnerabilities discovered included usage of weak or default passwords.

Another group of relevant threats to web applications in general are that of cross-site scripting attacks(XSS), as proven by the CERT.LV[31] report mentioned previously, where they found an XSS vulnerability that allowed for arbitrary code execution in a RTU.

By using STRIDE, some of the most relevant threats specific to the Ethernet TCP/IP interface and its communications are enumerated in table 3.1.

Spoofing	An attacker acting as the control system could forge control messages to field devices
Tampering	An attacker staging a man-in-the-middle attack to extract and modify information going to and from the RTU
Repudiation	An attacker using IP Spoofing to hide its identity in the case of a DOS attack
Information Disclosure	An attacker finding a cross-site scripting vulnerability could inject a script that steals credentials that users enter into the web server login form
Information Disclosure	An attacker sniffing on packets sent either by field devices or control center could be able to find information on how devices are being used or the data they gather
Denial of Service	An attacker causing a denial of service through one of the open interfaces, for example TCP or HTTP flooding
Elevation of Privilege	By cracking the web server password, an unprivileged attacker could gain privileged access to the system

Table 3.1: Threats to the Ethernet TCP/IP interface enumerated using STRIDE

SD card

In normal operation, the SD card is inserted in the RTU for it to be able to access firmware and read and write parameters. By just removing the SD card from the RTU, it shuts the device off, and an attacker would have performed a denial of service. After removing the SD card, it can be inserted into any laptop with a SD card reader, and its contents can be read and written. But as mentioned earlier, the SICAM CMIC with its inserted SD card is locked inside a metal box. Also worth mentioning is that in case of removing the SD card, administrators would most likely be alarmed, increasing the risk of being caught. So attacking the SD card would probably not be an attackers first option, however it will be considered in this report because this is all being done in a laboratory setting, with enough time and without worries of being discovered. Some known vulnerabilities of SD cards in general as well as directly connected to the SICAM CMIC will be discussed.

SD, short for Secure Digital is a proprietary non-volatile memory card that was developed for use in portable devices[33]. Although it was long viewed as a safe way to store data, a vulnerability was recently surfaced. A group

at the Chaos Computer Congress[34] found a vulnerability that enables arbitrary code execution on the card itself. Furthermore the group showed that the vulnerability found could allow attackers to stage a man-in-the-middle attack where they would be able to intercept and manipulate communications between the card and the device that uses it.

There's also the threat of an attacker trying to reverse engineer the operations of the RTU by systematically modifying, adding and removing files from the SD card to try to gain knowledge of what actually is going on under the case.

Another potential threat is an attacker scanning through the files of the SD card, where there are possibly files stored unencrypted that leak secret information about the hardware and software inside the RTU. This information could be the operating system, libraries, web servers that the RTU uses. By having this knowledge, an attacker can more easily sift through the long list of potential vulnerabilities. By using the STRIDE potential attacks to the SICAM CMIC via SD card are enumerated in table 3.2.

Spoofing	By staging a man in the middle attack, an attacker could act as the SD card to intercept and manipulate the communications between the SD card and the RTU
Tampering	An attacker simply modifying the contents of the SD card
Repudiation	An attacker modifying the contents of the SD-card to clear his tracks, i.e delete or modify logs
Information Disclosure	An attacker decoding the contents of the SD-card, as to disclose information about credentials, configuration parameters, sensor data, hardware and software used by the RTU
Denial of Service	An attacker using the SD-card as a mean to deny service, either by removing the card or corrupting the contents of the card.
Elevation of Privilege	An attacker modifying the contents of the SD-card to bypass authentication. This could for example be done by modifying authentication credentials or finding another way to run arbitrary code by injecting files into the SD card

Table 3.2: Threats to the SD card enumerated using STRIDE

3.3.3 Assess threats

Attacking the SD card, requires a little bit more effort in terms of actually taking physical action. In a real world scenario, the SD card would be placed inside the RTU, inside a locked metal box in a remote location. For starters, getting hold of the SD card requires intelligence on where the RTU is stored. Now getting through the locked metal box may be of varying difficulty depending on the location of the box. Placing the RTU in plain view obviously increases its security versus physical attacks. On top of that, removing the SD card shuts the system down, something that could alarm administrators. This makes the SD card a more risky target than just connecting to the RTU via one of its interfaces or through a switch or router. Thus, most of the focus will be put on remotely available attacks and exploits, specifically on the Ethernet TCP/IP interface. Starting off, vulnerability scanners targeting the web application will be used to see if automated tools can aid us in finding vulnerabilities. Two specific threats has already been identified and considered interesting enough for a more thorough search and these are password cracking and denial of service attacks. A shallow scan and exploration of the SD card will also be performed. To sum it up, the threats that will be assess more thoroughly are:

1. Cracking the password of the administrator account of the web application
2. DOS attacks over the Ethernet TCP/IP interface
3. Tampering and information disclosure on the SD-card

Chapter 4

Vulnerability scanning the web application

In this chapter, results from vulnerability scanners targeting the web application will be presented. Firstly in section 4.1, the tools used and their scan reports will be presented. Then in section 4.2 a discussion and analysis about the vulnerabilities found will be included as well as a conclusion regarding the findings.

4.1 Vulnerability scanning tools

Many different vulnerability scanning tools were tested, and their reports will be presented in this section.

Nmap Scripting Engine(NSE)

By using Nmap Scripting Engine(NSE), scans were ran against the target, and more specifically the scripts in the denial of service category[35]. This scan warn about the web server being vulnerable to a HTTP-slowloris attack. This will be covered more in depth in a later section.

OWASP Zed Attack Proxy(ZAP)

OWASP ZAP is one of the most popular web application scanners available, it is open-source and free-to-use[36].
A scan was ran using the traditional spider option with default settings. The results came back with 4 warnings as shown in the figure below.

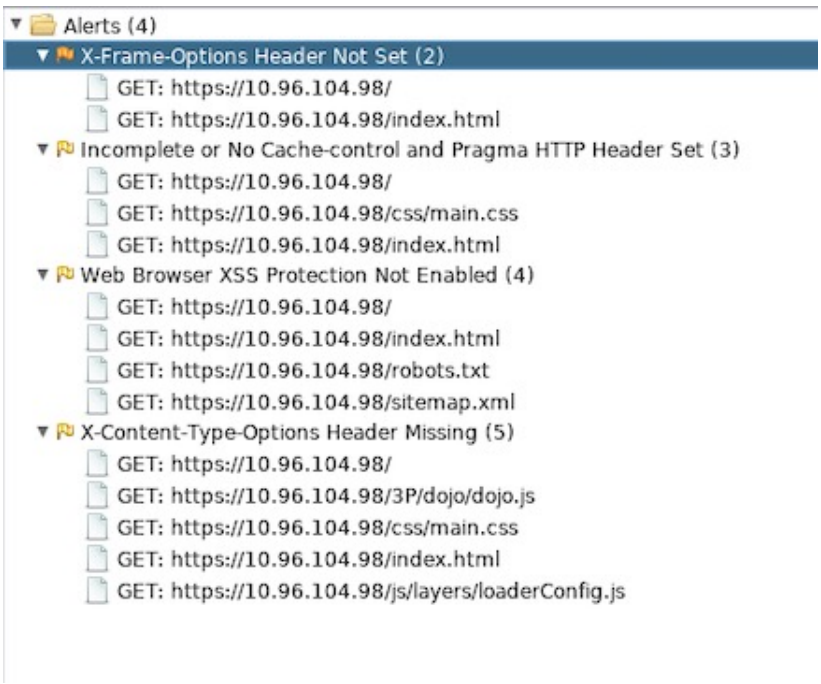

Figure 4.1: Scan report from OWASP ZAP

Nexpose

Nexpose is a network and vulnerability scanner that identifies active services, ports and applications on a target host. It also attempts to find vulnerabilities that may exist based on the known services and applications[37].

Multiple scans were run using different scan templates, including full audit, denial of service and exhaustive mode. All potential vulnerabilities that the scans flagged for are listed below:

- tlsv1_0-enabled
- ssl-self-signed-certificate
- ssl-cve-2016-2183-sweet32
- ssl-cve-2011-3389-beast
- ssl-static-key-ciphers
- ssl-3des-ciphers

Nikto

Nikto is an Open Source web server scanner that performs tests against web servers for over 6700 potentially dangerous files and programs and also checks for outdated versions and version specific problems. It also checks server configuration and will try to identify installed web servers and software[38].

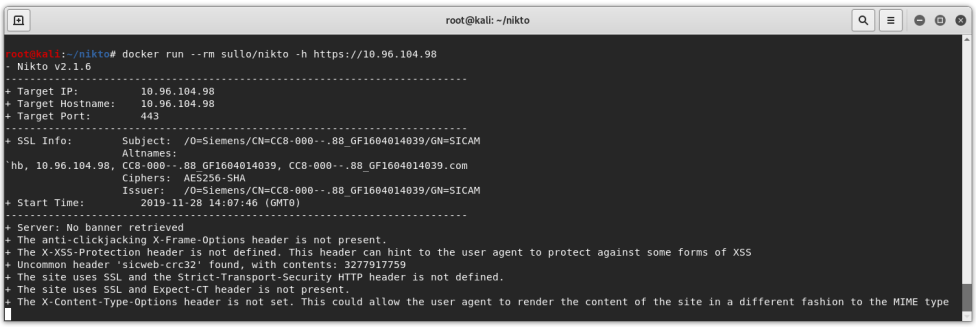

Figure 4.2: Scan report from Nikto

The tool came back with the following warnings:

- X-Frame-Options not present
- X-XSS-Protection header is not defined
- Uncommon header 'sicweb-crc32' found
- Strict-Transport-Security HTTP Header is not defined
- Expect-CT header not defined
- X-Content-Type-Options not set

Nikto was also unable to determine the web server version.

SQL Map

SQL Map is a penetration testing tool that can be used to detect SQL injection flaws[39]. The tool was used to test if the login screen of the web server was vulnerable to SQL Injections. The tool was unable to find any vulnerabilities concerning SQL Injections in login-form of the web application.

4.2 Summary of results

The automated vulnerability scanners helped show some potential vulnerabilities in the web server. One of the scans showed that the web server is potentially vulnerable to a HTTP-slowloris attack, this is something that will be covered more in depth in a later section where denial of service attacks will be presented.

Most of the scans came back to show vulnerabilities in HTTPS configuration, that is HTTP headers, TLS certificates, versions and cipher suites. These potential vulnerabilities will be explained and discussed more in depth in this section.

X-Frame-Options Header Not Set

The X-Frame-Options header is used to indicate whether a browser should be allowed to render pages in frames and embedded objects. By ensuring that content is not embedded into other sites, administrators can use this flag to avoid clickjacking attacks[40]. Clickjacking is a malicious technique of tricking users into clicking on something different than what they perceive. A

common example of this is hiding buttons underneath other content, potentially revealing confidential information or allowing attackers to take control of their browser[41].

Incomplete or No Cache-control and Pragma HTTP Header Set

The Cache-Control and Pragma headers are used to instruct browsers on how they are allowed to cache requests and responses[42]. For example telling the browser to never store information in the cache regarding the current site. The Pragma header is used for backward compatibility where the Cache-Control header is not yet present[43].

By allowing browser caching, many redundant requests and responses can simply be stored in memory, and reused to limit the bandwidth load. But it also leaves potentially confidential information stored unencrypted in the browser cache, this can include passwords and other valuable information.

Web Browser XSS Protection Not Enabled

The X-XSS-Protection header stops pages from loading when the browser detects a reflected cross-site scripting attack(XSS). This header is only used for legacy support in newer browsers where the Content-Security-Policy header is used to disable the use of inline Javascript[44]. Reflected cross-site scripting attacks occur when an attacker embeds a malicious script inside a HTTP query string. If a website displays the contents of the query string without sanitizing, it is possible for an attacker to trick unsuspecting users into running their malicious scripts[45]. One common example of this is search bars, where the search string is commonly displayed in the resulting page. If instead of a correct search string, a malicious <script> is inserted in the search field. The resulting web page will include that malicious script, and be ran as a result. When URL's containing malicious scripts are sent out to victims, this is considered a reflected cross-site scripting attack[45].

X-Content-Type-Options Header Missing

The X-Content-Type-Options response header is used by servers to indicate that the MIME types advertised in the Content-Type header should not be changed. This is a way to opt out of MIME type sniffing that is done by many modern browsers[46]. MIME or Multipurpose Internet Mail Extensions is a standard to describe the type and format of a document or file[47]. If a browser sends assets to a server without specifying the MIME type, browser

has the ability to sniff out the correct type. With type sniffing, some vulnerabilities arise. The vulnerability occurs when an attacker for example disguises a HTML or Javascript file as a different file type, e.g. a JPEG file. A poorly configured web site would perhaps accept JPEG uploads for profile pictures, but instead a malicious attacker injects Javascript code that would be ran instead, providing an opportunity to execute cross-site scripting attacks[48].

Strict-Transport-Security HTTP Header is not defined

The Strict-Transport-Security response header tells web browsers that the site should only be accessed with HTTPS as opposed to HTTP[49]. A vulnerability exists where a website accepts connections over HTTP initially and then redirects to HTTPS. This leaves an opportunity for eavesdropping and man-in-the-middle attacks[50].

Expect-CT header not defined

Certificate Transparency is a technology that aims to make it impossible for a certificate authority to issue a SSL certificate for a domain without the certificate being visible to the owner of that domain. By providing a certificate log that keeps track of records of certificates, the technology helps protect users from being duped by certificates that were mistakenly or maliciously issued[51]. The Expect-CT header is used by servers to tell browsers that they should enforce Certificate Transparency requirements to prevent the use of misissued certificates from that site from going unnoticed[52].

TLS v1.0 enabled

TLS or Transport Layer Security is a protocol that ensures communications are encrypted. HTTPS is basically HTTP using TLS. TLS 1.0 is a rather old and outdated protocol that dates back to 1999[20]. Several security standards like the PCI Data Security Standard and FIPS-140-2 requires a minimum of TLS v.1.1 and even recommends using TLS v1.2[53]. Some of the most prevalent vulnerabilities relating to TLS v1.0 includes the Heartbleed, POODLE, BEAST and CRIME vulnerabilities, all of whom has been used in notable breaches[54]. In general, vulnerabilities in the TLS protocol can allow for man-in-the-middle attacks and could endanger confidential information like credit card data, intellectual property and credentials.

Self-signed SSL Certificate

This is simply a warning that tells us that the web server is using a self-signed SSL certificate. What this means is that the certificate has not been issued by a CA(Certificate Authority) i.e. a trusted third party. However, this is common for internal sites just like the one that is being scanned, because of the cost and ease of configuration and maintenance.

SWEET32

TLS commonly use block cipher algorithms like AES, DES and Triple-DES to encrypt data between server and client. What differentiates block ciphers from other ciphers is that they operate on fixed-length blocks. The algorithm works by breaking the original data into blocks of fixed-length and each block is encrypted separately. Common block sizes include 64 and 128 bits and it is well-known that shorter block sizes makes block ciphers vulnerable to birthday attacks[55].

The birthday attack is named after the birthday paradox which says that in a set of n randomly chosen people, one pair of them will have the same birthday with a surprisingly high probability. By using the pigeonhole principle, the probability reaches 100% when the number of people in the group exceeds 367. But with just 70 people in the group, the probability that any two has the same birthday is 99.9%[56]. The same principle can be used to find hash collisions[57], something that attackers can use to decrypt encrypted information[55].

Because the web server allows for 64-bit Triple-DES encryption, it becomes vulnerable to the SWEET32 attack. Some of the mitigations include using 128-bit ciphers and not allowing long-lived connections[55].

BEAST

BEAST or Browser Exploit Against SSL/TLS is an attack that allows a man-in-the-middle attacker to extract information from an encrypted HTTPS session[58][59]. Many popular ciphers are block ciphers, but one issue with block ciphers is that they are prone to chosen plaintext attacks because a block of plaintext will always result in the same ciphertext for a given key[59]. That is why most secure ciphers use a specific mode of operation, the most common one being Cipher Block Chaining, where each block of plaintext is used in the XOR operation with the previous block of ciphertext. The first block is combined with an initialization vector, a preferably random block of data that

makes each message unique. What makes TLS v1.0 vulnerable to the BEAST attack is the fact that the initialization vectors are not chosen randomly[59].

Use of static key ciphers

Some ciphers use keys which are used for a long period of time, these ciphers are considered static key ciphers. The problem with static key ciphers are that they do not support forward secrecy[60]. Forward secrecy is a feature that assures that session keys will not be compromised in case that the server private key is compromised. It specifically protects past sessions in the case of future compromises of secret keys[61].

3DES Ciphers

TLS v1.0 include cipher suites based on the 3DES algorithm, which only provides an effective security of 112 bits. Many organisations recommends at least 128 bit security, although NIST still considers 3DES to be appropriate to use until 2030[62].

4.3 Discussion

The warnings were generally relating to either missing HTTP headers, or vulnerabilities arising from the usage of TLS v1.0. A missing HTTP header could be a threat, but a missing header alone will most often not cause a vulnerability by itself. For example XSS attacks require some type of user input being reflected in the response to become exploitable, which is not the case with the SICAM WEB login page, the only publicly accessible page from an attackers perspective. Another example is the Strict-Transport-Security header that is missing, but the SICAM WEB server never allows connections through HTTP, so the vulnerability regarding man in the middle attacks because of initial HTTP connections does not exist in this application.

The main vulnerabilities of the web server arises because of the usage of TLS v1.0 and insecure or weak ciphers. A set of known attacks has been launched because of these vulnerabilities like the Heartbleed, POODLE, BEAST and CRIME and SWEET32. Because these attacks are already publicly known and described in detail, it is of lesser value to try to recreate these to launch a identical attack on the SICAM WEB. As the application is to be used internally, issues regarding the certificate being self-signed will not be regarded as a high-risk threat.

4.4 Conclusion

The SICAM WEB application uses TLS v1.0, a version of TLS that is outdated and insecure. A series of vulnerabilities, as explained previously exist in the TLS v1.0 and therefore also in all applications that uses it, including the SICAM WEB application. The web application also supports a set of weak or static ciphers that cause vulnerabilities.

Chapter 5

Penetration testing: Password cracking

After the initial threat modeling and scanning, the scope is zoomed into testing the password authentication system, denial of service attacks and scanning the SD card. The following chapter will present theory, method and results regarding the password cracking attack.

5.1 Password cracking - Theory

By opening a browser and making a HTTPS GET request to the SICAM CMIC's IP address, it can be confirmed that indeed, the server is running a HTTPS server on TCP port 443. The first thing that the user is greeted by when browsing the web page is a login screen. This is obviously a designated entry-point into the system, protected by an authentication mechanism that is password protection. A theory study was performed on password cracking techniques, and some of them are performed and presented below.

5.1.1 Password authentication systems

Passwords has been used as a means of authentication since ancient times. Back then watchtower guards used to ask visitors for the password, and if the correct password was given, the visitor was allowed to pass. Since the inception of the Internet and the world wide web, which are publicly accessible to anyone with a modem connected to a Internet Service Provider, passwords has became a mainstay of HTTP/HTTPS authentication. Although a new trend of using two-factor authentication in place of just using passwords has emerged, many platforms still employ simple password authentication.

Password authentication over HTTP traditionally have the web server ask the client browser for credentials i.e. username and password via a web form. The credentials are then sent via HTTP to the server who then tries to verify by comparing the username password combination with the data it has stored in a database. One problem with this approach is that if the web server is compromised, the passwords that are being stored in plaintext can be stolen. That is why most modern systems employ password hashing. Instead of storing plaintext passwords in the database, the web server uses a cryptographic hash function to compute a hash of the original password, that it then stores in the database. Cryptographic hash functions are often called one-way functions because given an input, in this case a password, it is easy to compute the hash. But given a hash, it should be very hard to invert the function to retrieve the original password. With hashing employed, even if the server is compromised, only hashes of the original passwords can be stolen. While it is theoretically very hard to invert the operations of the hash function, it can still be possible to retrieve the original password given the hash. Simply put, if an attacker knows which hash algorithm is being used, he/she can simply try to input common passwords into the hashing algorithm and then compare its resulting hash with the stolen password hash.

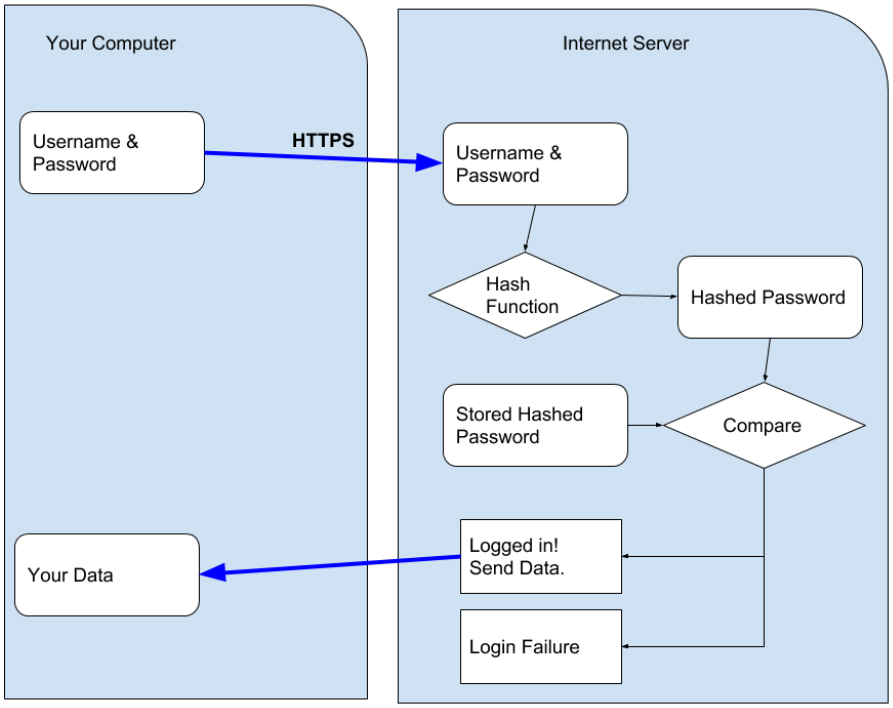

Figure 5.1: Typical login flow from [63]

Attacks where an attacker has came across one or multiple password hashes, are classified as offline attacks[64]. Because a password hash is already retrieved, the attacker no longer needs to communicate with the server to try different passwords, everything can be done locally. This means unlimited amount of attempts, and an unlimited amount of time . This differs from the scenario where an attacker does not have access to a stolen password hash, but instead have to rely on the server to verify passwords. The attack has to be carried out online, which is then classified as a online attack. Attacks that require active communication with the server are not as popular as offline attacks because of the various protection mechanism that modern web servers employ. Captcha images, maximum unsuccessful authentication attempts, firewalls and intrusion detection systems and limited bandwidth are some of the factors that often makes a online attack difficult to execute[64]. However, if these security mechanism are not properly configured, an online attack may still be possible.

5.1.2 Brute-force attacks

The simplest and most obvious way to crack a password is to just try every single possible password. Theoretically, this approach is guaranteed to succeed, given enough time to complete the attack[64]. One of the downsides of the brute-force attack is that the number of attempts required at worst grows exponentially with an increasing password length. This makes this attack rather ineffective against longer passwords[64], or against systems that employ a large minimum password length requirement. Having a lower bound on the password length is seen as a very viable way to strengthen password security against brute-force attacks. Because the longer the password is, the greater the search space for the attacker becomes[65]. All password authentication schemes has defined a set of permissible character sets that is allowed in a password, for example only allowing passwords to be made up of characters in the ranges A-Z and 0-9. Given a character set and an upper bound on the password length, the total amount of possible passwords can be calculated as shown in formula 5.1 where M is the number of characters in the character set, and N is the maximum amount of characters in the password.

$$\sum_{n=1}^{N} M^n \tag{5.1}$$

The time complexity of the brute-force attack is a polynomial of the size of the character set and exponential of the password length[65]. Therefore it is practically unfeasible to brute-force a password longer than 7 characters with a large enough character set[65].

5.1.3 Dictionary attack

The second password cracking technique is the dictionary attack, where instead of trying every possible password, a list of more likely passwords are tried instead. These dictionaries are usually crafted from leaked password lists from previous attacks and breaches. There are many password lists available on the net, one example is the RockYou data breach that occured in 2009. The company had stored their users passwords in plaintext, and when hackers found a vulnerability in their database, the passwords were recovered and leaked onto the internet[66]. An alternative approach to creating a password list is by analyzing the target to create a dictionary of words and phrases used on the target website, twitter feed, facebook feed or any other publicly accessible media. By starting off with a dictionary like explained above, it can be

extended by using a rule-based approach to appending numbers, special characters, or substrings to every password in the list. Ever since websites started to force users to use numbers in their passwords, many users simply append a single number at the end of their passwords. By using a rule to target these types of passwords, the original dictionary can easily be extended to fit these needs. There are many tools out there that can perform automated password cracking attacks or dictionary generation. One of these are John The Ripper[67], an open source password cracking tool that supports brute-force mode and is also able to apply rules to modify or "mangle" dictionaries.

5.2 Password cracking - Method and results

In this section, the methods used to launch a password cracking attack will be explained and its result presented.

5.2.1 Brute-force attack

In the SICAM CMIC's official manual[29], the rules on password selection state that a password should be of maximum length 6 characters, consisting of letters, digits and 31 special characters.

Given the fact that the target password is of maximum length 6, and is made out of 10 numbers, 52 letters and 31 special characters, which equals a character set of size 93. So first let's answer the question, how many possible passwords are there in this scheme, i.e. at worst, how many passwords will have to be tried to guarantee success. Using formula 5.1 this number can be calculated as:

$$\sum_{n=1}^{6} 93^n = 654022685443$$

So there are about 654 billion possible passwords, which sounds like an awful lot. Now that it is known how many passwords that will have to be tried, it must be considered how fast one attempt can be performed in order to determine the viability of this approach.

The initial tests were done using THC-Hydra, which is a network password cracking tool catered towards researchers and security consultants[68]. Default username(administrator, guest) was used together with Hydra's brute-force option with the specified character set targeting the login form sent out by the SICAM CMIC web server. Because THC-Hydra has no option to reuse

a single TCP connection, it has to create new TCP connections for each password attempt, something that brings a lot of overhead, both for the attacker, and of course for the server as well. The initial brute-force attempt failed because the connections started by THC-Hydra timed out. This is because Hydra by default sends out 16 parallel login attempts over 16 TCP connections, something that the web server does not seem to handle very well. Even going as low as using more than 4 parallel requests makes the server unresponsive. The web server becomes unresponsive and the SICAM CMIC even restarted at one point, although this was a bit inconsistent in that it only restarted some of the times this test was performed. So the server seems to have an issue with handling many concurrent TCP connections, either because it is reaching maximum capacity, or that it has some intrusion detection mechanism that only permits a small number of connections.

To bypass the issue with the server becoming unresponsive, attempts were made using the HTTP header Connection: Keep Alive, which allows for a HTTP session to be kept alive on a single TCP Connection. Using Python, a script was written that uses a variable number of threads to send multiple login requests to the server, using a single TCP connection. Using this approach, increasing the number of threads above 4, made the server respond with HTTP status code 500, which signifies Server Error. Using 2 threads gave stable results with a throughput of about 10.000 requests over a 135 second span. During this window, approximately 75 attempts per second was made which corresponds to 0.0135 seconds per request. Given the time it takes to send one request, and the number of total requests needed to guarantee a successful password retrieval the time it would take can be calulcated as:

$$654022685443 * 0.0135 = 8829306253 seconds = 279,8 years$$

So with the current throughput of 75 attempts per second, it would take almost 280 years to complete the full attack. Even if 10 concurrent TCP connections was used, it would not decrease the time required enough to make the attack viable. Despite this, during the testing of the password cracking script, all passwords of length 0-3 was tested with the administrator user unsuccessfully.

5.2.2 Dictionary attack

Because the brute force attack was deemed unviable, next up is the dictionary attack which makes use of the fact that most passwords are based on human language. Therefore, using a dictionary of common words and passwords as a

basis for the attack makes sense. One great candidate for a dictionary is the the RockYou password list[69], that was leaked during a breach in 2009. The list contains 14,341,564 unique passwords, and after manipulating the list to fit the SICAM CMIC's specified character set and maximum length, the resulting list consisted of 2186927 passwords. Combined with the default usernames "administrator" and "guest", the passwords were used by the same python script described in the brute force attack. None of the passwords contained in this password list was correct for any of the usernames.

Next up is using mangling rules to modify the password list to better fit our specified character set and secure password guidelines. But because of time constraints coupled with a rather slow throughput of passwords, this was not really possible. If the RockYou password list is pruned to only contain passwords that comply with the allowed character set, and maximum length of either 4 or 6, so to be able to append numbers and specials and still be shorter than length 6, this list came out to contain 20463 entries. When applying Korelogic's AppendNumbers_and_Specials_Simple rule[70] which appends numbers and specials to each word in the list, the grew to contain about 1.5 billion passwords, which would fall out of the timing of this project.

Instead of using the password list from RockYou, a custom dictionary was created from common swedish and english words, combined with words and terms that are connected to the SICAM CMIC, its brand name, model name, project name and more. This custom dictionary originally contained 44 passwords as seen in appendix B, but after using Korelogic's AppendNumbers_and_Specials_Simple rule[70], the dictionary was extended to 1590162 entries. With the resulting dictionary also mangled with the Korelogic's ReplaceLettersCaps, the total counts up to 5338401.

Using the same original custom dictionary, AppendNum_AddSpecialEverywhere was also used, resulting in a dictionary of 50840 passwords and extended with ReplaceLettersCaps to a total of 212660. Both these resulting dictionaries were tried unsuccessfully.

5.3 Discussion

Because brute-force attacks was deemed unviable, a fully randomized password would be rather secure in the current scheme, even though the existing limitations on password length. With a fully randomized password, it is very unlikely a dictionary attack would recover it successfully.

5.4 Conclusion

All in all, close to 10 million passwords were attempted, without success. Even though no login credentials were retrieved, these tests helped expose a potential denial of service vulnerability in the web server. These tests also help show that there are factors other than password length that can affect the viability of a brute-force attack.

Chapter 6

Penetration testing: Denial of service

After the initial password cracking attempts and vulnerability scans, it was clear that the SICAM CMIC was vulnerable to denial of service attacks over TCP Port 443(HTTPS). This chapter will start off by introducing denial of service attacks, as well as some of the most common attacks. Some of these attacks will be launched against the SICAM CMIC and their results will be presented as well.

6.1 Denial of Service - Theory

Any military or commercial application should have the goal of maintaining availability to its intended users. Denial of Service is a type of attack where an attacker tries to limit or deny users access to a service by interrupting the server's normal function[71]. The primary means of denying service is by over-saturating the capacity of the target machine[71]. Denial of Service(DoS) attacks differ from Distributed Denial of Service(DDoS) attacks in that a single computer is involved for the attack, rather than a group or network of computers. Denial of Service attacks can be placed into two categories, buffer overflow attacks and flooding attacks[71]. Because denial of service attacks are not limited to a specific layer of the TCP/IP model[72], some sources group attacks by where in the OSI-layer the attack takes place, for example Layer 3-4 versus in Layer 7[73].

6.1.1 Buffer overflow attacks

In computing, buffers are memory partitions used for storage of data within programs. Memory buffers are restricted in size and programs that use them must ensure they stay within the bounds of the memory buffers that are assigned to it. Without proper bounds checking, programs run the risk of "overflowing" its buffer by writing data that exceeds the capacity of the buffer[74]. This can cause memory of other buffers and programs to be overwritten, something that can cause errors, data loss, denial of service or even worse, execution of malicious code. Although buffer overflow attacks are not limited to cause denial of service, this is where the focus will lie within this report. When a buffer overflow is used to cause denial of service, it is usually because the target machine is forced to consume all available hard disk space, memory or CPU time[71].

6.1.2 Flooding attacks

By overwhelming the target machine with network packets, attackers can oversaturate the servers capacity to cause denial of service[71]. There are many different types of flooding attacks, but what they all have in common is that they force the server to spend all or most of its resources to deal with packets emanating from the attacker, causing it to be unable to answer to legitimate requests.

6.1.3 SYN flood

To better understand the SYN flood attack it is important to understand how TCP works, especially during the initial three-way handshake.

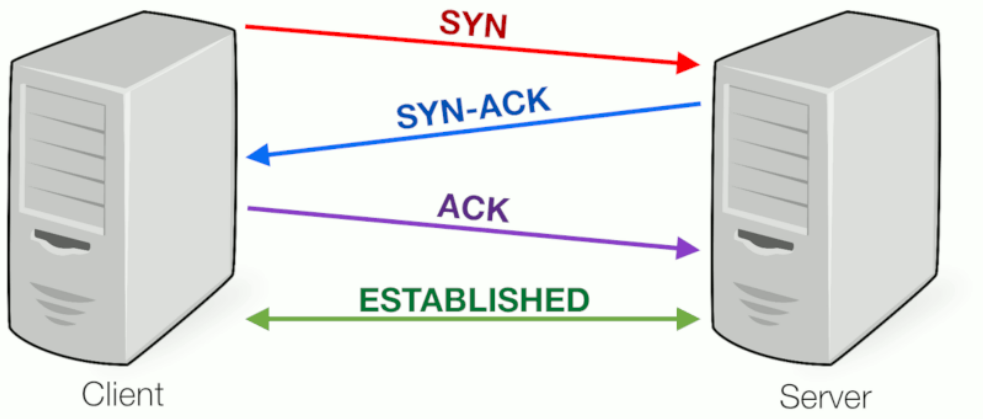

Figure 6.1: TCP three-way handshake from [75].

In short, a client sends a SYN packet to the server to start synchronization. The server will then respond with a SYN-ACK packet to acknowledge that it has received the initial SYN Packet. It then waits for an ACK packet from the client before the connection is properly established. It is the fact that the server has to wait for the last acknowledgement from the client that is exploited in the SYN flood attack[76]. While the server is waiting for the client, it still has to keep all the resources and information available about the client. The actual SYN Flood attack is performed by sending a high volume of SYN packets to the target. The server will respond to all of the attacks with a SYN-ACK packet, but then also keep a port open and ready to receive the response[76]. While the server waits for the final ACK packet, which will never arrive, the attacker keeps sending more SYN Packets. For each SYN packet the server receives, it is forced to keep a new port open. Once all available ports has been used, the server will not be able to make new connections and thus will seize to function properly.

6.1.4 Slowloris

Slowloris is an attack that works by opening multiple connections to a web server and then keeping those connections open for as long as possible[77]. What distinguishes the slowloris attack from many other attacks is that it can cause a lot of harm with very little bandwidth. Web servers only have a limited

amount of threads to handle concurrent connections. When the maximum has been exceeded, the server is not able to respond to new requests, and a denial-of-service will occur. The slowloris attack works by sending multiple partial HTTP request headers[77]. The web server will open a thread for each incoming request, and the the threads will only be closed once the connection is completed. Many modern web servers will close connections that takes too long to finish, as to help free up system resources. The slowloris attack combats this by sending partial headers, as to tell the server that more information is coming, to avoid having the server shut the connection down prematurely[77]. This way, the server is not able to free up any of its open connections and will eventually reach its maximum number of threads and cause a denial of service.

6.1.5 HTTP GET/POST flood

HTTP flooding aims to submerge web servers in enough requests that it has no time to complete a request before dozens more has arrived. It differs from TCP flooding in that it does not look to flood the connection pool of the target, but rather flood the web server with requests. Depending on what we're asking the web server to do, we may get different results. Just asking the web server for a HTML file may not cause the server enough overhead to cause it to slow down. However if the attack is distributed, even a simple request like asking for a file could shut the server down, if enough requests are sent.[78] POST requests on the other hand can be a lot more cost-intensive for the server, as it has to process the incoming data, access databases, create responses etc[78].

6.1.6 Billion laughs attack

The billion laughs attack is a very specific attack on XML parsers, also known as a XML bomb or exponential entity expansion attack[79]. By defining an XML entity that consists of 10 copies of the previous entity, the entity that in the end consists of a single instance of the largest entity will expand to one billion copies of the first entity. This procedure will consume a lot of resources on the server[79], in many cases enough to bring the server down.

This attack was performed by another penetration testing group, and it was shown to be successful in bringing the web server down, requiring a manual reboot. Only the web server was affected by this attack, other services were still working[13].

6.2 Denial of Service - Method and results

By using the THC-Hydra password cracker, as described in section 5.2.1, it suggested that the SICAM CMIC was vulnerable to flooding, however it was unclear at that point the exact extent of the problem. One of the early Nmap scans performed also warned about the SICAM CMIC being vulnerable to a slowloris attack. Thus, this section will try to make a deeper dive into the threat of denial of service attacks over the Ethernet port with focus on attacks on TCP and HTTP. An attack will be developed, and existing tools will also be utilized to stage a denial of service attack against the SICAM CMIC.

6.2.1 Developing a denial of service attack

Using python, the original password cracking script that was used to brute-force the login form for the web server in section 5.2.1, was rewritten to try to stage a denial-of-service attack. The script can be found in appendix A.

In short, the script tries to open up 150 HTTP sessions with the web server, and by continuously making new requests on that connection, the server is forced to keep it alive. Even though HTTP is mainly a connectionless protocol, it still accepts keeping connections alive using the HTTP header Connection: Keep Alive[80]. Using this script, the web server became slow and unresponsive, taking up to 15 minutes to recover. Although very inconsistently, the SICAM CMIC even rebooted at times, and kept rebooting as the attack was ongoing. Once it rebooted, it booted in safe mode. Even in safe mode, this script made the SICAM CMIC reboot. This behavior was very inconsistent, so other factors also have to be considered when thinking about the causes for the restarts.

6.2.2 Using hping3 to stage denial of service attack

Hping3 is a command-line oriented packet assembler and analyzer that supports ICMP, TCP, UDP, and RAW-IP protocols[81]. Using hping3, a SYN Flood attack was staged as shown below. This command caused the web server to become unresponsive for up to 20 minutes after the attack was finished.

Similar ICMP flood, TCP RST and FIN flood attacks were also staged, however without the same results or success. The web server remained responsive during all of these attacks, all performed on a single computer, sending 10000 packets.

```
root@kali:~# curl -k 10.96.104.98:443
curl: (52) Empty reply from server
root@kali:~# hping3 -c 10000 -d 120 -S -w 64 -p 443 --flood --rand-source 10.96.104.98
HPING 10.96.104.98 (eth0 10.96.104.98): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.96.104.98 hping statistic ---
297954 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@kali:~# curl -k 10.96.104.98:443
^C
```

Figure 6.2: SYN Flood using hping3

6.2.3 Recreating the Billion laughs attack

As mentioned in section 1.6, another research group had already made some penetration tests on the SICAM CMIC[13]. Specifically, they were successful in staging a billion laughs attack against the web server through the login form. The web server login endpoint accepts POST requests with an XML payload on the form:

```
<?xml version="1.0" encoding="UTF-8"?>
<Cmd_LogOn>
  <authentication user="admin" pwd="P@ssw0rd"/>
  <language id="en"/>
</Cmd_LogOn>
```

By changing the XML to include a billion laughs entity, the research group was able to cause a denial of service in the SICAM CMIC, requiring a manual restart of the device to recover from the attack[13]. These results were confirmed by sending the same request to the SICAM CMIC. After sending a billion laughs entity to the web server, it was unable to respond to further requests, ultimately requiring a manual restart to recover, just like described in the report[13] mentioned in section 1.6. No other services than the web server was affected by this attack.

6.3 Discussion

After running the Python script as described previously, the web server became unresponsive, but automatically recovered after the attack concluded. Even though the RTU restarted, it did so successfully restarting the web service. It is unclear if any other operations would be disturbed or lost during these restarts. The SYN flood attacks performed with hping3 never caused the RTU to restart, and it also automatically recovered after the attacks seized. What

differentiates the Billion Laughs attack from the other attacks performed, is that this attack actually requires a manual restart of the SICAM CMIC for it to recover.

6.4 Conclusion

The SICAM CMIC is vulnerable to TCP SYN flood attacks and the Billion Laughs attacks, although the consequences of the Billion Laughs attack are harder to recover from.

Chapter 7

Penetration testing: SD card

The SD card is the SICAM CMIC's non-volatile storage drive. Thus, most of the data and programs that is used by the RTU is stored on the SD card. Being able to understand the contents of the SD card could give an attacker the possibility to reverse-engineer its operations to expose vulnerabilities. Although there are some difficulties as the SD card is locked behind a metal container. Because of the difficulties in performing this in a real world scenario, and time limitations lead to only a shallow exploration of the SD card. A brief theoretical introduction and the tests performed with their corresponding results will be presented in this chapter.

7.1 SD card - Theory

SD or Secure Digital is a non-volatile memory card format developed by the non-profit organization SD Card Association that was formed by a group of electronics manufacturers. The standard was formed in 1999, but has since been extended and developed to include MiniSD and MicroSD cards as famously present in many smartphones.

Figure 7.1: SD card like the one found inside the SICAM CMIC. Image from [82]

7.2 SD card - Method and results

7.2.1 Scanning the SD card contents

After initial insertion, the SD card contained many files, most of them with file extensions .P, .F, .BIN and .xml, although client side web files were also found, including html and javascript files. Using Binwalk, a tool designed for identifying files and code embedded inside of firmware images[83] was used to extract content out of the .F and .BIN files present on the SD card. The resulting folders were scanned using the linux tools grep and strings to find any human-readable text of value.

The scans revealed information about third-party library files that is being used by the RTU. Some .xml files with <code> tags containing program code was also found. Another interesting finding was text files indicating that the device was running Micrium microC/OS-II. This was discovered by Emma Good, a colleague of mine that also works on penetration testing the SICAM CMIC.

The most interesting thing found was a RSA Private key. After comparing the RSA Private key to the server SSL/TLS certificate it was shown that they did not match, meaning the RSA private key was used for other means.

7.2.2 Tampering with client side web files

After finding the client-side files from the web server in .html and .js format, attempts were made to make modifications to the initial login page, to see if it was possible to modify the Javascript files as to insert malicious code in the login phase. The idea is that if it is possible to make changes to the client side files, an attacker could in theory, remove the SD card, insert into his/hers PC, make changes to the Javascript files so that, the username and password combination that is sent to the server actually gets sent to another location instead, perhaps the attackers email inbox.

Inside the folder where the web files resided, every .html and .js file has a corresponding .crc and .gz file. The .crc files contain Cyclic Redundancy Check checksums, used to verify data integrity of the files[84]. Most probably to protect against an attack just like this. After some initial probing, it seems like it is the .gz file that is actually being processed by the web server. However, when making changes to index.html and then compressing it using Gzip, the web server responds to requests with an error message. Most likely it is because the corresponding .crc has not been updated with a checksum that corresponds to the newly updated index.html.

So how to bypass the .crc protection? By just deleting the .crc files, the newly updated index.html was displayed to the users browsing the web servers main page. This means, that by simply deleting the .crc files, attackers are able to modify client side web files. This includes Javascript that is being ran on clients that connect to the web server.

7.3 Discussion

Although the scan of the SD card was rather shallow, only string searches was performed. There are still a huge number of tests that could be done to try to inject code in these files. Regarding the RSA Private key found it is still unclear what the use of this private key is, potential uses include encrypting communication between the engineering software and SICAM CMIC or SSL/TLS encryption with the SICAM WEB server, even though comparisons with the server certificate shows otherwise. The most powerful finding was

the fact that attackers are able to modify client side web files as to inject malicious code. One scenario where this could be useful is to inject Javascript code that sends the inserted credentials to another server, perhaps the attackers mail server. This way, an attacker could steal credentials from an unsuspecting authorized user.

7.4 Conclusion

The main findings was the fact that an attacker could bypass the .crc protection to change client side files to potentially steal login credentials from anyone using the web application. A RSA Private key was also found, although its use is unknown. Information regarding operating system and third-party libraries used are also leaked.

Chapter 8

Responsible disclosure

All the findings from this thesis has been responsibly disclosed following Siemens vulnerability handling and disclosure process[85]. The process follows four phases as can be seen in the image below.

Figure 8.1: Siemens vulnerability handling and disclosure process from [86]

1. Report - This is where ethical hackers and other users report their findings to Siemens before making their findings public as to avoid a "0-day situation"[85] that risk putting Siemens customers at risk.

2. Analysis - Siemens will try to analyze the reported vulnerability by recreating the situation it arises in.

3. Handling - Siemens performs internal vulnerability handling with its development groups. National and Governmental CERTs may be notified about the security issue in advance. Siemens will keep regular communication between Siemens and the reporting party, and the reporting party may be provided with pre-releases of software fixes for verification.

4. Disclosure - After the issue has been analyzed and handled internally, if a fix is deemed necessary to cope with the vulnerability Siemens will

release a Security Advisory containing the following information: Description of the vulnerability with CVE reference and CVSS score, identity of known affected products and software/hardware versions, information on mitigating factors and workarounds and location of available fixes.

After reporting my findings to Siemens as described in their official handing and disclosure process. Their response was that the Denial of Service attack described in chapter six has already been patched in newer versions of the firmware. The RSA Private key that was found in the SD Card as described in chapter 7 was a so called "dummy key" apparently not used for anything. Furthermore the SD Card has been further protected from these kinds of readings in the latest patch. As far as changing the client side code of the web application, according to Siemens this is not exploitable as the web application is not able to open connections to anyone outside of the client it is communicating with. When it comes to the vulnerable SSL ciphers that the web server offered to use, Siemens are going to fix this in a future patch.

Chapter 9

Discussion

As all of the attacks described in this thesis was performed against a SICAM CMIC RTU with outdated firmware, it must be considered that many of the weaknesses and vulnerabilities are already patched. This is the case for the Billion laughs attack[13] presented in section 1.6. Although newer versions of the firmware may be more secure, many RTU's in the field still house older and outdated versions of firmware, thus making them vulnerable to some of the attacks described in this thesis.

First off, the vulnerability scanners that targeted the web application all came back with warnings concerning missing headers protecting against cross-site scripting attacks and other forgery attacks. They also warned about the web server using TLS v1.0, a version that has several known vulnerabilities. Although none of these vulnerabilities was further investigated, it must still be noted that a possibility exists for these vulnerabilities to be exploited in the SICAM WEB application. However because of the ease of mitigating these attacks, it is very likely these weaknesses has already been adressed.

One potential weakness that was discovered in this thesis are the fact that the SICAM WEB enforces a password that has an upper bound of only 6 characters. Although it becomes practically infeasible to make a brute-force attack against the SICAM WEB login, a dictionary attack is still viable given that the password consists of words and phrases like many common passwords are. When talking about password cracking and security it becomes important to distinguish between weak passwords and weak security. Obviously weak security enforcement allows for weak passwords that can lead to security breaches. But even with weak password security enforcement like the case with the SICAM CMIC only allowing shorter passwords, other security mechanisms can help patch the problem of weak passwords.

The SICAM CMIC and specifically SICAM WEB server was vulnerable to flooding attacks, either causing the web server to become unresponsive, or even making the remote terminal unit restart. It is unclear if the restarting of the device resumes all of its previous operations. If this is the case the restarting of the SICAM CMIC could be viewed as a protection mechanism, and the severity of the attack limited.

A previously discovered XML Bomb attack was also tested, and was still successful in shutting the web server down, requiring a manual restart of the SICAM CMIC. This attack is a lot more severe than the SYN flood attack because it requires an engineer to visit the remote location, something that could be costly. The SD card that is used as main storage, stored unencrypted client side web files, that was able to be modified to potentially steal passwords and other information from authorized users. The SD card also stored a RSA private key, unencrypted in plaintext, as well as leaking information about the operating system and third-party libraries that is used.

It is clear that Siemens takes security of highest importance as many of these issues had already been fixed since, and those that had not probably will be in the near future.

9.1 Ethics, Sustainability and Socio-economic aspects

As anything else in science, ethics and sustainability should be considered. Not only from the perspective as a scientist, but also from the perspective of affected businesses and ecosystems. In the case of ethical hacking, some ethical considerations has already been introduced. But to cap things off, it is of high importance specifically to the vendor and its customers that no vulnerabilities are leaked prior to the vendor having handled the situation. As explained in chapter 8, this includes a period of time in between the discovery of a vulnerability and the disclosure to the public. This is to ensure that the vendor has time to develop mitigations to the vulnerabilities, so that they can keep their customers safe.

From a socio-economic perspective, the use of ethical hackers has opened up a new field of employment for computer scientists. It also opens up a new avenue for businesses trying to secure their products. They no longer have to fully rely on automated software tools to tell them whether their product is secure, and they don’t have to wait for the "bad guys".

But what happens when a larger portion of the population are aware of

hacking methodologies? How are we going to keep society safe when everyone is a capable cyber attacker? These are valid questions, but it can be argued that a capable attacker is equally capable of defending. And that is what society will have to aim towards, developing as many capable cyber security specialists as possible to secure critical infrastructures.

9.2 Future work

In order to understand the SICAM CMIC and its operations better, doing a more in depth and thorough search and testing of the SD-Card contents would be very valuable. Being able to understand how the SICAM CMIC works could surface more vulnerabilities that is not as easily spotted from a network standpoint. As all the tests in this project has been done to an isolated RTU, without simulating its normal communications with SCADA's and field devices, more testing could be done on its communications with other devices in normal operation. The SICAM CMIC is also able to run the Network Time Protocol(NTP), a protocol that has known vulnerabilities like the monlist command that potentially floods the client with a long list of previous clients[87]. Because of this it would be valuable to carry out tests against the NTP servers and clients on the RTU network.

Chapter 10

Conclusions

The most striking vulnerabilities found was the ones that required physical access to the RTU and its SD card. One of these vulnerabilities was the leakage of a unencrypted private key on the SD card. Another vulnerability that was discovered, again targeting the SD card, was the ability to modify client side web files to inject code. The SICAM WEB server runs an old and outdated version of TLS, a version that has known vulnerabilities that the SICAM CMIC therefor also suffers from. This thesis also shows that the SICAM CMIC is vulnerable to several denial of service attacks, not only the Billion laughs attack that was previously published, but also SYN flood attacks was able to interrupt the system. However because of how the SICAM CMIC is being operated, only connected to a few number of hosts, protecting against remote DOS attacks is perhaps not of a high priority as long as manual recovery is not required. All in all, the conclusions of this thesis must be that the system is rather secure, seeing as the tests made were unsuccessful in gaining access to the device. The thesis also shows that Siemens has already put effort into fixing these issues as some of them are not present in newer versions of firmware. Of those issues that had not already been fixed, Siemens vowed to fix them in a future update.

Bibliography

[1] Ramesh Sanga Aditya Gour et al. “Design and Development of FPGA and FPAA Based Remote Terminal Units for Nuclear Power Plants”. In: 2014.

[2] WaterWorld. *Purpose-built RTUs for Water, Wastewater*. 2018. URL: https://www.waterworld.com/technologies/article/16190193/purposebuilt-rtus-for-water-wastewater.

[3] Hudson Courses. *Ethical Hacker or Penetration Tester: What's the difference?* 2018. URL: https://www.hudsoncourses.com/ethical-hacker-vs-penetration-tester/.

[4] Adam Shostack. *Threat Modeling: Designing for Security*. John Wiley & Sons, Feb. 2014, pp. 1–86. ISBN: 9781118809990.

[5] Michael Wulff. “Making the Case for Ethical Hacking”. Ball State University, June 2009, p. 7.

[6] Linda Rosencrance and Michael Cobb. *Definition: Ethical hacking*. 2018. URL: https://searchsecurity.techtarget.com/definition/ethical-hacker (visited on 05/20/2020).

[7] “Strengthening the digital Achilles heel of the European Union: Make use of ethical hackers to find vulnerabilities in information systems?” Leiden University, July 2017.

[8] MITRE. *CVE and NVD Relationship*. Aug. 2019. URL: https://cve.mitre.org/about/cve_and_nvd_relationship.html (visited on 05/26/2020).

[9] Jr James F. Lea and Lynn Rowlan. *Gas Well Deliquification*. 3rd ed. Gulf Professional Publishing, 2019, p. 344. ISBN: 9780128158975.

[10] Siemens AG. *SICAM CMIC Brochure*. 2013. URL: https://static.dc.siemens.com/datapool/industry/smartgrid/siprotec5/EA-Notes/2013-12/IC1000-G220-A127-V1-4A00_CMIC_Broschuere_EN.pdf (visited on 05/20/2020).

[11] Jeff Hieb. "Security hardened remote terminal units for SCADA networks". University of Louisville, May 2008.

[12] Michael B Kelley. *The Stuxnet Attack On Iran's Nuclear Plant Was 'Far More Dangerous' Than Previously Thought*. 2013. URL: https://www.businessinsider.com/stuxnet-was-far-more-dangerous-than-previous-thought-2013-11?r=US&IR=T (visited on 05/20/2020).

[13] Nicolas Heiniger and Emanuel Duss. *Siemens SICAM A8000 Series Denial Of Service*. 2019. URL: https://packetstormsecurity.com/files/151217 (visited on 05/20/2020).

[14] Ludvig Christensen and Daniel Dannberg. "Ethical hacking of IoT devices: OBD-II dongles". School of Computer Science and Communication, KTH Royal Institute of Technology, 2019.

[15] OWASP. *Category:Threat Modeling*. 2018. URL: https://wiki.owasp.org/index.php/Category:Threat_Modeling (visited on 05/20/2020).

[16] Ida Matero. "Ethical Hacking: Research and Course Compilation". Helsinki Metropolia University of Applied Sciences, Nov. 2016.

[17] David Hafele. *Three Different Shades of Ethical Hacking: Black, White and Gray*. Tech. rep. SANS Institute, Feb. 2004. URL: https://www.sans.org/reading-room/whitepapers/hackers/shades-ethical-hacking-black-white-gray-1390 (visited on 05/20/2020).

[18] Johan Nilsson. "Vulnerability scanners". Department of Computer and Systems Sciences Royal Institute of Technology, May 2006. URL: https://pdfs.semanticscholar.org/c2f1/1c1f68a523a01604e967dc1fe0a3f1ac2pdf (visited on 05/20/2020).

[19] Nmap. *Nmap: the Network Mapper - Free Security Scanner*. 2020. URL: https://nmap.org/ (visited on 05/20/2020).

[20] Nmap. *Nmap: OS Detection*. 2020. URL: https://nmap.org/book/man-os-detection.html (visited on 05/20/2020).

[21] Thomas Glaser. *TCP/IP Stack Fingerprinting Principles*. Tech. rep. Sans Institute, Oct. 2000.

[22] Cyrus Peikari and Anton Chuvakin. *Security Warrior*. O'Reilly Media, Inc., Feb. 2004, p. 229. ISBN: 0596005458.

[23] Roger Christopher. "Port Scanning Techniques and the Defense Against Them". In: (2001).

[24] Nmap. *Nmap: Port scanning techniques*. 2020. URL: https://nmap.org/book/man-os-detection.html (visited on 05/20/2020).

[25] Ying Chen. *Confluence: Nmap*. 2018. URL: https://wiki.onap.org/display/DW/Nmap (visited on 05/20/2020).

[26] Russ Rogers and Jay Beale. *Nessus Network Auditing*. Syngress Publishing, Inc., 2004, pp. 88–236. ISBN: 193836086. (Visited on 05/21/2020).

[27] Netsparker. *The Problem of False Positives in Web Application Security and How to Tackle Them*. URL: https://www.netsparker.com/blog/web-security/false-positives-web-application-security/ (visited on 05/21/2020).

[28] Srinivas. *Process: Gaining and Elevating Access*. 2017. URL: https://resources.infosecinstitute.com/process-gaining-and-elevating-access/#gref (visited on 05/21/2020).

[29] Siemens AG. *SICAM CMIC User Manual*. 2016. URL: https://www.manualslib.com/products/Siemens-Sicam-Cmic-5898773.html (visited on 05/21/2020).

[30] IANA. *Service Name and Transport Protocol Port Number Registry*. 2020. URL: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=8 (visited on 05/21/2020).

[31] Eduard Kovacs. *Vulnerabilities Found in RTUs Used by European Energy Firms*. 2018. URL: https://www.securityweek.com/vulnerabilities-found-rtus-used-european-energy-firms (visited on 05/21/2020).

[32] Abhik Roychoudhury and Yang Liu. *Systems Approach to Cyber Security*. IOS Press BV, Feb. 2017, pp. 78–84. ISBN: 9781614997436.

[33] Wikipedia. *SD Card*. URL: https://en.wikipedia.org/wiki/SD_card (visited on 05/21/2020).

[34] Bunnie's Blog. *On Hacking MicroSD Cards*. 2013. URL: https://www.bunniestudios.com/blog/?page_id=3592 (visited on 05/21/2020).

[35] Nmap. *dos NSE category*. URL: https://nmap.org/nsedoc/categories/dos.html (visited on 05/21/2020).

[36] ZAP Dev Team. *OWASP Zed Attack Proxy*. 2020. URL: https://www.zaproxy.org/ (visited on 05/21/2020).

[37] Rapid7. *Vulnerability Scanning with Nexpose*. 2018. URL: https://metasploit.help.rapid7.com/docs/vulnerability-scanning-with-nexpose (visited on 05/21/2020).

[38] cirt.net. *Nikto 2*. 2020. URL: https://cirt.net/Nikto2 (visited on 05/21/2020).

[39] sqlmap. *Automatic SQL injection and database takeover tool*. 2020. URL: http://sqlmap.org/ (visited on 05/21/2020).

[40] MDN web docs. *X-Frame-Options*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options (visited on 05/21/2020).

[41] Wikipedia. *Clickjacking*. URL: https://en.wikipedia.org/wiki/Clickjacking (visited on 05/21/2020).

[42] MDN web docs. *Cache-Control*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control (visited on 05/21/2020).

[43] MDN web docs. *Pragma*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Pragma (visited on 05/21/2020).

[44] MDN web docs. *X-XSS-Protection*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection (visited on 05/21/2020).

[45] Portswigger web security. *Reflected XSS*. URL: https://portswigger.net/web-security/cross-site-scripting/reflected (visited on 05/21/2020).

[46] MDN web docs. *X-Content-Type-Options*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options (visited on 05/21/2020).

[47] MDN web docs. *MIME types (IANA media types)*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types (visited on 05/21/2020).

[48] KeyCDN. *What Is MIME Sniffing?* 2018. URL: https://www.keycdn.com/support/what-is-mime-sniffing (visited on 05/21/2020).

[49] MDN web docs. *Strict-Transport-Security*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security (visited on 05/21/2020).

[50] Vaijayanti Korde. *The Importance of a Proper HTTP Strict Transport Security Implementation on Your Web Server*. 2016. URL: https://blog.qualys.com/securitylabs/2016/03/28/the-importance-of-a-proper-http-strict-transport-security-implementation-on-your-web-server (visited on 05/21/2020).

[51] Certificate Transparency project. *What is Certificate Transparency?* URL: http://www.certificate-transparency.org/what-is-ct (visited on 05/21/2020).

[52] MDN web docs. *Expect-CT*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expect-CT (visited on 05/21/2020).

[53] Rapid7. *TLS Server Supports TLS version 1.0*. URL: https://www.rapid7.com/db/vulnerabilities/tlsv1_0-enabled (visited on 05/21/2020).

[54] Packetlabs. *TLS 1.0 and TLS 1.1 Are No Longer Secure*. URL: https://www.packetlabs.net/tls-1-1-no-longer-secure/ (visited on 05/21/2020).

[55] Karthikeyan Bhargavan and Gaëtan Leurent. *Sweet32: Birthday attacks on 64-bit block ciphers in TLS and OpenVPN*. 2016. URL: https://sweet32.info/ (visited on 05/21/2020).

[56] Wikipedia. *Birthday problem*. URL: https://en.wikipedia.org/wiki/Birthday_problem (visited on 05/21/2020).

[57] Seth Misenar Eric Conrad and Joshua Feldman. *CISSP Study Guide Book*. 3rd ed. Elsevier Inc., 2015, p. 244.

[58] Rapid7. *TLS/SSL Server is enabling the BEAST attack*. URL: https://www.rapid7.com/db/vulnerabilities/ssl-cve-2011-3389-beast (visited on 05/21/2020).

[59] Zbigniew Banach. *How the BEAST Attack Works*. 2020. URL: https://www.netsparker.com/blog/web-security/how-the-beast-attack-works/# (visited on 05/21/2020).

[60] Rapid7. *TLS/SSL Server Supports The Use of Static Key Ciphers*. URL: https://www.rapid7.com/db/vulnerabilities/ssl-static-key-ciphers (visited on 05/21/2020).

[61] Wikipedia. *Foward secrecy*. URL: https://en.wikipedia.org/wiki/Forward_secrecy (visited on 05/21/2020).

[62] Rapid7. *TLS/SSL Server Supports 3DES Cipher Suite*. URL: https://www.rapid7.com/db/vulnerabilities/ssl-3des-ciphers (visited on 05/21/2020).

[63] Tom Ritter. *Typical login flow*. [Online; accessed May 22, 2020]. Nov. 2018. URL: https://hacks.mozilla.org/2018/11/firefox-sync-privacy/.

[64] Chrysanthou Yiannis. *Modern Password Cracking: A hands-on approach to creating an optimised and versatile attack*. Tech. rep. Information Security Group, Royal Holloway, University of London, May 2013, pp. 5–7.

[65] Gongzhu Hu. "On Password Strength: A Surveyand Analysis". In: (2018), pp. 167–168.

[66] Jaikumar Vijayan. *RockYou hack exposes names, passwords of 30M accounts*. 2009. URL: https://www.computerworld.com/article/2522045/rockyou-hack-exposes-names--passwords-of-30m-accounts.html (visited on 05/22/2020).

[67] OpenWall. *John the Ripper password cracker*. URL: openwall.com/john/ (visited on 05/22/2020).

[68] vanhauser-thc. *THC-Hydra*. 2020. URL: https://github.com/vanhauser-thc/thc-hydra/blob/master/README.md (visited on 05/22/2020).

[69] William J. Burns. *Common Password List (rockyou.txt)*. URL: https://www.kaggle.com/wjburns/common-password-list-rockyoutxt (visited on 05/22/2020).

[70] Korelogic Security. *Korelogic rules*. URL: https://contest-2010.korelogic.com/rules.html (visited on 05/22/2020).

[71] Cloudflare. *What is a Denial-of-Service (DoS) Attack?* URL: https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/ (visited on 05/22/2020).

[72] National Cybersecurity and Communications Integration Center. *DDos Quick Guide*. Tech. rep. National Cybersecurity and Communications Integration Center, Jan. 2014. URL: https://www.us-cert.gov/sites/default/files/publications/DDoS%5C%20Quick%5C%20Guide.pdf (visited on 05/22/2020).

[73] Johan Arnör. "Domain-Driven Security's take on Denial-of-Service (DoS) Attacks". School of Computer Science and Communication, KTH Royal Institute of Technology, June 2016, pp. 7–9.

[74] SolarWinds MSP. *Buffer Overflow Vulnerabilities and Prevention*. 2019. URL: https://www.solarwindsmsp.com/blog/buffer-overflow-vulnerabilities-protection (visited on 05/22/2020).

[75] Catalin Tudose. *TCP Handshake*. [Online; accessed May 22, 2020]. URL: https://www.luxoft-training.com/news/building-java-client-server-applications-with-tcp/.

[76] Cloudflare. *SYN flood attack*. URL: https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/ (visited on 05/22/2020).

[77] Cloudflare. *Slowloris DDoS Attack*. URL: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/ (visited on 05/22/2020).

[78] Cloudflare. *HTTP Flood Attack*. URL: https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/ (visited on 05/22/2020).

[79] Daniel Tomescu. *XML Based Attacks*. Tech. rep. OWASP, p. 12. URL: https://owasp.org/www-pdf-archive/XML_Based_Attacks_-_OWASP.pdf (visited on 05/22/2020).

[80] MDN web docs. *Connection*. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Connection (visited on 05/22/2020).

[81] Kali Tools. *hping3 Package Description*. URL: https://tools.kali.org/information-gathering/hping3 (visited on 05/22/2020).

[82] Amazon. *Transcend 2 GB SD Flash Memory Card (TS2GSDC)*. [Online; accessed May 22, 2020]. 2020. URL: https://images-na.ssl-images-amazon.com/images/I/81J6dAN1D1L._AC_SL1500_.jpg.

[83] Kali Tools. *Binwalk Package Description*. URL: https://tools.kali.org/forensics/binwalk (visited on 05/22/2020).

[84] TechTarget. *CRC File Format*. URL: https://whatis.techtarget.com/fileformat/CRC-Cyclical-Redundancy-Check-checksum-file (visited on 05/22/2020).

[85] Siemens AG. *Siemens Vulnerability Handling and Disclosure Process*. 2020. URL: https://new.siemens.com/global/en/products/services/cert/vulnerability-process.html (visited on 05/26/2020).

[86] Siemens AG. *Siemens Vulnerability Handling and Disclosure Process*. [Online; accessed May 26, 2020]. URL: https://new.siemens.com/global/en/products/services/cert/vulnerability-process.html.

[87] Cloudflare. *NTP Amplification DDoS Attack*. 2020. URL: https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/ (visited on 05/20/2020).

Appendices

Appendix A

Denial of service attack written in Python

```
import time, requests
from timeit import default_timer as timer
from multiprocessing.dummy import Pool

start = timer()
sessions = []
amount = 150
url = 'https://10.96.104.98/'
responses = []
headers = {'Content-Type': 'application/xml'}
for x in range(amount):
    sessions.append(requests.Session())

def worker6(payload):
  for x in range(amount):
    responses.append(payload.get(url, verify=False))
pool = Pool(2)
results = pool.map(worker6, sessions)
pool.close()
pool.join()
```

Appendix B

Custom password list

sicam
cmic
sicweb
web
test
hej
hejsan
qwerty
mamma
kalle
gnaget
niklas
kungen
vtnfl
vttnfl
lol
rtu
upgrid
demo
up
grid
html
web
webb
horizon
siemens

admin
abc
abcdef
0123456
123
vatten
fall
xyz
guest
789
456789
password
user
1925
ostra
bjalke
bol
logon
device

TRITA -EECS-EX-2020:859