



AALBORG UNIVERSITY

STUDENT REPORT

Security Analysis of IP-cameras

AALBORG UNIVERSITY

Group DS101F21



AALBORG UNIVERSITY
STUDENT REPORT

Software 10: "Security

Analysis of IP-cameras"

Group DS101F21

May 28, 2021

Semester:

10. semester

Project theme:

Security

Project duration:

1/02/2021 - 28/05/2021

ECTS:

30

Supervisor:

René Rydhof Hansen

rrh@cs.aau.dk

Danny Bøgsted Poulsen

dannybpoulsen@cs.aau.dk

Authors:

Thomas Lundsgaard Kammersgaard

tlha16@student.aau.dk

Thue Vestergaard Iversen

tivers16@student.aau.dk

Abstract:

This project is about a security analysis of IP-cameras and developing a solution to improve the security through automated firmware updates. We found it implausible to perform automated testing on firmware images to detect known vulnerabilities through the analysis.

The solution developed through this project is an automated update framework that relies on the device manufacturers to develop and maintain firmware updates for the devices. Through said updates, the overall security of their devices would increase, but only if the end-user applies the updates. We propose a solution that can automate and streamline the update process on IP-cameras from multiple manufactures.

The overall conclusion of this project is that the current approach to firmware development, distribution, and consumption are all contributing factors to the security of IP-cameras in general. The solution developed in this project automates the update process for IP-cameras produced by Hikvision and Herospeed. This is done by selecting the correct firmware and making sensible choices such as calculating checksum matches, performing backups and reapplying user configurations.

Number of pages: 84

Preface

This report was written by two Software students from group DS101F21 at Aalborg University during their 10th semester in Software Engineering. The project was supervised by [René Rydhof Hansen](#) and [Danny Bøgsted Poulsen](#).

- *Group DS101F21*

Resumé

The increased usage of IP-cameras has resulted in more insecure installations of said IP-cameras. While the installation itself might be secure, the device manufacturers and the hacking community usually can find software bugs that would pose a security threat if exploited correctly. Such threats are often fixable in two significant ways; firmware updates and configuration settings.

Firmware is the term used for all software running on embedded devices, such as IP cameras. These software packages come in many different formats and are often different between manufacturers. Some manufacturers rely on compression and encryption to obfuscate their code and make it harder for attackers to reverse engineer and find vulnerabilities. This obfuscation, however, also makes it harder to validate the content and evaluate the security of the update. Forcing trust on the device manufacturers to make high-quality updates with security in mind.

While most manufacturers have their way of packaging updates, most rely on a similar workflow to apply the updates. Most IP-cameras provide a webpage that is used for all interaction with the IP-camera. These webpages provide the ability to watch the IP-camera stream, operate the pan/tilt/zoom functionality, change the general configuration and update the firmware. This webpage, however, often relies on old, poorly supported software stacks, which requires the installation of browser plugins and other similar solutions. These solutions have often been required for years, but as web browsers have continued, many of the old plugins have become obsolete and outright unsupported. Those unsupported and obsolete plugins make it both a security risk to use them while posing as an unnecessary barrier to the end-user, resulting in devices that are left misconfigured and outdated.

Through this project, we propose a tool that does not care about the device manufacturer and device model while applying the firmware updates provided by the manufacturers. The tool aims to streamline the update process to keep devices up-to-date while providing a sane update policy aiming to keep the users existing configurations between updates. Since each manufacture has its own firmware distribution method, the collection of firmware is complex. Currently, only the Herospeed component can automatically download the latest firmware. In contrast, the Hikvision relies on a folder containing all firmware images, which should be periodically updated when new firmware is released. The solution can create backups, apply firmware updates and restore user settings from the backup.

The project has solved a subset of IP cameras from Hikvision and Herospeed, where firmware updates are supported through an easily extendable framework, allowing

for supporting more device manufacturers. The tool has primarily been tested on Hikvision IP-cameras due to availability and is both deemed useful and stable. The project has several aspects which would increase the usability and capability by providing a graphical user interface and the ability to check the configurations of the IP-cameras. The ability to check user configurations would be beneficial to increasing the security of the devices by either notifying the user about insecure settings or suggesting changes that would increase the security or performance of the product.

Contents

Contents	4
1 Introduction	6
1.1 Initial Problem Statement	6
2 Security	7
2.1 Context	7
2.2 Definition of Security	10
2.3 Security in IP-cameras	11
2.4 Privacy	13
3 Problem Analysis	15
3.1 IP-cameras	15
3.2 Common IP-camera setup	18
3.3 STRIDE analysis	20
3.4 Best Practices	22
3.5 Virtual LAN	23
3.6 Web Components	24
3.7 Universal Plug and Play	25
3.8 Firmware	27
4 Firmware analysis	29
4.1 Hikvision firmware	29
4.2 Herospeed	35
5 Problem Definition	46
5.1 Revised Problem Statement	46
6 Tools	47
6.1 Hikpack	47
6.2 Metasploit Framework	48
6.3 Network Mapper (NMap)	50
7 Design	51
7.1 Project Goal	51
7.2 Components	51
7.3 Usages	52
8 Implementation	54
8.1 Autodiscovery	54

8.2	Interface	59
8.3	Automatic updater	61
9	Related Work	74
9.1	Hikvision - TFTP Automatic Update Tool	74
9.2	Herospeed - Search Tool	74
9.3	Viakoo - IP-camera Firmware Update Manager	75
10	Evaluation	76
10.1	Solution	76
11	Discussion	78
11.1	The Solution To The Problem	78
12	Conclusion	80
	Bibliography	81

Introduction

1

This project's goal was sparked by curiosity about how IP-cameras could become a threat to the cybersecurity of both private citizens and the corporate industry. Based on initial readings on the subject, both from online news outlets and research papers, it seemed plausible that there might be some open problems within the topic. Based on botnets such as Mirai, which consisted primarily of IP-cameras, which could attack and bring down some substantial businesses[1] our main focus would be on IP-cameras. Should such devices provide a security threat to cybersecurity, we want to evaluate why this might be the case and develop a technical solution to secure such devices.

1.1 Initial Problem Statement

Based on the initial thoughts of the problem, the following initial problem statement was created and used as the basis for further analysis:

How does the use of IP-cameras in both private and industrial installations pose a threat to cybersecurity, and how can the security of said IP-cameras be strengthened?

With this initial problem statement we will define what security is and research how IP-cameras are affected.

Security 2

In this chapter, we will describe what security is and why it is important to define. Because if security is not clearly defined it is impossible to analyse. In this section, the security context will also be defined to help further the analysis. The concept of security will also be described.

2.1 Context

The security context is defined into four categories of access to an IoT device. This is done because confining access to these four categories allows us to focus on them individually as they are uniquely defined.

The four types of access we are working with are the following:

- Physical access to the device
- Visual access to the device
- Local access
- Remote access

Physical access to the device will allow for pressing buttons, disassembly, tampering, installing malicious hardware, and/or breaking the device. This also allows for an attacker to interact with the device using Wi-Fi, Bluetooth, Ethernet, Infrared and most types of radio communication.

Visual access to the device is defined as the device is visible, and the information on any displays can be read, and LED's or other indicators are visible. However, there is no physical access to the device, meaning that the device is placed at a location that is not possible to physically getting to. An example of this could be an IP-camera mounted to the ceiling in a room where it is placed in the ceiling out of reach.

With visual access the device is within radio range meaning technologies such as Bluetooth, Wi-Fi and Infrared is available.

Local access is if the device is placed on the same network as an attacker. Local access can happen if the network the device is placed on is not properly secured.

Remote access is if the device is placed on a network that is accessible through the internet.

In the next part, we will analyse how severe the different access contexts are, how hard they are to get access to, and how easy an attack would be in the given situations.

2.1.1 Different access types

The analysis is a combination of how much effort is required to get access and how severe of an impact such access could have. This is then split into different colours where **Red** is a serious threats, **yellow** being a medium threats and **green** being low- or non-threats (see Figure 2.1).

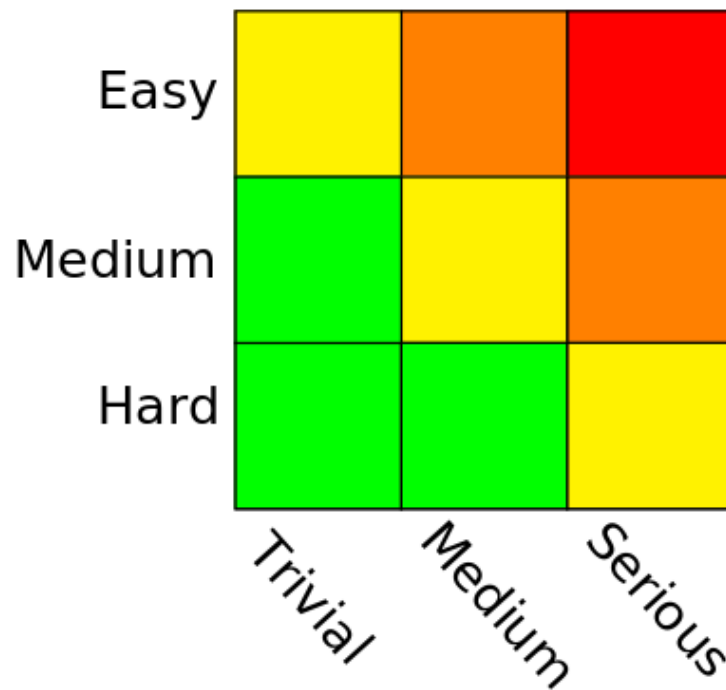


Figure 2.1. Security fields and their colours representing attack severity types.

Physical access to the device is considered a hard task to get, and very serious, which would place it in the green part of Figure 2.1. Such access would be considered a critical security problem if it is possible as it could allow an attacker to disassemble, destroy or steal the device. Physical access to the device could also allow an attacker to connect a malicious device that could function as MITM. An attacker would also be able to press any buttons on the device and connect to any available input ports on the device such as Network ports, USB ports, debug ports. It would also allow an attacker to cut any cables such as network, alarm or power connections to the device. Physical access would also gives the same options as Visual access. Physical access is assumed hard to get, but depends on how the device is located.

Visual access to the device is considered a medium task and could be of medium concern. This is because the device is visible, but an attacker cannot touch the

device, but could possibly interact with it using Wi-Fi, Bluetooth or Infrared. An attacker can also get visual feedback from any LED or displays showing various information while also being able to listen for any audio feedback from the device. This allows an attacker to get some feedback from their attempt at attacking the device which might be valuable. Such information could be an LED indicating that the device is rebooting or is turned off, produces an audible alarm or beeps error codes. All this gives an attacker some indications of what the device is doing. However, there are no guarantees that such attacks are possible, depending on what device is being attacked and what its capabilities are, it might not even have Bluetooth or Wi-Fi, making attacks even harder. This task could also be considered easy to get, depending on what type the device is. A type of device where visual access is considered an easy task is IP-cameras as their function is to record an area, which requires the device to be visible. Sometimes IP-cameras are placed in obscure locations to hide their location because it can be valuable for the IP-camera that people do not know where it is recording. However, the opposite can also be the case, where the IP-camera is placed visibly to clearly state that the area is being recorded. It all depends on where the device is located and how hard it is to get close to, and what the intention of the device is.

Local access is when an attacker is connected to the same network as the device. This means that there is no firewall blocking connections to the device, and only the device's own security is protecting it. Here any security vulnerabilities in the software can be utilised to attack a device. The difficulty of this type of access often varies as in some cases an insecure Wi-Fi password could potentially give this amount access. So it is hard to put place its difficulty on Figure 2.1 but the consequences are serious. Since most security vulnerabilities are published online, an attacker can just look up the device and find any known Common Vulnerabilities and Exposures (CVE) and use these to try to gain access to the device. Depending on the setup this is often considered a hard task to get with serious consequences.

Remote access as the device is considered an easy to medium task but with low to medium concern. This is because that if the device is exposed to the internet using a port forward, it can be easily reached but without it, it is often much harder. In some cases only some ports to a device is exposed limiting the surfaces of said device that can be interacted with to the ports that are forwarded. Remote access can become a concern if a device is using weak credentials that then could allow an attacker to breach a device which then can be used to attack other devices on the network. If a device is exposed to the internet, any device connected to the internet would be able to use known exploits to gain access to the device.

From these finding, serveral security contexts were found interesting subjects for further analysis.

2.2 Definition of Security

In this section, a definition of the system and what security is will be outlined.

2.2.1 System

A system is defined as the operational structure of the entire setup. The system is affected by inputs, the process of these inputs and some output. The input can be a sensor or the press of a button, power, or messages over a network. The output can be an LED turning on/off or blinking, or a message over a network, an alarm to a control system like SCADA or CCTV.

A system can have various tasks it should be able to perform given different inputs. An example of a system could be a sprinkler system, which should deploy water if there is a fire. The input of the system could be a smoke detector or a thermometer to monitor the temperature around the sprinkler, but another input is also the water to spread. The output of the system is if it should deploy water or not. The system's task is then to consider the input and decide what the output should be.

2.2.2 Security for a System

Security is defined "The ability of a system to satisfy its goals in the presence of an adversary". This means that security is achieved if the system can satisfy its goals even in the presence of an attacker. However, in our project security is dependent on the context, how is an attacker placed in relation to the system. This is important to keep in mind because what an attacker can observe or affect is dependent on how much access is available or obtainable.

Security is considered compromised when one or more properties of Confidentiality, Integrity and Availability (CIA) are compromised. However, in some systems, not all three parts of CIA can be considered critical to the system. More often than not only one or two of the properties of CIA are important to be upheld. An example of this could be a bank account. It is important that the system is kept confidential to ensure that personal information is not leaked, but it is also important to keep integrity to ensure that bank accounts cannot be tampered with to give a bank account additional funds. However, it is much more important to keep the bank account confidential, and the integrity than it is available. Put into context if a bank account is unavailable a transaction can be postponed and performed at a later time, but when confidentiality or integrity is breached it can be impossible to restore it.

So what integrity of the system is, is dependent on what the tasks of the system are, and which part of the system is considered critical.

If we consider the example of the sprinkler system, there are several inputs an attacker could affect to change the output of the system. The attacker could put

a lighter to the temperature sensor making the system think the temperature in the room is much higher than expected and make the system deploy water. The attacker could also cut the water supply to the sprinkler, making it ineffective in the event of an actual fire, since it would not have any water to deploy.

To protect the system from attacks, there could be taken measures such as adding multiple temperature sensors, which allows the system to have fault tolerance, meaning an attacker would have to affect multiple sensors to deploy the water.

2.3 Security in IP-cameras

There are different typical setups for IP-cameras, but some of the most common setups are shown below in Figure 2.2 The figure shows features such as Wi-Fi, Ethernet and GSM connectivity which is not present in all but only some models of IP-cameras.

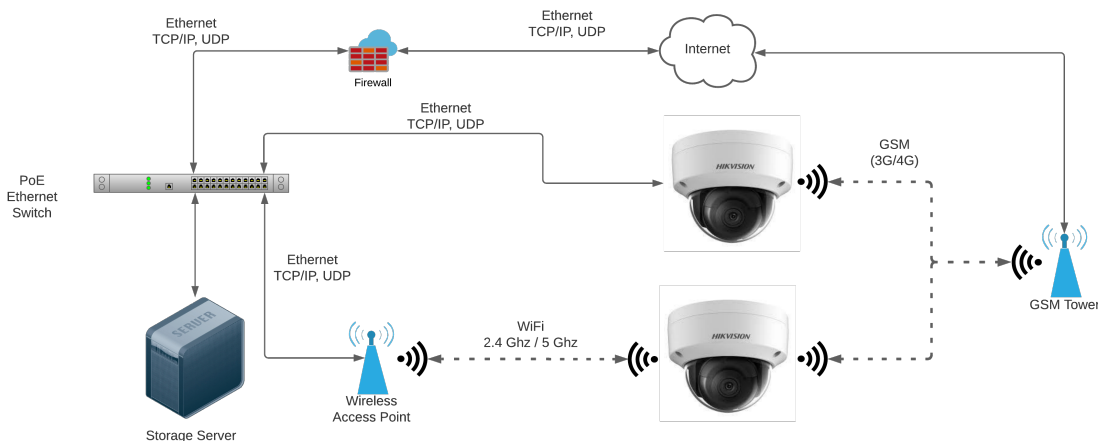


Figure 2.2. Network diagram showing common IP-camera setups.

It has to be in a visible location as it is an IP-camera, meaning that attacks from a visible view are difficult to protect against in software. There are attacks such as using infrared light to blast the IP-camera and make the image distorted. The rest of the attacks that can be done within the visual range are to gain local access to the IP-camera. There is also no way for software to prevent an IP-camera placed in a position where there is physical access to it to be tampered with. The software can detect tampering and trigger an alarm, but this is not a preventative measure but reactive measures. So in terms of context, the security contexts to look at is Local access and Remote access. This means that from the diagram in Figure 2.2 it would look like this in the presence of a local and a remote attacker as shown on Figure 2.3.

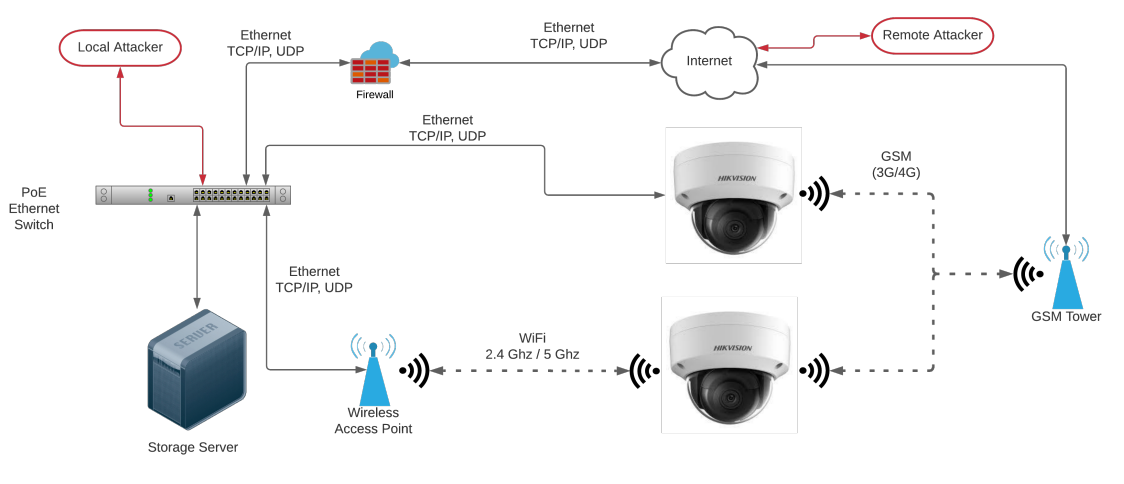


Figure 2.3. Network Diagram with attackers

The difference between a local attacker, and a remote attacker is that the remote attacker will have first to find a way to bypass the firewall. This can be done in several ways, but sometimes the devices are port forwarded to be reachable beyond the firewall. This could allow an attacker to reach any local device and its services. Another way for a remote attacker to bypass the firewall is to gain access to a port forwarded to beyond the firewall, which has a vulnerability that, when exploited, can allow access to the local network, becoming a local attacker. Compared to the local attacker, who would scan the local network and find devices without being limited by a firewall. Local attackers could be limited to scanning devices in the same VLAN as they are assigned, but this requires that the local network is set up using VLANs which is described in section 3.5.

If an attacker gains local access to devices, the devices are left with only the security implemented in the firmware. This means that if there is a vulnerability in the firmware, the device could be compromised, breaking the CIA principles(subsection 2.2.2).

If a IP-camera has a disclosed vulnerability, the manufacturers will often attempt to fix this in a firmware update that the end-user then uses to upgrade the IP-camera. This process is often manual and is the end-users responsible for keeping their devices updated as few devices can update themselves.

From a software perspective, the only way to improve security is by updating the firmware. However, it is often hard to inform the end-users of firmware updates as it often relies on the end-users finding the information themselves. This is because there is no good way for manufacturers to directly notify users of new firmware other than posting about it on their website.

2.4 Privacy

This section will look at the different consequences and risks of having a vulnerable IP-camera within the home.

2.4.1 Sensitive information

In today's digital age, people are more and more connected, and they share private information about themselves on social media. However, the users of social media are posting the information themselves. In contrast, a vulnerable IP-camera that is publicly available could give an attacker insights into its owners' everyday lives.

On the website [Shodan.io](https://shodan.io) we found an image that shows a naked man in a chair having a phone call. While the image itself might not be very incriminating, an attacker could use the image as blackmail.

2.4.2 Theft

While personal information might be harmful if leaked, even basic information may cause harm for the owner. While some information might not seem harmful, all data can be a valuable asset depending on the motives of an attacker. On [Shodan.io](https://shodan.io) we found multiple images from insecure IP-cameras that show a driveway in which a car is parked and with the house's entrance visible on the right. On another image we found another driveway, where both the driveway of the IP-camera owner, their neighbour, and the house across the street can be seen.

This information might not seem harmful nor essential, but an attacker with the intention of breaking and entering could benefit from such information. With the IP-cameras' information, an attacker can know their victims' schedule; they can see if their car is in the driveway and know if they are home. They can monitor the IP-camera during the theft and know if anyone is coming home and leave through alternative access before being caught.

2.4.3 Espionage

While IP-cameras are popular among private citizens, they have also been used in the corporate sector for many years. Most companies should have and use the resources to protect such IP-cameras. However, data leaks still accrue. Most recently, the breach of video feeds from 150,000 surveillance IP-cameras from inside hospitals, companies, police departments, prisons, and schools was leaked.[2] Such an attack could potentially violate police officers' safety, inmates' privacy in the prisons and schools and potentially breach the integrity of companies who might have a IP-camera at a position where sensitive information is discussed or disclosed.

On [Shodan.io](https://shodan.io) we also found an IP-camera installed in an office consisting of two workplaces looking to be executive. With access to such an IP-camera, fitted with

sound, an attacker could eavesdrop on the office and possibly intercept sensitive information.

With a definition of security and a clear definition of the context, we can analyse how security is impacted within the problem domain.

Problem Analysis 3

In this chapter, we will perform an analysis of IP-cameras from a security perspective. We will focus on how IP-cameras, in general, can be attacked and how to prevent such attacks from being successful. After the initial analysis, we focus on two brands of IP-cameras, namely Hikvision and Herospeed.

3.1 IP-cameras

An IP-camera is a camera that can provide its video feed over a network connection. IP-cameras have different connectivity that varies from model to model and brand to brand. There are pros and cons to using different connectivity, and as a result, it allows for different attacks. Below is a list of the connectors we have seen on IP-cameras

- Wi-Fi
- Ethernet
- Proprietary connector

Often a proprietary connector will be a combination of some form of network communication and power. In some cases, IP-cameras use proprietary connectors to force their product line from an NVR to the IP-cameras. These connectors are often unique and would require much more researching to understand. So for this project, we will focus on Ethernet and Wi-Fi connectivity.

Using Wi-Fi connectivity is used when using a cable is not possible. Wi-Fi is often not the primary choice because ethernet allows for faster and more reliable connectivity. Ethernet can also be used to provide power to the IP-camera using Power over Ethernet (PoE). Wi-Fi is also vulnerable to low-skill Wi-Fi jammers available on eBay and Amazon for less than 100\$. These jammers would prevent the IP-camera from uploading footage and would prevent anyone from viewing live footage.

3.1.1 Power

Powering the IP-cameras also varies from model to model and brand to brand. But there are three primary ways of supplying the IP-cameras with power.

- Power over Ethernet (PoE)

- Power brick
- Battery

Power over Ethernet (PoE) is the most common way of supplying power to IP-cameras as power and data can be transferred over the same cable. A few top-level IP-camera models that use PoE also have an onboard battery backup if there is a power outage. However, this is often a minimal supply that will only run the IP-camera for a short period. PoE has the problem that often, all IP-cameras are powered by the same PoE switch, meaning if it fails, so does all the IP-cameras.

Using a power brick is done when an IP-camera communicates over Wi-Fi because running ethernet cables is impossible, but power is. This is a good alternative as it provides a good source of power but allows a single IP-camera to fail because of its power supply failing. If a power brick fails, it can be harder to diagnose since the IP-camera would appear offline, which means it could just have lost its Wi-Fi connection.

There is a third option which is to power the IP-camera using battery power. This is often an option when the IP-camera is placed where no power or ethernet can be connected. The problem with relying on battery power is that the batteries have to be charged. This requires a person to manually take the IP-camera down, change or charge the battery and reinstall it. When talking about a permanent solution, this is often avoided because it requires much maintenance.

Battery-powered IP-cameras are sometimes used in residential houses where running cables are complex and expensive. Some IP-cameras use induction wireless charging, which is very easy for the end-user but takes the IP-camera offline for an extended period as charging using induction is often slow. The other option is simply having a charge station for batteries and then just swapping batteries once they are drained.

Batteries are often the "easy" option but often come with maintenance requiring much attention. If the batteries are not changed regularly, it can result in IP-cameras powering off, which is a problem for obvious reasons.

3.1.2 Storage

Storing IP-camera images and footage is often a problem and requires different solutions depending on where the IP-cameras are located and how they are used. Some IP-cameras record all the time, which results in a lot of footage that requires a lot of storage. In some cases, IP-cameras are set up to only capture when it detects motion, which significantly reduces the amount of footage captured. Some IP-cameras, that record all the time, use a low frame rate to reduce the amount of storage required. Some IP-cameras can be setup to only record in a specific timeframe or only on certain days of the week to decrease the amount of footage.

Depending on the IP-cameras' setup, different storage solutions are needed. Some

of the most popular storage solutions are listed below.

- Local storage
- Network Video Recorder (NVR)
- Storage server
- Cloud storage

Local storage is when an IP-camera has onboard storage, soldered on onboard flash memory, or an SD card. Some IP-cameras have no connectivity and rely only on using SD cards to store footage. This is a practice used a lot in trail IP-cameras used to capture pictures and video of animals.

Local storage is often used as a backup if the connection to the storage server or NVR is lost.

Network Video recorder is a device capable of pulling footage from an IP-camera using Real-Time Streaming Protocol (RTSP) to watch, record and save the footage live from the IP-camera. Depending on the NVR used, it sometimes can detect movement and only save footage with motion in them. An NVR is often expensive because they need to have powerful enough hardware to process large amounts of footage in real-time to keep up with the IP-camera footage. Older NVRs record to VHS, DVD or Tape storage, which is expensive, so it is rarely used in newer systems.

A storage server is a server that provides a network accessible storage such as NFS or Samba share that the IP-cameras can mount and write their footage to. This is a simple solution that offloads handling the footage to the IP-camera's processor, and the server provides the storage. However, this requires that the IP-cameras are powerful enough to support this feature as it requires the IP-cameras to have enough power to offload footage.

Wi-Fi connection can sometimes be poor and not have enough bandwidth to offload footage in real-time, but this depends on the footage's resolution and frame rate.

Cloud storage is the last option and is often a cheap solution that does not have a high start-up cost but works on a subscription basic. The IP-camera footage is uploaded from the IP-camera or an NVR to the cloud storage. A problem with cloud storage is that IP-cameras have to be connected to the internet. As stated in section 3.4 having IP-cameras connected to the internet is often a bad idea for a variety of reason.

When using cloud storage, investigating how the IP-camera's footage gets to cloud storage is important. Some questions you should ask yourself are:

- How is the footage transferred? Is it Encrypted?
- What kind of authentication is used, if any?
- How is the footage protected when it is stored in the cloud?

Transferring the footage encrypted to cloud storage is important because unencrypted data should be treated as public information. This is why it is crucial to ensure that the implementation of offloading the footage is done properly. It can be tough to know if it is implemented correctly since most providers do not disclose their implementation or source code.

Another thing to keep in mind is how the IP-camera authenticates itself to cloud storage. Is the authentication safe and encrypted, or is it sent across the internet in plaintext? If the IP-camera authenticates using credentials in plaintext, it could allow for MITM attacks, giving an attacker the option to "sniff" the credentials from the network traffic.

Some cloud providers advertise how the footage is stored and how it is secured. Others advertise the total capacity they give and not how it is stored. It is essential to know that the footage is stored encrypted so that if a hacker gains access to the storage, the footage cannot be tampered with.

So, all in all, IP-cameras vary a lot in connectivity and features such as battery backup. For this project, we will focus on IP-cameras using standard connectivity in the form of Ethernet and Wi-Fi. We do not have any IP-cameras with battery backup, so this is neither something that we will do any more research on.

3.2 Common IP-camera setup

When describing typical IP-camera setups, it is essential to distinguish between the context of the setup. Unlike the context described in section 2.1, this context is based on the business perspective where most configuration variations are present.

- **Home Surveillance**
- **Small Business Surveillance**
- **Medium/Large Corporate or Public Sector Surveillance**

IP-cameras provides different interfaces that cater to different users as described in section 3.1. Interfaces such as hardware alarms are rarely utilised by home users and small businesses, whereas large corporations are likely to integrate such interfaces into their existing alarm systems.

3.2.1 Home Surveillance

Most IP-cameras marketed to home users have many of the same features and interfaces as the ones used by large corporations and the public sector. Most home users have a single IP-camera installed focusing on a specific point of interest in the home, such as the driveway, main entrance, or living room.

Such setups primarily consist of the IP-camera connected to the home network via Wi-Fi. This uplink is most commonly to set up and control the IP-camera, while

it can also connect the homeowner's smartphone to the IP-camera to view the IP-camera's live stream. To save images, when the homeowner is not at home, most IP-cameras support installing an SD memory card where the footage can be stored. Such setups do not necessarily save any footage but provide some amount of mental safety for its owners.

3.2.2 Small/Medium Business Surveillance

Like home surveillance, small businesses usually deploy a minimal amount of IP-cameras without centralised control. Small businesses might differentiate from home surveillance because the IP-cameras often are placed outside of buildings monitoring entrances and different areas of interest.

Whenever an ethernet cable or jack is exposed, especially outside the user's control outside of buildings, network separation is even more key since all ports could be compromised. Different techniques, such as device authorisation and network separation, should be implemented but are likely not.

In some businesses where an IP-camera is located in an inaccessible location, wired networking might not be practical. Such IP-cameras will often operate with an SD card containing the footage, only retrieved once an incident has happened. Should a burglar be aware of the IP-camera, nothing stops them from simply removing the IP-camera's SD card. Once the SD card is removed, no footage from that IP-camera can be retrieved, often making it a poor choice.

Alternatively, such IP-cameras use a Wi-Fi connection to the business to save the footage onto a network drive or allow a user to watch a live stream from the IP-camera. Such setups protect the footage from physical destruction and theft while making it easier for them to store the footage elsewhere. The same rules apply to network separation and device authorisation on the network when connecting using Wi-Fi, like using ethernet. However, as with ethernet connected devices, such separation and authorisation are likely not to be present.

3.2.3 Large Corporate or Public Sector Surveillance

Unlike small and medium business, large corporate and public sector surveillance commonly consists of a large number of devices that are implausible to manage individually. Large corporations often have multiple sites to manage their surveillance, all with varying connectivity and accessibility. Likewise, the public sector uses surveillance in public areas, official buildings, and different interest points.

Large corporations, such as security companies like Verisure[3] and G4S[4] provide alarm and surveillance systems to home users, have different locations based on their customers' locations which may vary from downtown in a large city to more rural areas outside the main cities. Remote locations, such as pump stations for

utility companies which are often located outside cities, are essential for the utility companies to provide their services. Such locations may or may not have active video surveillance, but if they do, they require an internet connection to a centralised control centre, where if an alarm is triggered, a patrol can dispatch.

Large corporations can scale the number of devices to a vast number by centralising multiple IP-cameras' control. Such controls could be to watch the live feed from each IP-camera, adjust the IP-camera using the pan, tilt, and zoom functionality, enable and disable alarms, configure network connectivity, all without the knowledge of the people on the other side of the IP-camera. However, by centralised control, such centres become high-value targets for attackers, who could, instead of breaching thousands of IP-cameras individually, gain access to the same amount by breaching the central control centre. As such, security is paramount to keep the confidentiality of their customers safe.

At these large scales, the use of VLAN separation becomes more paramount since all devices attached to such a network are without reach for the administrators, and the users or people walking by can tamper with the IP-camera or network connection. Here the ability of the IP-camera to interact with a traditional alarm system is keen. The IP-cameras used should have alarm pins available, which allows the IP-camera to send alarm and heartbeats to the centre, making it easier to detect failures and false positives. There the IP-camera might experience a blackout, but without being tampered with by an attacker.

With the basis of these common setups, the analysis can focus on the different possible attack surfaces that apply to IP-cameras. Due to the group having access to a limited setup of IP-cameras, we can only emulate a home setup. For the remainder of this project, we will focus on a setup like that used for home surveillance as described in subsection 3.2.1.

3.3 STRIDE analysis

We want to perform a STRIDE analysis because it gives a good insight into an IP-camera's different attack surfaces and what could potentially be targeted in further analysis.

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

If an attacker gains access to an IP-camera, it is possible to replace the Real Time Streaming Protocol (RTSP) server so that instead of the live video feed, it is

replaying a video file instead, spoofing the footage. Such an attack would require shell execution on the device to be performed. It could also be considered tampering as an attacker is technically tampering with the footage.

Some IP-cameras do not have firmware signing, giving an attacker the option to modify the firmware and upload it to an IP-camera. If the IP-camera cannot dispute that the firmware is not from an authorised entity, running malicious firmware could have backdoors and other malicious stuff included, causing the IP-camera not fulfilling repudiation but allowing unsigned custom code to execute.

A denial of service attack can be performed in many ways. Still, the goal is always to result in an authenticated user being unable to connect and use the product as intended. One way is if the IP-camera is exposed to the internet, a package flooding attack known as a Denial of Service (DoS) or Distributed Denial of Service attack if multiple external connected devices are flooding an IP-camera. As the IP-camera has to process all the packages it is receiving; it will spend most if not all of its processing power on handling the network traffic, and will not have the processing power to allow an authenticated user to access the device as intended.

Suppose an attacker gains access to the IP-camera either by gaining access to controlling the device either from a web panel that does not have authentication or because of weak credentials. In that case, they could turn the live feed off, power off the device or change the password disallowing a previously authenticated user to be locked out. If the IP-camera has a motor to change where it is pointed, an attacker can change the orientation of the IP-camera to an angle where it is useless and not surveying the needed area. Some IP-cameras with a motor can run in "fan" mode where they are sweeping left to right. If an attacker gains access to the web interface, it could allow them to disable the "fan" mode, effectively severely limiting the effective surveillance area.

Another more straightforward denial of service attack could be to make the IP-camera stop working, which can be achieved by cutting the IP-camera's connection or just destroying it with a hammer. However, this sometimes gives an observer enough time to identify who is tampering with the IP-camera and often allows for action against an attacker. In contrast, it can be hard or impossible to find the people behind the attacks in other cases. This is why it is essential to make sure the wires connected to an IP-camera is not visible nor easily accessible. Most IP-cameras have options for the wires to be hidden behind the mount, which means getting access to the wires much harder.

Another attack that could destroy an IP-camera without it being visible on the IP-camera could be using a mobile electromagnetic pulse (EMP) device. EMP devices send out electromagnetic pulses that can interfere with electronics by causing bit flips. This often results in electronics rebooting or burning the circuit boards because electricity is flowing places, and at voltages, it should not. EMP devices can be made relatively simple or be bought for under 100\$ on eBay and Amazon.

Some IP-cameras can record at night or in low-light environments using infrared light as an invisible light that can not be seen with the human eye but can be seen with IP-camera sensors.

Information disclosure is often an issue with cheaper chinese IP-cameras that "phone home" sending information back to a server. There have been several cases where IP-cameras have sent the credentials in clear text back to the manufacture [5].

Elevation of privilege can happen in many ways. The most common ways are that a user account can access something that has a vulnerability but is overlooked. It already required authentication but gave regular users access to features that only admins should have. Elevation of privileges can also happen if a user somehow can factory reset the device to reset the administrator password, giving a user full access to the IP-camera.

If a vulnerability in a web interface is found, regardless if it is seen as an admin or a user could allow an attacker to gain shell access. With shell access, the device can be used for a wide variety of stuff, such as part of a botnet, mining cryptocurrency, or performing more attacks on the IP-camera's local network.

With a clearer definition of the attack surfaces, it is easier to identify countermeasures to mitigate them. There are simple measures that should be implemented, while some of which may have more significant ramifications for the administrator.

3.4 Best Practices

From the STRIDE analysis, described in section 3.3, we have created a list of the best practices that should be easy to follow and will make IP-cameras more secure.

- Make sure the IP-cameras are connected with a cable to increase stability, connectivity and protect against Wi-Fi jammers
- Ensure that the cables connected to an IP-camera are hidden or protected
- Make sure the IP-cameras are in an isolated network (VLAN)
- Make sure the IP-cameras do not have access to the internet
- Make sure not to use a default password and change it regularly
- Ensure that the IP-cameras are running the latest firmware and check for firmware updates on a regular basis (at least every 3 months)

These best practices pose some problems, especially if the IP-camera does not have an internet connection. Without an internet connection, the IP-cameras cannot offload their footage to online cloud storage, which might not be a problem if local network storage is used. Another problem to keep in mind when the IP-cameras do not have an internet connection is that synchronisation protocols like Network Time Protocol (NTP) are used for the IP-camera's time and date and requires access to an NTP server. Without NTP the IP-camera could come out of synchronisation, and

the date and time stamp on the IP-camera feed could be out of sync. This means that the recordings' timestamp will be different from the actual time the recordings occurred, making it much harder to find a specific event at a specific time. Another problem with time is that most IP-cameras can only record during specific time periods, resulting in the IP-camera not filming at the expected time.

There are options to solve these problems, and one of them is to have a local NTP server that can be synced to an external NTP server or have a locally accepted clock. This will allow the IP-cameras to synchronise the time without an internet connection.

3.5 Virtual LAN

A Virtual LAN (VLAN) is any broadcast domain that is partitioned and isolated within a computer network at the Data Link Layer (OSI Layer 2[6]). VLAN's are used to separate a network into two or more isolated networks, which is critical for separating different types of network-attached devices.[7]

A typical home network, seen on Figure 3.1, could consist of a single network, where all devices are accessible. Typical home network devices could be Printers, Smartphones, SmartTV's, and Personal Computers. On such networks, the number of devices is usually stable. New devices are mainly connected when guests are visiting and bring their own devices.

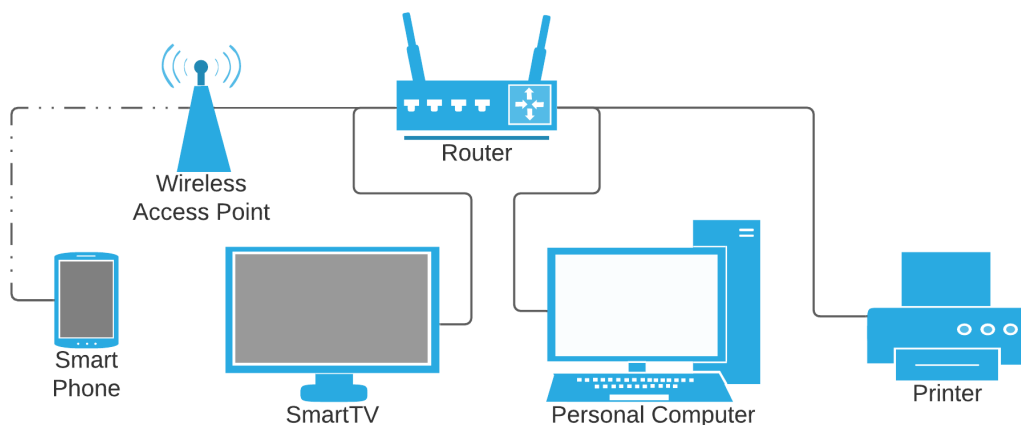


Figure 3.1. Network diagram with one single network with all devices.

From a security perspective, these devices should be isolated since most home users do not require their smartphone to connect to their Personal Computer, nor the other way around, and a single compromised device would not be able to compromise other devices further.

Some devices, however, are not very useful if not available to other devices. Printers and personal computers should communicate, and smartphones and smart TV's should also, while the smart TV and the printer make little sense. Such a network is shown on Figure 3.2.

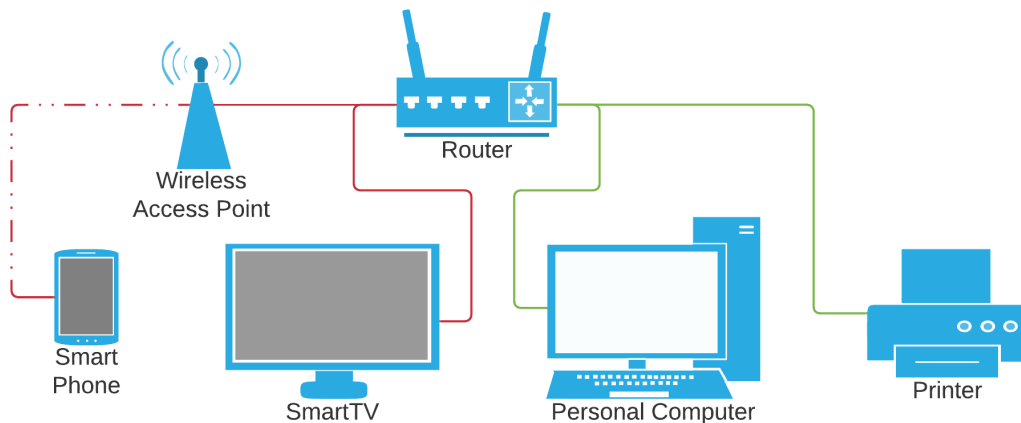


Figure 3.2. Network diagram with two networks for each category of devices. Different colours represent different VLAN's.

By introducing a new category of devices, IP-cameras, a new VLAN only containing said IP-cameras and possibly a Network Video Recorder (NVR) should be created. Said network should be separated from the rest of the devices and networks to contain the security risks in the case of a breach. This can be done using a VLAN to isolate them from other devices.

3.6 Web Components

This section will analyse web components used in IP-cameras within their web interfaces to allow additional functionality such as in-browser live video playback, firmware upgrading, and more.

3.6.1 ActiveX components

ActiveX is a software framework developed by Microsoft that has only been officially supported in Internet Explorer (IE) and is set to be discontinued along with IE 11 on 17 August 2021. Microsoft has shifted focus from IE to their new browser Microsoft Edge, based on the Chromium Engine by Google, which does not support ActiveX components. Microsoft presented ActiveX as an alternative to Java applets. The main selling point of ActiveX compared to Java was a better performance. ActiveX has the unfortunate downside of only being supported on Windows, meaning macOS and Unix cannot use ActiveX components, something that Java supports.

Because of the missing support on macOS and Linux, Java applets became much more popular. ActiveX also has a reputation of being insecure because of any exploitable vulnerabilities in the framework allowing an attacker to gain unrestricted access to the system as ActiveX components run within a web browser with full permissions.

As ActiveX is only supported in IE and as of February 2021, only 0.81% of the browsers used globally are IE compared to 63.59% using Google Chrome [8].

ActiveX was disliked from the start as it required becoming a "Microsoft developer", which cost a fee of 20\$/year for individuals and 400\$/year for corporations to allow for development. This was because, to have ActiveX components run in the IE browser, the developer would have to sign the code. Microsoft only distributed the certificates for signing code to trusted developers that signed contracts saying they would not develop malware. **ActiveX**

3.6.2 Netscape Plugin Application Programming Interface (NPAPI)

Netscape Plugin Application Programming Interface (NPAPI) is a programming interface initially intended to allow developers to develop plugins for web browsers that could be used with older Netscape Navigator browser. The first version of NPAPI was released in 1995 with Netscape Navigator 2.0. From Wikipedia: ("Because of the age of the API, security issues, and adoption of alternative technologies such as HTML5, software vendors began to phase out NPAPI support in 2013.")[9]. In May 2014, NPAPI was removed from Chrome 35 on Linux. In April 2015, NPAPI was disabled by default in Chrome for Windows and OS X with Chrome 42; however, in September 2015, it was removed entirely in Chrome 45. Firefox kept support until Firefox 85.0, which was released in January 2021. Until January 2021, Firefox had support for NPAPI, but it was disabled by default.

As of writing (April 2021), only Internet Explorer still supports NPAPI, where it is enabled by default.

There are browsers such as Waterfox, a fork of Firefox, that focus on supporting old discontinued technologies such as NPAPI [10], which could be used in the future to use pages with outdated web plugins.

3.7 Universal Plug and Play

Universal Plug and Play (UPnP) is a set of network protocols used by devices to discover other devices connected to the same network. UPnP works by sending a package using UDP to port "1900" using HTTP multicast (HTTPMU). UPnP has no security and therefore assumes that locally connected devices on the network can be trusted.

UPnP is used from a device to all other network-connected devices to inform of its capabilities. An example of this could be an IP-camera that informs the network which services it has, such as web interfaces, video streaming etc.

Most routers support UPnP in some way or another. The functionality supported and how UPnP is handled will vary as there is no standard for UPnP. Some routers handle the UPnP requests by allowing NAT traversal using the Internet Gateway Device Protocol IGD protocol, allowing devices to get their external WAN address, see port mapping and request port forwards. Adding a port mapping, a UPnP controller behind the IGD can allow an external device access to an internal client. This can be a problem if it is unmonitored; a device could expose its services to the internet that should be kept private under normal circumstances, even without its owner's knowledge.

Examples of this happening can be seen in section 2.4, where local IP-cameras can be accessed from the world wide web, but since the devices have weak or no credentials, allowing anyone with an internet connecting to see the local IP-camera.

However, UPnP also has its uses, allowing devices such as Google Chromecast, Amazon Alexa or Google Nest to be easily discovered by smartphones, allowing devices to cast audio and video. Printers can also use UPnP to make them easily discoverable and allowing for easier printing.[11]

One of the known vulnerabilities found in UPnP allows an attacker to mask packages as they appear to come from within the local network, while in fact, they originate from an external client. The vulnerability has been named CallStranger and can be used to make devices spam packages to a remote IP-address resulting in DDoS attacks. This vulnerability relates to the best practices defined in section 3.4 that if a device does not strictly need an internet connection, it should not have access to it. Open Connectivity Foundation (OCF) has published a fix for this, but since the vulnerability was found in 2020, most devices have yet to be updated to fix it. [12]

Since UPnP have no security, and the fact that devices can expose themselves to the internet makes the technology hard to recommend since the security risks and the negatives far outweigh the benefits. Newer versions of UPnP have attempted to resolve some of the security problems known to exists within the protocol. Most devices that support and use UPnP are not easily upgradeable, resulting in devices remaining vulnerable.[11]

Disabling UPnP is the best option to prevent it from being exploited, but since most people do not know what UPnP is, they also do not know how dangerous it can be. Since some UPnP exploits have been fixed and firmware updates have been made, all products that have not been upgraded are still exploitable, resulting in the firmware updates have little to no effect.

3.8 Firmware

Firmware is running on all IP-cameras, but it varies from brand to brand and product to product. It is common for IP-cameras to run ARM Linux since it is open-source and free to use. In this project, we will focus on IP-cameras running Linux as they are the most common, and it is what we find the most interesting to analyse.

3.8.1 Firmware compression

Firmware often comes packed in different archive formats and using different compressions. Compression can reduce the firmware's total size so that when the end-user has to download it, they use less data. The usage of data can be important for regions that have data caps or pay by MB.

Compression algorithms applied depends on what the manufacturer is using, but it is sometimes limited to a few compression algorithms because of IP-cameras' limited processing power. Based on our experience examining firmware, some of the most common compression algorithms used are the Gunzip (GZIP (.gz)) and Lempel–Ziv–Markov chain (LZMA (.xz)) algorithms.

The compression is combined with different archiving options such as Tarball (Tar) and Copy In, Copy Out (CPIO). Tarball is often known with the extension *".tar"* and is most commonly used with LZMA and Gunzip and is recognisable with the extensions *".tar.xz"* and *".tar.gz"*.

CPIO and Tar are the most commonly used because they do not require much processing power to handle. Both are supported by the official archiver "cpio" included in most Linux distributions on most platforms because it is often used when handling packages within the Linux system.

In some cases, the compression depends on the processing power of the destination device. However, sometimes compression is not an option because the device does not have enough processing power to extract files within a reasonable timeframe. If the device takes several hours to extract the firmware, it will result in the end-user not bothering to update the devices since they do not want to spend the time it takes.

Manufacturers sometimes ship firmware that has to be extracted before it is uploaded to the device circumvent this. This reduces the amount downloaded from the internet and utilises the more powerful processors found in laptops and desktops to perform the decompression.

3.8.2 Firmware Encryption

Some manufactures want to hide how their firmware works, relying on the old saying "security through obscurity".

Hikvision has used a series of XOR instructions on the entire firmware to obfuscate it from being read. However, reverse engineers have found out how this all works and created tools such as Hikpack (See section 6.1) that can reverse the XOR instructions to allow other tools such as Binwalk to examine and extract the archived files.

Other manufactures such as Herospeed use no encryption and simply ships a firmware archive that is neither compressed nor obfuscated in any way. The Herospeed firmware is extractable by binwalk without any problems and any issues.

3.8.3 Updating Firmware

Updating the firmware on IP-cameras is crucial since it is the only way for the manufacturer to fix vulnerabilities or implement new functionality after delivering it to the end-users. However, it is not an automated process with most devices and requires the end-user to do it manually.

Updating the firmware on IP-cameras requires the end-user to identify which firmware version it is currently running on the device and search the manufactures' website for a newer firmware if it exists. Most manufacturers do not announce when new firmware becomes available, and it is up to the end-user to find this information. Some manufactures have newsletters that contain information about if new firmware is available. However, these emails often land in the spam folder and are rarely read by any actual users.

Suppose an end-user has found a newer firmware than running on the device. In that case, the user has to use the sometimes complicated interfaces running on the device to find where firmware is uploaded to the device to update the firmware. In some cases, the interfaces require additional tools to be installed, such as ActiveX components(See subsection 3.6.1) or Custom NPAPI plugins (See subsection 3.6.2).

As described in subsection 3.6.1 and subsection 3.6.2, ActiveX and NPAPI are dying technologies, which means that devices using these older technologies will have some if not all their functionality made unusable soon.

Firmware analysis 4

This chapter will focus on two manufacturers Hikvision and Herospeed, and how they deliver their firmware updates. It will analyse how the firmware is packed and how it can be extracted and what research we have performed.

4.1 Hikvision firmware

Hikvision ships their firmware in ZIP compressed files, which has to be extracted by the end-user. The ZIP file only contains one file, which is named "digicap.dav". This file then has to be uploaded to the IP-camera to update the new firmware.

Analysing firmware from Hikvision IP-cameras yielded some valuable information about how the IP-cameras work.

The Hikvision IP-cameras run a custom ARM processor with extra hardware encoding capabilities for H264, H264+ and H265. Hardware encoding is critical to the IP-cameras to process the raw footage into a digital format that can be transferred over the network to a playback or recording device.

The processor running on the IP-camera is a custom ARM processor but runs a standard ARM Linux installation with some kernel modules to support the custom processor's unique capabilities. These kernel modules could be of interest as they could potentially have vulnerabilities that could be exploitable.

Below is a list of the kernel modules we have found. Some of these modules are pretty obvious of what they do as "*PTmotor.ko*" which is for the Pan/Tilt motor connected to the processor. A vulnerability in this kernel module could allow an unauthorised user to change where the IP-camera is facing, making the IP-camera less effective.

Another obvious module is the "*alarm.ko*". Most Hikvision IP-cameras have some additional pins used to connect an alarm cable used with CCTV systems to trigger system alarms. A vulnerability in this module could allow an unauthorised user to disable alarms. A picture and the alarm wire can be seen in Figure 4.1.

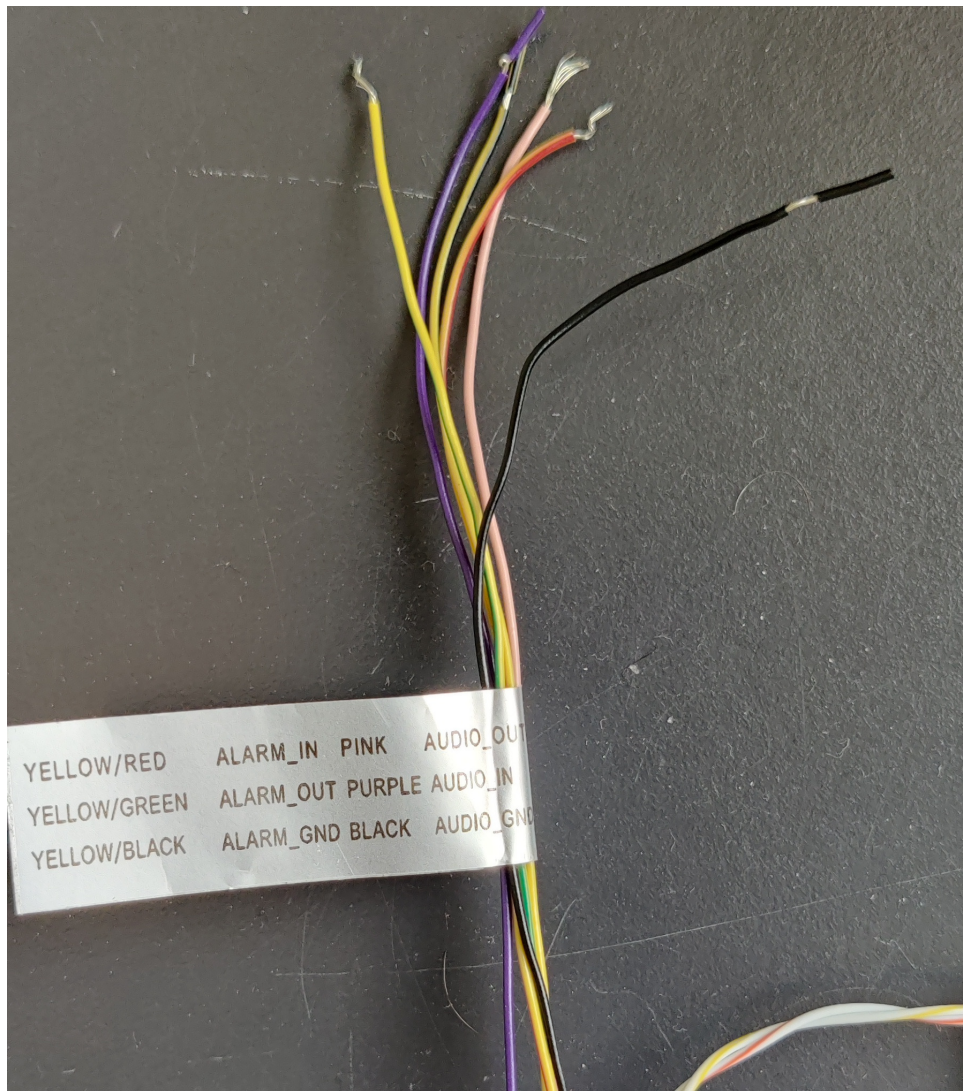


Figure 4.1. Hikvision alarm wires.

However, meaning that the kernel modules are vulnerable does not mean they are exploitable, so a vulnerability is not worth much without an exploit. Nevertheless, finding a vulnerability is still a problem because a vulnerability is just waiting to be exploited. So it is often essential to fix a vulnerability before an exploit is discovered.


```
1 8188eu.ko
2 PTmotor.ko
3 abf_trigger.ko
4 alarm.ko
5 an41908.ko
6 ap_pwr.ko
7 base.ko
8 bcm43362
9 bcm4356
10 bcmdhd_sdio.ko
11 bcmdhd_sdio_bcm4356.ko
12 bcmdhd_sdio_cy.ko
13 cdc7005.ko
14 cfg80211.ko
15 cy43362
16 detect_sensor.ko
17 dome_timer.ko
18 eth_debug.ko
19 event_notify.ko
20 hi_sensor_i2c.ko
21 hi_ssbi.ko
22 hiir.ko
23 hikcomm.ko
24 hitimer.ko
25 ipc_stm8.ko
26 mmz.ko
27 modules.dep
28 motor.ko
29 option.ko
30 sii9024.ko
31 smart_config.ko
32 spiflash.ko
33 uart232.ko
34 usb_wan.ko
35 usbserial.ko
```

Figure 4.2. Kernel modules from Hikvision R7 firmware.

4.1.1 Root password

Further examination of the firmware we found a passwd file that contains a single root user.

```
1 root:ToCOv8qxP13qs:0:0:root:/root:/bin/psh
```

Figure 4.3. Contents of /etc/passwd file found in the firmware.

The hashed password "ToCOv8qxP13qs" was quickly found by doing an online search for it and corresponds to the password "hiklinux". This means that if an attacker would be able to get SSH access to the device, the password for root is known beforehand. From our experience, the password is identical across all firmware versions we have examined.

4.1.2 SSH

The IP-camera does not appear to have port 22, which is commonly used for SSH, open on the IP-camera, so SSH is not an option. The Hikvision firmware indicates that it contains a Dropbear SSH server [13], which is a lightweight alternative to OpenSSH, commonly used in low powered IoT devices. The device has an SSH server but no obvious port to connect using SSH to the IP-camera. A Nmap scanning of the IP-camera shows that port 80, 554 and 8000 is open.

```
1      Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-07 12:36 CEST
2      Nmap scan report for 192.168.1.10
3      The host is up (0.0081s latency).
4      Not shown: 997 closed ports
5      PORT      STATE SERVICE
6      80/tcp    open  http
7      554/tcp    open  rtsp
8      8000/tcp   open  http-alt
10     Nmap done: 1 IP address (1 host up) scanned in 0.27 seconds
```

Figure 4.4. Nmap scanning of a Hikvision IP-camera.

Port 80 is the web interface (see section 4.1.3). Port 554 is used for Real-Time Streaming Protocol (RTSP), which requires authentication using username and password. The web interface and RTSP share the same username and password. Port 8000 is listed in Hikvision documentation as a "Service port", and some further research indicates this is used for Digital Video Recorders (DVR) and Network Video Recorders (NVR).

Port Name	Default Value	Configurable in Local	Port Property	Comment
Service Port	8000	Yes	TCP	
SDK UDP Live View Port	62000~65000	No	UDP	Multiplex with RTSP Server
HTTP Port	80	Yes	TCP	
HTTPS Port	443	Yes	TCP	
RTSP Server Listen Port	554/10554	Yes	TCP	
RTSP Server UDP Port	62000~65000	No	UDP	Multiplex with SDK UDP live view
Ezviz Command Port	9010	No	TCP	
Ezviz Live View Port	9020	No	TCP	
IPC Alarm Upload Port	30960~(30960+n-1)	No	TCP	
IPC Access Port	50000~52400	No	UDP	
SNMP Port	161	Yes	UDP	
DDNS Port	6555 7555 8245	No	UDP	
Multicast Port	62000~65000	No	UDP	

Yellow: The most commonly used ports, which need port forwarding to access DVR/NVR remotely.

Figure 4.5. Screenshot of Hikvision documentation showing what the different ports are used for.

4.1.3 Updating Firmware

There are multiple ways of updating the firmware running on Hikvision devices; the most common is using the web interface. There are also some other less known ways of injecting firmware updates, such as using Curl to push the firmware file through the IP-cameras API, but there is also the option to use TFTP to force the IP-camera to update.

TFTP works by an IP-camera being in factory mode where the IP-camera is hardcoded to pull a firmware from 192.168.0.128 using TFTP. This means that if an attacker somehow could factory reset the IP-camera, the IP-camera would revert any configuration while also forgetting anything about being in an isolated subnet and assume a static IP instead. Since there is no authentication on the TFTP server, the IP-camera will install whatever firmware is provided. However, resetting the IP-camera to factory defaults requires access to the web interface or physical access to the IP-camera. With physical access to the IP-camera, it has to be disassembled, and the reset button has to be held down for 10 seconds which is a rather long and cumbersome task.

We found some indication that the Hikvision IP-cameras does not validate firmware provided over TFTP, meaning there is a possibility of uploading modified firmware. However, we could not confirm this since we could not get TFTP to work in our setup.

Web interface

The Hikvision web interface is secured by a login page that requires both username and password, which to our knowledge, is not bypassable. Using the web interface does not require any additional components, but does require the user to use Internet Explorer or the error message shown on Figure 4.7 is shown. This message can be easily circumvented by just changing the browser's user-agent, making the web page think it is Internet Explorer 11, and the login page is shown instead. However some functionality in the web interface such as live-video playback, firmware upgrading, uploading/downloading of configurations require the "webcomponents.exe" to be installed which is only possible on Windows. The "webcomponents.exe" installs an ActiveX component (subsection 3.6.1) that only Internet Explorer can use.

Without using Internet Explorer and having the proprietary ActiveX components installed, it is impossible to use the "browse" buttons to select a firmware file. (Browse buttons are Marked red on Figure 4.6) Non-responsive buttons in the web interface mean that end-users cannot update the firmware using the web interface.

If the plugin is not installed, the message marked with a blue circle in Figure 4.6 is shown on all pages that require it within the web interface.

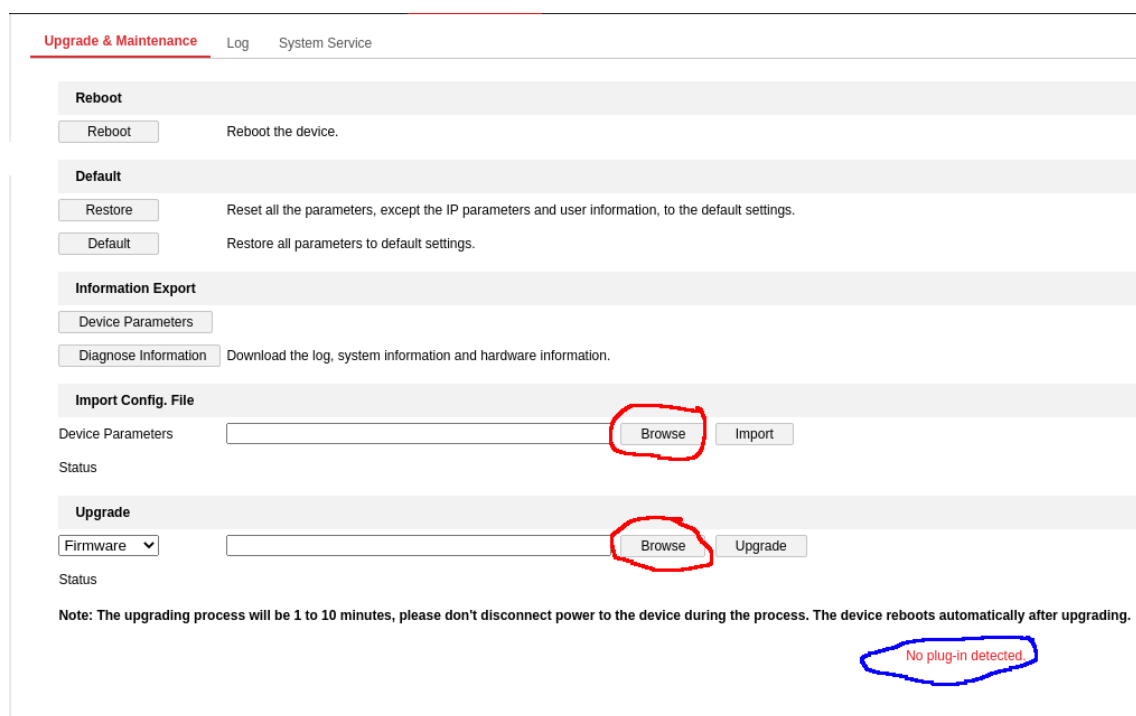


Figure 4.6. Hikvision web interface showing the Maintenance page.

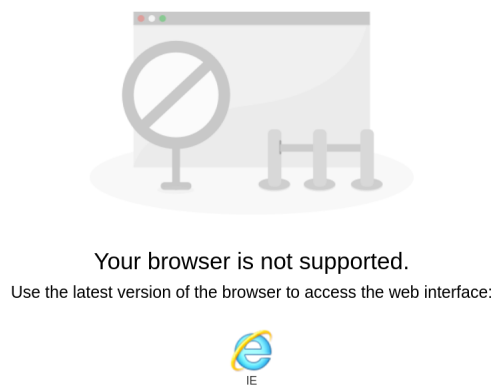


Figure 4.7. Hikvision web interface showing that the browser is not Internet Explorer.

Curl

Curl can be used to upload firmware to the IP-cameras. Updating the firmware using Curl utilises the Web API called "ISAPI" available on the IP-camera through the address "http://xxx.xxx.xxx.xxx/ISAPI/...." The API requires authentication using basic auth, which is the same as the username password combination used to login to the web interface. The API has an endpoint used to update firmware, described in the Hikvision "ISAPI" documentation. Using Curl to perform a PUT operation with a firmware file will result in the IP-camera trying to use that firmware to update. The command used to do this is shown in Figure 4.8.

```
1 curl -X PUT --digest -T digicap.dav -u *username*:password* ↵  
   http://XXX.XXX.XXX.XXX/ISAPI/System/updateFirmware
```

Figure 4.8. Curl command used to upload a firmware to a Hikvision IP-camera.

This allows an authenticated user with some knowledge of Curl to upgrade the firmware regardless of the operating system without installing the ActiveX plugin.

4.2 Herospeed

Herospeed is a software company that makes firmware for IP-cameras. The company makes firmware for their line of IP-cameras, but also third parties. Herospeed describes itself as a "Surveillance And Security Solution" focusing on IoT, Digital audio/video compression, storage and decompression.

Analysing the firmware from Herospeed gave an insight into how their devices work. Compared to Hikvision (section 4.1), Herospeed firmware is much simpler and does not feature any encryption or obfuscation.

4.2.1 Firmware analysis

Herospeed firmware is shipped in ZIP files, which the user has to extract before uploading the firmware through the web interface. Extracting the ZIP file results in a single file, an uboot image with the file extension ".fls". This file examined in binwalk looks like this:

	DECIMAL	HEXADECIMAL	DESCRIPTION
1			
2			
3	24	0x18	Unix path: /var/cfg/passwd
4	1816	0x718	Unix path: /var/cfg/Device.ini
5	2328	0x918	Unix path: /var/cfg/config_3M.ini
6	2584	0xA18	Unix path: /var/cfg/DevTypeEnableLdrTf.ini
7	346190	0x5484E	Executable script, shebang: "/bin/sh"
8	346261	0x54895	Unix path: /opt/app/bin
9	346511	0x5498F	Unix path: /opt/app/bin
10	346539	0x549AB	Executable script, shebang: "/bin/sh"
11	346582	0x549D6	Unix path: ↵
	/usr/local/lib:/usr/lib:/opt/app/lib"		
12	346642	0x54A12	Unix path: /var/cfg/update)
13	346703	0x54A4F	Unix path: /var/cfg/update)
14	346775	0x54A97	Unix path: /var/cfg/update)
15	346868	0x54AF4	Unix path: /var/cfg/update)
16	346932	0x54B34	Unix path: /var/bin/SystemDaemon.bak" ↵
]; then		
17	347017	0x54B89	Unix path: /var/bin/SystemDaemon
18	347046	0x54BA6	Unix path: /var/bin/SystemDaemon.bak ↵
	/var/bin/SystemDaemon		
19	347108	0x54BE4	Unix path: /var/bin/SystemDaemon
20	347143	0x54C07	Unix path: ↵
	/var/bin/LongseMiniServer.bak"]; then		
21	347236	0x54C64	Unix path: /var/bin/LongseMiniServer
22	347269	0x54C85	Unix path: ↵
	/var/bin/LongseMiniServer.bak /var/bin/LongseMiniServer		
23	347339	0x54CCB	Unix path: /var/bin/LongseMiniServer
24	347377	0x54CF1	Unix path: /opt/app/ko/exfat.ko
25	347644	0x54DFC	Unix path: /var/cfg/mountapp ; then
26	347676	0x54E1C	Unix path: /var/cfg/mountapp
27	347863	0x54ED7	Unix path: /opt/app/ko
28	348014	0x54F6E	Unix path: /var/cfg/startapp &
29	348091	0x54FBB	Unix path: /var/cfg/Ver.ini
30	348271	0x5506F	Unix path: /var/cfg/Ver.ini
31	348296	0x55088	Unix path: /var/bin/LongseMiniServer &
32	348340	0x550B4	Unix path: /var/bin/SystemDaemon &
33	361631	0x5849F	ELF, 32-bit LSB executable, ARM, ↵
	version 1 (SYSV)		
34	401907	0x621F3	Unix path: /var/cfg/update
35	402027	0x6226B	Unix path: /var/cfg/Ver.ini
36	403755	0x6292B	Unix path: /var/cfg/update
37	406031	0x6320F	HTML document header
38	406202	0x632BA	HTML document footer
39	411219	0x64653	Unix path: /opt/app/bin/AVServer &
40	412867	0x64CC3	ELF, 32-bit LSB executable, ARM, ↵
	version 1 (SYSV)		
41	491959	0x781B7	Unix path: /var/cfg/CustomTools.cfg
42	493583	0x7880F	Unix path: /var/cfg/DevTypeEnableLdrTf.ini
43	496503	0x79377	Unix path: /var/cfg/CustomTools.cfg
44	505863	0x7B807	Unix path: /var/cfg/Ver.ini
45	506783	0x7BB9F	Unix path: /var/cfg/Device.ini
46	515507	0x7DDB3	Squashfs filesystem, little endian, ↵
	version 4.0, compression:xz, size: 6956372 bytes, 482 inodes, ↵		
	blocksize: 65536 bytes, created: 2019-06-25 01:14:40		
47	7492947	0x725553	gzip compressed data, has original file ↵
	name: "u-boot.bin", from Unix, last modified: 2019-06-24 07:13:22		

Figure 4.9. Binwalk run on Herospeed firmware file.

Disassembling the firmware is easy and can be done entirely using Binwalk.

```
1 binwalk -e firmware_uboot.flr
```

Figure 4.10. Binwalk command to extract everything from the firmware

When extracting the firmware folder with the hexadecimal name is created with the folder that can be extracted. Here the Squashfs Filesystem and the gzip compressed u-boot.bin are valuable to look at since these are the largest files in the firmware. The Squashfs looks like a Linux filesystem, but there are missing a Linux kernel. However, it contains the files for the web interface and many kernel modules used to control hardware-specific components. The u-boot.bin contains the Linux kernel, and we speculate that part of the firmware upgrade process is first installing the Linux kernel followed by extracting the squashfs into the filesystem.

4.2.2 Root password

When analysing the firmware file using a HEX editor at address "110D" we found something that looks like a passwd file. The contents of the passwd file can be seen in Figure 4.11.

```
1 root:RnkKmz.XHnH1s:0:0::/root:/bin/sh
2 root:5mTFqDgojIl56:0:0::/root:/bin/sh
```

Figure 4.11. Contents of a passwd file found in the Herospeed firmware.

It is unknown which of the root passwords is the correct one since it is a little ambiguous from just the HEX editor. The format of the "passwd" file indicates that there is not used any specific hashing algorithm, so it is assumed that it is Data Encryption Standard (DES) [14]. Searching for the password hashes on Google did not yield any results. In an attempt to get the password, we will use the tool called "John the Ripper" (JtR), a password cracking tool that uses a dictionary and tries to crack passwords in a passwd file. JtR automatically detects which hashing algorithm is used and uses that hashing algorithm to crack the password. The dictionary we used is downloaded from crackstation.net [15] and consists of a list of passwords totalling 15 GiB in size. We ran JtR on the passwd file using the crackstation.txt file, which took just over 4 minutes to calculate all the hashes.


```
1 > john passwd --wordlist=crackstation.txt
2 Using default input encoding: UTF-8
3 Loaded 2 password hashes with 2 different salts (decrypt, ←
   traditional crypt(3) [DES 128/128 AVX])
4 Will run 8 OpenMP threads
5 Press 'q' or Ctrl-C to abort, almost any other key for status
6 0g 0:00:04:06 DONE (2021-04-07 14:41) 0g/s 3512Kp/s 7024Kc/s ←
   7024KC/s ??????????..?????????
7 Session completed
8 > john --show passwd
9 0 password hashes cracked, 2 left
```

Figure 4.12. Running JtR on the passwd file.

As seen in Figure 4.12 none of the two passwords was cracked. This means that the root password is not something that has been cracked and is not easily crackable, even with a huge dictionary. This indicates that Herospeed uses a good password. It only took 4 minutes to calculate hashes of 15 GiB of words because the hashing algorithm used in DES 128/128 AVX can be calculated quickly on modern hardware. So even though it was run on a quad-core laptop CPU, it was relatively fast.

It also has the option to run in incremental mode, which is a brute-force approach. Since the hashing used is the DES-128, the max length of the password is eight characters, but it may still contain any combination of characters, numbers and symbols. Cracking the password using the incremental brute force method is expected to take around ten years using an Nvidia GTX 1070. This can be seen on Figure 4.13

```

1  >sudo john --incremental passwd --format=descrypt-opencl ↵
2  --progress-every=3
3  Device 1@XPS: GeForce GTX 1070
4  Using default input encoding: UTF-8
5  Loaded 2 password hashes with 2 different salts (descrypt-opencl, ↵
6  traditional crypt(3) [DES OpenCL])
7  Warning: MaxLen = 13 is too large for the current hash type, ↵
8  reduced to 8
9  Press Ctrl-C to abort, or send SIGUSR1 to john process for status
10 0g 0:00:00:04 0g/s 0p/s 7216Kc/s 7216KC/s Dev#1:54C 123456..123456
11 0g 0:00:00:07 0g/s 9294Kp/s 27884Kc/s 27884KC/s Dev#1:54C ↵
12 51ruz..51ruz
13 0g 0:00:00:10 0.00% (ETA: 2037-02-04 09:27) 0g/s 13223Kp/s ↵
14 39670Kc/s 39670KC/s Dev#1:55C limbrook..limbrook
15 0g 0:00:00:13 0.00% (ETA: 2034-12-26 11:47) 0g/s 14891Kp/s ↵
47154Kc/s 47154KC/s Dev#1:55C asugabow..asugabow
0g 0:00:00:16 0.00% (ETA: 2033-12-06 00:56) 0g/s 16428Kp/s ↵
51337Kc/s 51337KC/s Dev#1:55C 8log09..8log09
0g 0:00:00:19 0.00% (ETA: 2033-04-18 19:26) 0g/s 17421Kp/s ↵
55749Kc/s 55749KC/s Dev#1:56C clpil1!..clpil1!
0g 0:00:00:22 0.00% (ETA: 2031-12-25 10:28) 0g/s 19309Kp/s ↵
57929Kc/s 57929KC/s Dev#1:56C 29bo5*..29bo5*
0g 0:00:00:25 0.00% (ETA: 2031-10-27 03:16) 0g/s 19870Kp/s ↵
59611Kc/s 59611KC/s Dev#1:56C juvyol12..juvyol12
0g 0:00:00:28 0.00% (ETA: 2031-09-11 19:56) 0g/s 19903Kp/s ↵
60880Kc/s 60880KC/s Dev#1:57C sc8uxx..sc8uxx

```

Figure 4.13. Running JtR with a GPU.

As stated by JtR, it is expected to finish in 2031, which means it is unlikely that the password will be cracked without some serious computing power and much time. We will not spend more time trying to crack the password as we wanted to check if easy to crack, which it is not.

4.2.3 Web interface

The Herospeed web interface is powered by a BOA web server, an old, outdated, and discontinued web server [16]. The web interface requires a plugin to be installed on the computer to function. The plugin can be downloaded from the web interface of any Herospeed IP-camera. The web plugin is available for macOS and Windows. As we do not have a Mac to test the web plugin, we will focus on running it on a virtual Windows machine.

When the Web plugin have been installed the following files are located in "C:

Program Files (x86)

HsWebPlugin" Figure 4.14.

```
1 |-- HSAGRAccess.dll
2 |-- HSAVPlayer.dll
3 |-- LibThread.dll
4 |-- npLsNetClientCtl.dll
5 |-- unins000.dat
6 |-- unins000.exe
```

Figure 4.14. Installed files by the HSWebPlugin.exe in Windows.

It appears that the plugin is only supported in Internet Explorer. Upon further research and reading the help page of the IP-camera, it is found that the plugin uses Netscape Plugin Application Programming Interface (NPAPI). As stated in subsection 3.6.2, NPAPI has been supported in most browsers until 2013, when it was decided that it should be discontinued and that support should be dropped for security reasons.

Using the web interface on a Linux machine without the plugin installed shows the login screen. The login screen is useless on Linux and has no functionality. This is because the javascript used by the web interface can only check if the operating system is either Windows or Mac.

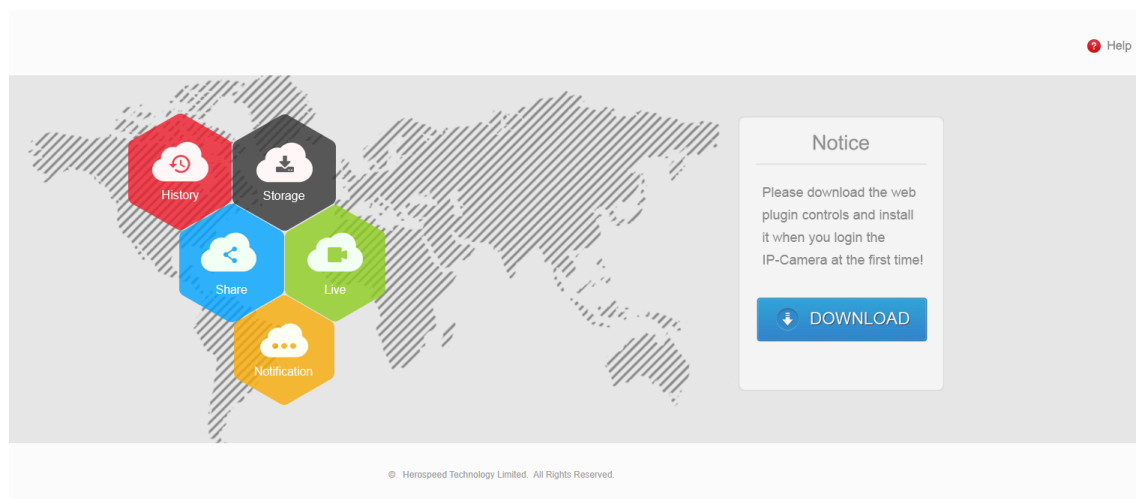


Figure 4.15. Web interface of a Herospeed IP-camera without the web plugin installed.

4.2.4 Web API

Herospeed IP-cameras are fitted with a hidden API that we found on a Polish FTP server (Figure 4.17) which contained some documentation about it, which was previously unknown to the group. The FTP server is open to access without the need for a username and password. On the FTP server, a file containing documentation of a Herospeed API and configuration endpoint on the IP-camera that is not visible in the firmware (see Figure 4.16).

```
1 http://xxx.xxx.xxx.xxx/ini.htm  
2 http://xxx.xxx.xxx.xxx/vb.htm
```

Figure 4.16. Hidden URL of Herospeed IP-cameras

The web interface pages are protected using basic auth with a username/password, which is identical to that of the administrator credentials on the IP-camera. When opening the ini endpoint, the content of an ini-file is shown. The page contains much information about the IP-camera model, firmware version, Linux version, IP-camera setup, IP-address and MAC address. The page can be accessed from any operating system and does not require the Herospeed Web Plugin to be installed.

The API is accessed by performing HTTP GET operations on the *"vb.htm"* endpoint. The API allows for rebooting the IP-camera, restoring it to factory settings, getting and setting the username/password and IP/port for the IP-camera's built-in FTP server, and getting/setting the username/password and IP/port of the SMTP server. All these API endpoints are described in the documentation that was downloaded from the FTP server.

We found no information in the documentation that the API on the *"vb.htm"* endpoint allows for updating the firmware of the devices.

```
1 ftp://kamery.delta.poznan.pl:21000/
```

Figure 4.17. The FTP server on which the documentation was found.

4.2.5 Live feed

When analysing the firmware, we found that there is a file in the root called *"snap.jpg"* that is a symlink to a file placed in the *"/tmp"* folder on the IP-camera. When trying to access the snap.jpg file, a snapshot from the IP-camera feed can be seen. When refreshing the page, the picture is updated to the latest.

Upon further analysis and searching for the *"snap.jpg"* within the root folder of the web interface, we found that it is referred to in another HTML file *"live.html"*. Accessing *"live.html"*, which is a simply html page with some javascript that keeps loading the *"snap.jpg"* every second. The entire content of the *"live.html"* can be seen in Figure 4.18.

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" ↔
   "http://www.w3.org/TR/html4/loose.dtd">
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5  <title>JPEG STREAM</title>
6  </head>
7  <body onload="init();">
8  

10 <script type="text/javascript">
11 var url=window.location.href;
12 url=url.split('\\');
13 var src='http://'+url[2]+'/'+'snap.jpg';

15 function init(){
16     var img=document.getElementById('showImg');
17     changeImg();
18 }

21 function changeImg(){
22     var img=document.getElementById('showImg');
23     var myDate = new Date();
24     var time=myDate.getTime();
25     img.setAttribute('src',src+'?'+time);
26     setTimeout('changeImg()',1000);
27 }
28 </script>
29 </body>
30 </html>

```

Figure 4.18. Content of live.html from Herospeed firmware.

Depending on the firmware version on the IP-camera, we have found that older firmware versions do not have any authentication on these files and can be accessed without authentication. However, updating the firmware to a newer version, this vulnerability has been fixed, and the files now require basic auth with administrator credentials.

4.2.6 Herospeed SDK

On the polish FTP server (Figure 4.17) we also found the source files and a manual for a Herospeed SDK that contains some header files that allow for compiling ones own software that can interact with the IP-camera.

Within the SDK, there are documented ways to upgrade the firmware using the SDK, which could allow us to write a plugin to upgrade IP-cameras' firmware. A snippet of the documentation describing how the *"HS_NET_Upgrade"* method with information about input and output can be seen on Figure 4.19.

```
1      Remarks: This API is used to device upgrades.  
2      (The progress and status messages via callback function )Parameters?  
3      [in] nLoginId  
4      User ID, the return value of HS_NET_Login  
5      [in] filename  
6      Url of file  
7      Return: Return 0 on success, -1 on failure.  
8      int HS_NET_Upgrade(int nLoginId, const char* filename);
```

Figure 4.19. The SDK documentation of the method for upgrading firmware.

The SDK has features that are not described in the API for the *"vb.htm"* endpoint. We do not know how the SDK communicates with the IP-camera, but we speculate that it utilises a hidden API that is undocumented.

4.2.7 Updating Firmware

To update the firmware running on a Herospeed IP-camera, the user needs to download the file from the software page, unpack the firmware and upload it through the Web interface. The device will then update itself. If the IP-camera runs a newer firmware, there is support for updating the firmware without the need for the end-user to download and upload the firmware. However, this has to be triggered by an end-user manually by logging into the web interface, navigating to the maintenance page. The IP-camera will then attempt to download the newest firmware from the Herospeed server.

In order to examine how the Herospeed IP-camera is provided new firmware files, we analysed packages sent to and from the IP-camera using Burp Suite. When uploading a firmware file, a post request is made to the *upgrade.html* endpoint. This file does not exist in the firmware, which got us thinking that this is just an alias for an endpoint in the Boa web server. Calling the endpoint from the browser gives a 404 page not found error. When doing a post from the software Postman using the same information as from the post shown in Figure 4.20 the IP-camera started updating, and a status 200 was returned.

The upgrade endpoint is not mentioned in the documentation, but we found references to it in the javascript files within the firmware. So we have an endpoint that is undocumented but appears to allow for updating the IP-camera.

```
1 POST /upgrade.html HTTP/1.1
2 Accept: */*
3 X-Requested-With: XMLHttpRequest
4 Content-Type: multipart/form-data; ↵
    boundary=-----7e52571480150
5 Referer: http://xxx.xxx.xxx.xxx:xxxx/views/main.html
6 Accept-Language: en-DK,en-US;q=0.7,en;q=0.3
7 Accept-Encoding: gzip, deflate
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; ↵
    rv:11.0) like Gecko
9 Content-Length: 10048380
10 Host: xxx.xxx.xxx.xxx:xxxx
11 Pragma: no-cache
12 Cookie: lang=en; userName=admin; pwd=admin; modifypsw=true; ↵
    updateTips=true
13 Connection: close
14 -----7e52571480150
15 Content-Disposition: form-data; name="file"; ↵
    filename="HEROSPEED_FIRMWARE.FLS"
16 Content-Type: application/octet-stream
17 **** FIRMWARE CONTENT ****
18 .....
```

Figure 4.20. The post made by the browser when uploading firmware.

With an analysis conducted against two different IP-camera manufacturers, the group has identified several problems, which, when solved, would improve the security of IP-cameras.

Problem Definition 5

This section presents a definition of the problem derived from the results of the problem analysis. The problem's essence is encapsulated in a problem statement defining what this project develops a solution for.

In the analysis (section 3.1), we analysed IP-cameras as the product it is, focusing on how an attacker would attack such a device. In the preliminary analysis of IP-cameras, we discovered many attacks, but more interesting is the attacks made possible by software vulnerabilities. Software attacks focus on attacking the IP-camera by exposed surfaces such as a web interface or an API.

The analysis focused on the firmware of IP-cameras as this is often where vulnerabilities and exploits are found, but it is also where the manufacture can patch and fix them.

Patching vulnerabilities in the software is often up to the manufacturer, but applying the new firmware relies on the end-user to manually keep themselves up to date with the latest firmware version and then manually install that firmware update to their devices. However, as described in subsection 3.8.3 updating the firmware sometimes requires using older deprecated and no longer supported technologies. This can make it impossible for generic end-user to secure their devices by updating the firmware.

As described in section 2.4, the consequences of having a vulnerable IP-camera could allow an attacker to gain knowledge that could be used for breaking and entering, theft, spying and blackmailing.

5.1 Revised Problem Statement

The problem is expressed in the following problem statement:

How can a software solution scan the local network for IP-cameras and inform the user of outdated devices, and perform firmware updates on the IP-cameras automatically?

With the revised problem statement, the project group identified a tangible problem that the project would aim to solve.

Tools 6

In this chapter, we will present some of the tools we have used in our work to evaluate Hikvision products and firmware. Tools which yielded no usable information are excluded.

6.1 Hikpack

Hikpack is a tool for packing and unpacking Hikvision firmware, and it is made by a user under the nickname "montecrypto" on the ipcamtalk.com forum the tool was published. Hikvision firmware is separated into different hardware platforms, and Hikpack supports the IP-cameras running on R0, R1, R6 and G0. Hikvision has a range of Network Video Recorders products that support recording and storing IP-camera footage. Some of these are also supported by Hikpack, namely the k41, k51.

The Hikpack tool is not open for public downloading and is only given when "montecrypto" is contacted directly. It was given to us through him on the condition that we would not redistribute it.

```
1 hikpack v2.5 Hikvision firmware packer/unpacker by montecrypto
2 *** No expressed or implied warranties of any kind. Use at your own risk ***
3 Usage:
4   hikpack -t <fwtype> -i <src_dav_file>                print dav file information
5   hikpack -t <fwtype> -x <src_dav_file> -o <dst_dir>    extract dav file into directory
6   hikpack [opts] -t <fwtype> -p <dst_dav_file> -o <src_dir> pack dav file from source ↵
7   directory
8   hikpack -t <fwtype> -d <src_crypted_file> -o <dst_file> decrypt file
9   hikpack -t <fwtype> -g <src_crypted_cfg> -o <dst_file> decrypt configuration backup ↵
10  file
11  hikpack -t <fwtype> -G <src_file> -o <rypted_cfg_file> encrypt configuration backup ↵
12  file (CRC adjusted if needed)
13  hikpack -t <fwtype> -e <src_file> -o <dst_crypted_file> encrypt file
14  -t option sets firmware platform type. Currently supported: IP-cameras: ↵
15  r0,r1,r6,g0 nvr: k41,k51
16  ----- The following options are used by the pack (-p) command:
17  -L <1,2>      set language id (1=EN, 2=CN)
18  -D <YYYYMMDD> set firmware date.
19  -V <ver>      set firmware version. Use hex number, e.g.: 0x05040003 for v5.4.3
```

Figure 6.1. Hikpack man page as shown when used in Linux.

The original intention of Hikpack was to modify the language of firmware as Chinese models of the Hikvision IP-camera will not accept an English version.

However, as Chinese IP-cameras are cheaper than English, many western people have bought inexpensive Hikvision IP-cameras from Chinese websites and flashed English firmware onto them. The IP-cameras cannot have their language changed by uploading an English firmware, but instead, the language is changed in the Chinese firmware. However, the tool can do much more than just changing the language of Hikvision firmware.

6.2 Metasploit Framework

Metasploit Framework is an offensive security framework developed and maintained by Rapid7 LLC. The framework is a part of the larger Metasploit project, a paid service, whereas the framework itself is free and open-source. The framework is included in the Kali Linux distribution and is widely known by cybersecurity personnel [17].

The Metasploit Framework contains several tools, which an attacker might find helpful during an attack. The standard tools are categorized into five categories; Auxillary, Encoders, Exploits, Nops, and Payloads. The framework serves as a utility between all these tools, allowing an attacker to have all the necessary tools available in one place. Currently, the framework contains 2106 exploits that take advantage of known vulnerabilities in many different software pieces. The exploits are ranked based on their effectiveness and are all searchable based on their efficacy, name, CVE and different keywords.

Once the user has decided on a target to attack and has found an exploit that might work on the target, the framework prompts the user for all required information to execute the exploit. Once all required options have been set, the framework can execute the exploit, and if the exploit was successful, a shell should have been spawned, giving an attacker control over the targeted system.

The framework integrates with other tools, such as Nmap (See section 6.3. If used correctly, the framework is a very potent tool. However, the learning curve for new users might be intimidating.

Although the user might not be proficient and experienced using the framework and techniques of the framework, all modules contain information and references, as shown in Figure 6.2, making it an excellent learning tool that gives the user all the necessary information.

If an attacker cannot find an exploit that might work, the attacker can try and execute all available exploits against the target and hope that one of them might work. Such an attack is known as a "Hail Mary", which might work but is very "loud/noisy" if a defender is monitoring the network.

```

1  msf6 exploit(windows/smb/ms17_010_eternalblue) > info
3      Name: MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
4      Module: exploit/windows/smb/ms17_010_eternalblue
5      Platform: Windows
6      Arch:
7      Privileged: Yes
8      Licence: Metasploit Framework Licence (BSD)
9      Rank: Average
10     Disclosed: 2017-03-14

12  Provided by:
13     Sean Dillon <sean.dillon@riskssense.com>
14     Dylan Davis <dylan.davis@riskssense.com>
15     Equation Group
16     Shadow Brokers
17     thelightcosine

19  Available targets:
20     Id  Name
21     --  ---
22     0   Windows 7 and Server 2008 R2 (x64) All Service Packs

24  Basic options:
25     Name          Current Setting Required Description
26     ----          -
27     RHOSTS         yes           The target host(s), range CIDR identifier, or ↔
28         hosts file with syntax 'file:<path>'
29     REPORT         445          yes           The target port (TCP)
30     subdomain      .            no           (Optional) The Windows domain to use for ↔
31         authentication
32     SMBPass        no           (Optional) The password for the specified username
33     SMBUser        no           (Optional) The username to authenticate as
34     VERIFY_ARCH    true        yes           Check if remote architecture matches exploit ↔
35         Target.
36     VERIFY_TARGET  true        yes           Check if remote OS matches exploit Target.

38  Description:
39     This module is a port of the Equation Group ETERNALBLUE exploit,
40     part of the FuzzBunch toolkit released by Shadow Brokers. There is a
41     buffer overflow memmove operation in Srv!SrvOs2FeaToNt. The size is
42     calculated in Srv!SrvOs2FeaListSizeToNt, with mathematical error
43     where a DWORD is subtracted into a WORD. The kernel pool is groomed
44     so that overflow is well laid-out to overwrite an SMBv1 buffer.
45     Actual RIP hijack is later completed in
46     srvnet!SrvNetWskReceiveComplete. This exploit, like the original may
47     not trigger 100% of the time, and should be run continuously until
48     triggered. It seems like the pool will get hot streaks and need a
49     cool-down period before the shells rain in again. The module will
50     attempt to use Anonymous login, by default, to authenticate to
51     perform the exploit. If the user supplies credentials in the
52     SMBUser, SMBPass, and SMBDomain options it will use those instead.
53     On some systems, this module may cause system instability and
54     crashes, such as a BSOD or a reboot. This may be more likely with
55     some payloads.

57  References:
58     https://docs.microsoft.com/en-us/security-updates/SecurityBulletins/2017/MS17-010
59     https://cvedetails.com/cve/CVE-2017-0143/
60     https://cvedetails.com/cve/CVE-2017-0144/
61     https://cvedetails.com/cve/CVE-2017-0145/
62     https://cvedetails.com/cve/CVE-2017-0146/
63     https://cvedetails.com/cve/CVE-2017-0147/
64     https://cvedetails.com/cve/CVE-2017-0148/
65     https://github.com/RiskSense-Ops/MS17-010

67  Also known as:
68     ETERNALBLUE

```

Figure 6.2. Metasploit Framework - Eternalblue Exploit Information

6.3 Network Mapper (NMap)

Network Mapper (Nmap) is a tool to map a network. The tool works by giving it an IP-address or an IP-address range, which it then scans for online hosts and open ports. It can also tell what ports are used for, which can be valuable when exploring devices.

Running an Nmap scanning on an IP-address is very simple and is just *"nmap xxx.xxx.xxx.xx"*. NMap will then scan the IP and tell which ports are available. On Figure 6.3 there is an example of running Nmap on a Hikvision IP-camera. The scanning shows that port 80, 554 and 8000 is open, which is identical to that described in section 4.1. Nmap even states what service is available on a given port. This makes it easy to scan a network, detect the devices, and the services running on them.

```
1      > nmap 192.168.1.10
2      Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-15 12:41 CEST
3      Nmap scan report for 192.168.1.10
4      Host is up (0.069s latency).
5      Not shown: 997 closed ports
6      PORT      STATE SERVICE
7      80/tcp    open  http
8      554/tcp   open  rtsp
9      8000/tcp  open  http-alt
11     Nmap done: 1 IP address (1 host up) scanned in 1.90 seconds
```

Figure 6.3. Running NMap on a Hikvision IP-camera.

Nmap has different options to scan, some of which require Nmap to be run as root or administrator. All the different options for Nmap scanning methods can be found here [18].

Based on the discoveries made during the problem analysis, this chapter will describe the design of the envisioned solution, which should solve the problem described in section 5.1

7.1 Project Goal

Through the problem analysis and resulting problem definition, a goal for the product development was needed to design a solution. Based on the revised problem statement, the project aims to develop a software solution that can:

1. Perform a network scan of a network to detect any online IP-cameras
2. Inform the users of the currently running version of the firmware
3. Perform a firmware upgrade if such an upgrade is available without breaking the configuration

With a clear goal for the product development, the design phase is much easier and should result in both a better design and an expandable product.

The goals of the solution can be categorized into the following components:

- Discovery
- Intelligence Gathering
- Firmware Crawler
- Device Upgrader
- User Interface

7.2 Components

In the following, we will describe each component more in dept and explain its responsibilities and expected behaviour.

7.2.1 Discovery

The discovery component is responsible for conducting the network scanning to find online IP-cameras throughout the network. Thus, the discovery components are an essential component since it provides the most basic information about the available IP-cameras, namely the IP-address and ports available for communication.

7.2.2 Intelligence Gathering

The Intelligence gathering component is responsible for gathering information from the different IP-cameras found by the discovery component. The intelligence component should obtain the information from the IP-cameras such as firmware version, hardware version, etc., making it easier to find and match available firmware to each device.

7.2.3 Firmware Crawler

The firmware crawler component is responsible for gathering and downloading different software versions from the manufacturer's resources and making them available for the relevant devices. The firmware crawler should maintain a small amount of the latest firmware versions available and obtain new firmware upon request. Since most firmware is specific to the targeted device, the firmware crawler should check if the targeted devices are within its cache; else, it should try to fetch a applicable version and add it to its cache.

7.2.4 Device Upgrader

The device upgrader component is responsible for conducting the actual upgrade of the IP-camera by uploading the latest version obtained from the firmware crawler. It is also responsible for ignoring the upgrade if the latest version of the firmware is of the same version as the one already installed on the IP-camera. Before the upgrade is performed, the device upgrade should obtain a backup of the device settings, which should be restored after the upgrade to maintain the user settings on each IP-camera.

7.2.5 User Interface

The user interface component is responsible for informing the user of the process and giving the user the control to start the process or execute individual components. The user interface should be easy to use and understand while providing the users with the necessary information and capabilities to effectively gain an overview of their network and the online IP-cameras.

7.3 Usages

Through the components identified, the design could allow the user to interact with the system in multiple ways. During our design phase, we identified two primary activity paths a user might use. On Figure 7.1 the five components can be seen connected with arrows representing information and actions going from each component.

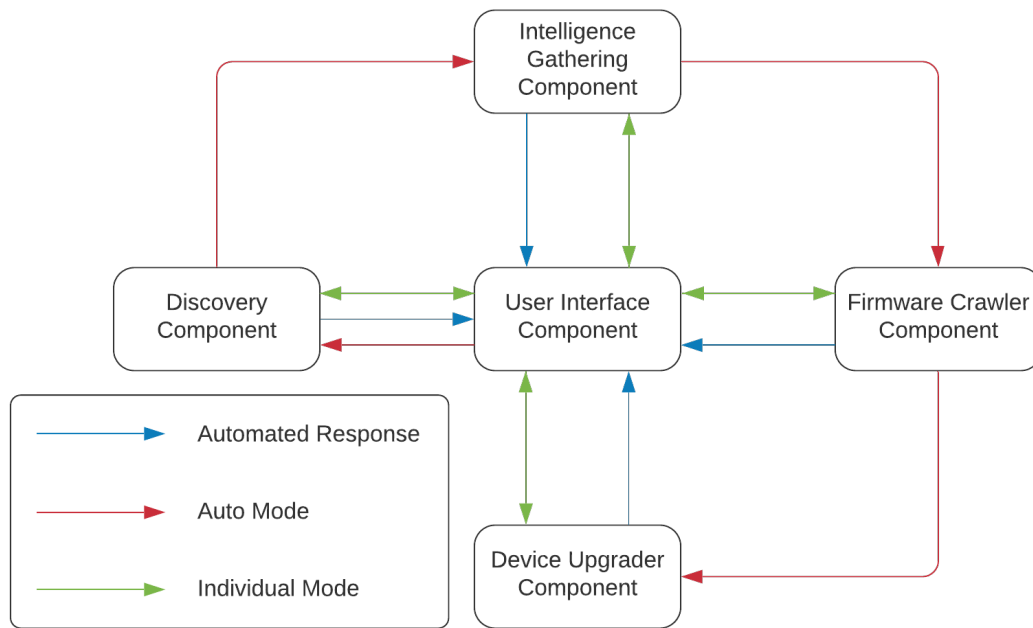


Figure 7.1. User Activity Diagram.

The user interacts with the User Interface Component and can choose two different paths. Either they can activate the path marked in red on Figure 7.1 where the solution will automatically follow the red arrows around the components. The blue arrow back to the User Interface Component represents information at each component once the different stages have completed. The other path is marked in green, where the user, through the User Interface Component, can activate each other component and have the result presented within the User Interface Component.

With a general idea of the design in place, the actual development of a solution could begin based on that design.

Implementation 8

The implementation is meant as a proof of concept and is built to support two different makers of IP-cameras. The first is Hikvision which is an expensive and professional-grade IP-cameras that are used throughout the world. The other Herospeed is a budget IP-camera that is commonly used in households and small businesses. By focusing on these two brands, we aim to provide a proof of concept, which should provide the reader with a better understanding of our concept, and it should lay the foundation for future work while providing a clearer picture of limitations and pitfalls. The following chapter will provide insights into the development of both the autodiscovery functionality, the user interface, and the automatic updater.

8.1 Autodiscovery

To find IP-cameras on the network, we created an implementation that uses the Nmap tool (See section 6.3). Using this tool, we can scan any IP-address or subnet for open hosts and open ports. Below in Figure 8.1 is shown the implementation running a scan on the 192.168.1.0/24 subnet. As can be seen, there are found devices responding on 7 IP-addresses in the subnet. We now have a list of available IP-addresses and which ports are open and listening for connections on them. From this, we can start testing the ports using our identification tool to find if there are any IP-cameras (In subsection 8.1.1).


```

1  [VALUE] IP = 192.168.1.5
2  [VALUE] Subnet = 192.168.1.0/24
3  [INFO] Scanning 192.168.1.0/24...
4  [INFO] Scan done!
5  [INFO] Host: 192.168.1.1 UP
6  Port      | State   | Service
7  -----
8  22        | open    | ssh
9  53        | open    | domain
10 80         | open    | http
11 [INFO] Host: 192.168.1.5 UP
12 Port      | State   | Service
13 -----
14 22        | open    | ssh
15 5000       | open    | upnp
16 [INFO] Host: 192.168.1.10 UP
17 Port      | State   | Service
18 -----
19 80        | open    | http
20 554       | open    | rtsp
21 8000      | open    | http-alt
22 [INFO] Host: 192.168.1.100 UP
23 Port      | State   | Service
24 -----
25 9999      | open    | abyss
26 [INFO] Host: 192.168.1.139 UP
27 Port      | State   | Service
28 -----
29 8008      | open    | http
30 8009      | open    | ajp13
31 8443      | open    | https-alt
32 9000      | open    | cslistener
33 10001     | open    | scp-config
34 [INFO] Host: 192.168.1.209 UP
35 Port      | State   | Service
36 -----
37 6668      | open    | irc
38 [INFO] Host: 192.168.1.241 UP
39 Port      | State   | Service
40 -----
41 6668      | open    | irc

```

Figure 8.1. Scanning the 192.168.1.0/24 subnet for hosts.

It should be noted that this scanning is not a complete scan, as we only scan the 1000 most commonly used ports.

8.1.1 Identification

Identifying the IP-cameras is an essential step because this allows us to handle Hikvision IP-cameras using the Hikvision ISAPI and Herospeed IP-camera using a custom made hook into their web interface. This is also applicable so that in the future, adding support for additional IP-cameras can be done the same way as with Hikvision and Herospeed.

When the scanning part is done, and we now know what ports are open on the devices, we can identify the services. Nmap attempts to identify the services, but this should be taken with a grain of salt as services are free to use any port they

like and not necessarily the correct ones.

To identify a device, we had to pull some information about running on the different ports. We have found that all Hikvision and Herospeed IP-cameras have their company name in the footer of their web interfaces during our analysis. So if we could perform an HTTP GET call to all the ports, and see if HTML is returned, we could examine the HTML and look at the footer.

However, we quickly found a problem with this. Hikvision and Herospeed both use a lot of javascript to set the HTML, which means that using curl to get the HTML resulted in a web page with missing information. So instead, we had to use a browser that supported javascript execution, so the information we needed to extract had been filled in. For this purpose, we decided to use a package provided by Composer named Symfony that provides a browser called Panther. Panther is a browser written in PHP that can run headless versions of Firefox, Chrome and Gecko browser. Panther is often used to scape web pages that use javascript to display data. Panther has a lot of functionality, but we just want to navigate to a URL, execute the javascript, and return the HTML for our purpose. The only problem with using Panther is that it is much slower than just using curl because it has to wait for the javascript to execute. It can still be done within a reasonable timeframe and often takes less than 3 seconds to get the HTML.

If HTML is returned, we attempt to parse it and look at the footer to see if it contains any copyright information. To navigate to the footer, we are using XPath, which is a way to tell where we want to retrieve the information in XML. For all Hikvision and Herospeed IP-cameras, their web interface contains copyright information in the footer that indicates who made the IP-camera. This allows us to identify Hikvision and Herospeed IP-cameras without the need for the end-user to supply any credentials.

Another way we considered doing the identification process was also using Nmap, but instead of looking at the web interfaces to determine if a device is a IP-camera, we could use the Nmap tool to identify the devices based on their mac address. The only problem with using the mac address instead of the web interface is that the mac address can be changed by the end-user, meaning if the mac address has been changed, this method will not work. The upside to this method is that it is much faster as it does not require scanning every port on a host but could ultimately yield false positives or missed IP-cameras. So in order to find more IP-cameras, we use the slower but more thorough method. A possible combination of using the mac-address and the Panther browser could be the most optimal way to solve the problem.

When the IP-camera has been identified as either a Hikvision, Herospeed or not an IP-camera, we filter the 'not an IP-camera' away and end up with only ports and IP-addresses belonging to valid Hikvision or Herospeed web interfaces.

We then have to ask for the credentials for these IP-cameras from the end-user to determine what firmware version is running on the IP-cameras and make changes to the IP-cameras. However, from our unfortunate experience, most IP-cameras use the default username and password against what is the right thing to do. As a result, we have implemented that if no username or password is provided, the default username and password for that device is used. For Herospeed, the default username/password is "admin/admin", and for Hikvision, it is "admin/password".

The next part of the identification process is getting the model and firmware version running on the IP-camera. This process is unique to each vendor as their IP-cameras work in different ways.

8.1.2 Hikvision Identification

For Hikvision IP-cameras, we utilise the "ISAPI", a web API that requires "digest" authentication with the username and password for the IP-camera's admin account. The Hikvision web interface depends on the API to get information rendered on the webpage using javascript injection.

We can use the same API to get information about the IP-camera. The API returns information in XML format, which can easily be converted to a key-value pair in PHP. Hikvision has released the documentation for the ISAPI, making it relatively easy to utilise[19]. All the endpoints require authentication, so we cannot use the API before we have the correct credentials provided by the user.

The endpoints we are interested in are the following:

- /ISAPI/System/deviceInfo
- /ISAPI/System/upgradeStatus
- /ISAPI/System/updateFirmware
- /ISAPI/System/configurationData

These endpoints are used to gather information about the device but are also used to update the device.

8.1.3 Herospeed Identification

For Herospeed IP-cameras, we found that there is a partly documented API endpoint at */ini.htm* that requires basic authentication to be accessed. This endpoint returns an ini style page that contains the configuration of the Herospeed IP-camera. This ini format is commonly used and can be easily converted into a key-value pair in PHP.

In some older version of Herospeed firmware, this endpoint does not require authentication, but we do not know exactly which firmware versions are vulnerable to this flaw.

```
1 textenable1=1
2 title=IPC resolution 2592 * 1944 20 720 * 576 20 hdr 0
3 videocodec=1
4 videocodecname=Single;Dual;Tri-Stream
5 videocodeccombo=1
6 mirctrl=0
7 mirctrlname=OFF;HORIZONTAL;VERTICAL;BOTH
8 timeformat=0
9 timeformatname=YYYY/MM/DD;MM/DD/YYYY;DD/MM/YYYY
10 bitrate1=4096
11 bitrate2=512
12 bitrate3=256
13 ratecontrol1=1
14 ratecontrol2=1
15 ratecontrol3=1
16 ratecontrolname=VBR;CBR
17 timestampenable1=1
18 textposition1=0
19 textpositionname=Top-Left;Bottom-Left
20 overlaytext1=Soggiorno
21 localdisplay=1
22 localdisplayname=Indoor NTSC;Indoor PAL;Outdoor
23 framerate1=20
24 framerate2=20
25 framerate3=20
26 livequality=0
27 qualityname=high;normal;low
```

Figure 8.2. Snippet of the returned data from a Herospeed IP-camera's ini endpoint.

During our analysis of Herospeed firmware (See section 4.2), we found that there is a hidden API endpoint that we were unable to find any documentation for. We have found the following endpoints:

- /api.html
- /params.html
- /upgrade.html
- /ini.htm

8.2 Interface

In this section, we will describe our implementation of the interface in which the user interacts.

Since we have implemented our solution in PHP, we are limited to the options PHP have to offer in terms of interactive options. PHP has very few frameworks that allow for creating desktop applications, but PHP can be used with HTML and CSS to create graphical user interfaces that work on Windows, macOS, Linux, Android, and iOS. However, PHP also allows for running as a console application, which is what we have chosen. The console interface was chosen as it is easier to implement and will work as a prototype. The downside is that it is not as user-friendly, but for our purpose, the demonstration that our prototype works are the most important. With PHP, it could be changed from a console application into a web application with an interface written in HTML or similar languages. This could improve the application's usability, but this is something we would look at in future works.

8.2.1 Interface implementation

We have implemented the application as a console application because it is easy and fast to implement ways to display simple information.

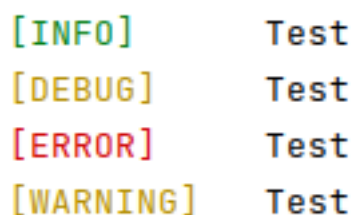
Since it is a console application, we have decided just to use the simplicity of *printf* function to print to the console, which allows us to print in different colours.

Inside the application, we have a class with static methods that can be called from any class in the application that allows for printing in different colours. We have methods used to print with different prefixes, which are the following:

- Info
- Warning
- Error
- Debug

These different prefixes have assigned different colours, indicating the amount of attention the user needs to put towards the messages.

In Figure 8.3 it can be seen what the colours look like when used in the program.



```
[INFO]      Test
[DEBUG]    Test
[ERROR]    Test
[WARNING]  Test
```

Figure 8.3. Interface colours showing the different message types.

The interface class also contains methods for reading input from the command line and methods for getting the user's confirmation.

8.3 Automatic updater

The automatic update is implemented to support two main cases; Hikvision and Herospeed IP-cameras.

8.3.1 Case - Hikvision IP-cameras

Updating a Hikvision IP-camera is done in 6 major steps. These steps are done to ensure that the device is upgraded successfully and that the device is not left misconfigured.

- Determine firmware and hardware version of the device
- Determine the latest firmware for the device
- Make a backup of the current configuration of the device
- Upload the new firmware to the device
- Check that the IP-camera has been updated to the new firmware
- Restore the configuration to the device to ensure all settings are identical

Determine firmware and hardware version of the device

The deviceInfo endpoint is the most important in the identification phase. The endpoint can be used with an HTTP GET request, which will return information about the device. Below in Figure 8.4 is shown the output running on Hikvision DS-2DE3304W-DE PTZ IP-camera. The important information to take from this is the Model, Firmware Version, Firmware Released Date and Firmware Version Info. Hikvision IP-cameras use different firmware depending on their hardware version. There are several hardware versions for Hikvision IP-cameras, and they are all listed below.

- E0
- E1
- E2
- E3
- E3L
- E3S
- E4
- E6
- E7
- G0
- G1
- G3
- G5
- H0
- H1
- H3
- H3+

- H5
- H7
- R0
- R1
- R2
- R3
- R4
- R6
- R7

The hardware version is part of the Firmware Version Info, which for the DS-2DE3304W-DE is "B-R-R7-0". Here R7 is the hardware version, which determines that this IP-camera uses Firmware for R7. This is important to know when checking for which firmware versions are available for the IP-camera, and the latest version for that hardware.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <DeviceInfo version="2.0" xmlns="http://www.hikvision.com/ver20/XMLSchema">
3   <deviceName>IP DOME</deviceName>
4   <deviceID>884bc000-de12-11b5-84c6-f84dfc65b45b</deviceID>
5   <deviceDescription>IPDome</deviceDescription>
6   <deviceLocation>hangzhou</deviceLocation>
7   <systemContact>Hikvision.China</systemContact>
8   <model>DS-2DE3304W-DE</model>
9   <serialNumber>XXXXXXXXXXXXXXXXXXXX</serialNumber>
10  <macAddress>f8:4d:fc:65:b4:5b</macAddress>
11  <firmwareVersion>V5.5.71</firmwareVersion>
12  <firmwareReleasedDate>build 180725</firmwareReleasedDate>
13  <encoderVersion>V7.3</encoderVersion>
14  <encoderReleasedDate>build 180320</encoderReleasedDate>
15  <bootVersion>V1.3.4</bootVersion>
16  <bootReleasedDate>100316</bootReleasedDate>
17  <hardwareVersion>0x0</hardwareVersion>
18  <deviceType>IPDome</deviceType>
19  <telecontrolID>88</telecontrolID>
20  <supportBeep>false</supportBeep>
21  <supportVideoLoss>false</supportVideoLoss>
22  <firmwareVersionInfo>B-R-R7-0</firmwareVersionInfo>
23 </DeviceInfo>

```

Figure 8.4. Output of calling the deviceInfo endpoint on a Hikvision IP-camera (The serial number has been hidden as it is used to reset the administrator password).

If the IP-camera runs older version firmware, the deviceInfo endpoint does not return which hardware version the device is using. We have created a file with a huge key-value pair array for these scenarios where the key is the IP-camera's model name, and the value is the hardware version. This allows us to determine the hardware version from the model name for IP-cameras with such old firmware that the IP-cameras do not provide which hardware version they are using. It is preferable to get the hardware version from the IP-cameras because this makes the implementation able to support new hardware versions without updating the code.

So by having this information locally stored allows us to upgrade most if not all Hikvision IP-cameras.

Determine the latest firmware for the device

When we know the current firmware version and which hardware version the IP-camera is running, we can then download firmware from the Hikvision download portal[20]. This would preferably be handled by a crawler that, when given the firmware version and hardware version, download the latest firmware from the portal. But for now, we just have a folder with all the firmware versions for that hardware version, and the program can select the correct firmware.

The Hikvision firmware downloading portal is a simple folder structure where one can choose folders based on the hardware version and are given a list of different firmware versions. A screenshot of the page can be seen in Figure 8.5. As the page is quite simple and easy to understand, it would be an easy task to create a crawler that could download the firmware.

File/Direcory	File Size	Date
..	-	2019-03-27
ANPR(DS-2CD4xxxx)	-	2020-11-06
E0 Civil platform (2U01,2U21,2Q01,2Q21)	-	2021-02-24
E1 platform(12X1,13X1)	-	2018-12-19
E2 platform (DS-2CD1X01,IDS-2CD6810)	-	2021-02-24
E3 platform(2X23G0-IW2 Wi-Fi,1X23G0,2X21G0,2X27G3E)	-	2020-04-30
E3L platform(1X13G0)	-	2018-12-19
E3S platform(1X23G0E,1X21(E))	-	2019-11-21
E4 platform(1X21,1X31)	-	2019-07-17
E6 platform [Wi-Fi (B) model, 2421]	-	2021-02-22
E7 platform (1X43G0E)	-	2021-02-24
G0 platform(1X43G0 1X53G0 2X47G3E 2X4'51G1-IDW1'2)	-	2020-04-30
G1 platform (H.265 DS-2CD2XX5,2XX3,3XX3,3XX5,XM67X6)	-	2020-09-21
G1+M platform (2XX6)	-	2019-06-13
G3 platform(2XE64X2F,2XC66X5G0)	-	2020-09-29
G3 platform(2XX6G2 3XX6G2)	-	2020-06-22
G5 (XS6A Solar Kit)	-	2021-04-02
G5 platform(2xx3G2 2xx6G2(C) 2xx7G2(C) 3xx6G2(C) 3xx7G2(C) 1x83G0)	-	2021-02-25
H0 platform(40X5)	-	2018-12-19
H1 DarkfighterX (5028 Customization Firmware)	-	2021-02-25
H3 platform(5xxx,7xxx,6924,6DX4,8426)	-	2021-02-25
H3+M platform (6425G0'F,63X5)	-	2019-06-18
H5 (DS-2CD69XXG0)	-	2021-04-02
H5 (DS-2CD8146)	-	2020-01-15
H7 Platform(DS-2CD6DX2,6825)	-	2021-04-20
H7 Platform(IDS-2CD7XXX)	-	2021-02-26

Figure 8.5. Hikvision download portal.

Here is an example of how we have indexed the firmware in the implementation (Figure 8.6.)

```

1  |-- hikvision
2      |-- G1
3          |-- IPC_G1_EN_STD_5.5.61_180718.zip
4          |-- IPC_G1_EN_STD_5.5.80_180911.zip
5          |-- IPC_G1_EN_STD_5.5.82_181211.zip
6          |-- IPC_G1_EN_STD_5.5.83_190221.zip
7          |-- IPC_G1_EN_STD_5.6.0_190507.zip
8          |-- IPC_G1_EN_STD_5.6.1_190604.zip
9          |-- IPC_G1_EN_STD_5.6.2_190701.zip
10         |-- IPC_G1_EN_STD_5.6.3_190923.zip
11         |-- IPC_G1_EN_STD_5.6.4_191224.zip
12         |-- IPC_G1_EN_STD_5.6.5_200316.zip
13         |-- IPC_G1_EN_STD_V5.5.0_Build170725.zip
14         |-- IPC_G1_EN_STD_V5.5.4_Build171206.zip
15         |-- IPC_G1_EN_STD_V5.5.51_Build180314.zip
16         |-- IPC_G1_EN_STD_V5.5.51_Build180326.zip
17     |-- R6
18         |-- IPC_R6_EN_STD_5.5.82_190909.zip
19     |-- R7
20         |-- IPC_R7_EN_STD_5.5.53_180719.zip
21         |-- IPC_R7_EN_STD_5.5.80_180911.zip
22         |-- IPC_R7_EN_STD_5.5.82_190117.zip
23         |-- IPC_R7_EN_STD_5.5.83_190218.zip
24         |-- IPC_R7_EN_STD_5.5.86_200108.zip
25         |-- IPC_R7_EN_STD_V5.4.4_Build161022.zip
26         |-- IPC_R7_EN_STD_V5.5.0_Build170726.zip
27         |-- IPC_R7_EN_STD_V5.5.0_Build170914.zip
28         |-- IPC_R7_EN_STD_V5.5.6_Build180328.zip

```

Figure 8.6. Folder structure of the firmware for Hikvision IP-cameras.

As can be seen, all the firmware is not named identically, but it is close enough that a simple regex expression can recognise and separate the necessary information. The regex that can extract the firmware version and build from the filename is as shown in Figure 8.7.

```

1  /.*((v)?(<version>(\d+)\.(\d+)\.(\d+)))*?(<build>(\d+)).*/i

```

Figure 8.7. Regex for extracting firmware version and build from filename.

When the information is extracted we can determine in which order the firmware is supposed to be placed. This is all done in a firmware compare function which is shown in Figure 8.8.

```
1 public static function cmp(firmwareVersion $a, firmwareVersion $b): int
2 {
3     if ($a->major-$b->major == 0) {
4         if ($a->minor-$b->minor == 0) {
5             if ($a->patch-$b->patch == 0) {
6                 return $a->build-$b->build;
7             } else {
8                 return $a->patch-$b->patch;
9             }
10        } else {
11            return $a->minor-$b->minor;
12        }
13    } else {
14        return $a->major-$b->major;
15    }
16 }
```

Figure 8.8. Compare function used to compare firmware versions and find out which is the newest.

This method compares firmware versions from Major, Minor, Patch and even Build versions of different firmware. Comparing the firmware down to even the correct build is important since Hikvision IP-cameras have to be updated chronologically and can not just be upgraded from whatever version it is running to the newest. This means that if a Hikvision R7 IP-camera is running firmware "V5.5.0 Build 170726", then the next firmware is not "V5.5.6 Build 180328" but "V5.5.0 Build 170914". Attempting to upgrade a IP-camera to a version that is not the next will result in the IP-camera attempting to upgrade the firmware but often failing since the current firmware cannot be upgraded to the given. There is no error message from the IP-camera telling that it was the wrong version or even failed. This problem was identified by reading the Hikvision manual for the IP-cameras, as it is documented in the section about firmware upgrading.

Hikvision implements it because the new firmware knows which older firmware versions it is compatible with. Every rule has exceptions, which is also the case with Hikvision. This exception to the rule of chronological order is broken with some of Hikvisions G1 IP-cameras, which are variable focal models, where the model name ends with 26, 27, 2H. These IP-cameras have their special firmware different from regular G1 firmware from version 5.4.5 to 5.4.6 but then returns to using the same firmware for versions 5.5.0 forwards.

This gives the solution some edge cases where some IP-cameras depending on the model, have to be upgraded twice compared to once to get from one firmware version to another, making it a bit more complicated. However, from our experience, this only affects a very select few IP-camera models and firmware, which in the scope of this project is not something we want to spend a lot of time customising to suit a few old firmware versions for. This could be looked at in future works of the product to handle these edge cases, but we will not spend time to implement it in our prototype as it is edge cases. On Figure 8.9 there is a screenshot of the download

portal showing the extra firmware.

■ V5.4.5_Build170124	-	2018-12-19
■ V5.4.6_Build170427 (only for vari focal models 26, 27, 2H)	-	2018-12-19
■ V5.5.0_Build170725	-	2018-12-19

Figure 8.9. Hikvision firmware for G1 IP-camera[20].

Make a backup of the current configuration of the device

Before we attempt to update a Hikvision IP-camera, we take a backup of the configuration on the IP-camera. According to the Hikvision ISAPI manual, this requires a password given to the IP-camera as the encryption key. This configuration file contains sensitive information such as admin username and password and information like credentials for SMTP. Since the encryption key has to be provided when uploading the configuration again, we need to set it to something that the program knows and can remember. For ease of it, since the user has already given us the password for the admin user, we use the password for the user as the encryption key. The user is informed about this and where the configuration file is stored to give the user a configuration backup.

To get the current configuration, we use Curl to make an HTTP GET call to the configuration endpoint */ISAPI/System/configurationData*. This endpoint takes a parameter "secretkey" which we set to the password, which the user provided; this key is the same that has to be provided when the user was asked for the password of the IP-camera. The return of this endpoint is configuration data in encrypted binary form. We then store the configuration in a folder with the structure shown below (Figure 8.10). The files are named using this format "*config_XXXXXXXXXXXXX.bak*", where X is the timestamp the file was created at. This gives us unique filenames and a way to find the latest config file.

The configurations backups are stored in a folder for the manufacturer, which has subfolders named the IP-address followed by the port. This is how the configurations are separated so that we know which files are for which IP-cameras. The folder structure can be seen in Figure 8.10.

```
1  |-- hikvision
2  |  |-- 127.0.0.1:8081
3  |  |  |-- config_1619445197.bak
4  |  |  |-- config_1619445271.bak
5  |  |  |-- config_1619448552.bak
6  |  |-- 127.0.0.1:8082
7  |  |  |-- config_1619452775.bak
8  |  |  |-- config_1619460226.bak
9  |  |-- 127.0.0.1:8083
10 |  |  |-- config_1619460884.bak
11 |-- 127.0.0.1:8084
12 |  |-- config_1619515832.bak
13 |  |-- config_1619518789.bak
```

Figure 8.10. Folder structure of configuration files.

Upload new firmware to device

After taking a backup of the configuration, we are now ready to begin the upgrade of the IP-camera. Hikvision has generously provided a simple way where using a Curl HTTP PUT call to upload a file to the IP-camera to the upgrade endpoint, and the IP-camera will start upgrading. The file that we upload is always called "digicap.dav" and needs to be extracted from the ZIP file of the corresponding firmware version. We have decided that extracting all the firmware beforehand was an unnecessary step as it would consume more disk space, and since the extraction process is quite fast, this is not a problem.

So the first step is to extract the correct firmware. This is done in the prepare firmware method, which can be seen in Figure 8.11. This method extracts the "digicap.dav" file to the "/tmp/" folder, which is a mounted folder of the ram. Because of this, the extracted firmware will not take up any storage space and will be automatically deleted on a system reboot. If the extraction is successful the function returns the path to the extracted "digicap.dav" file. This file is going to be uploaded to the IP-camera using the Curl HTTP PUT method.

```

17 private function prepareFirmware(firmwareVersion $nextFirmware): string
18 {
19     $zip = new ZipArchive;
20     $path = $nextFirmware->getPath();
21     $res = $zip->open($path);
22     if ($res === TRUE) {
23         printer::printInfo("Extracting $path...", false);
24         $zip->extractTo('/tmp/');
25         $zip->close();
26         printer::println("Done!");
27
28         if (file_exists("/tmp/digicap.dav")) {
29             printer::printInfo("Extracted file: /tmp/digicap.dav");
30             return "/tmp/digicap.dav";
31         }
32     } else {
33         printer::printError("Failed to extract $path");
34     }
35 }

```

Figure 8.11. The Prepare Firmware method used to extract the firmware.

After the firmware is uploaded to the IP-camera, the IP-camera will start updating. To keep track of how long through the upgrade process the IP-camera is, a call to the upgrade status endpoint is made, which returns if the IP-camera is upgrading and how far it has come in the process. Here is an example of the returned data from the IP-camera when the upgrade status endpoint is called Figure 8.12. As can be seen, the IP-camera is upgrading, and the upgrade is 54% done.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <upgradeStatus version="2.0" xmlns="http://www.hikvision.com/ver20/XMLSchema">
3     <upgrading>true</upgrading>
4     <percent>54</percent>
5 </upgradeStatus>

```

Figure 8.12. Data returned from the upgrade status endpoint during a upgrade.

We keep pulling information from the IP-camera during the upgrade while we wait for it to come back up. This is done by checking the status code that is returned from the login page; if the status code is 200, we know the IP-camera is up and running again. However, if a status code 404 is returned, we know that the IP-camera is not done updating or rebooting yet. In the event that the IP-camera does not complete the update and reboot within 5 minutes, we inform the user that the IP-camera failed to boot up and that they need to take a manual action at the IP-camera to ensure that it is functioning correctly. In this state the software will wait for the user to confirm that the IP-camera is back online and then check if the update simply took longer than 5 minutes, or if it failed.

This timeout has been built into the software to ensure that, if something has gone wrong with the IP-camera. The user should handle such errors to ensure that there are no cases where the IP-camera is left in a state where it is unable to record. The

timeout is an important feature since the IP-camera should be able to update its firmware within that time. Failing to do so within 5 minutes often means that the IP-camera experienced an error or the uploaded firmware file was corrupt.

This method results in the upgrade has to result in a functioning IP-camera, or the user being informed that the IP-camera needs the user's attention.

Check that the IP-camera has been updated to the new firmware

When the upgrade is finished, the upgrading value will return to false, and the percentage will be set to 0. However, at this point, the IP-camera is still performing the upgrade, and at this time, we will have to wait for the login page to become available again. So we keep trying to get the login page until we get an HTTP status code 200. When we get the status code 200, we restart the IP-camera again before we check the IP-camera if the firmware has been updated. This is because sometimes the devices has to be restarted before we get the actual new firmware version from the ISAPI. If the firmware version is the new we inform the user that the upgrade was successful. If the upgrade failed we inform the user and ask them to check if it was the correct firmware for that IP-camera. A potential reason for the upgrade to fail could be one of the edge-cases mentioned above, where a IP-camera requires a particular version of the firmware, or because the firmware is not compatible with this particular IP-camera.

Using a crawler to download the correct firmware is a "nice-to-have" feature, but this has not been implemented for this prototype.

Restore the configuration to the device to ensure all settings are identical.

Regardless of the firmware update being successful or failing, we upload the configuration file to ensure that the IP-camera has not lost its configured settings. This is done by making a Curl HTTP POST call to the configuration endpoint with the configuration file instead of earlier, where we made a GET call. The Hikvision IP-camera does not care about the file extension of the configuration file.

After the IP-camera has accepted the configuration, we ask the IP-camera to reboot, which is required for the configuration file to take effect. We wait for the IP-camera to come back online, and we check that we can reach it to ensure that it is in working condition.

At this point, the IP-camera should be working again with new firmware or accompanied by an error message saying that the IP-camera failed to upgrade. Since we cannot get error messages from the Hikvision IP-camera as to why the firmware failed to upgrade, we can only speculate that it was given an incorrect or incompatible firmware.

8.3.2 Case - Herospeed IP-cameras

Updating a Herospeed IP-camera is done in 6 steps to ensure that the IP-camera is upgraded correctly and to ensure that IP-camera configuration is applied correctly. The steps are in increasing order:

- Determine firmware and hardware version of the device
- Determine the latest firmware for the device
- Make a backup of the current configuration of the device
- Upload the new firmware to the device
- Restore the configuration to the device to ensure all settings are identical

Determine firmware and hardware version of the device

Depending on the maker of the IP-camera, each IP-camera from Herospeed behaves differently. This is because Herospeed only makes the firmware for the IP-cameras. If the device uses plain HTML as its web component, they are trivial to use and upgrade. If, however, it uses their Netscape plugin, the information is more tricky to retrieve. The different types of IP-cameras all require the users to authenticate before interacting with the IP-camera. The user's identity is proved by calling the API with a JSON object containing both the endpoint and the credentials for the user. The username is sent in plaintext while the password is computed into an MD5 checksum. On Figure 8.13 the body of the POST request is seen, and on Figure 8.14 the response from the IP-camera is shown. With this "sessionID" attached to all new requests, the requests will be authorized.

```
1 {  
2   "api": "login",  
3   "data": {  
4     "username": "admin",  
5     "password": "21232f297a57a5a743894a0e4a801fc3"  
6   }  
7 }
```

Figure 8.13. Body of login request.

```
1 {  
2   "code": 0,  
3   "data": {  
4     "Cookie": "sessionID=9dd3bcafb240f978c9e5f30536db177f"  
5   }  
6 }
```

Figure 8.14. Response on a valid login.

To gather the information from the IP-cameras, the API has an endpoint to fetch all known information from the IP-camera. The API requires the "sessionID" sent as

a cookie header on the post request. The information received from the IP-camera has the layout and information as shown on Figure 8.15.

```

1 {
2   "code": 0,
3   "data": {
4     "DevType": "0",
5     "PlatformType": "21",
6     "DevName": "IPC",
7     "DEVID": "XXXXXXXXXX",
8     "Manufacturers": "HeroSpeed",
9     "SoftwareVersion": "0.1.10.4",
10    "FirmwareVersion": "FH8856_K05_B1T1A1M0C0P1_W_20.1.10.4",
11    "Platform": "FH8856",
12    "DevSensor": "K05BD"
13  }
14 }
```

Figure 8.15. JSON response with device information from Herospeed IP-camera.

The information received contains both a "SoftwareVersion" and a "FirmwareVersion". However, the two version numbers are not identical, however very close. The reported "SoftwareVersion" has no documentation or references on the manufactures website and is unknown what it is used for, but we believe this to be an error. This error is corrected by overwriting the "SoftwareVersion" value with the information gathered from the "FirmwareVersion".

Make a backup of the current configuration of the device

Backing up the configuration on the Herospeed IP-cameras require using the API.

Thue API supports saving all parameters on the IP-camera into a file stored on the IP-camera. The API requires authentication with the "sessionID" set as a cookie. On a valid request, the information seen on Figure 8.16 will be received as a response.

```

1 {
2   "code": 0,
3   "data": {
4     "url": "/filedir/FH8856_K05_B1T1A1M0C0P1_W_20.1.10.4.zip",
5     "fileName": "FH8856_K05_B1T1A1M0C0P1_W_20.1.10.4.zip"
6   }
7 }
```

Figure 8.16. JSON response with device backup file.

After the endpoint is called, the data shown in Figure 8.16 is returned. The url indicates a location where the configuration is located. This url can then be used to download the configuration.

While the filename hints that it is a ZIP archive, the file is just a binary file containing the configuration, although obfuscated.

Determine the latest firmware for the device

All Herospeed IP-cameras have a feature within their web interface to check for the new firmware from the manufacture. By examining the network traffic generated by the IP-camera, a single outgoing request was noticed. The IP-camera sends a request to http://www.herospeed.net/hs/ipc/fuhan/FH8856_K05_B1T1A1M0COP1_W/updateinfo.txt where the link is divided into two dynamic sections. Firstly, in this case, Fullhan is interpreted as "fuhan" which can be derived from the currently running firmware version starting with "FH". The second part is the full device name which can be derived from the firmware version. The response from Herospeed contains all necessary information about the firmware as seen on Figure 8.17

```
1 MAGIC:16g1YHof7kSH2
2 VER:20.1.10.4
3 FILE:FH8856_K05_BASE_BD_W_20.1.10.4.FLS
4 MD5:8e9fbd72236961b4ac9cf1879cc8277f
5 LOG:update_log_v20.1.10.4.txt
```

Figure 8.17. Updated response from Herospeed.

Once the running firmware version and the newest available firmware is found, it is trivial to compare the two version numbers to determine whether the device is up-to-date or not. If the firmware version found on Herospeed is of a higher version number, the user is prompted for confirmation if they want to download and apply the update. From the response, see Figure 8.17, the file location and MD5 checksum is used to both download the firmware and verify that it matches the checksum. If the checksum does not match, the program will not apply the update, and the user will be notified.

Upload the new firmware to the device

Once the firmware update is downloaded, and the checksum has matched, the IP-camera has an API endpoint to upgrade the device. As with the other endpoints, the device requires the user to be authorized with a valid sessionID. With the sessionID and update file, the program will make a POST request to the IP-camera on the "upgrade.html" endpoint. If the file is accepted, the IP-camera will respond with a code 0, meaning everything is okay. While the upgrade is being conducted, it will enter a maintenance mode, where all requests will be met with a maintenance notice. Once the IP-camera has completed the upgrade, it will reboot and continue to normal operating mode. The IP-camera does not provide insights on progress of the update nor does it respond upon completion without interference from a user. Therefore, the user is incurred to check the IP-cameras web interface before continuing.

Restore the configuration to the device to ensure all settings are identical

Once the device upgrade has completed, the user can choose if they want to apply to settings from the backup created at the beginning of the process. If so, the file is pushed to the same endpoint, which also creates the backup. Once the IP-camera accepts the backup, the device does not require a reboot; the device will return a code 0 upon success.

Related Work 9

In this chapter, we will describe some related work conducted by others. We will describe tools made by both Hikvision and Herospeed for their IP-cameras and a tool made by Viakoo. These tools all provide some functionality similar to our solution, but they differentiate in some key aspects.

9.1 Hikvision - TFTP Automatic Update Tool

Hikvision has developed a tool they call TFTP Automatic Update Tool, which can update and recover Hikvision IP-cameras. The tool will only run on a Windows computer and requires the computer to be directly connected to the IP-camera. Once the program is running, each IP-camera will have to be rebooted, during which the program will detect the device and attempt an update. The computer running the program must be assigned to IP-address 192.0.0.128/24. The tool can only handle one version of firmware and, consequently, only one device model at a time. The tool might require the user to disable the Windows Firewall if the tool cannot connect to the IP-cameras.[21]

The IP-camera will then try to pull the firmware file from any TFTP server hosted on "192.0.0.128". This method is often used to recover Hikvision IP-cameras that have been bricked, or a firmware update went completely wrong. Using the TFTP automatic update tool bypasses any authentication on the IP-camera and does not preserve any running configuration.

9.2 Herospeed - Search Tool

Herospeed has developed their tool called Search Tool. The Search Tool can scan a network for devices and list them for an accessible overview. The tool only can find Herospeed products, such as IP-cameras and Network Video Recorders. The tool allows the user to select several devices and make mass changes, such as configuring the username and password, changing network parameters and more. The tool can also upgrade the firmware on the devices with a firmware upgrade located on the computer. The Search Tool can be installed on both Windows and macOS and does not require some special plugins to function.[22]

9.3 Viakoo - IP-camera Firmware Update Manager

Viakoo is a company that focuses on IoT device security, physical security and compliance. Their main product is a web dashboard that manages devices for companies. The IP-camera Firmware Update Manager (CFUM) makes it easier for system administrators to manage devices and schedule updates. The CFUM can manage devices from different manufactures. However, it is unclear which and how many they support. It works by creating profiles for each device model and assigning parameters within each profile, such as selecting the intended firmware version. It cannot download nor notify the users of newer firmware made available by the manufacturer but instead relies on the user to know that new firmware is available. The CFUM focuses primarily on the compliance part of device management. The product works by having a user set rules for devices. These rules could be that the firmware has to be above 5.7.0. Then the program can inform which IP-cameras have older firmware than 5.7.0. [23]

Evaluation 10

This chapter will evaluate the developed solution regarding its current state and feature set to a plausible deployment. Lastly, we will describe the following development options, which should be developed later.

10.1 Solution

The solution developed, as described in chapter 8 has multiple unique abilities when compared to the alternative solutions described in chapter 9. It is essential to notice that the developed solution is purely a proof-of-concept meant to validate the project's premise and should thus not be seen as a finished product. The solution can find online IP-cameras on an accessible network. It can identify IP-cameras made by both Hikvision and Herospeed and access device information with credentials provided by the user. The solution can find the newest available firmware for Herospeed IP-cameras and select the latest firmware from a folder containing Hikvision firmware. With the newest firmware identified, the solution can perform a firmware upgrade on IP-cameras from both manufacturers and restore the user settings based on a backup.

The solution is not perfect in any way and has only been tested on a minimal amount of devices. It is unable to download new firmware for Hikvision IP-cameras. The current implementation of the Herospeed firmware upgrade does not validate once the upgrade has finished. The solution does not keep track of IP-cameras, meaning that the same IP-camera might have changed IP-address between two scans, which might be helpful information to a user. The current implementation does not provide its users with a graphical user interface, which results in a system that can be hard to use.

We see our solution not as a stand-alone tool but rather as an option that would integrate nicely into existing solutions such as pfSense, an open-source firewall and router operating system[24]. By integrating into pfSense as a plugin, it would ideally continuously monitor and upgrade IP-cameras automatically. The solution is built as a modular system, where each IP-camera manufacture is isolated and would be easy to extend for enabling support for other manufactures.

10.1.1 Future Works

For future versions of the solution, several aspects would need further work. The current solution is written as a command-line interface application in PHP without any graphical user interface. Creating a graphical user interface would make the solution more approachable to novice users while providing a better overview to all types of users. The solution should also enable monitoring and have the ability to compare different searches to highlight changes to configurations and settings.

The solution would also be able to read the configurations taken as part of the backup process and give feedback to the user of any insecure settings, such as weak passwords and exposed ports. The user should be able to change the settings directly from the solution and potentially push the changes to multiple IP-cameras.

The Herospeed IP-cameras do not report an upgrade process, but it responds to all requests with an upgrade notice. The implementation lacks any integration with this notice. Once implemented, it would eliminate the chance for the backup to be uploaded to the device prematurely. Lastly, it would be very beneficial to add additional support for more IP-camera manufactures.

From our experience, most IP-cameras have web interfaces that rely on SDKs or APIs, which could be reverse engineered to allow our solution to update IP-cameras from almost any manufacturer.

This chapter will discuss our solution and compare it to the problem definition that the project aimed to solve. Furthermore, we will also compare our solution to existing tools.

11.1 The Solution To The Problem

The problem which this project aimed to solve was described in section 1.1 and later revised in section 5.1:

How can a software solution scan the local network for IP-cameras and inform the user of outdated devices, and perform firmware updates on the IP-cameras automatically?

The solution developed during this project, which is described in chapter 8, was proposed as a solution to the problem. The solution can scan a local or remote network for IP-cameras manufactured by either Hikvision and Herospeed. The solution can detect and inform users of outdated devices and perform updates on the IP-cameras with acceptance from the user. Since the product is a proof of concept to solve the problem, we believe that the tool in a complete state would be a valuable tool for both novice and professional users.

The proposed solution relies on the device manufacturers to develop and maintain firmware updates for their devices to secure their products. By analysing the firmware images from Hikvision, it is infeasible for end-users to validate the consequences of a firmware update, let alone write them. Therefore, it is paramount that the device manufacturers prioritise security updates and make such updates available to the users.

Our solution areas mentioned were only able to update device firmware and fix security issues from that perspective. However, misconfigurations are often a significant factor for security issues on the devices. An alternative approach or extension would be validating the device configuration from the backup file and proposing an alternative optimised configuration. For example, some IP-cameras will prompt the user with a warning if the password is considered weak. Such functionality would and should also be a part of the solution. Since we have the

configuration, the solution could be extended with functionality like a password manager that would be able to create long and unique passwords for each IP-camera but remember it for the user. Other configurations, such as clock synchronisation, open ports, etc., would also be very potent to check on behalf of the user. Some IP-cameras feature service ports and potential backdoors, which could also be essential to inform the end-user about and not have in the first place.

Our solution is, when compared to the other solutions, rather unique. Compared with the tools from Hikvision and Herospeed, our tool supports firmware upgrades for devices from both manufacturers but does currently not support changing devices settings and configurations. Then compared with the tool from Viakoo, our tool can find new updates for Herospeed IP-cameras remotely and identify the latest available version for Hikvision IP-cameras from a local directory. The main focus point for Viakoo seems to be the ability to verify firmware versions from a compliance perspective while having the ability to manage said firmware via profiles specified by the administrator. Each IP-camera type and model needs to have a separate profile that manages the version of firmware the IP-camera should use, which provides the administrator with control over the firmware versioning. The firmware specified in each profile is chosen and uploaded by the user, and the system cannot find new firmware from the manufacturers and does not help the users identify outdated devices. The user only knows that a device is running a particular firmware version which might not necessarily be the latest.

Conclusion 12

Through this project, we have analysed IP cameras and typical setups. The analysis resulted in a revised problem statement as described in section 5.1, which we see as a plausible problem that, if solved, would result in a generally more secure IP-camera culture thus limiting the abilities of botnets and prevention of data leakage.

The proposed solution, which is a proof of concept, provides a working solution to solving the problems statement while still leaving some features to be desired. The solution solves the problem by providing a framework through which firmware updates can be applied to multiple IP-cameras from multiple manufactures at once. The solution is deemed necessary since each IP-camera manufacturer provides different interfaces and workflows to perform the firmware upgrade. While most manufacturers also provide similar tools for updating their products, they are limited to only their products. The main problem with these tools is that they only work with a single product, and they are rarely made cross-platform. Compare to our solutions which is fully cross-platform, making it available on Windows, macOS and Linux. With our solution, the overall security of IP-cameras will be increased by providing a trivial way of updating IP-cameras. This approach relies on the IP-camera manufacturers to write firmware updates and security patches for their supported devices.

Only a limited amount of IP-cameras has been tested throughout the project due to the lack of access to IP-cameras The solution was developed with access to Hikvision IP-cameras through the dorm security system, to which one group member has access. Through the solution, the dorm could update their entire IP-camera system, which had previously been a challenge. The IP-camera setup consists of 9 Hikvision IP-cameras with multiple hardware versions spread across four different buildings with isolated networks.

Bibliography

- [1] *Understanding the mirai botnet*, <https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-antonakakis.pdf>, Accessed: 17-12-2020.
- [2] W. Turton, *Hackers breach thousands of security ip-cameras, exposing tesla, jails, hospitals*, <https://www.bloomberg.com/news/articles/2021-03-09/hackers-expose-tesla-jails-in-breach-of-150-000-security-cams>, Accessed: 24-03-2021.
- [3] Verisure, *Videoovervågning*, <https://www.verisure.dk/alarmsystemer/videoovervagning>, Accessed: 13-04-2021.
- [4] G4S, *Danmarks største beredskab*, <https://www.g4s.dk/om-g4s/beredskab>, Accessed: 13-04-2021.
- [5] EMILY HODGKIN. “Smart doorbell urgent warning: Cheap versions ‘sending passwords to china’.” Accessed: 20-06-2021. (2021), [Online]. Available: <https://www.express.co.uk/life-style/property/1363457/smart-doorbell-doorbells-uk-news-which>.
- [6] “Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model,” International Organization for Standardization and International Electrotechnical Commission, Geneva, CH, Standard, 1994, ISO/IEC 7498-1:1994(E).
- [7] *Ieee standards for local and metropolitan area networks: Virtual bridged local area networks*, eng, 1999.
- [8] Statcounter, *Browser market share worldwide*, <https://gs.statcounter.com/browser-market-share>, Accessed: 17-03-2021.
- [9] *Netscape plugin application programming interface*, <https://en.wikipedia.org/wiki/NPAPI>, Accessed: 12-04-2021.
- [10] *Waterfox*, <https://en.wikipedia.org/wiki/Waterfox>, Accessed: 12-04-2021.
- [11] *Universal plug and play*, https://en.wikipedia.org/wiki/Universal_Plug_and_Play, Accessed: 29-03-2021.

- [12] *Upnp callback vulnerability*,
https://en.wikipedia.org/wiki/Universal_Plug_and_Play, Accessed:
06-04-2021.
- [13] *Dropbear wiki*, [https://en.wikipedia.org/wiki/Dropbear_\(software\)](https://en.wikipedia.org/wiki/Dropbear_(software)),
Accessed: 07-04-2021.
- [14] *Data encryption standard*,
https://en.wikipedia.org/wiki/Data_Encryption_Standard, Accessed:
07-04-2021.
- [15] *Crackstation.net wordlist*, <https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm>, Accessed: 07-04-2021.
- [16] *Boa (web server)*, [https://en.wikipedia.org/wiki/Boa_\(web_server\)](https://en.wikipedia.org/wiki/Boa_(web_server)),
Accessed: 15-04-2021.
- [17] A. Singh, *Metasploit Penetration Testing Cookbook, 2nd Edition*. Oct. 2013, ISBN: 1782166785.
- [18] Nmap Documentation. "Nmap network scanning - port scanning techniques." Accessed: 20-06-2021. (2021), [Online]. Available:
<https://nmap.org/book/man-port-scanning-techniques.html>.
- [19] *Hikvision ip surveillance api*, <https://usermanual.wiki/Pdf/HikVision20IP20Surveillance20API20RaCM20Part20User20Guide.1552796388/view>, Accessed: 28-05-2021.
- [20] *Hikvision downloaod portal*,
<http://www.hikvisioneurope.com/portal/?dir=portal>, Accessed:
28-05-2021.
- [21] *Tftp automatic update tool user manual*,
<https://www.hikvision.com/UploadFile/File/201433116851896.pdf>,
Accessed: 11-05-2021, Hikvision Europe B.V.
- [22] *Search tool - user manual for win & mac*,
<http://www.herospeed.net/en/ver//Manual/Search%20Tool/Search%20Tool%20User%20Manual%20for%20Win%20%20Mac.pdf>, Accessed:
11-05-2021, Herospeed Technology Limited.
- [23] *Camera firmware update manager*,
<https://viakoo.zendesk.com/hc/en-us/articles/360010814132-Camera-Firmware-Update-Manager-Overview>, Accessed: 11-05-2021,
Viakoo, Inc.
- [24] Electric Sheep Fencing LLC. "Take a tour of pfsense." Accessed: 14-05-2021. (2021), [Online]. Available: <https://www.pfsense.org/about-pfsense/>.