Bachelor of Science in Computer Science
February 2017

Implications of vulnerable internet connected smart home devices

Felix Hellman
Pierre Hellmann

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science. The thesis is equivalent to 10 weeks of full time studies.

Contact Information:
Author(s):
Felix Hellman
E-mail: feha15@student.bth.se
Pierre Hellmann
E-mail: pihe15@student.bth.se

University advisor:
University Lector Hans Tap
Department of Creative Technologies

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background. With the rise of Internet of Things and Internet connected devices many things become convenient and efficient but these products also carry risks. Even though a lot of people own devices like this not so many think of the consequences if these devices aren't secure.
Objectives. Given this our thesis aims to discover the implications of vulnerable devices and also at what rate there are insecure, unpatched devices compared to the patched, secure counterpart.
Methods. The approach implemented uses Shodan to find these devices on the internet and also to find version information about each device. After the devices are found the objective is to calculate a CVSS score on the vulnerabilities and the exploit that can abuse the vulnerability, if there exists any.
Results. What we found was that 71.85% of a smart home server brand was running an insecure version. As to the consequences of having an insecure device, it can be severe.
Conclusions. We found that, for instance, an attacker can without much difficulty shut off alarms in your smart home and then proceed to break into your house.

Keywords: Vulnerability; Shodan; Internet of Things (IoT); Patching

Acknowledgments

We want to thank Mikael Lagström of Truesec for inviting us to do our thesis work under the supervision of Truesec. We also want to thank Davide & Erik at Truesec for helping us determine some of the more complicated real world impacts of some vulnerable devices. Finally we want to thank our supervisor Hans Tap for giving valuable feedback and good ideas about our work during the whole project.

Contents

Abstract i

Acknowledgments ii

1 Introduction 1
1.1 Research Questions . . . 3

2 Related Work 4

3 Method 6
3.1 CVSS . . . 9

4 Results 12
4.1 Loxone Denial of Service . . . 12
4.2 Loxone Phishing attack . . . 12
4.3 AGEFO Custom Firmware Engineering . . . 14
4.4 PowerLogic Scanning . . . 14
4.5 C4Max Scanning . . . 14

5 Analysis and Discussion 16
5.1 Tracking cars . . . 16
5.2 Disable alarms and sensors . . . 16
5.3 Reading power information . . . 17
5.4 Finding out if anyone is home . . . 17
5.5 Unpatched . . . 17
5.6 Method Discussion . . . 19

6 Conclusions and Future Work 20

Bibliography 21

Chapter 1

Introduction

Security in internet connected devices is important because if there is poor security in these devices that get deployed on a massive scale they can be exploited in the same scale as they are deployed. Between January and October 2014 there were more than 5 billion downloaded Android apps that was vulnerable to remote attacks and 96% of mobile malware is targeted against Android [1]. We investigated security flaws in internet connected devices for everyday use under the following categories; smart home servers, internet connected cars and smart electric meters. To find out what security flaws had what real world implications of a potential cyber or cyber assisted attack we also wanted to investigate the update frequency of the chosen devices and compare them with other market segments.

The first category we investigated was cars connected to the internet to enable new features, like remote tracking. A connected car is a car that is capable of being connected to the internet which allows the car to be able to receive and transmit information[2]. Estimates show that more than 50% of new cars in 2019 will be smart, internet connected. And by 2020 they expect this to have grown to at least 75%[3]. Other estimates show that not only new, but all cars in 2025 will be connected to the internet[4].

The second category we investigated was Smart meters. Smart meters are electrical meters capable of measuring real time electrical information at a greater detail than non-smart electrical meters. Furthermore, they allow communication and also to be measured and controlled remotely. One use of this is to make billing customers for the correct amount of electricity easier [5]. In the EU, use of smart meters are quite common e.g. in Sweden. It already has a wide-spread deployment, which is defined by over 80% of electrical meters being smart meters. In the same data analysis it's estimated that the lifetime of smart meters is expected to be about ten years[6]. Smart meters also enables the central system to send data and instructions to the smart meter.

The third and final category we investigated was smart home devices, which are IoT devices brought to the home to enable new functionality and to automate simple tasks, an example layout can be seen in figure 1.1. Smart homes are computer systems that uses sensors and artificial intelligence to draw conclusions about the state of a home [7]. A Smart home server is the central hub in such a system which allows different types of smart home devices to communicate with each other. A 2016 survey show that 67.5% of the population in the USA have a general idea about what smart homes are but 30% don't think that there is a risk for loss of control[8].

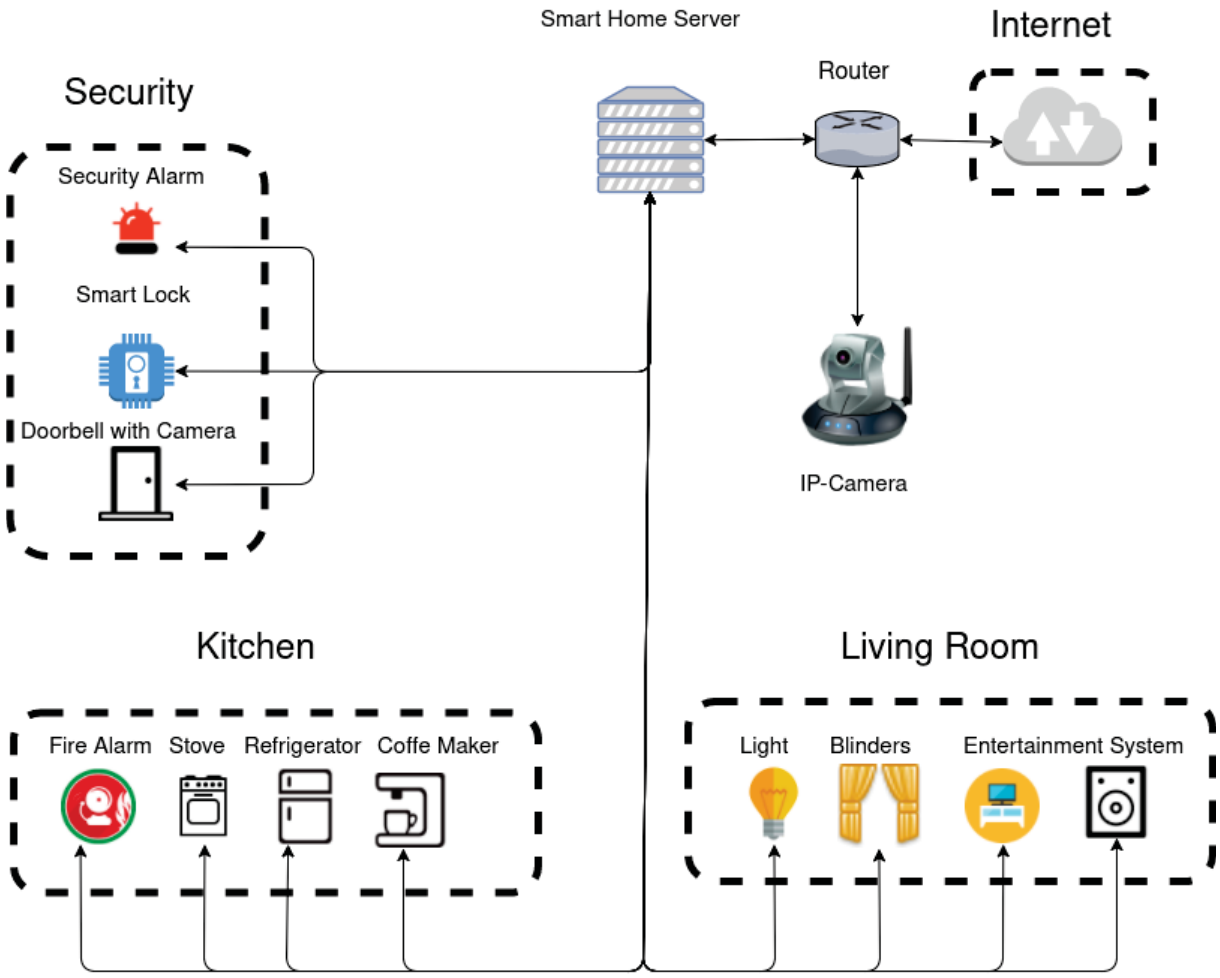

Figure 1.1: Example of Smart home topology

After implementing the method to find vulnerable devices, the following devices was found:

- C4Max smartbox, a device which can be connected to almost any car in order to give it internet and GPS access.
- Loxone smart home server, a centralized hardware server placed inside a home which controls smart home devices and receives signals from sensors in the smart home.
- AGFEO smart home server, similar use case to the Loxone.
- Smart electric meters from Schneier Electric, electric meters used to keep track of electrical usage to simplify billing.

1.1 Research Questions

In this paper we investigate the following questions:

- What real world implications do the selected vulnerable internet connected devices have?
- How frequent are unpatched internet connected devices compared to patched devices?

Chapter 2

Related Work

Whilst we try to touch upon the practical implications of smart home devices and statistics about currently used systems we need to highlight that others have done a very similar thing with medical devices by doing scans on Shodan and assessing targets with a vulnerability scanner [9], which is our main reason to not include medical devices. It's also important to know both the technological and legal challenges that arise when developing a smart home server, like developing aggregation schemes without trusted aggregators and developing specialized key management schemes. Without solving these issues, the whole infrastructure can be exploited leading to economic collapse and loss of lives[10].

When introducing IoT into the smart home it also introduces more risks and vulnerabilities both from the inside and the outside, like man in the middle attacks, information theft, code injection vulnerabilities to make devices behave in malicious ways and watching people inside their homes via their security cameras. One of the main issues with implementing strong security is that these smart home devices often have weak computational power and a low amount of storage making it unfeasible to use strong security standards [11].

With cars connected to the internet, cars can create networks with other vehicles to help direct the flow of traffic [3]. It's also possible to gain access to the airbag, warning lights and anti-theft system of a car by injecting CAN-BUS messages [12]. Which is something we are able to do with the vulnerabilities we found on the C4Max. This can result in a loss of life if the air bag system is activated whilst the driver is going down a highway.

A potential problem with smart meters is the privacy leakage from smart meter readings. Yanan Sun et al. have proposed a cost effective solution to avoid privacy leakage from smart meters. This was done with the use of existing thermal appliances and energy storage units for household load hiding[13].

In our discussion, we speculate upon why people don't patch their systems. In some cases it's because there are no patches available, but sometimes it's based on a lack of trust from the users of the system [14]. Since 96% of all mobile malware is targeted to Android we compare the patching statistics of iOS and Android mobile systems to see if newer versions of software correlates with less targeted malware.[1][15][16]

Chapter 3

Method

To find information about what searches to do we read full-disclosures from seclists. Seclists is a forum where a discussion of vulnerabilities and exploitation techniques can be held. It is also public and vendor neutral which is important for us[17]. In Figure 3.1 the work flow of the method is displayed.

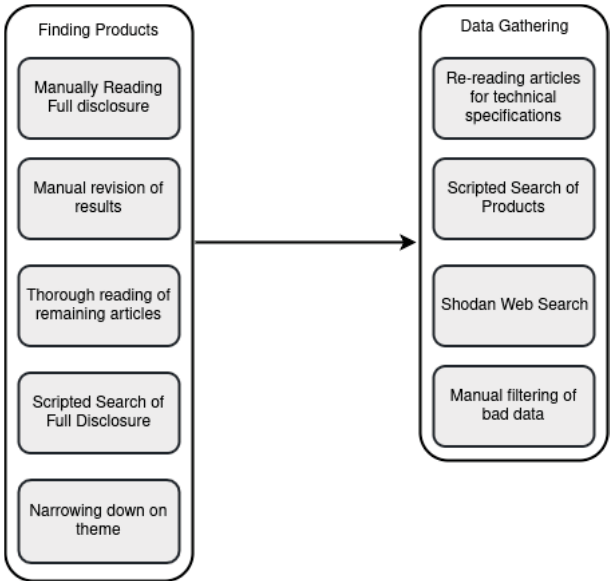

Figure 3.1: Work flow in the method

We started our search from the year 2010 and forward, but there was over 20.000 entries to cover so in order to not miss information we developed a search script capable of filtering headlines by keywords. The different keywords came from manually reading full-disclosures. Some other keywords used was "Smart", "Home", "IoT", "Smart meter", "Scada" and also more specific words like the company names that developed products that fit our criteria. These search words were chosen to get us the most amount of interesting and relevant products while still removing software only related vulnerabilities since we want to focus on the devices. This path led us down to 510 entries which we then manually vetted by doing initial reads of them to conclude if they: A) had the right information, B) was related to a specific product in

the chosen category. This manual revision led us down to 310 results. This set was filtered down to a final seven by enforcing our more strict theme of only home related devices with cars included. Out of these seven, two was the same product and another two yielded no results. This led us to the final four products to investigate. After finding the products we did a more thorough read through the full-disclosures from every product, what vulnerability's there were, how they worked, what could be done by exploiting them. Investigations into the products themselves was also done to see how they worked and where they were being sold.

To find devices on the internet, we used Shodan. Shodan is a search engine for internet connected devices with a billion-record database. Access to Shodan can be done with their web interface or API. Finding devices on Shodan can be done via multiple filters, you can filter on plain text responses, headers, ports, ip address and more. In figure 3.2 there's a picture of the Shodan web interface, and some of the data shown by a free text search

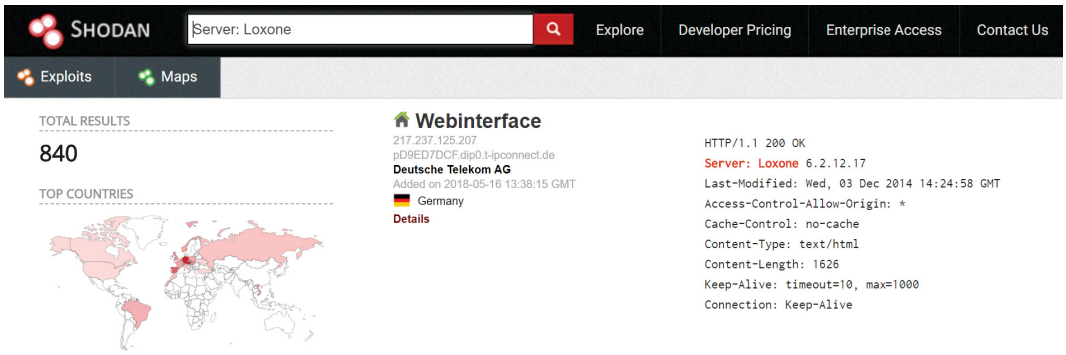

Figure 3.2: Search on Shodan.io

Finding products was done by crafting search queries using information about the products like, response text, ports, headers, protocols and encryption key fingerprints see Table 3.1. One source already included a search query for how to identify the device on Shodan. Searching was done in an automated fashion using a python script that could fetch all of the data in one go with the help of the API.

Device	Query	Type of Data
Loxone Smart Home Server	Server: Loxone	HTTP headers
AGFEO	ssl.cert.serial:"10293758115057549292"	HTTP headers, SSL Certificate
C4MAX	port:23 gps "on console"[18]	TCP Telnet
Powerlogic Smart Meter	PowerLogic ION	HTTP Response Text

Table 3.1: Shodan Search Queries

Example, script used for finding Loxone version statistics LoxoneSearch.py, all code can be found at github.com/felix-hellman/shodansearch

```
import Auth
import shodan
import re

api = shodan.Shodan(Auth.KEY)

Header = "Loxone Smart Home Server\nIp\tVersion Number\n"
SearchString = 'Server: Loxone'
FileName = "LoxoneSearchResult"

class LoxoneResult:
    version = ""
    ip = ""
    def printFormat(self):
        return str('' + self.ip + "\t" + self.version + "\n")


def findDigit(inputString):
    digit = ['0','1','2','3','4','5','6','7','8','9']
    for x in range(0,len(inputString)-1):
        for d in digit:
            if inputString[x] is d:
                return x

def Search():
    results = ''
    resultList = []
    try:
        results = api.search(SearchString,page=1)
        pages = int((results['total'])/100)+1
        for x in range(1,pages+1):
            results = api.search(SearchString,page=x)
            for result in results['matches']:
                r = LoxoneResult()
                r.ip = result['ip_str']
                serverIndex = result['data'].find('Server:',0,len(result['data']))
                endIndex = result['data'].find("\n",serverIndex,len(result['data']
                stringsearch = result['data'][serverIndex:endIndex:]
                search = findDigit(stringsearch)
```

```
                r.version = stringsearch[search::]
                resultList.append(r)

    except (shodan.APIError, e):
        print ('Error: %s' % e)
    return resultList
```

After searching and gathering results, some manual vetting had to be done to ensure that the data was correct. This was done to verify the data set. Only the Loxone data set had shortcomings with no version number on some entries, this due to false positives of the search. The invalid data was removed from the data set.

By doing a case study on the Loxone server we could see to what degree Loxone home servers were patched.

3.1 CVSS

For every vulnerability and corresponding exploit that can be applied on that vulnerability we have calculated a score with the help of common vulnerability scoring system (CVSS). CVSS is used in academia, industry and tool vendors [19]. The scoring system consists of eight different variables as seen in Figure 3.1. Depending on the choices on these eight variables you get a score between 1.0-10.0, each of these scores also belong to a written scale. 1.0-3.9 is Low, 4.0-6.9 is Medium, 7.0-8.9 is High and 9.0-10.0 is Critical.

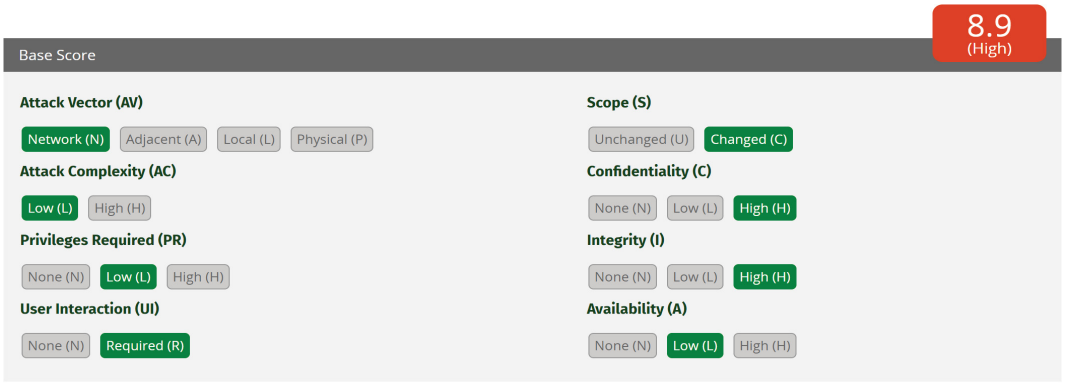

Figure 3.3: Example for calculating a CVSS [20]

Evaluating each vulnerability and an exploit that can be executed by entering the corresponding options on each of the variables in the calculator.

The variable attack vector (AV) means how the attacker can gain access to the product. Network (N) means that the product can be accessed via the OSI layer 3 (the network layer). This means that you should be able to access the vulnerable product via the internet. Adjacent (A) is that you need to share the same physical or logical network. This means that you should be able to access the vulnerable device via a WIFI, local IP network or via Bluetooth. Adjacent can't cross a router boundary either. Local (L) means that the vulnerable product is accessed via read/write/execute. It can also be that the attacker relies on the user to execute malicious files on the product. With Physical (P) access it means that the attacker needs to physically handle the vulnerable product.

Attack complexity (AC) describes conditions that the attacker can't control and that exists for the vulnerability to be exploited. Low (L) means that there is no such conditions and that the attacker can repeat the exploit again and again with the same results. For the complexity to be High (H) the exploit can't be executed at will. There has to be measurable time or effort invested into the preparation or execution of the exploit before a successful attack can be carried out.

Privileges required (PR) checks whether or not you need any privileges to carry out the attack. None (N) means that the attacker can be unauthorized prior to the attack and also that (s)he doesn't need access to settings or files. With Low (L) privileges that the attacker needs is to be an authorized basic user that can usually only affect setting and files owned by the user. High (H) means that the attacker needs to be an authorized user usually root or administrator that can affect most of the settings and files on the vulnerable system.

User interactions (UI) describe if the user needs to have any interaction in order for the attack to be successful. None (N) means that the attacker can exploit the vulnerability without any interactions from the user while Required (R) means that the user needs to have some kind of interaction in order for the attack to be successful.

Scope (S) of an attack is whether or not the attacker can impact more than just the vulnerable component. An Unchanged (U) scope means that the attack on the vulnerable component will only affect that component while a Changed (C) scope means that the attack on the vulnerable component can impact other components.

Confidentiality (C) refers to limiting the information access and disclosure to authorized users and to prevent access and disclosure to unauthorized users. So None (N) on confidentiality loss means that there is no confidentiality loss. Low (H) confidentiality loss means that there is some loss of confidentiality but the attacker has none or limited control over the kind and amount of the information. When there is High (H) there is total loss of confidentiality meaning that the attacker has access to all the information or only access to some restricted information but that this information leads to direct serious impact.

Integrity (I) refers to the trustworthiness and the veracity of the information. None (N) means that there is no loss to the information integrity. Low (L) means that the attacker can modify some data but can't control the consequences of the modification and that the modification doesn't have a serious, direct impact to the component. This however indicates that there is some integrity loss. High (H) means that there is a total loss of integrity because the attacker can modify any and all files protected by the component.

Availability (A) refers to the loss of availability of the component such as networked services, e.g. Email, website, database. But since availability also refers to accessibility to information, attacks that consume resources like network bandwidth, CPU cycles and disk space also impact this metric. None (N) means that there is no impact to the availability. Low (L) means that there is some loss of availability, for example the impacted components' resources are sometimes available or some of the resources are always available, but the attacker can't completely deny access to the component. High (H) means that there is a total loss of availability meaning the attacker is able to fully deny access to the resources in the component.

Chapter 4

Results

Using Shodan we found devices which are proven to be vulnerable with proof of concept exploits in the full-disclosures. Presented below is each device and how severe the exploit is via a calculated CVSS score.

We wanted to investigate patching status of the different devices but we only found resulsts for this in our searches for Loxone. Which also turned out be our second largest population after C4Max, which doesn't even have the availability of any patch.

4.1 Loxone Denial of Service

The DoS attack on the Loxone smart home server is available through network access and the attack itself is not complex. The attacker doesn't need to have any privileges or user interaction in order to execute the exploit. The scope is changed because if the smart home server is DoS attacked the sensors are in turn compromised. There is no loss of confidentiality with this exploit since you don't actually breach the product or system. The integrity of the data is also intact since the attacker can't access the system. There is a total loss of availability when executing a DoS attack. In figure 4.1 you can see the statistics from our Shodan search on the Loxone smart home server. It displays the percentage of devices that run patch 6.4.5.12 or newer and that run patches older than 6.4.5.12. [21] shows that this exploit is possible and how it is done.

4.2 Loxone Phishing attack

In order to control the smart home server the attacker needs to make the user click a URL link in a forum, e-mail, etc. It's not a very complicated attack. Since the attacker can attack remotely, the attack vector is Network. The attacker doesn't need any privileges but as former stated the user do have to click a link so there is interaction required from the user. The scope is also changed as the attacker can access more than just the smart home server. Here there is a total loss of confidentiality and

integrity since the attacker has access to the system. The attacker could also shut the user out since (s)he has total access to the system resulting in loss of availability. Since this is the same product as discussed before the same statistics apply as in figure 4.1. [21] shows that this exploit is possible and how it is done.

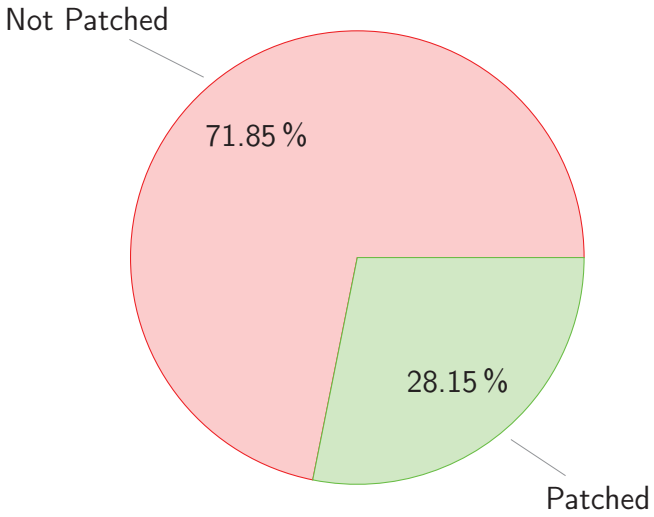

Figure 4.1: Loxone, Sample size of 700

4.3 AGEFO Custom Firmware Engineering

The exploit for the AGEFO smart home server requires the attacker to write custom code for the exploit. This attack can be carried over remotely so the attack vector is network, but the complexity is high. The attacker doesn't need any privileges or need any user interactions. Since this is a server that talks to other devices the scope has changed and the confidentiality loss is high since the attacker will gain access to the entire system. Both the integrity loss and availability loss is high. This is because the attacker has access to the system and could shut the user out of it and when (s)he has access to everything the user can't trust the data anymore. [22] shows that this exploit is possible and how it is done.

4.4 PowerLogic Scanning

Reading PowerLogic smart meter can be done remotely and the complexity is quite low. The attacker doesn't need any privileges or any user interaction for this to be possible. Since everything the attacker can access is what's on the smart meter the scope is unchanged. The confidentiality loss is high since the attacker can access everything that the smart meter has access to. The integrity loss is low because the attacker can only change the settings in the meter and therefore misrepresent the data. The attacker can't limit the availability with this vulnerability so there isn't any availability loss. [23] shows that this exploit is possible and how it is done.

4.5 C4Max Scanning

It's possible to execute this exploit remotely leading to the attack vector network and the exploit has low complexity. The attacker doesn't need any privileges or any user interactions. The scope is unchanged with this exploit. There is high confidentiality loss and high integrity loss because the attacker gains access to all the C4Max's data and (might be able to remove data). Though there is no availability loss, the attacker can't limit or remove access to the product. [18] shows that this exploit is possible and how it is done.

Table 4.1 shows the results from all the Shodan searches. The different results are: the number of found devices from all our products, the CVSS score that we calculated for each vulnerability and corresponding exploit and the different exploits that could be run on the vulnerable product.

Category	Products	Exploit	CVSS	CVSS Ranking	Number of devices
Smart home server	Loxone	DoS	8.6	High	700
Smart home server	Loxone	Phishing attack	9.6	Critical	700
Smart home server	AGEFO	Custom Firmware	10.0	Critical	120
Smart meter	PowerLogic	Scanning	8.2	High	14
Internet connected car	C4Max	Scanning	9.1	Critical	963

Table 4.1: Result from Shodan with CVSS score

In 4.2 the different inputs, for calculating the CVSS, for each vulnerability and corresponding exploit is comprised.

Product	AV	AC	PR	UI	Scope	Conf	Int	Avail
Loxone DoS	Network	Low	None	None	Changed	None	None	High
Loxone Phishing	Network	Low	None	Required	Changed	High	High	High
AGEFO	Network	Low	None	None	Changed	High	High	High
PowerLogic	Network	Low	None	None	Unchanged	High	Low	None
C4Max	Network	Low	None	None	Unchanged	High	High	None

Table 4.2: Result from Shodan with CVSS score

Chapter 5

Analysis and Discussion

After analyzing the results from the Shodan searches and reading about the different vulnerabilities and how they work, we found that the following scenarios are possible.

5.1 Tracking cars

Tracking cars locations and storing information in a database to build a record of routes can be done with a simple python script. Connect over port 23 to an ip address decided by a list of results. Execute the command to read out the GPS and store the information. Disconnect from the unit and repeat the process.

With this information an attacker could see when the car is away from the home and therefor when the owner isn't home, using this to break into homes. An attacker could also use this information to follow people or collect data over time to see patterns of the owner. With access to the CAN-BUS, as the C4Max has, it could be possible to remotely deploy the airbags, doing this when a car is on the highway can cause catastrophic damage and loss of life. One can imagine this on a bigger scale with 100 cars or even 1000 cars.

5.2 Disable alarms and sensors

Since there is a DOS vulnerability in the Loxone home server, it can easily be disabled which in turn also renders the alarm systems and sensors useless. The DOS attack can be carried out over network via a syn-flood which is very easy to perform.

This is one of the worst scenarios we found. There are many consequences to this, one is being able to just disable alarms and walk into a house when the owner isn't home. Another is to disable fire alarms and set fire to the house without the owner noticing. One not as serious is disabling the sensor and in turn locking the owner out of his/her's home.

5.3 Reading power information

With the vulnerability in the smart meters it allows the attacker to go into the smart meter and read the data that it has gathered. It also allows the attacker to change settings about the data that the smart meter is gathering. This can skewer the data that the company is gathering and the data that the customer might look at.

This scenario isn't as serious as many of the others. But an attacker could gather information from the smart meter and analyze it to maybe find out when the owner isn't home and then break into the home. The vulnerability itself won't help the attacker break in but can give information of the owners habits.

5.4 Finding out if anyone is home

After performing a social engineering attack, as described before, it's not only possible to control all connected devices; it's also possible to see which devices are currently being used. This can be exploited to find out when anyone is at home.

With this vulnerability the attacker can do everything (s)he can to with the DoS attack and more. Being able to freely start connected devices also gives the ability to start the stove in the middle of the night while turning of the fire alarm, all this while the owner is away, or even worse when the owner is at home.

5.5 Unpatched

The Loxone results show that approximately 72% of the devices running insecure versions (earlier than 6.4.5.12). This was surprising, that a strong majority was running three year old insecure versions. If you look at Verizon's 2015 data breach investigation they report that 71% of attacks that exploit vulnerabilities had a patch released over one year ago and that the majority had a patch released within months prior to the attacks[1]. This shows that most of the exploits on vulnerable system are carried out on systems with vulnerabilities that have had patches ready for at least a year.

The Verizon 2016 Data breach investigation shows that in 2015 the most exploited vulnerabilities are from 2007; this means that the vulnerabilities at the time was eight years old[24]. This also shows that it is not the zero day exploits that are the ones to be worried about; it's the old ones with users that don't patch. In Verizon's 2015 data breach investigation report they found that 99.9% of all exploited vulnerabilities were compromised more than a year after the CVE was released[1]. All this data points to how big a problem unpatched systems with vulnerabilities are and it also suggests that installing new patches as they come along can greatly decrease

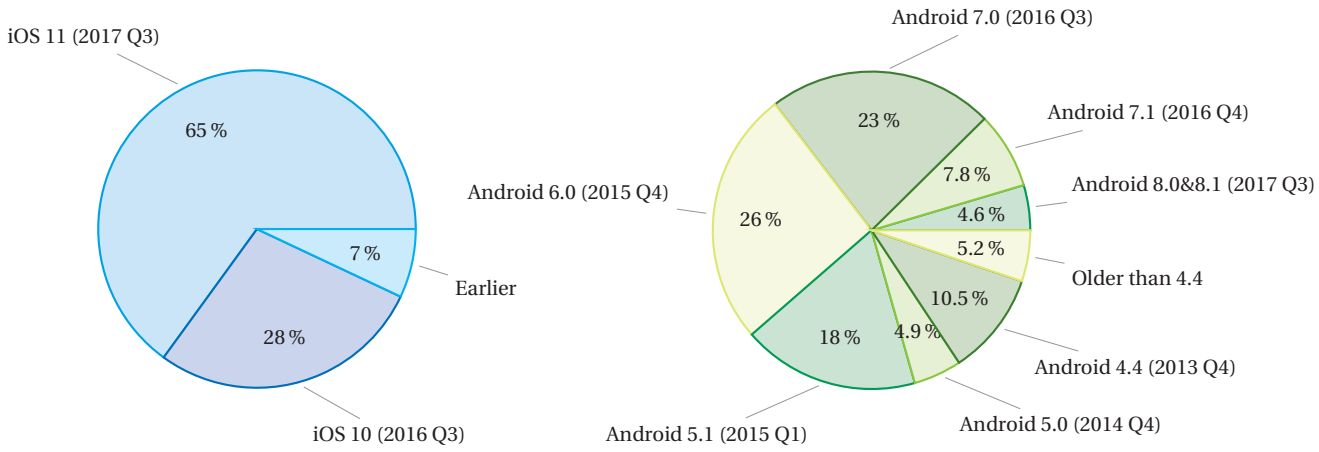

Figure 5.1: iOS[16] and Android[15] user version statistics

the chances of your systems getting exploited by a vulnerability in that system.

One question to ask is why are there so many unpatched systems if there are patches readily available? One answer to this can be that users can be confused and annoyed with update messages and that users have a negative attitude against software and update messages[14]. Another similar answer is that users don't install patches because they can't really tell if it's worth installing the patch due to not knowing what it does, confused about why the patch is necessary and that the program already works so why install a patch[25].

If you look at patching in the mobile phone sector we can see that it differs immensely between the biggest two; Android and iPhone. In Fig 5.1 we can see that the majority of iPhone users is running iOS 11 that was released less than one year ago. You can also see that the majority of Android users is running Android 6.0 or older; a version being at least three years old. Then looking at the data stated earlier, 96% of mobile malware is targeted against Android, we might see why. When the majority of the users are running older less secure versions, it is more profitable to craft malware and exploits for these versions. This is connected with what we found about Loxone, an over three year old version that is exploitable and it's still being used. When security professionals find vulnerabilities they contact the company to give them time to release a patch to fix this. But after the patch is released they disclose this information to the public, giving attackers information to craft exploits and sometimes offer already working exploits for these vulnerabilities. This raises the risk of being exploited when running an old version even higher since now attackers might not even have to craft their own exploits making it easier.

One thing that we noticed is that iOS is developing all their own hardware and software, this means that they can test their patches on all the phones. Android on the other hand is manufactured by Google and although they do have some hardware there are countless other devices that run Android. Meaning that it's up to the hardware developers to test the patches. If we compare this to our devices, we noticed that the same manufacturer did the hardware and the software for the device. So even though they can test their software with their hardware there is still a lot of vulnerabilities.

5.6 Method Discussion

As with all methods there are pros and cons, things that could be done differently. Some of the weaknesses in the method we used is that the connected cars aren't always connected, this means that depending on time of day or night the number of connected cars can differ. This can change the results if someone where to do the search at a later date. This is something to consider when using Shodan.

Another thing to point out is that the vulnerabilities all came from one source, meaning that there can be vulnerabilities that would have fit our criteria and search parameters but weren't listed in our source. But including several sources for vulnerabilities also takes time due to writing new scripts for the sites and manually going trough results to remove false positives. The good thing with using the one source for finding our vulnerabilities is that we could develop a script for the site. Meaning that we wouldn't miss any vulnerabilities on the site.

One thing we could have done differently is to use a vulnerability scanner, like Nessus as done by E. McMahon et al. [9], to find the vulnerabilities on the different products. This is done by feeding the IP addresses of the vulnerable products to the scanner resulting in getting the vulnerabilities for a product faster. But using a good scanner is often expensive and by going through the full disclosures we got a better understanding of how the products work and what these vulnerabilities could result in.

What we thought about early in the work with this method was honeypots, what we meant is that some of the results that we are getting could be honeypots. There isn't much we can do to remove these results without actually hacking the devices to find out if they are honeypots. With a vulnerability scanner we might find the honeypots since they find all vulnerabilities on an IP address. So if a IP has several vulnerabilities this might indicate that its an honeypot. We did some research and from what we could find there wasn't any honeypots for our devices, this however doesn't mean that there aren't any or that someone hasn't made his or her own.

Chapter 6

Conclusions and Future Work

This thesis aimed to find the real world implications of vulnerable internet connected devices and also find the frequency of vulnerable device compared to patched devices. The results from our study and from the data from Shodan shows that there are some serious real world implications that come with these vulnerabilities. One example of this being; disabling the alarms and sensors in a smart home and thus being able to carry out cyber assisted crimes. We could also find out that on the Loxone products there were 71.85% running a three year old insecure version and that 28.15% running a patched secure version.

For future work there are several interesting directions. First, research how to solve the problem with people running several years old patches that are insecure and make the system vulnerable to attacks. Maybe look into other industries and research how they solve patching problem, if they do. Second, conduct a similar thesis with different products or with software vulnerabilities and see if they get similar results.More research in this would bring attention to this area, and could lead to solving the patching security flaw.

Bibliography

[1] Verizon. *2015 Data Breach Investigations Report.* 2015.

[2] Melanie Swan. Connected car: Quantified self becomes quantified car. *Journal of Sensor and Actuator Networks*, 4(1):2–29, Feb 2015.

[3] Riccardo Coppola and Maurizio Morisio. Connected car: Technologies, issues, future trends. *ACM Comput. Surv.*, 49(3):46:1–46:36, October 2016.

[4] GSMA CONNECTED LIVING. 2025 every car connected: Forecasting the growth and opportunity'. *GSMA, white paper*, 2012.

[5] Guido Pepermans. Valuing smart meters. *Energy Economics*, 45:280 – 294, 2014.

[6] EU. Deployment statistics. `http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1403084595595&uri=SWD:2014:188:FIN%20`, 2014. Accessed: 2018-04-23.

[7] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan. Casas: A smart home in a box. *Computer*, 46(7):62–69, July 2013.

[8] Charlie Wilson, Tom Hargreaves, and Richard Hauxwell-Baldwin. Benefits and risks of smart home technologies. *Energy Policy*, 103:72 – 83, 2017.

[9] E. McMahon, R. Williams, M. El, S. Samtani, M. Patton, and H. Chen. Assessing medical device vulnerabilities on the internet of things. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 176–178, July 2017.

[10] N. Komninos, E. Philippou, and A. Pitsillides. Survey in smart grid and smart home security: Issues, challenges and countermeasures. *IEEE Communications Surveys Tutorials*, 16(4):1933–1954, Fourthquarter 2014.

[11] Bako Ali and Ali Ismail Awad. Cyber and physical security vulnerability assessment for iot-based smart homes. *Sensors*, 18(3), 2018.

[12] Pierre Kleberger. *On Securing the Connected Car Methods and Protocols for Secure Vehicle Diagnostics.* PhD thesis, 2015. Accessed 2018-04-19.

[13] Y. Sun, L. Lampe, and V. W. S. Wong. Smart meter privacy: Exploiting the potential of household energy storage units. *IEEE Internet of Things Journal*, 5(1):69–78, Feb 2018.

[14] Michael Fagan, Mohammad Maifi Hasan Khan, and Ross Buck. A study of users' experiences and beliefs about software update messages. *Computers in Human Behavior*, 51:504 – 519, 2015.

[15] Google. Userstats. `https://developer.android.com/about/dashboards/index.html`, 2018. Accessed: 2018-04-19.

[16] Apple. Userstats. `https://developer.apple.com/support/app-store/`, 2018. Accessed: 2018-04-19.

[17] Full Disclosure. Full disclosure. `http://seclists.org/fulldisclosure/`, 2018. Accessed: 2018-04-16.

[18] Jose Carlos Norte. Hacking industrial vehicles from the internet. `http://jcarlosnorte.com/security/2016/03/06/hacking-tachographs-from-the-internets.html`, 2016. Accessed 2018-04-19.

[19] P. Johnson, R. Lagerstrom, M. Ekstedt, and U. Franke. Can the common vulnerability scoring system be trusted? a bayesian analysis. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2017.

[20] Cvss Calculator. `https://www.first.org/cvss/calculator/3.0`, 2018. Accessed: 2018-04-19.

[21] Johannes Greil. Sec consult sa-20150514-0 :: Multiple vulnerabilities in loxone smart home (part 2). `http://seclists.org/fulldisclosure/2015/May/55`, 2015. Accessed 2018-04-19.

[22] T. Weber. Sec consult sa-20170712-0 :: Multiple critical vulnerabilities in agfeo smart home es 5xx/6xx products. `http://seclists.org/fulldisclosure/2017/Jul/17`, 2017. Accessed: 2018-04-19.

[23] Karn Ganeshen. Multiple vulnerabilities - powerlogic/schneider electric ionxxxx series smart meters. `http://seclists.org/fulldisclosure/2016/Sep/8`, 2016. Accessed 2018-04-19.

[24] Verizon. *2016 Data Breach Investigations Report*. 2016.

[25] Kami E. Vaniea, Emilee Rader, and Rick Wash. Betrayed by updates: How negative experiences affect future security. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 2671–2674, New York, NY, USA, 2014. ACM.