Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Spyware detection based on network traffic

TDK

Készítette
György Péter

Konzulens
dr. Holczer Tamás

October 28, 2019

Contents

Kivonat

A mai világban, ahol az információ egyre nagyobb érték, és szinte mindennap hallani különböző adatlopásokról, fontos feladat megvédeni az adatainkat. Ezek az adatok értéket képviselnek a támadók számára is, és különböző módokon próbálják ezeket megszerezni. Az egyik ilyen módszer, hogy kémprogramot telepítenek a kiszemelt áldozat gépére, vagy becsapással ráveszik az áldozatot, hogy önmaga telepítse fel a gépre a kémprogramot, amit akár egy hasznos programnak is álcázhatnak. A kémprogramok naplózhatják a billentyűket amiket lenyomunk, képeket készíthetnek a képernyőnkről, lekövethetik az egérmozgásunkat, és akár még a mikrofonon és a kamerán keresztül képet is rögzíthetnek rólunk.

A dolgozatban a fő hangsúlyt az úgynevezett keylogger detekcióra fektettem. Ezek közül is azokra a megoldásokra, amik egy távoli szerverre töltik fel a rólunk készült logokat. A keylogger detekció a leggyakrabban magán a host gépen történik. Munkám során egy olyan programot készítettem el, amivel a hálózati forgalom alapján tudom megmondani, hogy van-e az adott gépen keylogger. Ennek az az előnye, hogy egy nagy hálózat esetén nem kell külön minden gépre telepíteni egy szoftvert, hanem a kifele menő forgalmat tükrözzük egy olyan gépre, ahol fut az analizáló. Több ingyenes és fizetős keyloggert is kipróbáltam, különböző beállításokkal, és készítettem egy sajátot is, hogy olyan beállításokat is kipróbálhassak, amik nem voltak elérhetűek a korábban kipróbált programokban.

A méréseket virtuális gépek segítségével végeztem el. Az eredmények alapján, függetlenül a keylogger gyártójától, ha periódikusan küldte el a szervernek a rólunk készült adatokat, azt rövid időn belül tudtam detektálni. Ugyanerre az eredményre jutottam olyan keyloggereknél is, amik bizonyos mennyiségű lenyomott karakter után küldik fel az adatokat. Ez a két kategória lefedi az elérhető keyloggerek döntő többségét.

Abstract

In today's world where information is becoming more valuable than ever, we can hear of data breaches every day. It is becoming an important task to protect our data which is highly valuable for the attackers, and they try to get these using different methods. One of this method is using spyware. The spyware can be installed many ways, one frequently used method is phishing, where the attacker sends an email to the victim promoting the spyware as a legit program that will help him. These spyware can log every key we press, take screenshots of our desktop, track our mouse movement and even can gather data about the outside world using our microphone and camera.

In this research, I focused mostly on keyloggers, especially on those which sends the logs to a remote server. Keylogger detection is usually implemented on the host itself, I tried to come up with a proof of concept solution that can detect keyloggers from network traffic. This way it is much easier to detect traffic in an enterprise network since we need to add only one server. Then tee the outgoing traffic into it, and then the analyzer can detect keyloggers from the traffic. We don't need to install a detector on every machine, so this is a much easier and scalable solution. I've tried freeware and proprietary software with different configurations and even created a keylogger for configurations, that wasn't implemented in the previously mentioned software.

I've used virtual machines for my research. The results showed that keyloggers that send the data periodically can be detected just very quickly (of course this depends on the period itself). Keyloggers that sends logs after every n character pressed is also detectable. These types of keyloggers are the most commonly used nowadays.

Chapter 1

Introduction

In today‘s world where information is getting valuable day by day, it is an important task to protect this information from the attackers. If the information falls in the wrong hand that can lead to harmful events. This is important for both companies and individuals. The companies can lose competition advantages and customer trust which leads to losing money. Individuals can leak out their credit card numbers, login credentials and also can be victims of identity theft.

Spying has been always part of our life, everyone wants to know more information about everybody else. Companies hire black hat hackers to help them steal data from rival companies. These hackers use various techniques to achieve their goals. These techniques include social engineering, phishing, and the use of spyware programs. To stop them we need strict security policies and anti-spyware programs to detect malicious activity.

There was already numerous famous spyware attack for example "CoolWebSearch" [12], "Gator" [13] or "Hot as Hell" [21]. Using spyware is often part of a bigger attack to gather information. For example, the Stuxnet-like malware called Duqu which was discovered by the CrySyS Lab also included one [5].

How much trouble can spyware do? To answer this question just look at the previously mentioned famous spyware. "Hot as Hell" was in the dial-up internet era. It dialed toll numbers to subscribe to pornographic websites and even disconnected the user from the local internet provider, and connected to a more expensive international provider. "Gator" is one of the most famous adware program, which displayed advertisements based on the user‘s behavior. "Gator" is also famous for it‘s spreading technique, it was bundled in freeware software, "Kazaa" a popular file-sharing program also helped the spreading of "Gator". "CoolWebSearch" was used for browser hijacking, it messed up the user‘s Internet Explorer settings. We can see that spyware can cause financial damage, and can be annoying.

Keyloggers are a type of spyware, they are continuously logging our activity, they can monitor our keystrokes, mouse movement, create screenshots and even can record the surroundings of the computer using the webcam and microphone of our device. There are even hardware-based keyloggers, which are small devices installed between the keyboard and the computer. Nowadays there is also a huge market for wireless keyboards, but we

can also find numerous gadgets on web stores that are capable of capturing our keystrokes.

Keyloggers are not only used by hackers, but there are also examples of legal keyloggers that are used by parents for monitoring children and employers monitoring their employees. Freeware and proprietary keyloggers exist on the market as well.

Software-based keylogging works differently depending on the operating system we are using. There are keylogger solutions for Windows and Linux systems, and in the last few years, there has been a huge increase in keyloggers made for macOS.

Detecting and removing spyware is important but not a trivial task. The host-based keylogger detection mechanism is the most common. For example, on a Windows system, we can detect keyboard hooks, and eliminate bad processes. However, some games or macros also need to use keyboard hooks, which can result in false positives. Host-based detection has a huge disadvantage, which is scalability, we need to install it on every machine. Imagine a huge company with thousands of workstations, installing and maintaining software like this requires a lot of resources. As I mentioned earlier keyloggers can work differently based on the operation system, which means we need different host-based detectors for each os which increases the complexity of the maintenance.

I tried to come up with a solution to detect spyware based on network traffic. This way we don‘t have to install software on every computer, instead we need to copy the outgoing traffic into a machine that runs the analyzer. This approach has many advantages, it is easy to maintain, doesn't use the host computer's performance and it is placed on a single machine. However, this solution also has disadvantages one of the biggest is that it can only detect spyware that is logging to a remote server.

The proposed algorithm works only as an IDS, so it doesn‘t try to intercept or deny the malicious traffic, it just creates logs for further more examination.

The remainder of the paper is organized as follows: Chapter 1 is about spyware in general. In Chapter 2 I introduce the approaches used for spyware detection. In Chapter 3 we can see how the algorithm works on different attack scenarios, while in Chapter 4 I will conclude my work.

Chapter 2

Spyware

2.1 Introduction to spyware

The term spyware has been with us for a long time. The first recorded use of the term spyware occurred on October 16, 1995, in a Usenet post that poked fun at Microsoft's business model [20]. Spyware at first denoted software meant for espionage purposes. However, in early 2000 the founder of Zone Labs, Gregor Freund, used the term in a press release for the ZoneAlarm Personal Firewall. Later in 2000, a parent using ZoneAlarm was alerted to the fact that "Reader Rabbit," educational software marketed to children by the Mattel toy company, was surreptitiously sending data back to Mattel. Since then, "spyware" has taken on its present sense [22]. We can find multiple definitions for the term spyware online, but I found the following the most accurate:

Spyware is unwanted software that infiltrates your computing device, stealing your internet usage data and sensitive information. Spyware is classified as a type of malware - malicious software designed to gain access to or damage your computer, often without your knowledge. Spyware gathers your personal information and relays it to advertisers, data firms, or external users [7].

There is much spyware in the wild, with different working mechanisms, and different purposes. There is much spyware for targeted attacks and of course, there is much for gathering as much information as possible, from as many sources as possible. Some spyware is used by their developer to create revenue, some of them are sold on the dark web. Complex spyware can be very pricey, for example in 2018 a senior programmer at NSO who had access to the firm's development systems including its source code repositories, tried to sell the company's new surveillance software for $50 million [4].

Spyware is made for smartphones, workstations, but in recent years there have been some huge milestones in the evolution of spyware software with the appearance of spyware programs specifically designed for macOS.

In 2007 a survey showed that 90% of U.S. home computers have been infected with spyware at some time [2]. The following figure shows us how much money gets stolen, and how many identity thefts there are just in the United States [1]. This number has been decreased over time, however, it is still a very big threat. There is still much spyware

out there, according to a paper by Security Magazine one in 50 emails contains some form of malicious content. Digital Trends estimates that 10% of all malicious emails contain malware such as ransomware, spyware adware or trojans [8].

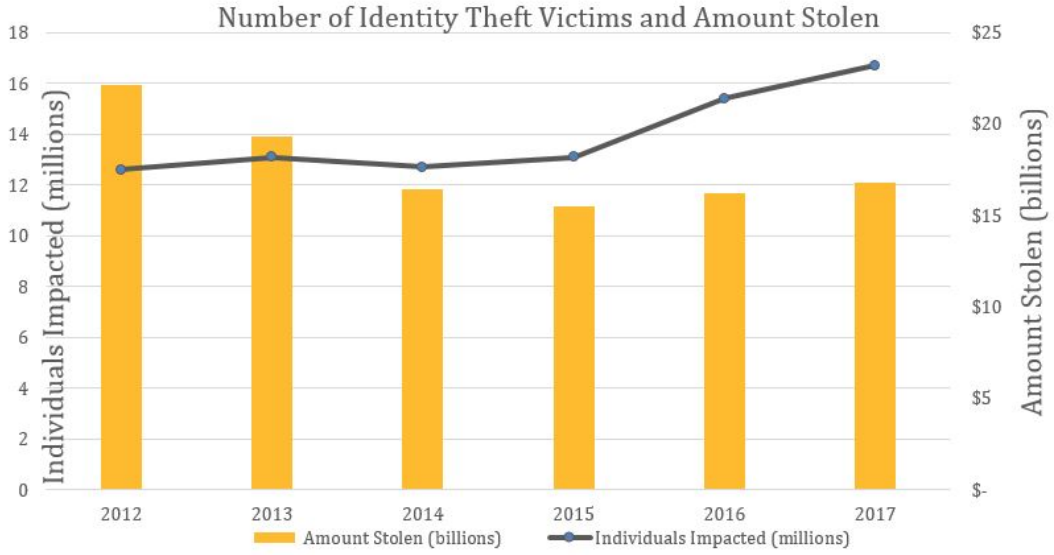

Figure 2.1: *Spyware casualities in the US.*

In recent years the number of spyware for mobile is increasing year by year. There is also commercial spyware which is legally downloadable surveillance software. In recent years there is a major increase in spyware made for smartphones. In 2019 there has been an issue, with the famous messaging application called "WhatsApp", where a vulnerability lets the attackers inject spyware on the phone. "WhatsApp" is used by 1.5 billion users worldwide. This spyware was called Pegasus and it was developed by an Israel-based cyber-intelligence company called NSO [18]. By some source, this is considered one of the biggest zero-day bugs in recent times. This vulnerability affected Android, iOS, Windows Phone, and Tizen.

There are many different kinds of spyware in the wild. They behave differently but mostly share the same purpose which is information theft and fund producing for the creator.

2.2 Classification of spyware

There are many different classifications for spyware. I found the list from Malwarebytes the most accurate, however, I think the list was missing an important item which is Adware [16]. In my opinion, spyware can be classified into 5 major categories.

- Password stealers
- Banking trojans
- Information stealers
- System monitors / Surveillance software

- Adware

2.2.1 Adware

Adware is a software that is made for gathering information about the victim, and then show customized ads to the user. This way the user is more likely to click on the ad. The information gathered by the adware can be the browsing history or the victim's browsing behavior.

Adware generates money for its creator on a per-ads-displayed or a per-click basis. They often come with shareware software bundled into the installer.

Some sources classify adware to PUP (potentially unwanted software). Adware gathers information without the victim's knowledge, so according to my point of view, adware is spyware.

Adware can be quiet or aggressive, quiet ones pop just a few ads, aggressive ones can overlay the screen with advertisements. In the following figure, we can see a rather aggressive adware. The picture was taken on an Apple computer running macOS, which shows not only Windows users are affected.

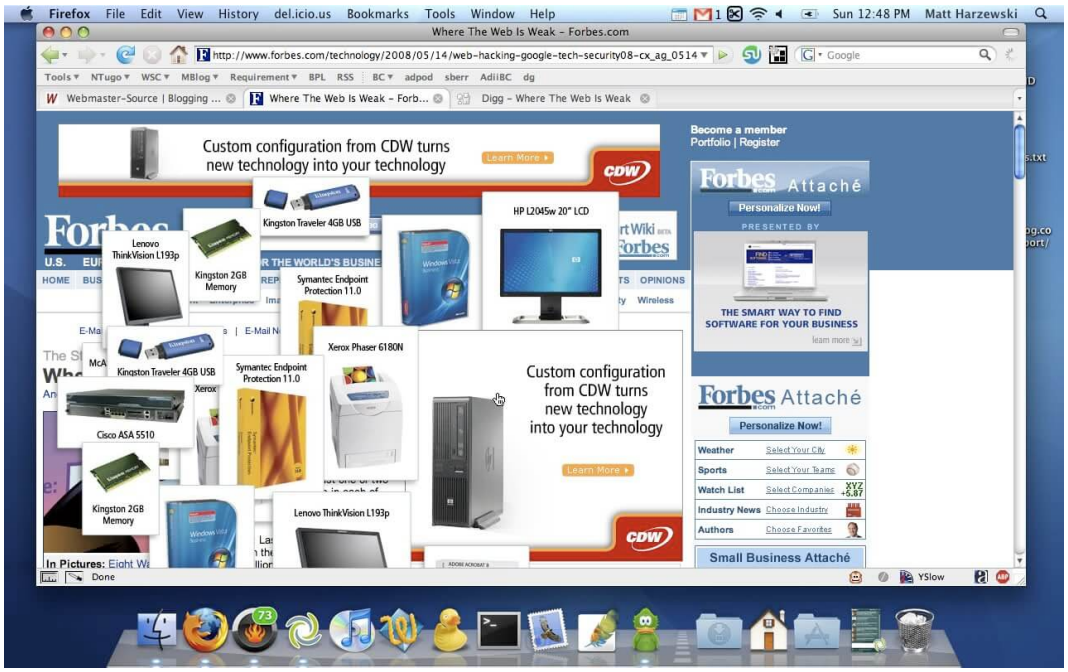

Figure 2.2: *Enourmous advertisement storm on a macOS.*

2.2.2 Password stealer

Password stealers are malicious programs, which are designed for recording usernames and passwords. The stolen credentials can be uploaded to a remote server or it can be stored on the infected machine for personal retrieval. Password stealers are considered dangerous because they can compromise email accounts and bank accounts. It can steal the saved passwords from the browser, from plaintext files or even from your password manager app if your vault is unlocked. Password managers can help against this type of attack, but it

isn't a complete solution, because the password manager can be vulnerable, and even if it is locked attackers may still gain access to the stored credentials.

The motivation behind password stealers is money as always. Password stealers can generate revenue by stealing credit card details, or they might let the attacker blackmail the victim. Password stealers also let the attacker to impersonate the victim online, which can lead to very high damages, for example, ruin personal relationships or getting the victim fired from their job.

In most cases, password stealers are installed with bundled shareware software.

2.2.3 Banking trojan

Banking trojans are specially designed to harvest bank account credentials or credit card details. Some of them are also capable of crafting or modifying transactions. This way the attacker may gain access to digital wallets and online financial portals. They are invisible to the victim and for the web application as well. They use vulnerabilities in browsers to manipulate the sites of official financial institutes. They are also capable of sending our credentials to the attacker's server.

This type of spyware exists because they can gather money very easily, and in a rapid way.

This spyware often spreads through emails.

2.2.4 Information stealers

Info stealers are similar to password stealers, with the big difference that they not only aim at getting the credentials from the user, but also gather documents, pictures, and other media files. They are very harmful because there is a huge chance it will find something confidential on our machine. It can save passwords and browsing history from the victim's browser, loop through every file on the victim's machine, and store confidential files. The gathered data can be uploaded to a remote server, or it can be placed somewhere on the victim's machine to let the attack retrieve it personally.

Info stealers can generate money by giving the attacker a chance to blackmail the victim, and also it can retrieve the victim's credit card credentials. The further damage it can cause is huge, if there is a picture of our identity card, it may let the attacker impersonate us, and this can result in identity theft.

This spyware can come with bundled freeware or shareware software, or it can arrive as a trojan.

2.2.5 System monitors / Surveillance software

This type of spyware is meant for giving the attacker a chance to gather information from the victim in real-time. It can record voice or video, log victim's keystrokes and their mouse movement. Not every surveillance software is malicious, we can find many legal system monitors, which are made for parents or employers to monitor children or employees. The legality of these software provides access to spyware software for people that don't have

knowledge in these fields, which makes attacks via spyware programs practical. Keyloggers are a type of this software which are used to capture keystrokes.

System monitors can generate revenue by stealing credentials. Legal surveillance software generates revenue based on purchases since they are mostly proprietary.

This software might be already installed on our workstation in case of legal use. When it comes to illegal data theft, in most cases it comes as a trojan, promoting itself a useful application.

2.3 What is spyware doing

There are many different kinds of spyware in the wild, with different purposes. However, they can be classified by behavior. Spyware is stealing our data and generates funds for the developer this way. There are different kinds of spyware that are not always easy to detect.

Adware is one very common type of spyware, not every adware can be considered as spyware, but those which gather information about the user's behavior and then generates advertisement designed for the user is spyware. It collects information without the user's consent, and deliver advertisements. Luckily these programs are more annoying than harmful.

Banking spyware is specially designed to steal credit card credentials.

Password stealers and Infostealers are a broad range of spyware which scans the user computer for log files, browser history, login credentials, documents, and other media files. They are also capable of uploading the stolen information to the attacker's remote server.

Keyloggers are created to log every activity the user takes, this includes mouse tracking, saving keystrokes, creating screenshots and even recording media through the computers, microphone, and webcam.

The spyware can store stolen information on the victim computer for personal retrieval or can upload it to the attacker's remote server.

There are also legal keyloggers, these are meant for monitoring employees or children, however, this software also can be used with malicious intent.

2.3.1 How spyware is spreading

There are many ways spyware can spread to our machine, this includes, emails, security vulnerabilities, phishing, trojans, and bundled with other software. Spreading is often based on user gullibility, in some cases when it comes to installing software using one or more zero-day vulnerability it is possible that the user doesn't have to do anything to be affected by the spyware.

2.3.2 Spreading via Email

This method is usually used by attackers to find possible victims. Nine out of ten cyberattacks start with a simple phishing email, and these emails are getting both more numerous

and more sophisticated [19]. 14.5 billion spam emails are sent out every day and it is estimated that 2-4% of all email traffic is malicious [6].

In this attack type, the attacker has to rely on the victim, where the attackers try to forge legitimate emails, with a malicious attachment. For example, the attacker disguises himself as a bank and sends a corrupted document, as soon as the victim opens the document becomes vulnerable to the attacker.

2.3.3 Spreading via vulnerabilities

This type of spreading in not so common, because it requires a higher knowledge, and in some cases an understanding of the victim's system. This kind of attack is harder to avoid, and in some cases, there is no need for user interaction.

Emotet famous spyware which was first detected in 2014, but some versions still exist and came back in September of 2019, used the EternalBlue/DoublePulsar vulnerabilities for spreading. This is the same vulnerability used by WannaCry and NotPetya [15].

2.3.4 Spreading via phishing

This method fully relies on the victim. Sending specially crafted emails can be considered one kind of these attacks. The main idea behind this is to get the victim, to install the malicious software on his/her device. Sometimes it requires social engineering, in some cases, it is enough if the attacker just lays down flash drives everywhere in the city, some people pick up those and are fool enough to put them inside their device and open documents, or programs which are copied on the flash drive.

2.3.5 Spreading via trojans

The malicious program disguises itself as useful software, but when the victim installs it on his/her device, it also performs spying alongside the promised functionality. This kind of malware is hard to detect and hard to remove. Deleting the program doesn't help in most cases because the spyware duplicates itself to different locations, and starts when the system starts. The famous spyware Campi was spreading this way and has stolen 500,000 online bank accounts.

2.3.6 Spreading via software bundles

The user is always in a rush when it comes to installing software the average user just wants the software already installed, rushes through the buttons always press next and expects the software to be installed. Well if the user is not prepared, can install spyware alongside the so wanted software. Many software comes with an installer that installs not only the software the user wanted but other software as well which can be spyware. The term bundleware is used for this kind of software.

2.4 Keyloggers

In this thesis, I am focusing on the detection of keyloggers so I introduce them in detail in the following section.

2.4.1 Introduction to keyloggers

Keylogger's main functionality is to record the victim's every keystroke. This log file can be sent to a remote server, or stored locally. Keyloggers are often used legally, by employers and parents to monitor their employees and children.

There is a huge market of keyloggers for sale online, from basic functionality to fully customizable loggers which are capable of recording, not just the keystrokes but even take screenshots or track the mouse movement. Some of the high-end keyloggers provide a much cleaner log for the spy, not only the keys pressed but also a timestamp and it can provide information about the windows the victim typed in.

Keyloggers can be recording keystrokes, from very low level to high level, for example, there is a device which needs to be placed between the keyboard and computer and captures keys there.

Software-based keyloggers are fairly easy to develop. Online repositories offer a lot of open-source loggers, for different operating systems.

Keyloggers work differently on different systems. The methods written below are just one type of possibilities, there are other methods to make them work.

2.4.2 Keyloggers on Microsoft Windows

Most keyloggers are made for Windows because this is the most frequently used operating system on computers. Windows keyloggers can work using the Windows API, which lets them hook into "SetWindowsHookEx" and after this, we can poll the "GetAsyncKeyState" to gather information about the keys pressed. As soon as the key is retrieved it is up to us, what we want to do with it. To create a keylogger one doesn't have to use low-level languages to interact with the Windows API, for example, there are Python modules called "pywin32" and "pyhook" which lets us interact with the API and this way we can create a keylogger in Python.

2.4.3 Keyloggers on Linux

The most common Linux keylogger is called "logkeys" it is an open-source keylogger that makes it easy for everyone to customize a bit and add to their malware. It is an advanced keylogger, it can handle modifier keys alongside with normal keys. This keylogger relies on the event interface of the Linux input subsystem [14]. Linux keyloggers can be easily written in Python or any other languages.

2.4.4 Keyloggers on macOS

Apple's security for macOS is generally held to a fairly high standard. A keylogger to work has to exploit a vulnerability in Apple's security controls. In recent years spyware made for macOS is becoming more frequent. The mechanism is almost the same as on the other operation systems. Keyboards use the HID API to work which has to endpoint IN and OUT, watching these endpoints makes it easy for the attacker to capture keystrokes. A paper from 2018 promotes a keylogger written in swift that doesn't need permissions to run [10].

2.4.5 Classification of keyloggers

Keyloggers can be specified to different categories, there are different classifications, but I found the following the most accurate [9].

- Hardware keyloggers
- Software keyloggers

Hardware keyloggers

Hardware keyloggers have the advantage it doesn't rely on any other installed software. They are also capable of record BIOS password, or password for the disk encryption software. There are more types of hardware keyloggers. All of them have a microcontroller and some kind of non-volatile memory where the data can be saved. Hardware keyloggers share a large disadvantage against software-based which is the physical presence, the attacker has to be close to the victim, which increases the risk of getting caught.

Regular hardware keylogger A regular hardware keylogger is just a small device that is a man-in-the-middle attack between the keyboard and the computer. It captures the user's keystrokes and stores them locally. It is hard to notice if already installed. See the following figure:

Figure 2.3: *The figure shows the device itself and the same device in action*

Wireless keylogger Today many users have a wireless keyboard, they usually communicate using a Bluetooth adapter. A wireless sniffer can capture this traffic, if it isn't encrypted or the encryption key can be cracked then the attacker can steal our data.

Acoustic keylogger This method is more like a proof of concept method, and it isn't used in real life, however, I found this method very interesting. Recording the sounds the keystrokes make with a high-end microphone lets the attacker analyze it. Every key makes different sound, which gives the chance to the attacker to use statistical methods such as frequency analysis to pair the keys with the sounds [17].

Software keyloggers

Software keyloggers are programs made for capturing keystrokes. They are classified based on the logging mechanism. There are those which store the log locally, and there are those which upload the captured data to a remote server. I focused on those which uploads to a remote server. This spyware can be classified into three more classes based on the logging mechanism. The most frequently used logging protocols are SMTP, FTP, and HTTP.

- Periodic loggers
- Buffer based loggers
- Trigger key loggers

Periodic loggers This type of malware uploads its log files to a remote server in a periodic behavior. Let's say it sends an email to the attacker every 3 minute containing the log files, and screenshots created by the spyware.

Buffer based loggers Buffer based loggers add the keystrokes to a buffer, and when the buffer is full it can be uploaded to a remote server. For example, let's say the buffer is 500 bytes when it is full, then the spyware uploads the 500 bytes/characters to an FTP server.

Trigger key loggers This is an interesting type of spyware, it defines a trigger key or trigger word. When the trigger key or word is typed, the spyware uploads the logs to the remote server. In a real-life scenario, this can be implemented by a keylogger which uses HTTP POST request to send the logs when the Enter key is pressed.

Now that we are aware of the basics of keyloggers, we can proceed to detection.

Chapter 3

Detection

There are already a lot of detection mechanism out there. Most of them are host-based approaches. I wanted to create an algorithm that enables us to detect spyware by sniffing the network traffic. I have read a lot about a possible solution and concluded that an Intrusion Detection System like approach would be the best. This way even if there is a malfunction on the algorithm it doesn't have an impact on the network traffic.

However this approach has limitations, we can detect only remote logging keyloggers, so hardware-based and software-based keyloggers that store the capture locally is not possible to detect this way.

There are many advantages to this approach, it is lightweight and doesn't need to be installed on every machine. This solution is very scalable because if we add a new computer to the network we don't have to do anything else, the algorithm will do the work.

I implemented a proof of concept Python script, that analyses traffic from the given interfaces. When it finds a possible spyware threat its IP and some other details are printed on to the standard output. The program uses tshark and pyshark for packet analysis.

Currently, the program is focused on SMTP, HTTP/HTTPS, FTP and FTPS protocols. As I mentioned earlier there are three types of remote keyloggers, each requires a different detection mechanism.

While developing the algorithm I created pcap traffic captures, that can be used for testing the algorithm. I made different captures some of them had spyware in it, some don't, the spyware always used different logging mechanism. I used different keyloggers while creating the captures, with different settings. The pcap files were captured in the Asherah infrastructure, which can be seen in Chapter 3.

3.1 Detecting periodic loggers

This type of keylogger uploads the captured data to the remote server in a periodic way. For example, it sends logs in every 2 minutes to a remote FTP server.

However, I couldn't find statistics about the dispersion of remote logging keyloggers, I think this is the most common type of remote loggers. Every keylogger I tried during this

research support periodic logging.

There are 2 different solutions for this kind of loggers: one is based on statistics, the other one builds up a binary signal and searches for the period. The development process looked the same for both of these approaches. I created a basic script, then tested it, refined it, then tested it again and this loop took many iterations.

In the following subsections I will define my detection algorithms, while the performance analysis of the solutions is described in Chapter 3.

3.1.1 Statistic based analysis

The algorithm works on the captured traffic, and the output is the host that is suspected to be infected by spyware. The algorithm loops through the following steps, but before it could run we need to define some constants, one important is the address range of the local network. The other one is the protocols we are looking for, currently, it is HTTP, HTTPS, FTP, FTPS, and STMP.

The simplified description of the algorithm looks like this.

1. We are only interested in outgoing traffic, so if the source address is not in the local address range we skip that packet.

2. We need to separate the packets based on the source address, this way we have the packets for each host. It is important to note that we pick 1 packet of each TCP stream.

3. Then we need to separate the traffic based on protocols, we are interested in.

4. Sort the packets by capture time.

5. For a given packet we need to calculate the time difference for every packet that comes after it. The difference is added to a list.

6. From the difference list we can build up a dictionary and we calculate which time difference is the most frequent. Because there can be errors, we provide an error margin, and if the difference is in that range we increase the counter for the dictionary and recalculate the key according to the new difference. The destination IP address is also added, so we can guess the attacker's server.

7. We have to check if the packets that seem suspicious, are constantly present in the traffic.

8. If none of the checks fail, then we print out that we found a possible spyware victim, and print out the details of the suspected packets.

The algorithm is not complicated at all, but it also includes some checks that are not included above.

It might become more clear if we check a real-life example. Let us say there is a periodic keylogger on our computer that sends logs every minute through SMTP, but we send out emails during the examined period.

The examined period is 5 minutes and the sent out packets are the following.

```
00:36 - User sent out an email
01:00 - Spyware sent out an email
01:07 - User sent out an email
01:21 - User sent out an email
02:00 - Spyware sent out an email
02:47 - User sent out an email
03:00 - Spyware sent out an email
03:36 - User sent out and email
03:55 - User sent out an email
04:00 - Spyware sent out an email
04:11 - User sent out an email
04:29 - User sent out an email
04:48 - User sent out an email
05:00 - Spyware sent out an email
```

Let's do the algorithm from the fifth step. If we do everything correctly the following frequency dictionary will be present.

```
01:00 - 4
01:24 - 3
00:53 - 3
02:00 - 3
03:00 - 3
00:49 - 2
00:24 - 2
00:31 - 2
03:53 - 2
02:29 - 2
00:19 - 2
02:48 - 2
02:11 - 2
...
```

The result shows us that even with this huge noise, the result of the algorithm was the correct period of the keylogger. The real algorithm would also check for the duration of the packets that seem suspicious to the duration of the whole capture and if the logging seems continuous then it is printed to the standard output.

3.1.2 Signal based analysis

We also wanted to create a signal processing based solution. The main idea was to create a binary signal which is 1 when there is outgoing traffic and 0 when there is not. Our first idea was to use Welch's power spectral density estimate, however, due to the high signal-noise ratio, this didn't provide good results. We also tried several Fast Fourier transform-based algorithm but none of them seemed to work. We asked Dr. Rucz Péter, and he gave us the idea to try using the cepstrum of the signal. A cepstrum is a result of taking the inverse Fourier transform (IFT) of the logarithm of the estimated spectrum of a signal[15].

The equation for calculating cepstrum:

$$|\mathcal{F}^{-1}\{\log(|\mathcal{F}\{f(t)\}|^2)\}|^2$$

The simplified description is presented below.

1. We are only interested in outgoing traffic, so if the source address is not in the local address range we skip that packet.
2. We need to separate the packets based on the source address, this way we have the packets for each host. It is important to note that there is a maximum of 1 packet per TCP stream.
3. Then we need to separate the traffic based on protocols we are interested in.
4. Then we need to sort the packets by capture time.
5. Build up a binary signal from the packets, if there is no packet the value is 0 if there is the value is 1, the algorithm uses 1 second long intervals.
6. Calculate the cepstrum of the previously created signal.
7. Check the spikes of the results, if there are spikes it means we suspect that there is malware on the computer and we print the result to the standard output.

The following figures show the difference between cepstrum and Welch's transform.

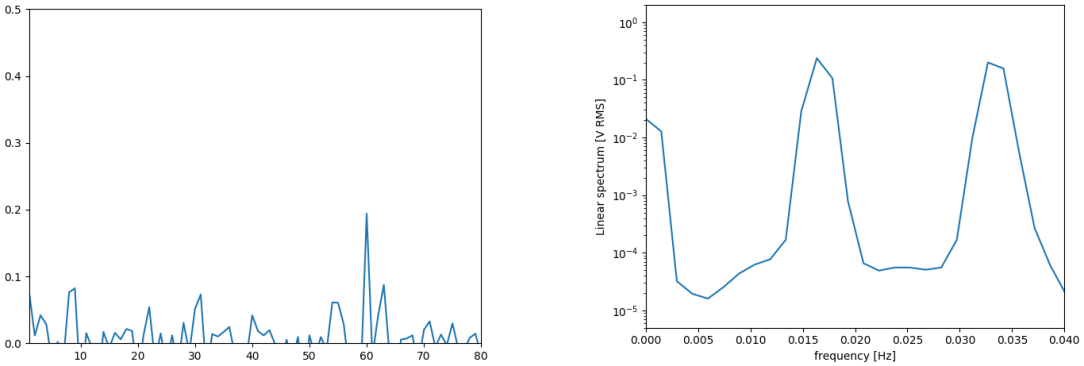

Figure 3.1: *The result of Cepstrum (left), and Welch(right) on a signal where the SNR is high.*

This figure shows us that both methods found the period, which was 60 seconds, there is a big spike on the cepstrum plot at 60, and there is also a big spike on the Welch's transforms plot at 0,0167 which is 1/60.

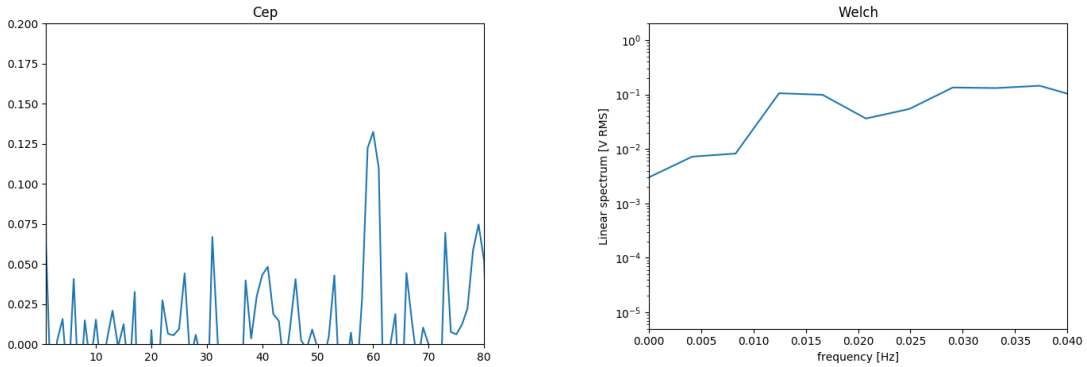

Figure 3.2: *The result of Cepstrum (left), and Welch(right) on a short and noisy signal*

The logging period of the spyware was 60 seconds, however, the recorded traffic was just 4 minutes long, and there was a very huge noise. We can see the spike at 60 on the cepstrum's plot. The other plot goes up at 1/60, but it would be difficult to conclude from this.

The following figures show us the result of cepstrum and Welch's transform when there is no spyware on the network.

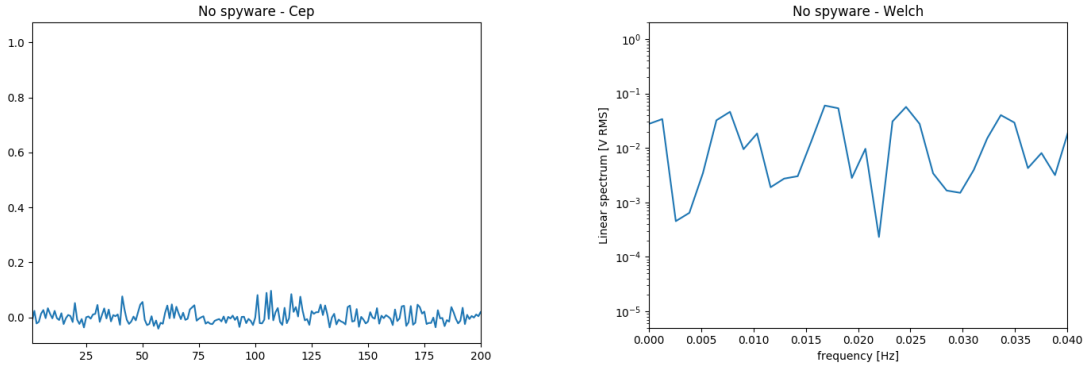

Figure 3.3: *The result of cepstrum (left) and Welch's transform (right).*

Here we can see that there are no big spikes on either of the plots, which is the expected behavior.

3.2 Buffer based keylogger

Buffer based keyloggers specify a buffer size, and when a key is pressed, it adds it to their buffer. When the buffer reaches its limit that initiates the uploading process, then the buffer gets cleared. The base idea was that if the payload is the same then the packet sizes shall be the same for a given protocol. This statement seemed true, but when the traffic was encrypted, we couldn't find the data packet. So the final idea was to sum the whole TCP stream size in bytes and then compare the stream sizes. However, errors can appear here as well, for example, a packet needs to be resent which can affect the stream size. The algorithm uses only statistics based approach the simplified description can be read down

below:

1. We need to separate the traffic based on the source address which results in a dictionary where we can access the traffic for each host.
2. We need to split the traffic based on the protocols, if a packet matched for the protocols we provided then the whole packet stream is copied to a list
3. We calculate the size of each stream.
4. Look for streams that are the same size, and check if they look the same, average packet size, packet count, etc...
5. The result gives us streams that are very similar to each other.
6. We put these streams on a timeline and compare it to the capture time, based on this we can decide if the streams need to be declared as suspicious.
7. We print out the IP addresses of the host that have been found suspicious and other details like the destination addresses of the suspicious TCP streams.

3.3 Trigger key-based keyloggers

This type of keyloggers can't be detected using the previously mentioned methods because the upload time, and packet size depends on the user.

Imagine a real-life scenario where the trigger key is the Enter key. This makes sense because most people use this to submit online forms and at the end of each log our password will appear. The time and the keystrokes among each press of the Enter key are variable. There can be uploads that contain many characters and uploads that contain only words, or characters.

These packets could be analyzed if we could take a look at the payload, which is not possible when the data is encrypted and transmitted through a secure protocol.

Because of the above, the proposed algorithm is not capable of indicating trigger-based keyloggers.

Trigger-based keyloggers aren't too common, I used many keyloggers during this research and just a few of them were capable of trigger based keylogging.

Now that we are aware of the algorithms used for detection, we can test them on real-life scenarios, see the next chapter for the test cases and results.

Chapter 4

Evaluation

I had the luck to test my solution on the upper levels of the Asherah topology. The Asherah infrastructure was created for the International Atomic Energy Agency, and it is a virtualized nuclear power plant. This infrastructure has working email servers, gitlab server, DNS servers and many more, the workstations authenticate through an active directory. With this network, I can test my solution in a fully working industrial control system.

4.1 Asherah infrastructure

Creating Asherah was an international project of the IAEA. The CrySyS Lab participated in this project, and we created security levels 4 and 5. The infrastructure is virtualized on VMware vCenter and the deploy script is written in Ansible.

Ansible is a Python based program that reads configurations and tasks from .yaml files and then executes them. It is used for automatic infrastructure deployment, and the playbooks, which contain the configurations can be run multiple times. This feature gives us a good chance to test different kinds of attacks, and if we want to test a new scenario, we can rollback easily to the default state of the infrastructure.

The two levels used in my scenario can be seen in the following figure.

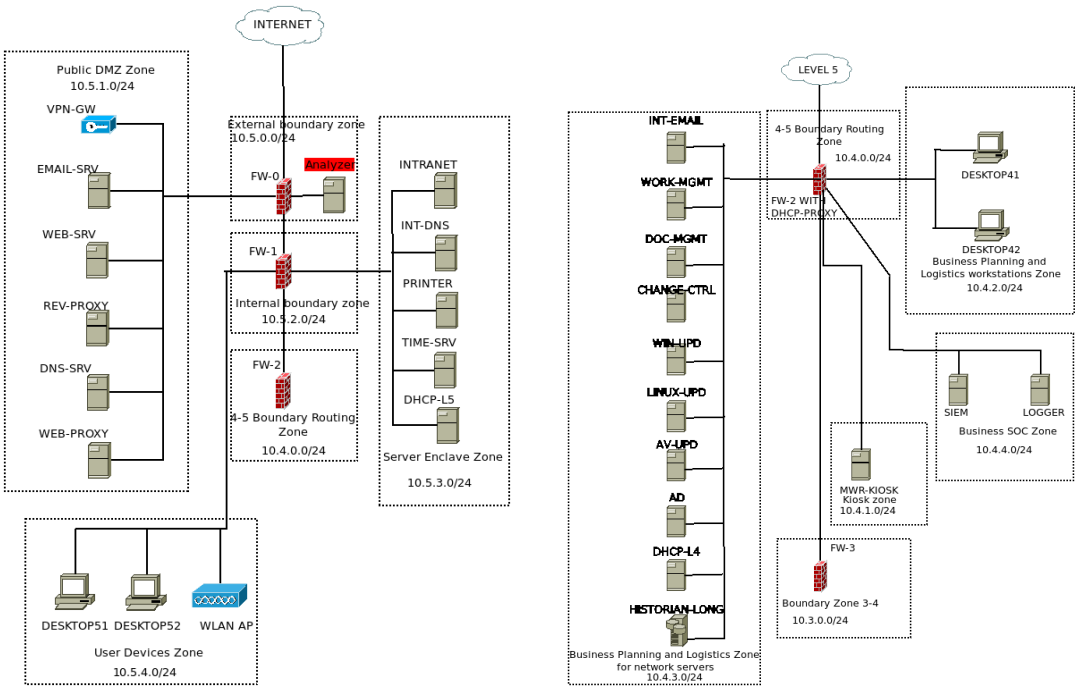

Figure 4.1: *The topology of Asherah SL 5 (left) and 4 (right)*

4.2 Inserting the analyzer into the infrastructure

As mentioned earlier we use an IDS based solution where the new virtual machine doesn't affect the performance of the network. We have to create a new virtual machine, and then we need to copy the outgoing traffic to the new virtual machine.

We can do the cloning in two different ways, one of them is to use vCenters built-in mirroring function which is called VMwareVspanSession. During my previous work at the CrySyS Lab, I was given the task of creating an Ansible module for managing VMwareVspanSessions [11]. Using this module we can create not just the infrastructure but even the port mirroring in ansible, and we won't have further work with this. The other method is independent of the virtualization method, this is done with an iptables rule. An example iptables configuration looks like the following:

```
iptables -t mangle -A PREROUTING -j TEE --gateway 10.5.0.200
```

Where 10.5.0.200 is the analyzer machine and the TEE target creates a copy of the traffic.

4.3 Setting up scenarios

I used several different keyloggers for testing. The keyloggers used can be seen on the table below, however, the results were independent of the keyloggers. To provide the same testbed, I used a user emulator developed by Gnandt Balázs [3], this way the computers had always the same input.

Before testing the scenarios I set up an FTP Server and an Apache Web Server where stolen keystrokes can be uploaded. This 2 server was also virtualized on vCenter and they

are outside of the Asherah's network range.

The workstations which act as victim in the scenario run Microsoft Windows 10.

The user emulator does the following tasks.

- Searches for a picture and downloads it.
- Uploads the picture to the Intranet.
- Browse Reddit.
- Reads StackOverflow.
- Searches on google.

The time amount which is spent on each task is randomized, and the tasks are also randomized this way there will be no periodic event which is generated by us. The emulator works with a recording script, which records the keystrokes and mouse movements, and then replays them. Modifying that script enables me to randomize the wait time.

The keyloggers I tested had a free trial period, or the developers granted me a trial license, I even built my keylogger for better customizability from snippets found online. I came to the conclusion that the keylogger used doesn't affect the detection. This type of keyloggers doesn't encrypt the captured data, the encryption depends on the upload protocol.

Table 4.1: *Table of tested keyloggers.*

Name	Periodic	Buffer	Trigger
All In One Keylogger	yes	no	yes
REFOG Keylogger	yes	no	no
Revealer Keylogger	yes	no	no
Best Free Keylogger Lite	yes	no	no
Keylogger created by me	yes	yes	yes
Keymail	no	yes	no

4.3.1 Example scenarios

The following scenarios are selected based on importance, I wanted to show one for each protocol. The most interesting is the HTTP traffic since that is the most frequent. We generate a very clear signal from SMTP traffic because SMTP traffic normally doesn't come from the Workstation. FTP traffic is rare as well, and the protocol type doesn't change the outcome of the detection.

I did more than 20 different scenarios, with different settings, and the detector always alerted the presence of a spyware.

Scenario 1

In this scenario, DESKTOP41 was already infected with malware, which records keystrokes and uploads them to the attacker's FTP server every 2 minutes.

The keylogger used in this scenario was the All In One Keylogger, the configurations can be seen in the following figure.

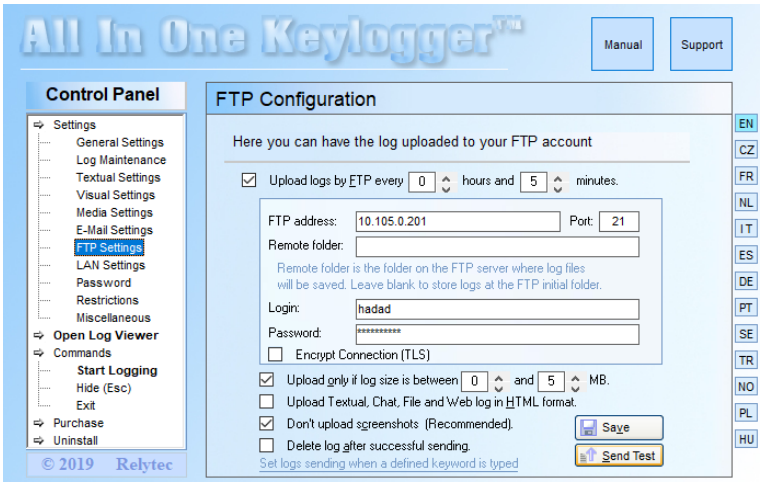

Figure 4.2: *The FTP settings for All In One Keylogger*

For analyzing we use the cepstrum based approach, the simulation took 20 minutes to run, and the analyzer runs every 5 minutes.

The calculated cepstrum can be seen in the following 4 figures. The difference among the figures is the amount of time elapsed, which is written on the top of each plot.

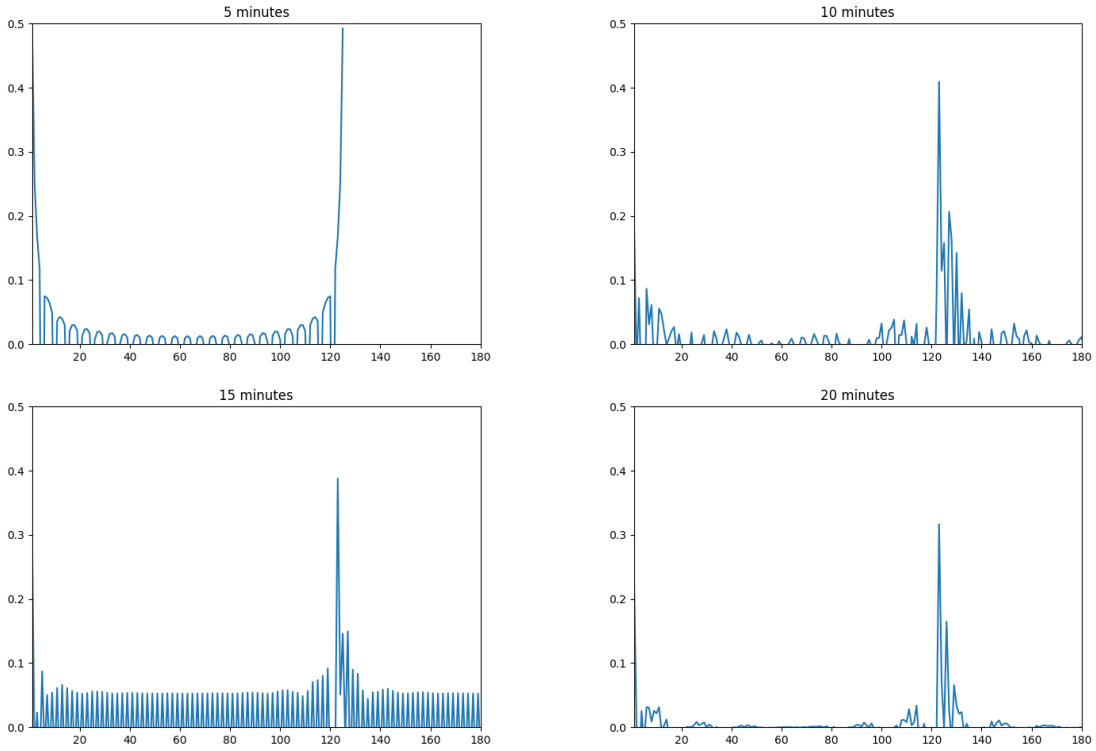

Figure 4.3: *The cepstrum plots for the Scenario*

It can be seen here that the more data we get the more accurate the plot gets, the actual

period is around 120 seconds. There is just one big spike, so it is easy to conclude from this figure that there is a spyware present on the computer that logs every 120 seconds. It is also nicely presented on the plots, the more data we have the easier the noise filtering gets.

Scenario 2

In this scenario, DESKTOP41 was already infested with a keylogger, which uploads the logs using HTTP POST request every minute. Note that there will be a huge HTTP traffic noise as well.

The keylogger used in this scenario was the one that was made by me.

The detection method used here was the statistics-based approach. The script will be handled 10 minutes, of the network traffic.

The end result is presented below:

```
PERIODIC CHECK
Host: 10.4.2.100
HTTP
Spyware Found!
Period: 0:01:00.010657
Attacker's possible logging server(s):
10.105.1.139
```

The algorithm can sense the presence of a spyware, it also logs the protocol used for logging in the third line. The logging period, and the destinations of the possibly malicious packets are printed out as well.

Scenario 3

In this scenario, DESKTOP41 was already infected with a keylogger, which sends the logs in an email every 5 minutes.

The keylogger used in this scenario is the REFOG Keylogger.

The simulation here runs for 1 hour, and the detection script runs every 10 minutes. Here the cepstrum based approach is used.

The results can be seen in the following figures. The difference among the figures is the amount of time elapsed, which is written on the top of each plot.

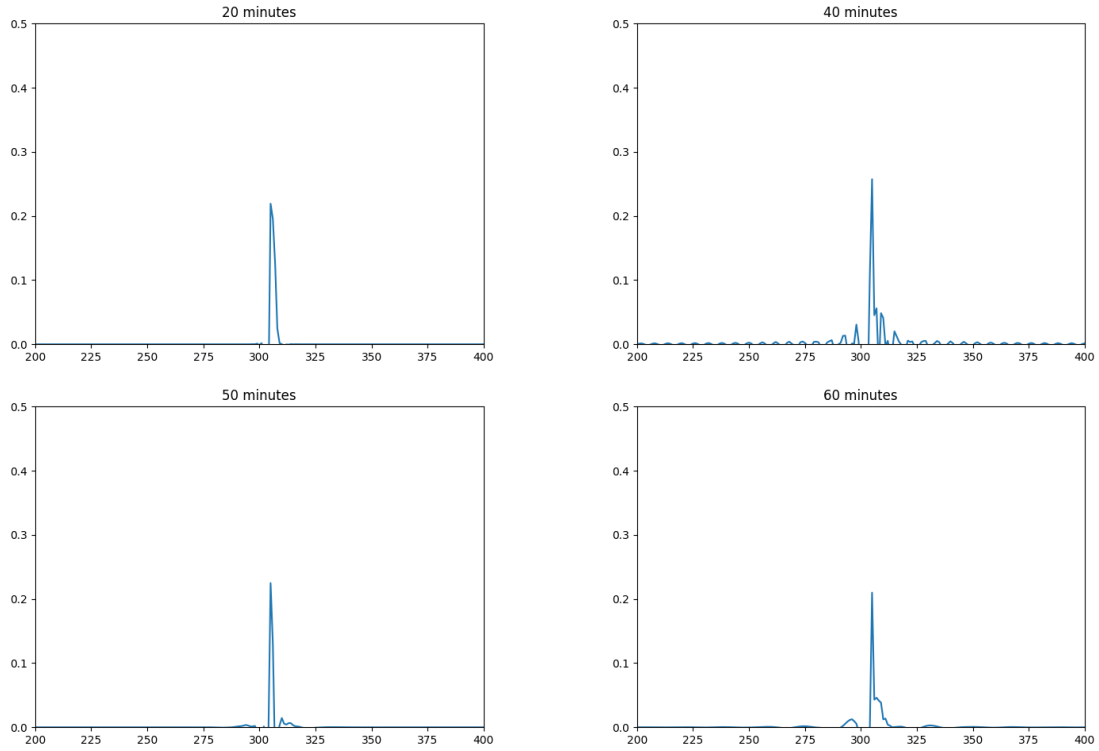

Figure 4.4: *The cepstrum plots for the Scenario*

The figure shows that the keylogger can be detected from the 20th minute, the problem with the signal-based approach was that it requires a minimal dataset which wasn't present at the 10th minute, so the script couldn't run the cepstrum calculation. As soon as the minimal dataset is present the keylogger can be easily detected.

Scenario 4

In this scenario, DESKTOP41 was already infected with a keylogger that logs key after every 100 characters pressed and sends the logs via email.

The keylogger used here was the Keymail Keylogger, and the email is encrypted (SSL/TLS). The intresting thing in this scenario is that, the encrypted emails, still share the same size, and this comes in a result which can be detected.

The output is presented below.

```
Scanning...
74.125.140.109
[1957, 1957, 1957, 1957]
Spyware Detected!
64.233.167.109
[1957, 1957, 1957, 1957, 1957, 1957, 1957, 1957]
Spyware Detected!
```

This snippet found Spywares, the printed IP addresses are the log servers of the attacker, in this case both belong to Google. Under the IP addresses we can see a list that contain numbers, these numbers are the packet size of the whole TCP stream in bytes.

It is important to note that the packet sizes can be different, it really depends on that

how the attacker decides to encode them. It also depends on the connectivity, if there is a packet lost, then the packet is transmitted again, increasing the size of the TCP stream.

For example, there can be an attacker who encodes everything as a string, there are special characters which aren't part of the ASCII table. This has an increased size in bytes and therefore even it is just one character it is more than 1 byte, which results in an increased packet size. When the packet sizes have small difference the algorithm still detects the spyware, but as soon as there is a bigger difference (15%-20%) the algorithm fails to detect the presence of the keylogger.

Scenario 5

This is a real-life scenario, where one of the computers is infected, and the script runs on the analyzer machine. The keylogger sends a log every minute to the attacker's web server.

The questins releated to this scenario was the following.

- How many logs are sent before we could detect the spyware?
- Which method detects the spyware first?

To answer these questions, we had to run the test and the results were the following:

lista 4.1: *The results for statistics based algorithm*

```
Analyzed 4 minutes
Spyware Detected!
Period:0:01:00.700317
Attacker's server: {'10.105.1.139', '216.58.208.46'}

Analyzed 5 minutes
No spyware

Analyzed 6 minutes
Spyware Detected!
Period:0:01:00.184344
Attacker's server: {'216.58.208.46', '10.105.1.139'}

Analyzed 9 minutes
Spyware Detected!
Period:0:00:59.441849
Attacker's server: {'172.217.19.110', '216.58.208.46', '10.105.1.139'}
```

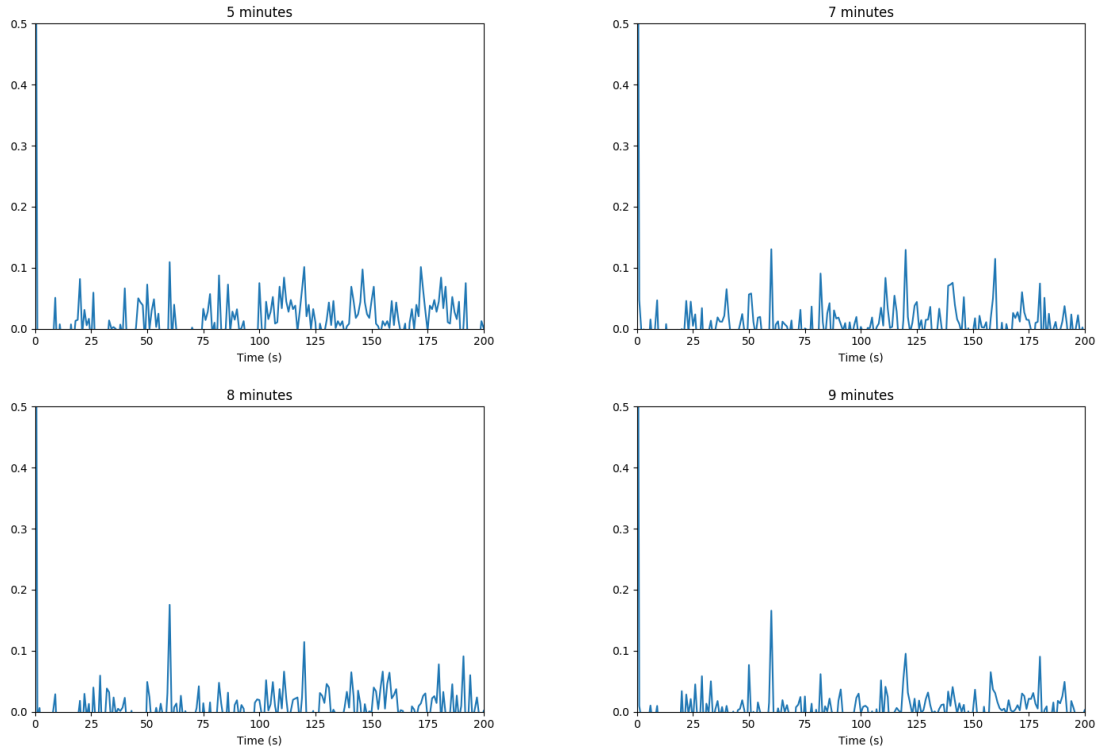

Figure 4.5: *The results for the cepstrum-based approach*

To summarize the results, the statistics based could detect the spyware for the first time just in 4 minutes, however, due to some interference in the 5th minute, the results came back negative. From the 6th minute, the spyware is detected every minute. Due to the small sample size, the algorithm gives false information for the Attacker's server which was 10.105.1.139. The cepstrum-based approach could detect the presence of the spyware from the 8th minute. The figure shows that with more sample we get less noise which results in stable detection.

4.3.2 Performance

Speed

While developing the scripts the first problem with the performance was memory usage. Analyzing a small pcap file resulted in enormous memory usage. This problem was resolved using our Packet class which keeps only the details we were interested in. After this problem was resolved, the next bottleneck was the CPU. I measured the analyzing speed using the following method.

First I was thinking about generating a huge load of traffic, but then I realized that it wouldn't be good for testing the capabilities of the algorithm because the packet size does not affect our algorithm since it uses our custom Pacet class, which doesn't contain the payload. Then I was thinking about a Packet/Second based solution since there are 3 different approaches to detect spyware I will benchmark all of them.

I created a lot of .pcap files that contain different amounts of packets the algorithm is

interested in. Then I will run the algorithms, and calculate a packets/second speed. These results depend on our CPU power.

The speed of each algorithm is present in the following figures. The tests were running on my laptop which is a Dell XPS 13 9370 and has an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz.

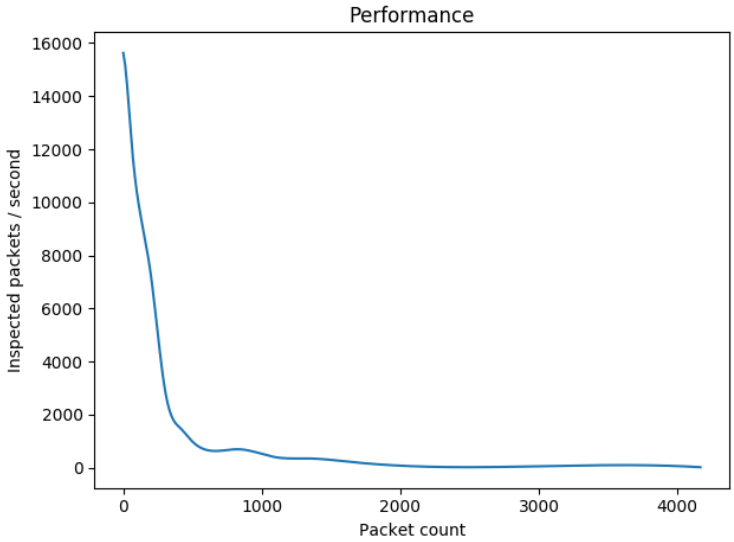

Figure 4.6: *Speedtest for detecting periodic keyloggers based on statistics.*

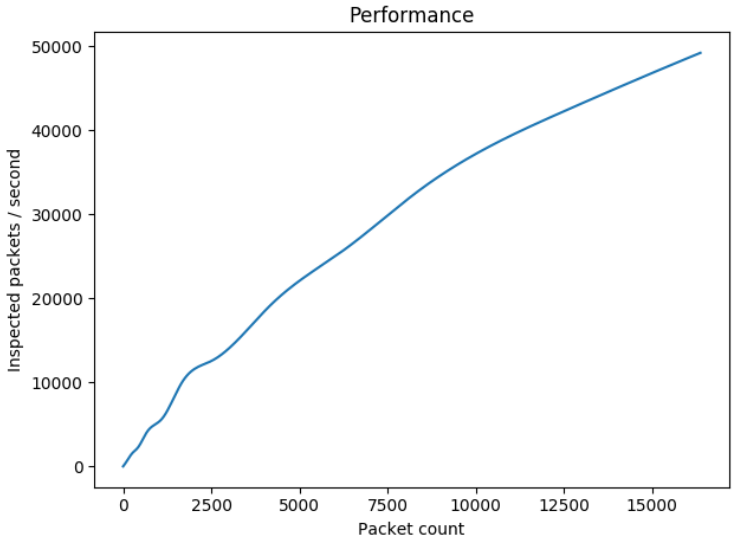

Figure 4.7: *Speedtest for detecting periodic keylogger based on the cepstrum of the signal.*

The speed for buffer-based keylogger can vary on the packet count of each different TCP stream.

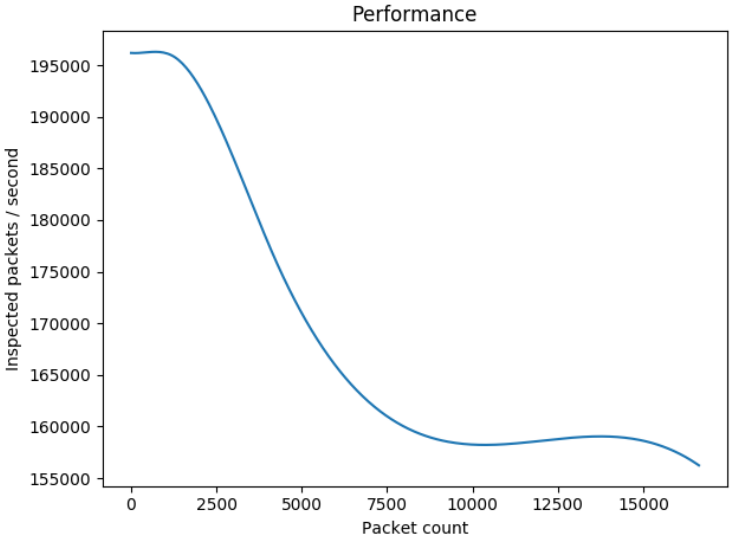

Figure 4.8: *Speedtest for detecting buffer-based keylogger based on statistics.*

This measurement showed that statistics-based approach for detecting periodic keylogger is not a viable solution on a network where the outgoing packet/second rate is high.

The cepstrum based may have disadvantages in the short run, but in the long run it is a viable solution because high packet count doesn't slow the algorithm.

The results for buffer based solution showed that the packet count affects the algorithm but due to its simplicity it can be applied even on huge amount of traffic

Chapter 5

Conclusion and Further Improvements

Spyware is frequently used by attackers to gather information, detecting them is an important task to minimize data loss. The detection through network traffic makes the detection mechanism more scalable and maintainable.

In my opinion, the fact that the algorithm detects keyloggers from different vendors makes it clear that this solution can be used for detecting malicious spyware. However, due to the constraints that it is unable to detect other than remote logging spyware, I would advise to use it along with a host-based approach. To increase scalability I would advise to use host-based detection on critical infrastructure and also apply the proposed solution.

After the performance test, I realized that I should have merged the solutions for detecting periodic keyloggers. When there is a small amount of sample data the detection should use the statistics-based approach and in other scenarios use the cepstrum-based approach.

There is still a lot of development needed for the detector script, but I think it worth time investments because the results clearly showed that we can detect keyloggers.

To increase the accuracy of buffer-based keyloggers, I should implement a mechanism that detects re-sent packets, and these packets won't be added to the final stream size.

To give the script a better and faster detection rate on periodic loggers, the statistics and cepstrum-based solutions need to be merged in a way these methods can reach the best results.

The statistics based solution for detecting periodic keyloggers needs an upgrade to counter the effect of packets bursts.

In summary, I showed that detecting keyloggers based on network traffic is possible, and an enhanced version of my solution can be a core technology used in enterprise environments.

Acknowledgement

I want to say thanks to dr. Holczer Tamás who guided me through not just this research but also the last two years.

I also want to thank CrySyS Lab for allowing me to do this research during my internship.

I want to say thanks to dr. Rucz Péter who helped with the signal-based approach, and mentioned us to try calculating the cepstrum of the signals.

I also want to say thanks to Gnandt Balázs for his awesome user emulator, and for helping me to create the users in this emulator.

During the testing of the algorithm, many keyloggers were used. I want to say thank to Refog Team, for giving me a license for REFOG Personal Monitor.

Finally, I want to say thanks to one of my best friend Körmendy Bertalan for helping with his enormous vocabulary to increase the grammatical quality of this paper.

Bibliography

[1] Adware. 10 cyber security facts and statistics for 2018. https://us.norton.com/internetsecurity-emerging-threats-10-facts-about-todays-cybersecurity-landscape-tha html.

[2] Adware. Spyware statistics. https://www.adaware.com/knowledge-database/spyware-statistics.

[3] Gnandt Balázs. Felhasználók emulációja csapdarendszerekben. https://diplomaterv.vik.bme.hu/hu/Theses/Felhasznalok-emulacioja-csapdarendszerekben.

[4] BBC. Dark web sting leads to arrest of alleged spyware thief. https://www.bbc.com/news/technology-44725446.

[5] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. Duqu: A stuxnet-like malware found in the wild. Technical report, CrySyS Lab, October 2011.

[6] Mailbird Christin. 5 facts on email security threats in 2019. https://www.getmailbird.com/5-facts-email-security/.

[7] Symantec Corporation. What is spyware? and how to remove it. https://us.norton.com/internetsecurity-how-to-catch-spyware-before-it-snags-you.html.

[8] Casey Crane. 80 eye-opening cyber security statistics for 2019. https://www.thesslstore.com/blog/80-eye-opening-cyber-security-statistics-for-2019/.

[9] Reiner Creutzburg. The strange world of keyloggers - an overview, part i. 2017.

[10] Skrew Everything. Hacking: Keylogger for macos. *no permissions needed to run*. https://medium.com/from-the-scratch/hacking-one-of-its-kind-keylogger-for-macos-no-permissions-needed-to-run-684ff32025

[11] Péter György. Create or remove a port mirroring session. https://docs.ansible.com/ansible/latest/modules/vmware_vspan_session_module.html.

[12] Gabriel E. Hall. Coolwebsearch - the most infamous browser hijacker. https://www.2-spyware.com/coolwebsearch-the-most-infamous-browser-hijacker.

[13] Nishantha Karunaratne. Understanding the gator ? a brief introduction to spyware. July 2003.

[14] kernc. Logkeys. https://github.com/kernc/logkeys.

[15] Malwarebytes Labs. Emotet. https://www.malwarebytes.com/emotet/.

[16] Malwarebytes Labs. Spyware. https://www.malwarebytes.com/spyware/.

[17] Michael LeMay and Jack Tan. Acoustic surveillance of physically unmodified pcs. In *Security and Management*, pages 328–334, 2006.

[18] Sukesh Mudrakola. Whatsapp spyware attack: Everything you need to know. http://techgenix.com/whatsapp-spyware-attack/.

[19] Andrew Sanders. 15 malware statistics, trends and facts in 2019. https://www.safetydetectives.com/blog/malware-statistics/.

[20] Finjan Team. The past and present state of spyware. https://blog.finjan.com/the-past-and-present-state-of-spyware/.

[21] TechTarget. Top 10 spyware threats. https://searchcio.techtarget.com/tutorial/Top-10-spyware-threats.

[22] Wikipedia. Spyware. https://en.wikipedia.org/wiki/Spyware#History_and_development.