Bern University of Applied Sciences

Bachelor Thesis

DarkComet Tracker

Students: Rosalie Truong, Nils Stampfli, Sandro Tiago Carlao
Professors: Dr. Endre Bangerter, Reto Inversini
Expert: Dr. Igor Metz

Declaration of primary authorship

We hereby confirm that we have written this document independently and without using other sources and resources than those specified in the bibliography. All text passages which were not written by us are marked as quotations and provided with the exact indication of its origin.

Rosalie Truong
rosa.g@windowslive.com
Signature:

Nils Stampfli
nils.stampfli94@gmail.com
Signature:

Sandro Tiago Carlao
tiago.sandro@hotmail.com
Signature:

DarkComet Tracker

Rosalie Truong, Nils Stampfli, Sandro Tiago Carlao

Bern University of Applied Sciences,
Quellgasse 21, CH-2501 Biel/Bienne

gerbr2@bfh.ch, stamn2@bfh.ch, tiags1@bfh.ch

Abstract

In year 2017, the paper "To Catch a Ratter: Monitoring the Behavior of Amateur Dark Comet RAT Operators in the Wild", has been published and provides the first reference for the work we are doing today as our Bachelor Thesis.

RATs are the so called Remote Access Trojans. They allow the people behind them, also called Operators, to remotely access to a victims computer, that has previously been infected. This can then be spied on, manipulated or totally taken over. Several RATs are available for free or at low prices on the Internet. The main topic of our work is the RAT DarkComet.

RATs like DarkComet are mostly used because of their usability and diversity.The user interface is developed in a way that allows people without further knowledge on technology to use them. On the one hand, it has been used by teenagers, "just for fun". On the other hand, it has been used by by intelligences, in a context like the war in Syria. Over all, there is relatively little detailed and systematic knowledge about the use of RATs or the behavior of their Operators.

Remote Access Trojans should not be confused with the Remote Administration Tools, like TeamViewer, which represent the legal side of the application possibilities and will not be part of this work.

Keywords: RATs, Malware, Analysis, Scanning, DarkComet, Behavior, Cuckoo, VirusTotal

Table of contents

Page

1 Introduction **1**
1.1 Structure of this Thesis document . . . 2
1.2 Summary of paper ”To Catch a Ratter” . . . 2
1.3 Security of the environment & Privacy . . . 3

2 Background **4**
2.1 Remote Access Trojans / Tools (RAT) . . . 4
2.1.1 Indicator of Compromise . . . 5
2.1.2 Packed samples . . . 6
2.1.3 Infection process . . . 7
2.2 DarkComet . . . 8
2.2.1 Network encryption . . . 8
2.2.2 DarkComet Configuration . . . 9
2.2.3 DarkComet Network Handshake . . . 11
2.2.4 Operator side . . . 12
2.3 Volatility . . . 13
2.4 VirusTotal . . . 14
2.5 YARA Rules . . . 15
2.6 Cuckoo Sandbox . . . 15
2.7 Virtual Machine Hypervisor . . . 16
2.7.1 Honeypots . . . 16
2.8 Scanning Tools . . . 16
2.8.1 Nmap . . . 16
2.8.2 Masscan . . . 17
2.8.3 ZMap / ZGrab . . . 17
2.8.4 Other Worldwide Scanner . . . 17
2.8.5 Shodan . . . 18
2.8.6 Censys . . . 18
2.9 How to recognise DarkComet? . . . 19
2.10 Ethical & Judicial Aspects . . . 21
2.10.1 Analysis . . . 21
2.10.2 Scanning . . . 21

3 Design **23**
3.1 Host System . . . 23
3.1.1 Cuckoo Sandbox - Setup and Configuration . . . 24
3.2 Guest System . . . 24
3.2.1 Victim . . . 25
3.2.2 Virtual Network . . . 25
3.2.3 Virtual Machine Hardening . . . 26
3.2.4 Virtual Webcam . . . 27
3.3 Data Flow . . . 28
3.3.1 Reverse Engineering . . . 29
3.3.2 Scanning . . . 29
3.3.3 Live Analysis . . . 30
3.3.4 Statistics . . . 31
3.4 Automation . . . 32
3.5 DarkComet commands documentation . . . 32
3.6 Scanning . . . 33
3.6.1 Approches éthiques . . . 33

3.6.2 Défis liés au réseau internet . . . 35
3.6.3 Différents types de scans . . . 38
3.6.4 Infrastructure de scanning . . . 40

4 Implementation 42
4.1 Ubuntu Configuration . . . 42
4.2 Cuckoo Configuration . . . 42
4.2.1 Iptables Rules . . . 44
4.3 Start the Cuckoo System . . . 45
4.4 Submit a sample . . . 45
4.5 Folder structure . . . 46
4.6 Scripts implementation . . . 47
4.6.1 Volatility . . . 47
4.6.2 Decrypt Network Communication . . . 48
4.6.3 Scanning-Scripts . . . 50
4.6.4 apscheduler.py . . . 51

5 Experiments 54
5.1 Reverse Engineering . . . 54
5.1.1 Results . . . 54
5.2 Scanning . . . 61
5.2.1 Résultats des samples . . . 61
5.2.2 Résultats obtenus avec Shodan . . . 65
5.2.3 Résultats des scans . . . 69
5.3 Online-Analysen . . . 71
5.3.1 Übersicht . . . 71
5.3.2 Aktivität der Operatoren . . . 73
5.3.3 Vergleich der virtuellen Maschinen . . . 76
5.3.4 Befehle & Aktionen . . . 77
5.3.5 Befehls- & Operatorkategorien . . . 81
5.3.6 Mehrfachanalyse von Operatoren . . . 84

6 Discussion 86
6.1 Reverse-Engineering . . . 86
6.2 Scanning . . . 86
6.3 Live Analysis . . . 87

7 Conclusion 88

8 Appendix 89

Glossary 89

Acronyms 90

References 93

List of Figures

1 Their data collection and processing workflow. Each box displays the number of corresponding entries after each step. [3] 2
2 Pyramid of pain for IOCs [11] 5
3 Original, packed and unpacked executables 6
4 Infection Process [3] 7
5 Encryption with RC4 [15] 8
6 DarkComet v5.3.0 Samples Configuration 10
7 DarkComet Network Handshake 11
8 DarkComet Menu Entries 12
9 DarkComet Operator's GUI 13
10 Possible responses when a port is scanned 19
11 Password RC4 encryption in order to obtain the banner 19
12 IDTYPE package in Wireshark 20
13 Network Environment 26
14 Systeminfo done on Virtual Machine (VM) after hardening 27
15 DarkComet Tracker Data Flow 28
16 Environment for production of DarkComet documentation 33
17 Site web sur Wix [53] 34
18 Speed Test sur le réseau Swisscom 37
19 Aperçu des différents types de scans 38
20 Message transmis par Nmap qui indique qui nous sommes 40
21 Infrastructure du scanning (Version finale) 41
22 Evolution du nombre d'opérateurs actifs par jour 64
23 Activité des opérateurs par jour de semaine 64
24 Activité des opérateurs par heure 65
25 Shodan - Evolution du nombre d'opérateurs distincts actifs par jour 68
26 Shodan - Activité des opérateurs par jour de semaine 69
27 Shodan - Activités des opérateurs par heure 69
28 Aktivität der Operator 73
29 Ausschnitt der Aktivität der Operator 74
30 Differenzierung der Operator nach Aktivität 75
31 Befehlsgruppen 79
32 Sankey-Diagramm bezüglich den Befehlen 80
33 Befehlskategorien 81
34 Operator-Verteilung bezüglich Befehlskategorien 82
35 Operatorkategorien 83
36 Operatorkategorien (ohne I/O-Control) 83
37 Anzahl erfolgreicher Analysen pro Sample 84

List of Tables

1 Default passwords for the various versions of DarkComet [41] . . . 20
2 Host-System hardware configuration . . . 23
3 15 Most used password . . . 54
4 15 Most used ports . . . 55
5 Results of default port combined with no password . . . 56
6 15 most used process names to hide DarkComet . . . 57
7 Most common versions of DarkComet in the wild . . . 58
8 Usage of dynamic DNS providers, private addresses and other IP / domains into DarkComet samples . . . 59
9 Results of Firewall Baypass by sample configuration extraction . . . 59
10 Results of keylogger active by default . . . 60
11 Pays des opérateurs . . . 62
12 Fournisseurs d'accès à Internet (FAI) des opérateurs . . . 62
13 Pays qui comptent le plus d'opérateurs actifs . . . 63
14 Shodan - IDTYPE trouvés sur Shodan . . . 66
15 Shodan - Ports utilisés par les opérateurs . . . 66
16 Shodan - Pays des opérateurs . . . 67
17 Shodan - Fournisseurs d'accès à internet (FAI) des opérateurs . . . 67
18 Shodan - Pays qui comptent le plus d'opérateurs actifs . . . 68
19 Anzahl durchgeführter Analysen, bez. ihrer Erfolgsrate . . . 71
20 Dauer der Analysen, bez. Erfolgsrate und Operator-Aktivität . . . 72
21 Verteilung der Samples und Erfolgsrate der virt. Maschinen . . . 72
22 Auflistung benutzer Befehle . . . 77
23 Operator-Verhalten über Zeit . . . 85

1 Introduction

Malware is the better known name for Malicious Software. It is a software containing a sequence of instructions that perform malicious activity on a computer. The history of malware started with "Computer Virus", a term first introduced by Cohen in 1986 [1]. Today, malware includes viruses, worms, Trojans, rootkits, backdoors, bots, spyware, adware, scareware and any other program that exhibits malicious behavior. In this work we are focused on a precise type of malware, namely RAT (Remote Access Trojan / Tool), and a precise family of RATs, namely DarkComet.

DarkComet is a RAT that runs on Windows platforms, starting from Windows XP all the way up to Windows 10 which means it could potentially affect more than the 80% of the desktop devices in the World [2]. We are interested to understand the spread and use of this RAT. As developed in 2008, DarkComet has already "10 years of history".

The subject of interest, DarkComet, has been chosen because of its high usage flexibility. So it has been used by script kiddies, unskilled people who use scripts or programs developed by others to attack other parties. On the ohter hand is has also been used in 2014 by the Syrian Government to spy on its civilians.

In this thesis, we aim principally to reproduce the experiment described in the paper "*To Catch a Ratter: Monitoring the Behaviour of Amateur DarkComet RAT Operators in the Wild*" [3] done by *Brown Farinholt, Mohammad Rezaeirad, Paul Pearce, Hitesh Dharmdasani, Haikuo Yin, Stevens Le Blondk, Damon McCoy* and *Kirill Levchenko*.

The objective for us is to build an environment that permits to scan and track operators: the people using the RATs. We want to let the system turn during several weeks to detect, record and analyse DarkComet RATs attacks. This will give us the possibility to track and categorise the behaviour of the Operators. To do that we split the work in three main parties, namely Reverse Engineering, Scanning and Architecture. In the first part, we extract useful information from a DarkComet sample, such as hostname, port and password to encrypt the connection. In the second part we scan the Operators in order to know when and where they are active. The third part concerns all the hardware and software infrastructure of this work, but also the live analysis where we want the operators to attack our virtual machines in order to study their behaviour.

1.1 Structure of this Thesis document

This Bachelor Thesis is done by three students of the Bern University of Applied Sciences in Biel/Bienne. As Biel/Bienne is a bilingual city and we are students who speak three different languages (German, French and Italian) we decided, requesting the necessary permits, to write this document in three languages (English, German and French). Clearly, the most important parts will be in English.

In this document we start by doing an introduction, we continue by explaining basic concepts of malware analysis which are useful to follow and understand the rest of the work. We explain then how we designed our system, including which challenges we had during the implementation that has as a result to changes in the design. After this part, we describe how we implemented the whole system, and we describe the experiments that have been done in order to enhance the quality of the results. For each experiment we present then which results came out. Of course a discussion over the total results cannot be forgotten as well as a final conclusion to finish this document.

1.2 Summary of paper by "To Catch a Ratter"

Like said in the section 1, this work is about to reproduce the great experiment described in the paper "*To Catch a Ratter: Monitoring the Behaviour of Amateur DarkComet RAT Operators in the Wild*" [3].

They collected 19,109 samples of DarkComet malware found in the wild, and in the course of two, several week-long experiments, ran as many samples as possible in their honeypot environment. By monitoring a sample's behaviour in their system, they are able to reconstruct the sequence of operator actions, giving them a unique view into operator behaviour. They report on the results of 2,747 interactive sessions captured in the course of the experiment. During these sessions operators frequently attempted to interact with victims via remote desktop, to capture video, audio, and keystrokes, and to exfiltrate files and credentials.

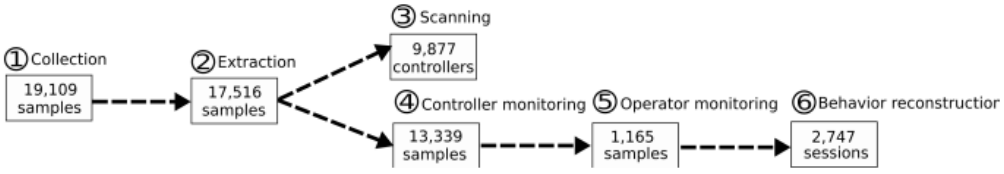

Figure 1: Their data collection and processing workflow. Each box displays the number of corresponding entries after each step. [3]

As per Figure 1, they obtained 19,109 total unique DarkComet samples over the course of the study. VirusTotal enables them to retrieve the geolocation of the Internet Protocol (IP) addresses used to upload these samples. The two most popular sources of their samples are Russia and Turkey. In total, they were able to automatically unpack 17,516 samples. For each unpacked sample, they extract its configuration information using an open-source RAT decoder [4]. Of the 17,516 samples from which they were able to extract configuration information,

13,339 were configured with valid addressing information - domain name(s) or IP address(es). Over the course of the experiment, they discovered and monitored 9,877 unique DarkComet controllers across the Internet. Over the course of the two experiments they made, the executions of 1,165 unique samples resulted in 2,747 total runs for analysis with an operator.

1.3 Security of the environment & Privacy

In this work we are dealing with different types of threats, like RATs, which are basically viruses, but not only. So it is very important to know where those threats can pose a problem in order to mitigate them, or at least try to limit the amount of damage they can do. Just as a kind of disclaimer, we confirm that no operation of this thesis has been done on the Bern University of Applied Sciences's network. For this work we become 2 separate networks, one from Sunrise AG [5] where we have done the malware analysis and one from Swisscom AG [6] where we performed the scans. Swisscom was aware of the work we were doing, but Sunrise not. That is not a problem because unless operators try to make a Denial of Service (DoS) attack, no dangerous operations should be done in that network. Moreover, operators have control of our machines over a limiter time, so not big damages should be done. In the Swisscom network we set up a web server saying what we were doing and showing an email address of contact, in case anyone would like to be cut off from the scans. No one wrote us. Moreover, the samples of DarkComet we got, were provided by Virustotal [7], so no privacy violation has been made.

2 Background

In order to understand the following chapters, we need to introduce some basics of this topic. Because of that, the following section is about to explain the basic tools and concepts used afterwards.

2.1 Remote Access Trojans / Tools (RAT)

A Remote Access Trojan / Tool (RAT) is a sub type of a Trojan that introduces a "backdoor" and allows remote control over the infected machine with administrative functions. The control commands are sent by a human operator via network connection, most of times encrypted, and are seen by the system as if they were done by the victim over physical access. Many RATs, like DarkComet, can be included in legitimate executables, so many times the victim installs a RAT while installing an other software. The victim then, has no knowledge of the presence of the malicious software. An other important propriety of RATs is that because they can be so invasive, and therefore detectable by anti-viruses, they try to hide themselves and its operation, for example by trying to "impersonate" an other process.

Given that the RAT has administrative permissions over the victim machine, it makes possible for the operator to perform practically any type of activity on the infected system without the victim noticing it.

RATs are principally used on targeted attacks to get access to well defined information. In fact, not only hackers or script-kiddies are using this type of malware, but also law enforcement authorities such like governments or police [8] [9]. The reason for that is that now-days cryptography in network communication is so strong that even if governments or law enforcement would like to break it, they can not. So, RATs are employed to overcome to this problem in surveillance strategies.

Here below is a list of the most common commands doable with the help of RATs:

- Activate victim's webcam to spy or supervise
- Monitor user behaviour using keyloggers or other spyware
- Access and steal sensitive data like passwords, usernames or credit card numbers
- Do a lateral movement, which means to use the already infected victim to infect or steal information from an other, most difficult to hack, victim
- Navigate into the victim's system accessing, downloading or modifying user's data
- Format or encrypt drives

2.1.1 Indicator of Compromise

With the increasing number of attacks during the years, antivirus enterprises started to invest to improve their products and try to detect as much malware samples as possible. So, they introduced the analysis by Indicator of Compromise (IOC), artefacts left by an attacker, and today many tools and antivirus use IOCs to analyse evidence. Here below is a definition of an IOC:

> "*Indicators of Compromise (IOC) are pieces of forensic data, such as data found in system log entries or files, that identify potentially malicious activity on a system or network*" [10].

Organisations can detect attacks quickly and therefore try to prevent breaches from occurring by using IOCs in their antivirus or in their Intrusion Detection System (IDS) as they are seen as red flags that indicate a potential running threat. IOCs are not always easy to detect as they can be very simple data structures or unbelievably complex part of malicious code. A good technique to use, is to correlate various IOCs in order to enhance the quality of the results and discard some false positives or false negatives, depending on how secure/usable the system needs to be.
Some examples of IOCs can be IP addresses, domain names, ports, hashes of malware executables, dropped files, logs entries, registry entries, mutexes, process names, usual outbound network traffic, large number of requests for a file, unusual or faked Domain Name System (DNS) requests, signs of DDoS activity, antivirus signatures or YARA signatures (that are explained in chapter 2.5).

As everything in Information Technology (IT) field, some discovered IOCs are easily changeable and some others are more difficult.

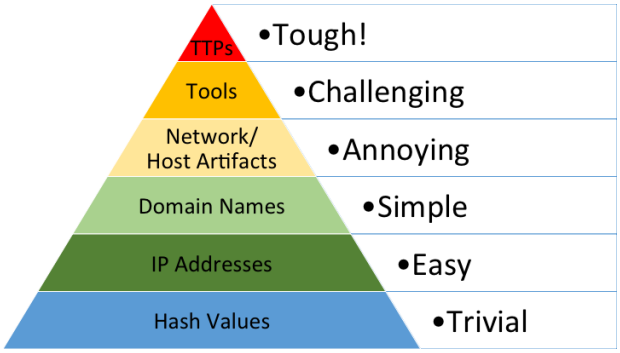

Figure 2: Pyramid of pain for IOCs [11]

Figure 2 shows how much pain it causes to an adversary when someone is able to recognise his malware using the IOCs listed in the pyramid. The level of pain is meant as how difficult is for an adversary to overcome to the detection over this IOC. For example, if a sample is detectable using an IP address, he just needs to change IP and communicate it to the already deployed samples. As an IOC is detected, hackers and operators tend to rebuild their samples and update their already distributed samples, in order to reduce the probability of get caught.

2.1.2 Packed samples

To overcome to the problem of being detected by easy IOCs, hackers and operators started to use the technique described in this section: packing.

A packer is a software that permits to encrypt and compress executables with the goal of evading antivirus signatures and prevent or slow down statistic analysis. When an executable is being packed, a new executable is created containing the original executable compressed and encrypted, and the information to do the opposite operation: decompress and decrypt. Executing a packed sample will first execute the unpacking code to decompress and decrypt the original sample on the memory of the victim and then let the unpacked sample turn into memory. Figure 3 shows the structure of the original sample, the packed sample and the unpacked one.

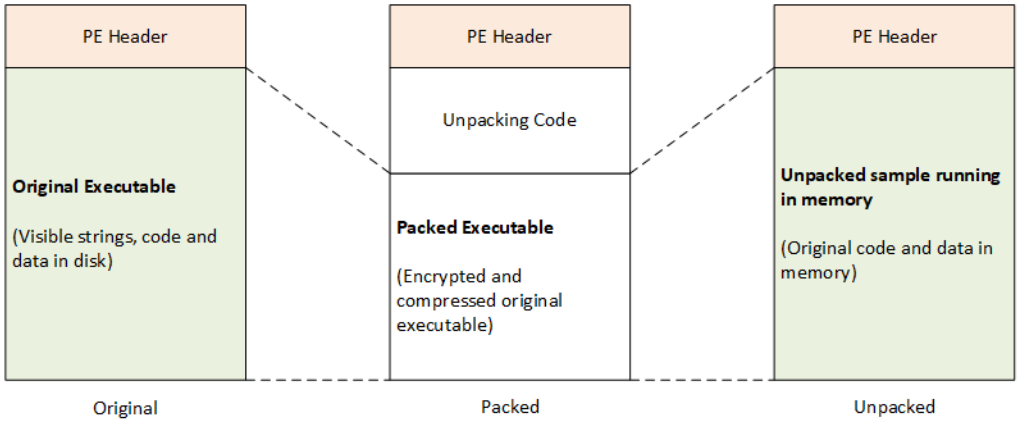

Figure 3: Original, packed and unpacked executables

As a good encryption algorithm should introduce randomness, so if a sample is packed more times, a different output should come each time. This will help to reduce the probability of detection over hash on sample IOC, as the hash of the packed executable will always be different. This technique is often called obfuscation as it literally obfuscates the contents of the original executable preventing direct disassembly and making it more difficult and costly to reverse engineering the packed sample.

By default, during the build of a DarkComet sample, the operator can choose a packer between UPX [12] and MPRESS [13], but of course they can also choose to pack it themselves with other packers, so we cannot say that all DarkComet samples are packed either with UPX or MPRESS.

2.1.3 Infection process

An important and somewhere difficult part of an attack is for sure the infection process. In fact the operator needs to have or create a channel to send the malicious executable to his victim. By creating this channel it might happen that he his exposed and easily traceable.

RAT operator are no exception to this technique, once they have built their sample, they need to deliver it in a such way that it can be lately executed. Figure 4 illustrates the four steps to perform a good attack using a RAT.

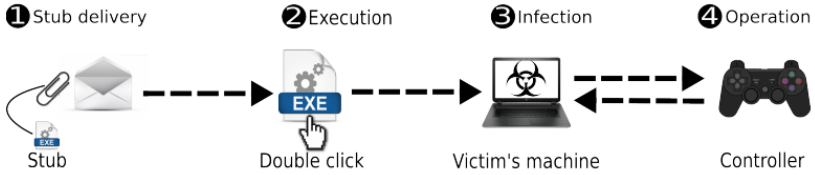

Figure 4: Infection Process [3]

1. The built sample needs to be delivered in a such way that it is not easily traceable by its target. It could be simply phishing a download link to the victim email, or an impersonation of an other software that the victim would like to download, or even simply by USB stick but of course in this case the operator needs physical access to the victim's PC.

2. As second step, the sample, packed or not, needs to be manually executed by the victim. In most of the cases it will be enough to double click the executable to be "successfully" infected.

3. As soon as the executable is launched, it will infect the victim usually by installing itself in a determinate position on the system. Moreover it will read its configuration and try to connect itself to the operator so that he knows that the infection has been successful. It is important to remark that in many cases, before any kind of infection, the code of the sample (packed or no) is configured to check determinate properties of the system. This is done to try to avoid virtual systems or debuggers where people like us can analyse the sample.

4. The last step is really the step where the operator acts his strategy to accomplish his goals. He normally executes commands that are sent via network connection and receives the information required.

2.2 DarkComet

DarkComet is one of the still most present RATs in the World, and it is also the main subject of this Thesis. It has been developed in 2008 but it started to appear in the wild only from 2011. What surprised us is that after almost ten years since its development, it is still widely present in the world. More versions are available, but the last one in the wild is the v5.3.0, which in some cases is also called version 5.1.

DarkComet operates as most of the actual RATs in a reverse socket form, which means that the victim is the server and the operator, or controller, is the client. The version 5.3 samples builder is relatively easy findable in the Internet, as we found it in quite few hours.

We basically searched for a sample builder in order to build our own sample, which means that we know the configurations inside it. That is very convenient when we started to reverse engineer DarkComet, as we already knew the information we want to find, we just had to figure out where does it was placed.

2.2.1 Network encryption

Using this technique we discovered that the network connection between operator and victim is encrypted using the algorithm Rivest Cipher 4 (RC4), also known as ARC4 or ARCFOUR [14], a very widely used stream cipher in the past as the WiFi encryption known as Wired Equivalent Privacy (WEP) [15] was based on it. As demonstrated in 2001 in this great article [16], WEP and more specifically RC4 are quite insecure as one of the major problems is the key-stream reuse.

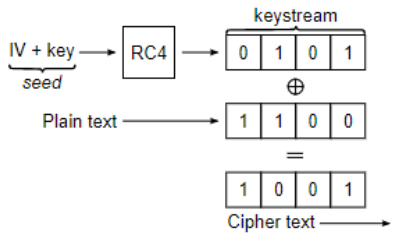

Figure 5: Encryption with RC4 [15]

In fact if we define P as the plain-text, IV as an initial vector, key as the encryption key and C as the cipher-text, we can understand how RC4 works. Figure 5 shows a visual view of this formula.

$$C = P \oplus RC4(IV, key)$$

So if $C_1 = P_1 \oplus RC4(IV, Key)$
and $C_2 = P_2 \oplus RC4(IV, Key)$
then $C_1 \oplus C_2 = (P_1 \oplus RC4(IV, Key)) \oplus (P_2 \oplus RC4(IV, Key)) = P_1 \oplus P_2$

Which means that in quite few time we could be able to decrypt the two plain texts without need to know the password.
Fortunately, we do not need to use this technique in order do decrypt the network connection between victims and operators using DarkComet, as we discovered that the key is stored inside of the sample.

The encryption key is based on two sub-keys: a string defining the version of DarkComet and a second string known as the operator defined password while configuring the sample. The two strings are afterwards concatenated in order to create an unique encryption key. When we were configuring our sample, we defined our custom password as "*testpassword*" which gave us finally this encryption key: #KCMDDC51#-890testpassword. In the case we did not set any password, the encryption would have been only #KCMDDC51#-890 as it is the standard key for version 5.3.0 (or 5.1 as explained)

Searching in the debugger, one more security feature DarkComet came out: the operator defined password is encrypted inside of the sample in the following way:

$$encrypted\_key = operator\_key \oplus RC4(IV, \#KCMDDC51\# - 890)$$

This means that the password defined by the operator, "*testpassword* in our case, is encrypted in the sample using RC4 with #KCMDDC51#-890 as key which basically gives the equation below:

$$825D8C6B6EdFfCC143B95979 = testpassword \oplus RC4(\#KCMDDC51\# - 890)$$

That is the reason because if we search with a string searcher inside the sample, we will not find the whole key "*#KCMDDC51#-890testpassword*", but only "*#KCMDDC51#-890*", as the rest is encrypted. It is therefore very important to understand how passwords are concatenated, in order to understand how to get them from memory once we want to rebuild what commands have been done by an operator attacking one of our virtual machines.

2.2.2 DarkComet Configuration

Like almost all other RATs, DarkComet also has a configuration. But in order to understand DarkComet's configuration, we need to explain what is the configuration of a RAT.

Almost each RAT has some different parameters configurable by the operator such as the IP or domain name which the sample will try to connect itself, the port used to create the network socket, some kind of password in order to secure some sensible data, or standard installation path in the victim system. DarkComet makes no exception to that, and stores only the information configured by the operator. It means that if there are some settings that are not configured by the operator, will not be present during the extraction of the configuration. Moreover, an other time, the configuration is RC4 encrypted with the default key of the version of DarkComet which in our case is still *#KCMDDC51#-890*.

In figure 6 we present 2 configurations extracted from two different samples. Sub-figure 6a comes from a sample downloaded from VirusTotal and sub-figure 6b is the configuration extracted from our sample. Now, as we tried to build the most easiest sample possible in order to easily reverse-engineer it, it is clear that the configuration is quite shorter compared to that of the real sample.

```
#BEGIN DARKCOMET DATA --
MUTEX={DC_MUTEX-KHNEW06}
SID={Guest16}
FWB={0}
NETDATA={test213.no-ip.info:1604}
GENCODE={F6FE8i2BxCpu}
INSTALL={1}
COMBOPATH={10}
EDTPATH={MSDCSC\runddl32.exe}
KEYNAME={MicroUpdate}
EDTDATE={16/04/2007}
PERSINST={1}
MELT={0}
CHANGEDATE={1}
DIRATTRIB={6}
FILEATTRIB={6}
SH1={1}
CHIDEF={1}
CHIDED={1}
PERS={1}
OFFLINEK={1}
#EOF DARKCOMET DATA --
```

(a) Complete Sample Configuration

```
#BEGIN DARKCOMET DATA --
PWD={testpassword}
MUTEX={DC_MUTEX-0XKU922}
SID={Guest20}
FWB={0}
NETDATA={cometa0.ddns.net:100}
GENCODE={STKzkP9NAjb5}
OFFLINEK={1}
#EOF DARKCOMET DATA --
```

(b) Our Sample Configuration

Figure 6: DarkComet v5.3.0 Samples Configuration

Starting from Figure 6a we can see that no password has been set by the operator as the field "$PWD = ...$" is not present. A default Mutex is also configured. It could be useful for us to recognise a DarkComet sample as it usually starts with "DC_MUTEX-" followed by a random sequence. FWB states for "firewall bypass" which gives more possibilities of attack to the operator, but it also gives more changes to be detected and afterwards caught. An other very important field is the NETDATA field, as it contains the addresses and ports that the sample will use to try to create a connection using a network socket. In this case it is configured to use the DarkComet's default port **1604**. We can then see that the INSTALL flag is set to one, which means that PERSINST and EDTPATH are also set. The malicious RAT sample will be installed into the following path `C:\Users\$User\AppData\local\temp\MSDCSC\runddl32.exe`, in order to be persistent. Once the computer is restarted, it will automatically be executed and the operator will still have control of the victim. If this three parameters are not set, once the sample thread is terminated (killed or due to a reboot), the operator will no more have control on the victim, unless he executes one time more the malicious executable. Last but not least, OFFLINEK set to one is intended to activate the offline keylogger. In fact, DarkComet permits to record each keystroke done by the victim and store it into a file on the victim's machine. Once the operator will access to those information, he just needs to ask a "copy" of that file. The alternative is to have an online keylogger that sends each keystroke to a FTP server defined by the operator. Using this technique, all keylogger files are always up to date.

Coming now to our sample on figure 6b, we can see that we set the password has been set to "testpassword" as explained in chapter 2.2.1. The mutex is as for the other sample standard and firewall bypass (FWB) is disabled. The host we choose, is "*cometa0.ddns.net*" on port 100. This has been very useful, because by changing DNS entries, we could handle a change of IP address without need to rebuild the sample. Offline keylogger, as per default, is then enabled.

In order to achieve our work for this project we need to be able to extract this configuration from a DarkComet sample, in particular, to find the password and the NETDATA information. Some other things like firewall bypass or offline keylogger are interesting topics to do some statistics, but not strictly necessary.

2.2.3 DarkComet Network Handshake

One of the important thing when it comes to analysing RAT's behaviour is the manner they use to contact the operator and create a connection. This can be done, as we did, by using a network sniffer like wireshark [17] or Tcpdump [18]. This kind of software basically records each packet sent from and to a determinate network card so that the communication to a determinate host, traffic on a determinate port or the usage of a certain protocol can be traced.

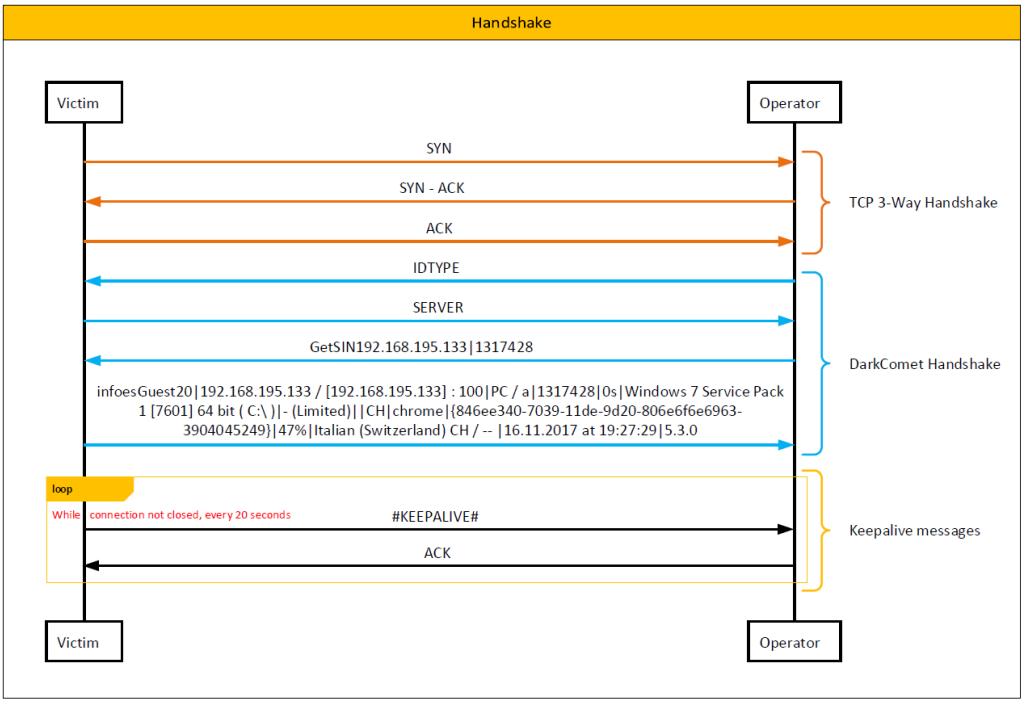

Figure 7: DarkComet Network Handshake

In case of DarkComet, the server (victim side) will try to create a network socket to the configured host and port, which if not modified is 1604 by default. By creating a socket the two parties will do a Transmission Control Protocol (TCP) three-way handshake [19] as presented in figure 7 using the orange arrows.

After this point, DarkComet starts its own handshake. All traffic during the DarkComet handshake and almost all traffic during normal operation is first RC4 encrypted and then sent through the just opened channel. For instance, the IDTYPE message and all other cyan messages are sent using this technique. So, as RC4 does not apply randomness, if the commands are encrypted with the standard password (no password set by the operator), then a message like IDTYPE will be the same for all operators that follow this condition. A more detailed description of this statement is done on section 2.9.

Following a series of synchronisation messages, the victim sends the first message with important information for the operator: it contains for example the operating system used, the country, the default browser, ram usage, keyboard keymap and version of DarkComet. Having this information, the operator can have a quick overview of the victims and estimate its "hacking-value".

Finally in yellow we see that a keep-alive message is encrypted and sent each 20 second in order to keep the network connection open, otherwise it could happen that it would be closed.

2.2.4 Operator side

When it comes to operator site, he has a nice and usable DarkComet Graphical User Interface (GUI), like the most RATs. This is very practical, as the operator does not need to be a very skilled person in order to use the software, as well as he does not need to send commands via command line or something in that style.

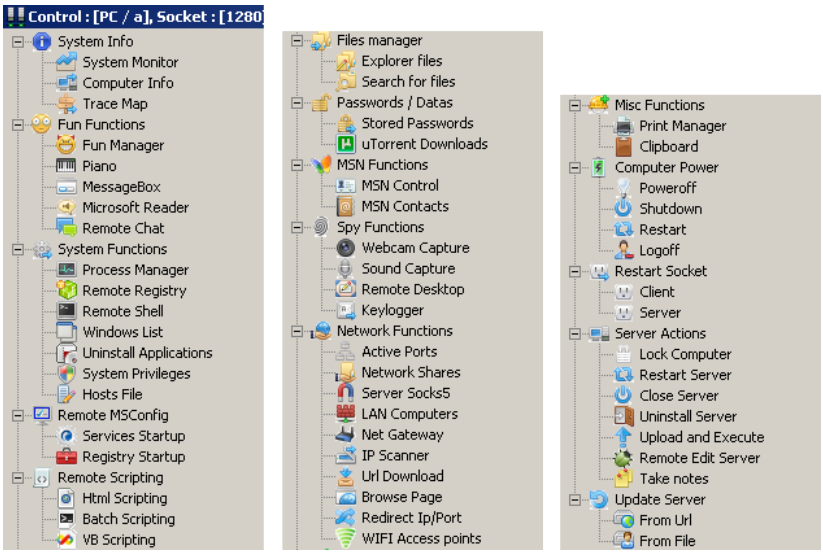

Figure 8: DarkComet Menu Entries

Figure 8 represents a photographic montage of all menu tabs of DarkComet expanded and all sub-menu items. As plainly visible, commands are subdivided in categories, represented by tabs and menu items names, or commands, are quite self-explanatory.

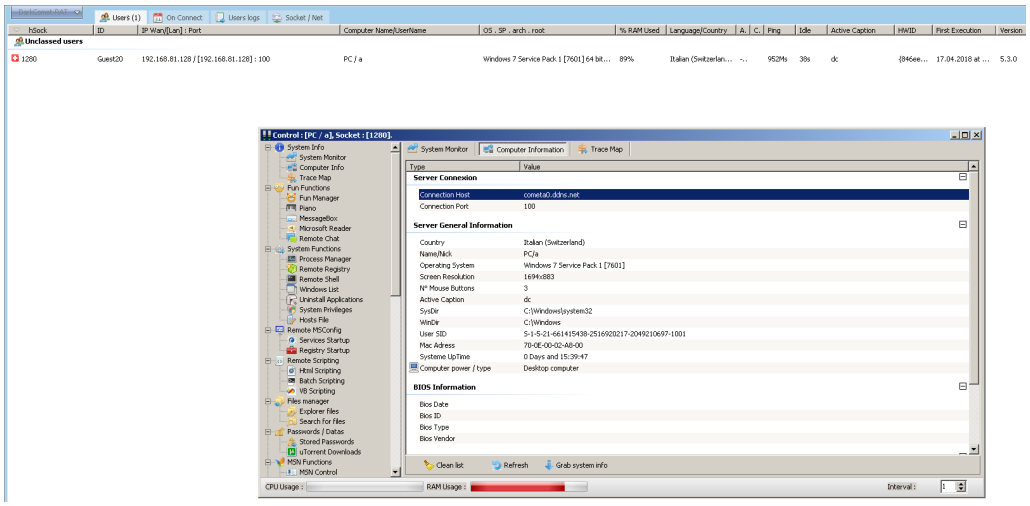

Figure 9: DarkComet Operator's GUI

Figure 9 shows a typical view from operator's perspective when he is interacting with a victim. In the background he has the main program containing a list of victims that are infected with one of its DarkComet samples, in this case only one victim is present as the figure comes from a demo of us.

By clicking on a victim the operator can open a control window where in the left side has all the menu entries presented in figure 8 and in the right part the actions and configurations for a determinate command. In the specific case of this figure, it is the view for the command *System Info* → *Computer Info* where the operator can see all the most important information about the system employed by the victim, such as operating system, system manufacturer and model, processor model, amount of ram, network cards installed and other details.

2.3 Volatility

Volatility [20] is one of the most unbelievable command line forensic tools we know! In fact the virtual memory allocated in a computer's Random Access Memory (RAM) can be a goldmine and volatility permits to analyse it. It has great plugins by default and many others are available in the internet. In order to use volatility, one must take a memory dump at a given time. For that many software are available like Winpmem [21] or Dumpit [22], or in case of using virtual machines as we did, snapshots are basically memory dumps.

Virtual memory is so important to digital forensic analysts because in contains most of the times passwords, encryption keys, decrypted data such like files, emails, chats or can even contain unsaved documents. When it comes to RATs, most of the times it not only contains the whole configuration of packed samples, but also those that were unpacked. So, imagine to be able to access to those information without need to know how the packer works because it does the work for us.

In the case of DarkComet, if the sample is packed and the system passes all the checks, in order to avoid being analysed and debugged, it will unpack itself and load the configuration information into the virtual memory. If the sample is not packed, by executing the sample it will also directly load the whole configuration into memory.

Here is when volatility comes into play. We do not use most of the great plugins provided by volatility, but we wrote our volatility plugin that uses "yarascan" to search in memory dumps using YARA rules, explained in 2.5, or regular expression [23].

2.4 VirusTotal

VirusTotal [7] was founded in 2004 as a free service that analyses files and Uniform Resource Locator (URL) for viruses, worms, trojans and other kinds of malicious content. Their goal is to make the internet a safer place through collaboration between members of the antivirus industry, researchers and end users of all kinds. Fortune 500 companies, governments and leading security companies are all part of the VirusTotal community, which has grown to over 500,000 registered users.

VirusTotal inspects items with over 70 antivirus scanners and URL/domain blacklisting services, in addition to a myriad of tools to extract signals or signatures from the studied content. Any user can select a file from their computer using their browser and send it to VirusTotal. VirusTotal offers a number of file submission methods, including the primary public web interface, desktop uploaders, browser extensions and a programmatic public and private Application Programming Interface (API). Submissions may be scripted in any programming language using the Hypertext Transfer Protocol (HTTP) based public or private API.

Virustotal private API is payware and it can give access to "Virustotal Intelligence" and "Virustotal Hunting" that are services designed to make queries over samples or analysis done by Virustotal, so one can sort only the information he wants. For instance, with the help of our professor Dr. Endre Bangerter, we can do some queries using the private API in order to download mostly samples designed as DarkComet samples. The people in the paper "To catch a Ratter" [3] had a collaboration with Virustotal, so they used "Virustotal Hunting" as researchers that permits to search for samples using a set of YARA rules, but we do not, so it is quite hard to compare results with them. In the next chapter we explain why those rules are so important to us.

2.5 YARA Rules

Basically the slogan of YARA says all we need to know:

The pattern matching Swiss knife for malware researchers (and everyone else)

YARA [24] is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples. With YARA you can create descriptions of malware families (or whatever you want to describe) based on textual or binary patterns. Each description, also known as rule, consists of a set of strings and a boolean expression which determine its logic. Let's see an example of a DarkComet rule:

```
rule DarkComet_5 : RAT
{
 meta:
  maltype = "DarkComet RAT"
  author = "https://github.com/reed1713"
  description = "Malware creates the MSDCSC directory, which is a common path used by DarkComet, as
                                                well as the mutex pattern."
 strings:
  $type="Microsoft-Windows-Security-Auditing"
  $eventid="4688"
  $data=/AppData\\Local\\Temp\\MSDCSC\\.+\.exe/

  $type1="Microsoft-Windows-Security-Auditing"
  $eventid1="4674"
  $data1=/DC_MUTEX-[0-9A-Z]{7}/
 condition:
  ($type and $eventid and $data) or ($type1 and $eventid1 and $data1)
}
```

Each YARA rule is divided into 3 sections: meta, strings and condition. Meta states for meta-data and it contains only a human readable description of the rule including author. The Strings section contains some known strings or IOCs such like created folders, IP addresses, synchronisation mutexes and many other. Finally the condition section, as the name says, gives the possibility to make groups, and try to define some better patterns.

In the concrete case of the rule above, we either want a sample that creates a folder and an executable on `C:\Users\$User\AppData\Local\Temp\MSDCSC` or a sample that contains the mutex called "DC_MUTEX-" followed by a sequence of 7 characters (numbers or letters from A to Z).

The reason why YARA rules are so important, is because many malware analysis tools use it to get information and classify samples. In the case of this work, YARA rules are heavily used with Volatility and with Cuckoo Sandbox.

2.6 Cuckoo Sandbox

Cuckoo Sandbox is *"the leading open-source automated malware analysis system"* [25]. In fact, it uses a series of utilities, such as virtual machines, virtual memory tools, network traffic analyser and other, to analyse and report malware samples.

Normally based on a physical host-system, the Cuckoo Sandbox installation forms the host and basis of the analysis environment. The guest systems, in our case the virtual machines, which represent the victims, get handled from there. The

same for all the necessary software, to record, dump, decode, decrypt and scan the samples and the traffic they generate during an analysis. There are different options how to set up and use the analysis system. So you can just use the basis and implement your own procedures with proper scripts. In the configuration files you can adjust the whole system as you like. Finally you submit a malware sample directly from the host system. The so called cuckoo agent act as a cross platform. He handles the communication and exchange of data between the host and the guest. After an analysis, the system creates a report with all results. In our case, we don't need that reporting-mechanism, because we do our proper analysis, evaluation and storage.

2.7 Virtual Machine Hypervisor

The virtual machine hypervisor is an interface between the host and the guest-system[26]. Normally the host is based on a physical system. The hypervisor act as a bridge between the hardware, with the installed operating system and the virtual machine, in the virtual network on the other side. The hypervisor manages the virtual environment and emulates the hardware resources, like Central Processing Unit (CPU), RAM and peripherals. This allows the installation of an operating system on a virtual machine. Additional there exists the possibility to make a so called snapshot of the actual state of the machine. Whatever happens to the machine, you can always come back to the saved state of the snapshot.
We use VirtualBox from Oracle as our hypervisor.

2.7.1 Honeypots

A Honeypot is an emulation of a target[27]. The basic idea behind honeypot-systems is, that an attacker would focus on the honeypot, instead of the real target. With monitoring and analyses, an attacker can be detected before successful intrusion. In another more offensive context, they are used to analyse malware, just as in the Cuckoo Sandbox. In our work we talk mostly about virtual machines and victims, instead of honeypots.

2.8 Scanning Tools

In this section, different scanning tools are briefly introduced.

2.8.1 Nmap

Nmap is a scanning program with a wide application area. It is often used by network administrators to deeply inspect the networks for which they are responsible. They can so discover security holes or connected devices without authorisation. Nmap is frequently used to scan a large amount of ports for a small group of IPs. Thus it is not designed for large-scale scans, where it would be much too slow.

Nmap, as it is not a specific scanner, contains an incredible number of parameters that are presented on a website [28] or in a book [29]. It also offers the possibility to use our own scripts written in Lua or to choose some that are available in the library [30].

2.8.2 Masscan

Masscan was designed for worldwide scans. The creators claim they can make them in less than 6 minutes for a port, on the condition that it sends 10'000'000 packets per second. According to them, they would have created the fastest program currently available [31]. In order to avoid any disruption of the targeted networks, Masscan automatically mixes the addresses. It even proposes to create random lists of IPs without scanning them. By default, the transmission rate is limited to 100 packets per second, which adds additional security for scanned networks. Unlike Nmap, Masscan does not adapt its transmission rate to the network. So there is always an error rate in sending packets that must be kept as low as possible by choosing the right speed.

Masscan has more possibilities than just telling us if a port is open or not, it also propose to grab the banners. This give us the possibility to know which program or service works on that port. This information can be used in a good way, for example by a network administrator, or maliciously by searching vulnerabilities in the system in order to attack it.

Masscan's designers have also thought about Nmap users by offering specific help in order to facilitate the learning of the program.

2.8.3 ZMap / ZGrab

ZMap [32] was created in 2013 at the University of Michigan to perform global scans. It has been used in a lot of research. Some of them are listed on Zmap's website [33], including the paper" *To Catch a Ratter: Monitoring the Behaviour of Amateur DarkComet RAT Operators in the Wild*" [3].

As the title shows, Zmap does not work alone. In reality, it is a set of tools available to people who carry out research on large networks. ZGrab was also mentioned because its function is to collect banners, once ZMap has defined the ports that are open.

2.8.4 Other Worldwide Scanner

In addition to ZMap and Masscan, which are the most popular and used scanners by researchers, there are others such as Unicornscan or Scanrand. Since they are less known, there are fewer websites that talk about them and provide examples of their use. It is also interesting to note that often tutorials on the use of scanners for research or ethical hacking are next to the ones who show how to carry out an attack or at least a malicious use.

2.8.5 Shodan

Shodan [34] is a search engine listing devices connected to the Internet, ranging from computers and servers to webcams, traffic lights and other connected objects. Shodan is mainly used by people working in security and researchers. It can therefore be used to discover security holes in a company or provide data for a study. In addition to the website, including a search bar and a map, an API is available.

For our Bachelor thesis, we received an academic access including following features [35]:

- Access up to 20 million results/month (API), including partial access to the real-time stream
- 100 Export Credits
- All add-ons (HTTPS, Telnet, view up to 10,000 search results)
- Shodan Maps and Shodan Images
- Shodan Book written by John Matherly

2.8.6 Censys

Censys [36] was designed by ZMap developers, see chapter 2.8.3. He became a competitor to Shodan. However, he's more specific in his search for vulnerabilities [37]. During our work, in late 2017, Censys left the University of Michigan to become a company. This decision was made in response to the increasing demands she was facing.

In reality Censys provides the data of the scans carried out and Google, thanks to BigQuery, has the necessary infrastructure for the requests. Censys makes its results available to researchers and students who complete an application [38]. However, there's still the BigQuery fee [39]. The invoicing is done by tera of searched data. Even if it seems a lot, it is very quickly exceeded. Tables often contain thousands of columns. For example, the following query will already use several Terabytes [40]:

```
select * from ipv4_public.current where ip = '8.8.8.8'
```

In offering the results, Censys requests in the final document a BibTex quotation as follows [38]:

```
@InProceedings{censys15,
author = Zakir Durumeric and David Adrian and Ariana Mirian and Michael Bailey and J. Alex Halderman,
title = A Search Engine Backed by Internet-Wide Scanning,
booktitle = 22nd ACM Conference on Computer and Communications Security,
month = oct,
year = 2015
}
```

We have received access from Censys, but unfortunately they are not scanning the main port we need in our work. In addition, as written before, they do more research on specific problems, such as heartbleed vulnerable devices.

2.9 How to recognise DarkComet?

Before starting a scan, it is still necessary to know how to recognise an operator. Port 1604 is the default port. However this one can be changed for any other port. In addition it is also possible that another program runs on this port. In fact, in our work, we had to deal with someone who, along the way, changed the port assignment from DarkComet to an openSSH server. So it's important to know what program works behind it. Figure 10 shows the different possible answers, when we scan a port.

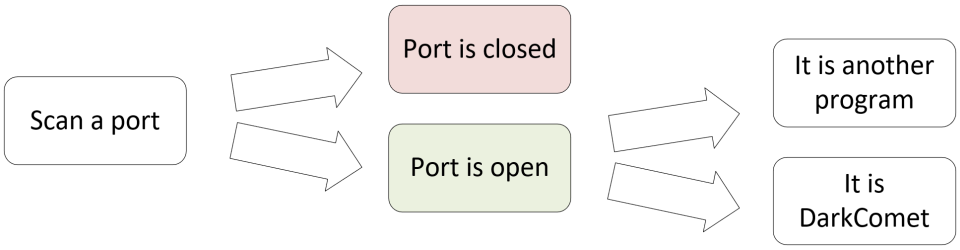

Figure 10: Possible responses when a port is scanned

When we look at the RC4 encoding of the password, explained in detail in the chapter 2.2.1, we notice that the result is a series of hexadecimal characters. This is always 12 characters long. The figure 11 gives an overview of this encrypted password that gives the IDTYPE or more generally the banner that interests us. A banner actually represents information about a program that can be collected after establishing a TCP connection or the response obtained during a User Datagram Protocol (UDP) exchange with a device connected to the internet.

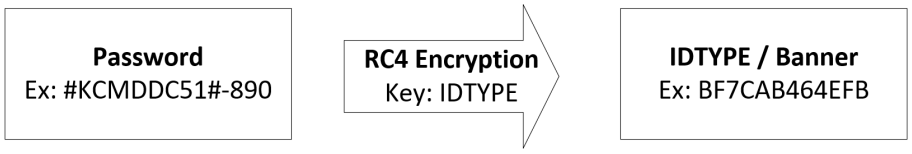

Figure 11: Password RC4 encryption in order to obtain the banner

The operator's DarkComet program sends, after a TCP connection is established, the encrypted password. The image 7 shows the transmission of these packets and this IDTYPE that interests us. In figure 12, we can also see it with Wireshark.

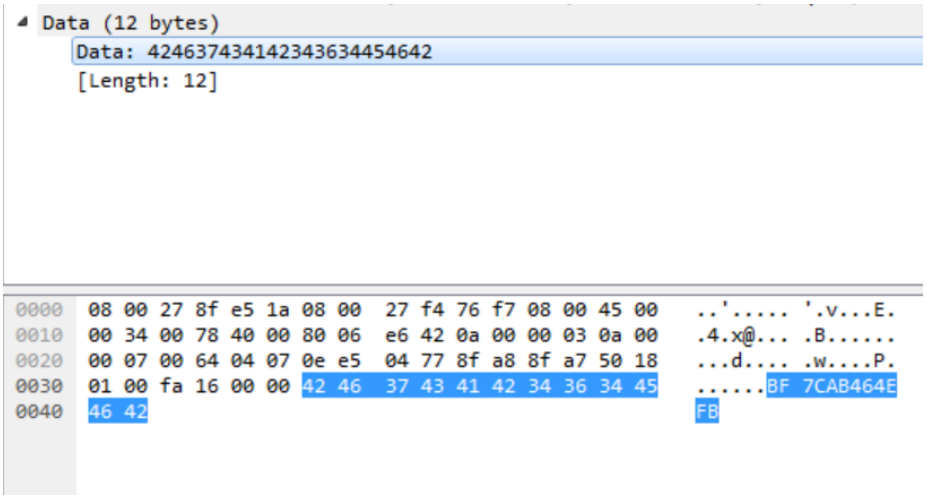

Figure 12: IDTYPE package in Wireshark

The table 1 shows the default password IDTYPE of the different versions of DarkComet [41].

Version	Default Password	Default Banner (IDTYPE)
2.x + 3.x	#KCMDDC2#-890	8EA4AB05FA7E
4.1	#KCMDDC4#-890	B47CB892B702
4.2	#KCMDDC42#-890	C7CF9C7CD932
4.2F	#KCMDDC42F#-890	155CAD31A61F
5.0	#KCMDDC5#-890	1164805C82EE
5.1+	#KCMDDC51#-890	BF7CAB464EFB

Table 1: Default passwords for the various versions of DarkComet [41]

It is important to note that this IDTYPE is only sent if the operator's computer is switched on and the DarkComet program is open. It is therefore easy to know when an operator has a good chance to connect to our environment. However on the other hand it is more difficult to find new operators by scanning networks because you have to do it when it is active. In short, you have to be in the right place at the right time.

2.10 Ethical & Judicial Aspects

2.10.1 Analysis

For the analysis and the creation of the guest system, there have to be set some boundaries. The guest system will be configured and prepared as real as possible and contains a lot of specific and pseudo personal user data. Such as personal data like photos, videos are questionable. We don't want to attack someone's privacy and abuse of their online data. So, we took a randomly generated name for the fake user and put only data on the machines which doesn't refer to a real person. Besides, all the collected data from the analyses, will not be passed over to other people or published online.

2.10.2 Scanning

As already mentioned before, scanning software can be used in a good way. They can be helpful to inform companies of a security vulnerability or bring new knowledge through research. However there is the other side of scanning where it is a malicious act whose purpose is to discover vulnerabilities to harm a business such as stealing data or sabotaging infrastructure.

Historically, the perception of scanning has changed. In the beginning, network protection was rather based on the principle of hiding. It was thus badly perceived to scan another part of Internet than its own network. There was also a greater risk that someone would complain about it. However, these measures did not stop malicious people and as time went on, scans became commonplace. Meanwhile the devices connected to the internet are regularly scanned and more and more listed. However, there are still not all people who accept them. They'll let you know it, by sending an abuse request to the Internet Service Provider, writing directly to the person who is performing the scans or by attacking back his network [42] [43].

In Switzerland, from a legal point of view, scanning is situated in the grey area. It is therefore quite understandable that Swisscom AG could not officially give us its approval. However, they tolerated our work [44] and made us aware that it is imperative to know and respect article 143bis1 of the Swiss Criminal Code [45]:

> ***143^{bis1} Unauthorised access to a data processing system [45]***
> *[1] Any person who obtains unauthorised access by means of data transmission equipment to a data processing system that has been specially secured to prevent his access is liable on complaint to a custodial sentence not exceeding three years or to a monetary penalty.*
> *[2] Any person who markets or makes accessible passwords, programs or other data that he knows or must assume are intended to be used to commit an offence under paragraph 1 is liable to a custodial sentence not exceeding three years or to a monetary penalty.*

Before starting to scan, it is also important to focus on the ethical aspect of this approach. We have therefore taken several measures to do this in the most respectful manner possible and to avoid any problems. They will be presented in a more detailed version under 3.6.1, because they are already an integral part of the design. The measures were certainly useful as we have not received any complaints.

3 Design

In this section we describe how we designed our whole system. We present the architectural design of our host and guest systems. We describe also our data flow and how we designed the automation of the processes.

3.1 Host System

The host system, with the Cuckoo Sandbox installation, is based on Ubuntu 17.10, a Linux operating system. During the project, we changed the underlying physical device. We started in the exploration phase with a desktop computer and changed during the analysis phase to a more powerful server. The more samples we had to analyse, the harder it was, with just a few virtual machines. In addition we changed our analysis cycle, for reason of quality and performance. This will be discussed more in detail in section 3.3.3. The server has the advantage to be upgradable and finally has more power, which offered us new options.

	CPU	**RAM**	**Disk**
Computer	Intel i7	16 GB DDR4	1 TB SSD 2 TB HDD (backup)
Server	2x Intel Xeon	32 GB DDR3	1 TB SSD 500 GB SSD 2 TB HDD (backup)

Table 2: Host-System hardware configuration

As soon as we changed from computer to server, we set up our 9 Machines in a strategic way.

We installed all machines with a odd number on the main Solid State Drive (SSD), where Ubuntu is installed. All machines with an even number were installed on the second SSD. We did our automation script in such a form, that it submits one sample to machine 1, the next to machine 2 and so on, sequentially. So, in the case we need to submit 2 samples, the system will choose a machine on one SSD and a machine on the other, reducing a lot the I/O traffic.

Since most malware is made for Windows, with a Linux operating system, we already got a basic malware resistance for our analysis environment. In addition we installed Suricata IDS, to log all the HTTP(S), DNS and TCP traffic. And so, in the case of an attack on our environment, we support confirmability. Another security factor is, that we worked on a separate Network, to not endanger anyone.

3.1.1 Cuckoo Sandbox - Setup and Configuration

There are several options, how to use the Cuckoo Sandbox [46]. The standard Cuckoo setup works with a virtual machine as guest. There is also the possibility to setup Cuckoo with a physical machine as guest. And finally there is the distributed Cuckoo version, with a REpresentational State Transfer (REST) API, for at least two Cuckoo-nodes. This makes sense, if you providing a farm of honeypots, for big-scale analyses.
We chose the standard version for our project. We got one physical host and several virtual guests on it. The version with physical host doesn't fit to our ambit, because we have to restore the snapshots of the victims, after every analysis. The design of the distributed Cuckoo is very scalable, what is a great benefit, but to heavy for our project.

We configured the Cuckoo environment as light as possible. So we don't lose computational capacity doing unused reporting-mechanisms. We just use the virtual memory dump and the TCP dump. With proper scripts we decode, decrypt and analyse these files. More details about that process in the chapter 3.3.3 and about the implementation of the proper scripts or how the configuration has been done in chapter 4.

3.2 Guest System

The virtual machines are based on the VirtualBox hypervisor. Alternatively KVM, VMware Workstation or XenServer could be used. As the operating system we use Windows 7. In the wild it's still a very naturally occurring version of Windows. If we want to have a real-looking victim, another factor is the power. You almost want find today computers, with only one CPU-kernel and less than 4GB of RAM. So, by reason of authenticity, the machines must have enough power. So we designed the model for our victim as follows:

Windows 7, 64-bit
4GB RAM, 2 CPU (kernel)

Since our victim should be as real as possible, we defined a user with personalised software, accounts, folder-hierarchy and data. Described in more detail in the next chapter.

There exist 9 virtual machines, based on this model. Machines 1 to 6 for the online-analyses and machines 7 to 9 for the offline analyses.
For reasons of performance and disk space, the 3 virtual machines of the offline analyses have been designed to 2GB of RAM only. So, the generated memory dump will be also 2GB and it takes less time to analyse it and also it takes less disk space.

3.2.1 Victim

The victim is David Munoz. We choose randomly a first and last name. We created a standard Windows-user under this name. We installed all the standard software, you could find today on a users machine.
List of installed software:
Adobe Acrobat Reader, Adobe AIR, Adobe Shockwave Player, Electrum, Euro Truck Simulator 2, Google Chrome, Java 8, Microsoft .NET Framework, Microsoft Office 2016, Microsoft Silverlight, Microsoft Visual C++ (2005, 2008, 2010, 2012, 2013, 2015), Mozilla Firefox, Python 2.7, qBittorrent, Skype, Spotify, Teamviewer, VLC Media Player, WinRAR.

We created a Google account, also used for Skype and Spotify:
dave.munoz1990@gmail.com
We installed the Electrum Bitcoin Wallet, created a local account and generated some key pairs. Then we archived a file in a private folder, with the restore data. This data is needed to restore the wallet, in the case you lose your password. Or it can be used to crack it.
We placed some "personal" data, such as a Curriculum Vitae (CV), essays, photos and so on.
The basic idea was, that we offer something to discover for the operators. They should have the feeling to be on a real victim-machine, with real sensitive data.

The superficial difference of the first 6 machines is the actual activity of the victim. One victim is surfing in the Web, another reading an article or watching a clip. In addition we set a different desktop background for every victim. This would be one of the first things a remote desktop operator notices.

Machine 6 is our exception. We installed additional a virtual webcam service, named Magic Cam. For operators with spy ambitions, they would have something to connect. More details in section 3.2.4.

3.2.2 Virtual Network

All the virtual machines are working in a virtual network, in host-only mode. This mode is recommended by Cuckoo documentation [46] and has the advantage of more flexibility in its configuration. Thus, this means that no machine can communicate to external networks. During the online analysis, the operator must have connection to the infected machine over the internet. In the host-only mode, the machines have no connection to the external, neither to the internet. With our iptables rules we forwarded the machines from the host-only network, to have real internet access. This is going to be explained more in detail in chapter 4.2.1.

Every machine has it own ip address and mac address. This is a must, because the network module of the hypervisor, would not work correctly, if several machines with the same addresses are running.

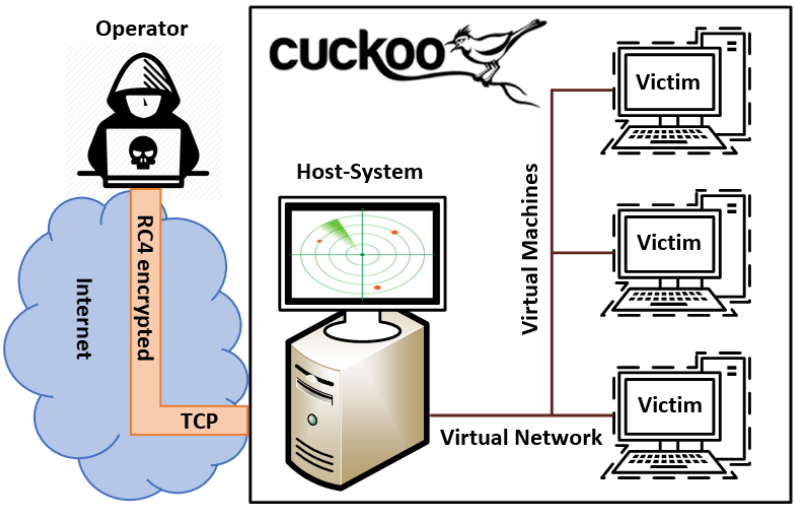

Figure 13: Network Environment

3.2.3 Virtual Machine Hardening

When it comes to malware analysis, one needs to think on what those malware will do. In fact, many RATs will check for some evidence that they are in a virtual environment, and if that is the case they will not do anything malicious. Otherwise they will proceed with the exploit. Some checks that malware will do can be such like Media Access Control (MAC) address check, as the first 6 digits of a MAC address, indicate the network card vendor. VirtualBox is also considered as a virtual network card vendor, so they have their own 6 digits. An other example could be some registry keys in windows that need to be modified. Or even default IP ranges done by virtual environments like VirtualBox that gives 192.168.56.0/24 to host-only networks by default.

Fortunately someone already had all those problems. After a search in the internet and a speech with our adviser Reto Inversini, we found this great tutorial [47], so that we can mitigate the probability of virtual machine detection by a sample. As very good described in the tutorial, two scripts are created using an other computer. The first one is a bash script to give properties to the virtual machine, such like change their MAC address using a virtual box command. The second one is a PowerShell script to use inside of the virtual machine, once it has been installed. It will change some registry keys, change the computer name and the username, etv. You can see some result by checking "systeminfo" into the virtual machine as described in figure 14, in our case we emulated one of our private Hewlett-Packard (HP) laptop.

Item	Value
OS Name	Microsoft Windows 7 Professional
Version	6.1.7601 Service Pack 1 Build 7601
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	HOMESTATION
System Manufacturer	Hewlett-Packard
System Model	HP Pavilion dv6 Notebook PC
System Type	x64-based PC
Processor	Intel(R) Core(TM) i7-2630QM CPU @ 2.00GHz, 2261 Mhz, 4 Core(s), 4 Logical ...
BIOS Version/Date	Hewlett-Packard F.1B, 05.10.2011
SMBIOS Version	2.5

Figure 14: Systeminfo done on VM after hardening

A good method to check if a virtual machine is enough hardened, is using Pafish (Paranoid fish) [48]. It will check the common patterns that known malware will check to see if they are in a virtual environment or not. Then it shows a OK/Fail report for each action, so one knows where needs to work.

3.2.4 Virtual Webcam

For the Machine06, we configure the virtual webcam service Magic Cam. We record a video of an empty workplace, in front of a computer. On the machine, we play this video via VLC-Player and Magic Cam shows it like a normal webcam stream. To hide the two running applications on the desktop, we use Taskbar Hide. The only lack is, that on the operator side, while listing all the webcams, the VLC-Player is also showed as a "webcam". This happens most likely because we must install a dll that is an add-on for the VLC-Player, to ensure the functionality of the virtual webcam service.

3.3 Data Flow

Three people working on a such project can pose some problems like priorities and dependencies. So we decided to design a data flow to clarify who needs what and when. A graphical overview is described by figure 15.

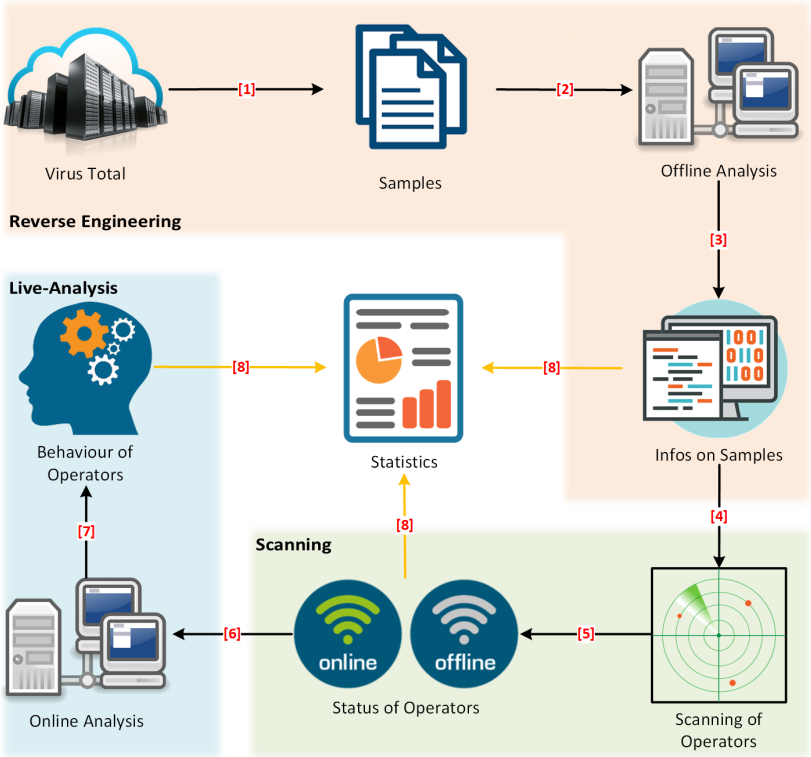

Figure 15: DarkComet Tracker Data Flow

1. Regularly download samples from VirusTotal.
2. Check whether they are DarkComet.
3. Extraction of the configuration of the samples.
4. Regularly scan extracted operator's addresses.
5. Check if operator is online.
6. For online operators we submit their sample to the Cuckoo system and let them take control of our virtual machines.
7. The analysis sessions were recorded and in a second step we decrypt the network traffic, to analyse which commands have been executed.
8. Based on this data we categorise operator's behaviour.

3.3.1 Reverse Engineering

Reverse Engineering is the whole part from the download of the samples, until the extraction and storage of the configurations of DarkComet.

We created a volatility plugin that is called by cuckoo in order to search in memory dumps for DarkComet configurations.

VirusTotal Download(1)
We imagined this step as a script of us that does the query, downloads the results and decompresses it to the right folder. In reality, that unfortunately it did not work that way.

We started this work thinking that sooner or later we would receive an account from Virustotal to use their "Hunting" app. Or at least we thought we would receive a private API key in order to perform our own queries and download the samples. Unfortunately this never happened, so in accordance with our professor Dr. Endre Bangerter, more or less every day we became a variable number of samples from him in a compressed format. Given that, we "just" had to decompress the samples into the right folder. Of course we discussed during the meetings kind of query to make on virustotal, so that we could maximise our results considering the the means we had.

Offline Analysis (2)
Once we have the samples in the right folder, we submit them to our Cuckoo Sandbox environment for a so called "Offline Analysis" even if the virtual machine still has internet access. This kind of analysis is very short, like less than a minute, so that the samples turn in our virtual machines, unpack themselves if they are packed and we can afterwards detect if they really are DarkComet samples. In the positive case we try extract their configuration.

Information on Sample (3)
As already mentioned in the last step, we try to extract the configuration for sample that we know are DarkComet. We store then the information coming from the configuration of the samples like hostname, port, password, firewall bypass, offline keylogger and DarkComet version into a Comma-separated Values (CSV) file. Moreover, in the same file, we store the country from where the hostname comes (if resolved), the sample hash and the name of the process where the configuration was found in the virtual machine.

3.3.2 Scanning

The scanning takes the results obtained by the reverse engineering in order to provide the active operators for the live analysis. Operators can then potentially connect to our machines.

Scanning of Operators (4)
The results of the new samples, obtained by reverse engineering, are added to the target list, before beginning scanning. In this step we are most interested in the hostname and port, which are essential elements to start a scan and identify an operator. Of course we need also to keep track of the sample, so we can transmit it later for live analysis. It is important in this step to sort the results received, as it is common that the same operator appears several times. There are also often private addresses that are used and that we cannot exploit.

Once the To-Scan list is ready, each operator is scanned with Nmap using the customised script, presented in point 3.6.4. Any domain name that does not exist anymore, is placed in the Trash-File, in order to keep a list up to date. Before placing them in the trash, the state of the network is checked to make sure that there is not an interruption that would lead to deleting the whole list.

Status of Operators (5)
Through Nmap's customised script, it is possible to determine if an operator has the DarkComet program open or not. For those where this is the case, they are added to an Online file that is transmitted for live analysis in order to record interactions. On the other side, those who are not active, remain in the To-Scan list, in the expectation that the operators will be there on the next scan.

Operators that have been transmitted for live analysis will wait in an Analysed file for a defined time interval, which is currently at least 4 hours, before being transferred back into the To-Scan file. We thus avoid a sequence of interactions with the same person and hope that we will not be perceived too quickly as a decoy.

3.3.3 Live Analysis

Live analysis describes the whole process, where we let an operator connect to our virtual machine during a online analysis and record everything he does. The stored data is used to evaluate the behaviour of the operator.

In the early exploration phase an analysis took 15 minutes. We changed that to 60 minutes. So an operator has enough time to notice, that a infected victim is online and can then follow his malicious behaviour. Up to 6 machines running in parallel is supported for a online analysis.

Online Analysis (6)
The online analysis starts with the submission of the RAT sample. The virtual machine starts up with a safe snapshot and load, via the cuckoo-agent, the RAT sample, an executable. Then this RAT get executed and the infection procedure happens. Once the victim is infected, the RAT tries to contact the operator over the internet, for establishing a connection. From here on the operator can connect, overtake the machine and do what ever he like to. The machine stays in this state for 60 minutes.

Data Acquisition and Evaluation (7)
During the online analysis TCPdump dumps all the network traffic to or from the machine. As the analysis is completed, the whole log is stored in a Packet CAPture (PCAP) file. With tshark [49], a command line version of wireshark, we decode and filter this PCAP file. Only the traffic corresponding to the port-number of the RAT will be taken. The output is stored in the human readable JavaScript Object Notation (JSON) format. The relevant information is stored in each data segment of this JSON file. This data segment holds all the information about the commands, which the operator executed and the data transferred. It's encrypted with the RC4 cipher and the corresponding password of the RAT, detected in the offline analysis. Every JSON data segment has it own timestamp. So we can determine, what the operator does and when he does. This information get filtered out with proper scripts and get stored in result files. This files are the basis to do analysis, categorisation and statistics relating to the behaviour of the operators. More details about the scripts in chapter 4.6.2 and 4.6.2.

3.3.4 Statistics

Reverse Eng. Statistics (8)
We want also check what kind of configurations of DarkComet are used the most in the wild. For this, we take trace of the configurations and in the end a statistic over some interesting fields will be done.
At this stage, interesting fields could be versions of DarkComet, password types and if they set one or not, understand if they use more hard-coded IPs or they prefer dynamic domain names, what kind of ports are used apart from the default one (1604), and so on.

Scanning Statistics (8)
First, statistics on the basic information of the operators will be done. This includes pieces of information, such as country, port or ISP that are used. To get country information we used geoiplookup. For ISPs, we found ipinfo.io [50] , that provides an API. With the paid version, it would still be possible to differentiate the type of Internet Service Provider (ISP), such as hosting provider, business,...
If we need additional information about an address, we use Réseaux IP Européens (RIPE) [51], the IP addresses coordination centre in Europe.

During the scans, the active operators are listed in an activity file. This should allow us to discover a little bit better the behaviour of operators. With this data, statistics such as activity days or hours can be made.

Live Analysis Statistics (8)
The basic idea is to get an overview of the operators and finally to determine the categories of RAT users. We expect a lot of so called script kiddies, which use this kind of malware for reason of fun. They would probably use more the fun-functions of DarkComet, while blackmailer would probably use more often the spy and data-theft functions. Some operators are more passive and another more active or even aggressive. We gonna try to split up this different groups of interest and behaviour. And we want to evaluate in general, how the analyses worked. Such as used commands, interaction time or the difference between the machines.

3.4 Automation

The whole automation has been implemented in the "*apscheduler.py*", which is described in detail, in chapter 4.6.4. The idea of the apscheduler is to automate the whole procedure. All three parts, meaning the reverse engineering, the scanning and finally the livw analysis, should be done automatically. Like that, we achieved to let work the whole analysis-environment during night and day, even the weekend.

Every 60 minutes, a scan gets started. After the whole scanning is done, every operator that is online will be logged in the activity file. We check that the operator has not been analysed during the last 4 hours and we put it into a list of online operators. At the end of the scanning, the online analysis part gets called.

In the online analysis part, the list with the online operators is used. Up to six RAT samples can be submitted to the Cuckoo system and then analysed.

On the other side, also the offline analyses are started from here automatically. Every day at 02:00 in the morning, in theory there would be a download from Virus Total. Every day at 04:00 in the morning, the offline analysing gets started. Every new sample gets submitted to the Cuckoo system and so analysed to extract its configuration.

The convertion, decoding and decryption are part of the "pcapparser.py", which is described in chapter 4.6.2.

3.5 DarkComet commands documentation

A big part of this work for the reverse engineering part, was also the creation of a documentation of the commands doable with DarkComet. We had full choice on how to do it, and then we thought, as there are quite a lot of commands, one of the best and simplest methods was to simulate it in a virtual environment and then build some sequence diagrams in order to describe which command are sent by whom and what impact do they have.

To do this, we have installed on a private computer two virtual machines, one that simulates an operator (so with the DarkComet client and GUI) and the other that simulates a victim. We installed then Wireshark in the victim side, so that we could sniff the whole communication on a given port. As attack sample we used the examples sample described in this document.

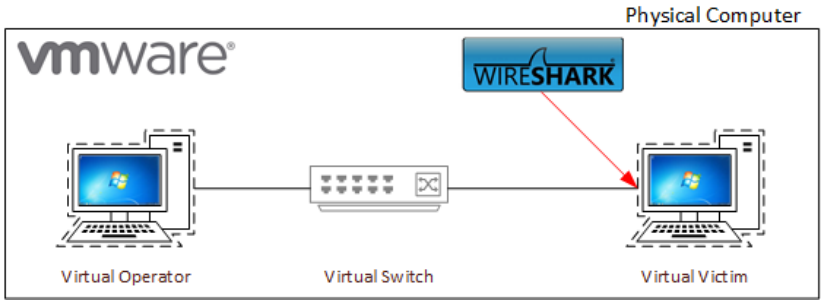

Figure 16: Environment for production of DarkComet documentation

In figure 16 we show a representation of the environment used for the production of the documentation. Wireshark creates a network dump which can be converted into a JSON file with tshark. For this, a script has been done so that the network dump could be converted in JSON, and the encrypted contents in the JSON file could be decrypted. Once done that we store the whole decrypted communication captured by wireshark on a given port, and we let the script create a text file in a such way that this great website [52], can build a sequence diagram for us. Unfortunately they do not have an API so that the whole process could be scripted, because of that we let the script generate the text file and then it is enough to copy and paste it to the website. Moreover, in the documentation some screenshots are provided in order for the reader to be clear of what functionality of DarkComet we speak.

The whole documentation is available on Appendix A. As some diagrams are quite huge, this appendix is available only in PDF version, because it would be difficult to print.

3.6 Scanning

3.6.1 Approches éthiques

Nous avons appliqué différentes mesures pour garantir des scans les plus éthiques possibles. Voici ce que nous avons mis en place:

Informer le fournisseur d'accès à internet
En informant le fournisseur d'accès à internet, celui-ci est au courant de ce qu'il se passe sur le réseau et la raison pour laquelle les scans sont effectués. Il peut aussi nous transmettre les éventuels "abuse requests" qu'il reçoit pour que nous enlevions ces personnes de nos scans. C'est d'ailleurs souvent cette méthode qui est employée pour en demander l'arrêt. En communiquant notre projet au four-

nisseur d'accès à internet, il existe cependant le risque que celui-ci le refuse tout bonnement. Il faut alors trouver une autre alternative. C'est principalement pour cette raison que nous n'avons pas informé Sunrise SA. Nous avons cependant fait la démarche chez Swisscom SA, où la majorité des scans ont été effectués.

Communiquer le but des scans aux cibles
Le scanning peut être utilisé pour de bons comme de mauvais buts, il est donc nécessaire de communiquer notre identité et nos intentions aux cibles. Les administrateurs des réseaux savent ainsi qu'ils ne sont pas en train de subir une attaque. Nous avons fait cela à travers un site internet, en réalité même deux. En premier, nous avons mis en place un site sur WIX [53]. Ce choix a été fait car il n'était d'abord pas possible de configurer notre réseau pour avoir notre propre serveur, comme expliqué au point 3.6.2. L'image 17 donne un aperçu de ce que les personnes peuvent voir. Nous avons choisi une adresse aussi explicite que possible: `https://bfhthesisnoscan.wixsite.com/noscan`.

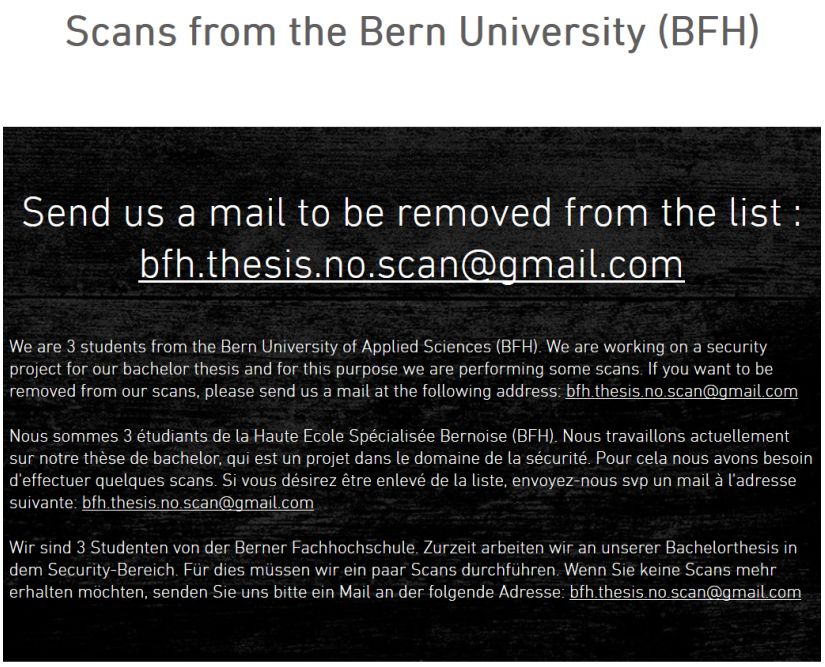

Figure 17: Site web sur Wix [53]

Par la suite, nous avons mis en place un serveur Apache, version 2.4.27. Comme cela, lorsqu'ils reçoivent un scan, ils peuvent voir sur le port 80 de notre adresse IP le site Web. En effet, les administrateurs réseaux effectuent des scans en retour pour essayer d'identifier les auteurs.

En plus du site web, nous avons ajouté un nom de domaine explicite:
`bfhnoscangmailcom.hopto.org`. Il leur permet de deviner qu'il s'agit d'une adresse mail et peuvent ainsi nous contacter. Le nom de domaine est fourni par no-ip [54]. Il doit être renouvelé tous les 30 jours dans la version gratuite.

Cependant, il y a de fortes chances que personne ou très peu de gens l'ai vu. Les fournisseurs d'accès à internet donnent par défaut un nom de domaine tel que `233.95.195.178.dynamic.wline.res.cust.swisscom.ch`. Étant donné que les gens reçoivent, lors des scans, l'adresse IP, ils ne vont pas avoir tous les noms de domaine liés à l'adresse. En effet, ils vont en recevoir que une. Le concept des noms de domaines permet une résolution rapide du nom à l'IP, mais n'est pas conçu pour le sens inverse. Ceci n'est pas grave, étant donné qu'il y a au minimum le site internet qui est à disposition.

Donner la possibilité de quitter les scans
A travers le site internet, les personnes peuvent nous envoyer un e-mail pour nous demander de retirer leur(s) adresse(s) IP. Nous avons aussi réfléchi à la possibilité de mettre à la place de l'adresse e-mail un formulaire sur le site web. Nous avons cependant balayé cette idée en pensant à la possibilité qu'une personne puisse "s'amuser" à remplir des formulaires qui ne concernent pas son réseau. Nous n'aurions ainsi plus la possibilités de trier les vrais des fausses demandes. Il est encore important de noter que dans notre travail, nous n'avons effectué aucun scan répété en-dehors des adresses utilisant ou ayant utilisé DarkComet.

Définition claire de chaque scan
Pour chaque scan, il est clairement défini quelles IP-Ranges et quels ports vont être ciblés. En faisant cette réflexion, on évite le plus possible des scans trop intrusifs ou trop intensifs.

Effectuer des scans aléatoires et pas incrémentals
Afin d'éviter d'utiliser trop de ressources sur les réseaux scannés, pouvant d'ailleurs même aboutir à une attaque DOS, il est impératif de mélanger les adresses IP scannées. En plus de cette mesure, il peut même être envisagé de diminuer la vitesse du scan. Les adresses aléatoires et une vitesse réduite permettent aussi d'être moins vite repérés par les administrateurs réseaux.

Surveiller les scans
En surveillant le scan, nous pouvons intervenir, à tout moment, pour l'arrêter, si nous découvrons un problème. Pour pouvoir y accéder en-dehors des heures de cours et réagir au plus vite, nous utilisons TeamViewer [55].

3.6.2 Défis liés au réseau internet

Durant notre travail, nous avons dû faire face à différents défis liés à l'infrastructure. Ils sont présentés ici car ils nous ont aussi coûté du temps, de l'énergie et influencé nos choix.

En premier, une demande a été faite à l'école pour effectuer les scans sur le réseau de la Berner Fachhochschule (BFH). Celle-ci a été refusée et il a donc fallu trouver un autre moyen pour essayer NMap. N'ayant donc aucun réseau à disposition, nous avons d'abord utilisé les services d'un fournisseur Virtual Private Network (VPN). Nous avons travaillé un mois avec Express VPN [56]. Il a l'avantage d'offrir une fonction qui arrête tout trafic sur internet si la connection

VPN se coupe.

Par la suite, nous avons eu le réseau de Sunrise SA [5]. En attendant que le service informatique le patche jusqu'au laboratoire, où nous avons effectué notre travail, il a été possible de déjà commencer nos essais dans le bâtiment Research Institute for Security in the Information Society (RISIS). Malheureusement sur ce réseau, personne n'a pu nous communiquer le mot de passe du routeur installé. Nous ne pouvions donc pas mettre en place un serveur Web indiquant l'origine des scans. En contactant Sunrise SA pour voir si il y avait une possibilité de réinitialiser le mot de passe, il s'est avéré que la ligne était tellement ancienne qu'ils ne voyaient même plus le routeur dans leur système. Elle va d'ailleurs être coupée cette été et nous avons eu de la chance de pouvoir terminer notre thèse sur ce réseau.

Nous avons aussi très vite dû faire face à des coupures de réseau, même si nous ne possédions pas un autre scanner que Nmap à ce moment là. Celui-ci ne demande pourtant pas beaucoup de ressources comparé à Masscan ou ZMap. Vraisemblablement, le routeur ne pouvait pas faire face à ce genre de requêtes. Par contre, les scans étaient encore plus lents et instables à cause du DNS qu'Ubuntu utilisait automatiquement de manière locale. Nous avons alors finalement utilisé ce réseau pour effectuer les analyses des samples. En l'utilisant que pour un but, la qualité était acceptable, d'ailleurs, nous avons eu la sensation qu'elle s'est même un peu améliorée au fur et à mesure du temps.

Pour le réseau de Swisscom SA [6] nous avons dû faire face à une longue attente jusqu'à son activation. Durant cette période, nous avons fait des recherches pour avoir un accès à travers un serveur dédié virtuel (VPS). Nous avons finalement eu notre connexion activée chez Swisscom SA aux environs du 18 avril 2018. Swisscom SA s'étant d'ailleurs excusé du délai d'attente.

Nous avons dû très vite nous rendre à l'évidence que le routeur fourni par Swisscom ne répondait pas à nos exigences. C'est l'Internet-Box light qui nous a été envoyée. L'interface de celle-ci ne nous permettait plus ou moins aucune configuration et il n'était donc toujours pas possible d'ouvrir le port et de le rediriger vers le serveur Web. Nous avons d'abord essayé de résoudre ce problème en prenant un routeur privé. Cependant, s'il n'est pas fourni par Swisscom, il a besoin d'un paramètre beaucoup plus spécifique, l'option 60 du Dynamic Host Configuration Protocol (DHCP), afin de recevoir une adresse du réseau Swisscom SA. N'ayant pas ce paramètre, nous sommes passés dans un Swisscom shop où ils nous ont généreusement échangé le routeur contre l'Internet-Box standard. A ce moment-là, nous avons pu mettre notre serveur web en place et démarrer avec Masscan, tout en gardant des scans éthiques.

Vitesse des réseaux

Les réseaux de Sunrise SA et Swisscom SA ne sont les deux pas très rapides. En terme de vitesse d'upload et de download, ils ont environs les mêmes valeurs. L'image 18 nous montre les résultats du test de vitesse sur le réseau de Swisscom SA. Nous sommes éloignés des taux de transfert employés par les chercheurs qui sont souvent de minimum 1 Gbps [43].

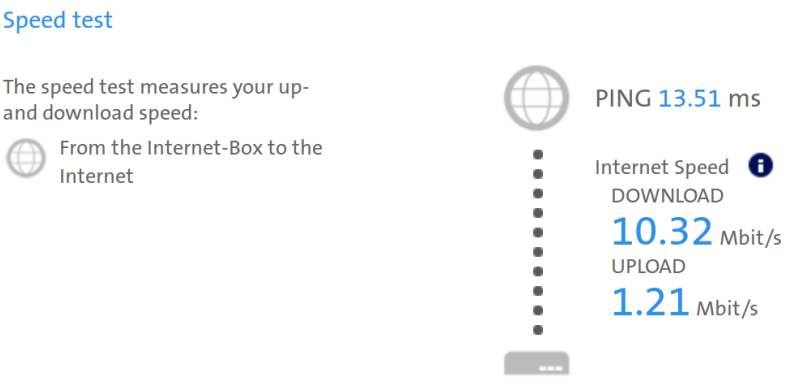

Figure 18: Speed Test sur le réseau Swisscom

3.6.3 Différents types de scans

Dans le cadre du scanning, le travail se sépare en plusieurs parties. En effet, ce n'est pas un, mais plusieurs scans qui sont effectués pour atteindre différents buts établis. Ci-dessous, ces différents scans sont présentés, un aperçu visuel est visible dans la figure 19.

Scans des opérateurs provenant des samples

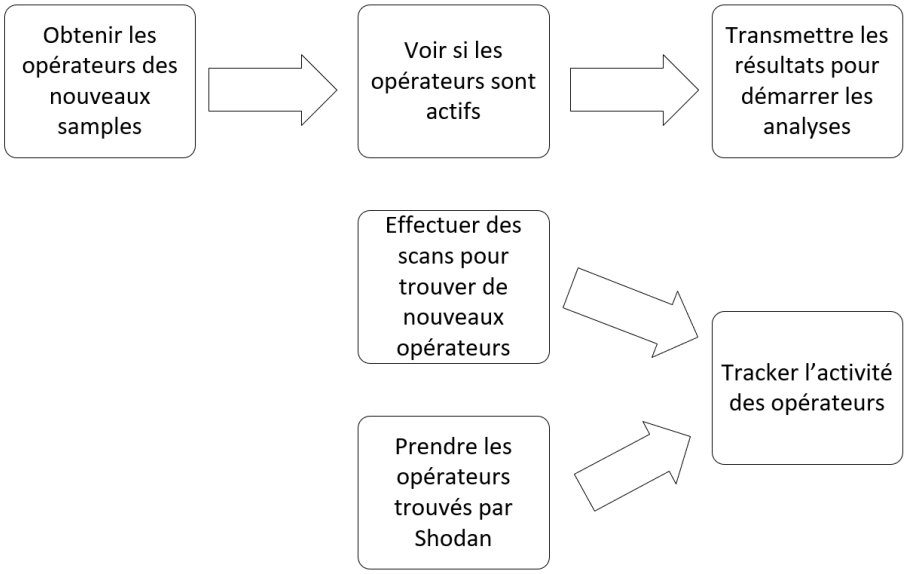

Figure 19: Aperçu des différents types de scans

Celui qui est le plus évident et qui dépend du reste du projet, est le scan des opérateurs trouvés lors de l'extraction des données des samples. Grâce à ces scans, comme dit dans le chapitre 2.9, il est possible de définir si un opérateur est actif, ou du moins s'il a le programme ouvert. Il est ainsi possible de cibler les analyses lives qui vont être démarrées. Nous pouvons donc maximiser nos chances que la personne se connecte et effectue des commandes.

Chercher des opérateurs sur Shodan

Un autre but de ce travail est de trouver de nouveaux opérateurs dans le monde. Shodan est un des moyens les plus simples pour faire cela, offrant une grande base de données.

La recherche des opérateurs peut se faire de deux manières. La première est la recherche d'une banner spécifique. Dans le tableau 1, les banners par défaut, que nous recherchons, sont listées. La deuxième manière est d'effectuer directement une recherche de produit, tel que Apache, DarkComet,... Au début de notre travail, seule la banner de la version 5.1 était vue comme un produit DarkComet. Cependant, durant notre thèse de Bachelor, cela a évolué et toutes les banners par défaut sont maintenant perçues comme un logiciel DarkComet.

Tout d'abord, la rechercher des nouveaux opérateurs s'effectuait de façon manuelle,

ce qui demandait un investissement de temps conséquent. Par la suite, la recherche a été automatisée et s'effectue maintenant une fois par jour à travers l'API de Shodan. Pour démarrer automatiquement le script permettant de récolter les opérateurs, Crontab, le planificateur de tâches d'Ubuntu a été utilisé.

Scans pour trouver de nouveaux opérateurs
Finalement, il existe la possibilité de chercher soit même de nouveaux opérateurs aux moyens de scans. Cela est un travail de fourmi car il faut scanner beaucoup d'adresses dans l'espoir de découvrir un opérateur.

On peut attaquer cette tâche par deux manières. La première, la version "Brute-force", qui consiste à scanner simplement toutes les adresses qui existent. On enlève bien sûr les adresses privées ou réservées, ainsi que les IP-Ranges connues qui ne souhaitent pas de scans. Il est aussi préférable d'enlever au mieux les adresses gouvernementales et militaires pour éviter les réclamations. Avec cette façon de faire, on attend finalement jusqu'à la fin de la thèse de Bachelor, pour voir ce que l'on a obtenu. Il est alors trop tard si l'on désire ajuster un quelconque paramètre. Avec cette méthode, il y a de fortes chances que l'on scanne beaucoup plus d'adresses pour obtenir ne serait-ce qu'un résultat.

L'autre possibilité, celle pour laquelle nous avons opté, est de cibler les scans. Nous avons donc la possiblité d'essayer différentes façons d'approcher le problème et de modifier la démarche selon les résultats obtenus. Ce choix est aussi plus logique, vu l'infrastructure que nous disposons et les problèmes auxquels nous avons dû faire face, qui sont expliqué dans le point 3.6.2.

Nous avons essayé différentes démarches pour trouver au mieux les opérateurs, les résultats de chaque tentative seront ensuite présentés au point 5.2.3.

1. **Scan par Autonomous System Number (ASN)**: En premier, nous nous sommes basés sur les numéros AS des adresses IP découvertes par Shodan. Comme cela, nous ciblons des réseaux où un opérateur a déjà été aperçu. Pour avoir tous les réseaux d'un Autonomous System Number, nous avons effectué des requêtes sur RIPE [51], qui est le centre de coordination des adresses IP en Europe. Pour encore mieux cibler notre recherche, nous avons trié chacune des adresses avec geoiplookup [57], pour ne scanner que celles qui se trouvent dans le pays de l'opérateur.

2. **Scan par Pays**: Nous avons aussi choisi de scanner tout un pays. Pour cela, nous avons utilisé la base de données GeoLite2 Country [58] mis à disposition par MaxMind. De plus, nous avons aussi ajouté un port supplémentaire, le 81. Il est le 2ème plus utilisé après le 1604, qui est le port par défaut.

3. **Scan par Hosting Provider**: Un hosting provider, OVH, apparaissait souvent dans les résultats de Shodan. Nous avons donc débuté par ce réseau puis nous nous sommes étendus en nous basant sur un article présentant les hosting providers qui comptent le plus de malwares [59] en se disant qu'ils peuvent potentiellement aussi abriter des opérateurs DarkComet.

Tracker l'activité des opérateurs trouvés
Comme pour le scan des opérateurs des samples, nous regardons si un opérateur est actif ou non. Pour cela nous scannons toutes les heures, aussi grâce à crontab, les adresses trouvées par Shodan ou à travers les scans. Les résultats serviront, par la suite, à créer des statistiques et découvrir un peu plus les personnes cachées derrière DarkComet.

3.6.4 Infrastructure de scanning

Comment nous avons choisi les programmes de scanning
Le choix des programmes pour le scanning a été en grande partie influencé par l'infrastructure que nous disposions. N'ayant tout d'abord pas la possibilité d'indiquer qui nous sommes à travers un serveur web, il a fallu trouver un autre moyen. Nmap s'est imposé à nous, offrant la possibilité d'ajouter une string lors de l'échange des paquets TCP. Il nous permet ainsi de laisser un petit message indiquant qui nous sommes. L'image 20 nous montre un aperçu de ce fameux paquet.

```
5c dc 96 98 3d ea 28 b2  bd af bc ac 08 00 45 00   \...=.(. ......E.
00 56 6e ba 00 00 27 01  a2 29 c0 a8 01 70 c0 a8   .Vn...'. .)...p..
00 03 08 00 71 5e ca d7  00 00 42 65 72 6e 20 55   ....q^.. ..Bern U
6e 69 76 65 72 73 69 74  79 20 68 74 74 70 73 3a   niversit y https:
2f 2f 62 66 68 74 68 65  73 69 73 6e 6f 73 63 61   //bfhthe sisnosca
6e 2e 77 69 78 73 69 74  65 2e 63 6f 6d 2f 6e 6f   n.wixsit e.com/no
73 63 61 6e                                        scan
```

Figure 20: Message transmis par Nmap qui indique qui nous sommes

Nmap nous offre également un éventail de scripts dans sa bibliothèque [30]. Le script Banner [60] nous permet ainsi déjà de récolter toutes les banners envoyées dans les 5 secondes par la machine scannée. Ceci nous sert à reconnaître un opérateur. En le modifiant, nous avons pu ajouter des messages qui indiquent que c'est un DarkComet. Pour les mots de passe par défaut, la version est également ajoutée.

Le gros problème de Nmap réside pourtant dans le fait qu'il est trop lent pour des scans de grandes envergures. En effet, cela nous a pris plus d'une heure pour scanner 16'384 adresses depuis un réseau VPN. C'est d'ailleurs le maximum d'hôtes possibles si on désire le faire de manière aléatoire [61]. Nous avons tout de même continué d'utiliser Nmap sur le réseau Sunrise en attendant l'activation du réseau Swisscom. En effet, un scan lent est toujours mieux que rien, si nous tenons à respecter le côté éthique.

Nous avons ensuite choisi Masscan par rapport à ZMap dans le fait qu'il se composait simplement d'un programme pour récolter les banners. Cependant, cette récolte de banners est plus poussée que le script d'Nmap qui attend simplement de recevoir quelque chose. M. Inversini a jugé cette méthode trop agressive et nous avons donc simplement utilisé Masscan pour nous indiquer si un port est ouvert.

Masscan offre aussi la possibilité de générer une liste d'IPs sans les scanner, ce qui est très utile dans notre cas. En effet, une fois qu'un port ouvert a été découvert, il n'est pas possible d'attendre trop longtemps avant de vérifier si un opérateur s'y trouve. Il ne faut pas oublier que nous interagissons finalement avec des humains. Plus le laps de temps entre les deux est grand, plus il y a de chance que la personne ait fermé entre temps le programme et que nous n'allons donc pas la trouver.

Infrastructure finale

Au final, des scans sont effectués sur les deux réseaux. Comme dit dans l'introduction, la majorité des scans se déroule pourtant sur le réseau de Swisscom. C'est là que nous effectuons la recherche de nouveaux opérateurs, une fois que nous avons généré la liste des IPs à scanner, selon une des trois approches présentées au point 3.6.3. Nous alternons ensuite les scans avec Masscan et Nmap. En premier, Masscan scanne 60'000 adresses IP pour voir si le(s) port(s) est(sont) ouvert(s). Avec la vitesse de 100 paquets par seconde, qui a été convenue avec M. Inversini, cela nous prend environ une dizaine à une quinzaine de minutes pour un port. Les ports qui sont découverts, sont ensuite scannés par Nmap pour déterminer s'il s'agit d'un opérateur. Ces deux étapes sont ensuite renouvelées jusqu'à ce que la liste d'IP générée soit vide.

Sur ce réseau, nous avons également la recherche quotidienne des opérateurs sur Shodan puis le tracking de tous les opérateurs découverts. Pour cela nous utilisons simplement NMap qui regarde pour chacun s'il est actuellement actif. Pour ceux où c'est le cas, ils sont ajoutés dans un fichier csv qui en garde une trace.

Dans le réseau Sunrise, nous avons finalement que le scan des opérateurs provenant des samples qui suit le même principe que celui du tracking. L'image 21 donne une vue d'ensemble de cette infrastructure.

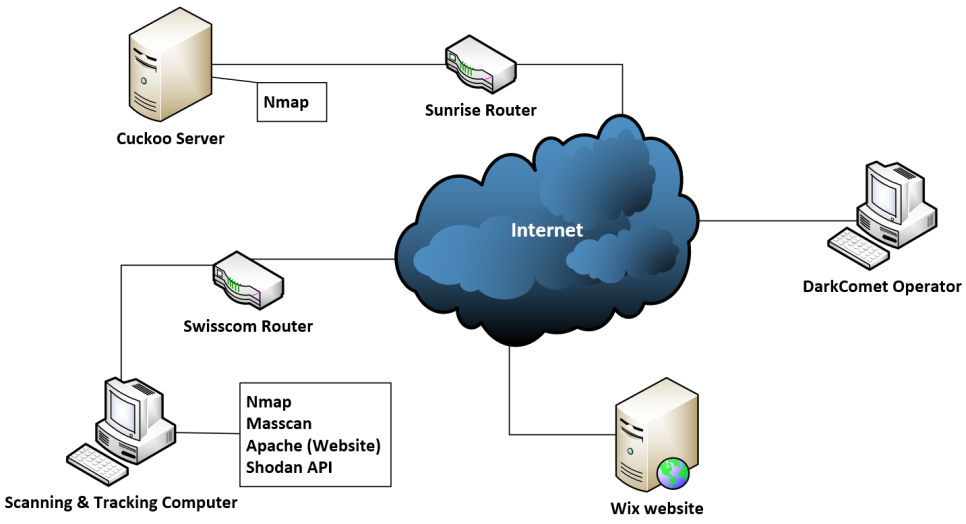

Figure 21: Infrastructure du scanning (Version finale)

4 Implementation

This section aims to explain some important details of our implementation. Although it does not describe the whole process of installation and configuration because it would be boring and not useful. The python code present in this section will be available on github on this address: https://github.com/tchax95/DarkCometTracker.

4.1 Ubuntu Configuration

For this work we installed Ubuntu 17.10 on one SSD. The second SSD is used for the virtual machines and the third drive (HDD) is used as backup.

4.2 Cuckoo Configuration

The installation of Cuckoo Sandbox, Virtualbox 5.2, TCPdump, Volatility and the libraries of python have been done following the Cuckoo installation documentation [46]. As advised by the documentation of Cuckoo we created a virtual environment of python to install the packages.

In the following lines we explain what we changed in the configuration files of Cuckoo.

cuckoo.conf
In this central file we set the following parameters:

```
[cuckoo]
machinery = virtualbox
memory_dump = no
terminate_processes = no
reschedule = yes
process_results = no
max_analysis_count = 10000
max_machines_count = 6
max_vmstartup_count = 2
freespace = 60024

[resultserver]
ip = 192.168.2.2
port = 2042

[database]
connection = mysql://cuckoo:cuckoo@localhost:3306/cuckoo
```

We set which virtual hypervisor we used, so VirtualBox. We decided to use VirtualBox because it is the most documented when used with Cuckoo. Then we disabled memory dump principally because of storage space problem. For each online analysis the system would create a 4GB memory dump that would not be used. We need it only when we want to extract the configurations in the offline analyses. The maximal machines running at same time is set to 6, of course if an hardware upgrade would be disposable we could handle that to increase the simultaneous analysis. From this 6 machines, only 2 at time can start up together, so that the system remains quite stable and not under too much pressure. Then we set almost 60GB of free space. Once the system has less than 60GB free in the primary disk, Cuckoo waits until some place is freed in order to continue. Of course we also set the address of the result server as our own IP address which was 192.168.2.2. Then, for performance and stability reasons, Cuckoo advice not to use its default database that is sqlite3 but rather to install an other database

engine. We then installed mysql and of course we changed the parameter in the configuration file. The other parameters in this configuration file were left as default.

virtualbox.conf
As we use Virtualbox as hypervisor, we configured it by changing the following parameters:

```
[virtualbox]
mode = headless
interface = vboxnet0
machines = Machine01,Machine02,Machine03,Machine04,Machine05,Machine06,Machine_Offline01,
                                                    Machine_Offline02,Machine_Offline03

[Machine01]
label = Machine01
platform = windows
ip = 192.168.200.101
snapshot = Snapshot1
interface = vboxnet0


[Machine_Offline01]
label = Machine_Offline01
platform = windows
ip = 192.168.200.201
snapshot = Snapshot1
interface = vboxnet0
```

We could choose between headless and gui for the virtual machines interaction. We decided to set it to headless so that the system does not start a new window each time a machine start. If we want, we are always able open it manually and interact with the virtual machine. Then we set which interface to use for network communication and a list of all our machines present in Virtualbox.
As explanation, we show the configuration for one machine that is used in online analysis and one that is used in offline analysis to extract the configurations of DarkComet. The most important is to give the right name to the machines, set up the right IP address and set up the right snapshot name.

memory.conf
The memory.conf file is basically a configuration given to Cuckoo in order to know how to use volatility.

```
# Volatility configuration
[basic]
guest_profile = Win7SP1x64
delete_memdump = yes

[darkcometconfigdump]
enabled = yes
filter = no
```

In this file we set the guest profile which basically is the operating system installed in our virtual machines (Windows 7 Service Pack 1, 64 bit). Then we say to Cuckoo to delete the memory dump once it has been analysed without errors, so that the space can be freed. As we created a volatility plugin which the name is "darkcometconfigdump" we need to create a section and enable it in this file. All other volatility plugins have been disabled so that we can spare some time during the analysis.

processing.conf

The Cuckoo processing module is responsible to launch all the analysis operations over the data produced by an analysis.

```
[memory]
enabled = yes

[pcapparser]
enabled = yes
```

We set that a memory dump can be done if required, and then analysed with volatility. We have done a processing module for Cuckoo so that we can take the network dump produced by Cuckoo with the help of tcpdump, and decrypt the whole network communication. For that we need to enable this module in this file.

Agent modification

A problem relating to our virtual machines we noticed, was the unsynchronised time. A time difference of 2 minutes, to the actual time zone is not a big issue, but a time difference of several hours is. Some malware or also some operators will check the right time according the time zone.

To do that we thought that the simplest way was to modify the Cuckoo agent, running in the virtual machines, in order to execute the time synchronisation just before to execute the submitted sample. Here below the command we implemented in the agent's code.

```
w32tm /resync
```

4.2.1 Iptables Rules

As our virtual machines are in an host-only network, we need before all to do a Network Address Translation (NAT) and to forward the network traffic from the network 192.168.200.0/24 to the internet. For doing this we need the so called Iptables rules, that define what network traffic is allowed and what not.

```
# delete all and reset

iptables -F && iptables -X
iptables -t nat -F
iptables -t nat -X

iptables -t mangle -F
iptables -t mangle -X
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

iptables -I FORWARD -o eno1 -i vboxnet0 -s 192.168.200.0/24 -m conntrack --ctstate NEW -j ACCEPT
iptables -I FORWARD -m conntrack --ctstate NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -I POSTROUTING -t nat -o eno1 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward
```

This script needs always to be executed as root just after a system boot, so that the system knows which rules to use. Basically the first lines will cancel all previous configurations done automatically by Iptables, and then it will configure the forwarding and the masquerading of the network packages.

4.3 Start the Cuckoo System

In order to successfully start the Cuckoo system it is very important to follow a well ordered sequence of operations. This is to avoid miss-functionalities. So the list below explains which steps need to be done.

1. Set up iptables rules as described in chapter 4.2.1. The commands need to be done as root
2. Start VirtualBox, restore to the right snapshot one machine and start it. Once the machine is powered on, shut it down. This is done to initialise the vboxnet0 interface. From now, all the terminal commands need to be done in the virtual environment
3. Start the Cuckoo rooter by executing:
   `cuckoo rooter --sudo`
4. Start Cuckoo by executing in a new tab:
   `cuckoo -d`
5. Start the processing threads so that the data produced during the analysis can be evaluated. This step can be repeated as many times as we want with different names. We used to do it 6 times in 6 different terminal tabs, so 6 threads evaluated our data. The command is reported below:
   `cuckoo process p_name`
6. Start our automation-script that will schedule the scanning, submission of samples etc.
   `python apscheduler.py`

In order to avoid to create the 6 processing threads, the solution would be to use Supervisord: "Supervisor is a client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems." [62]. Cuckoo already has a predefined configuration usable with supervisord. The problem is that we had no view on what was happening, because it starts the processes as daemons. So, for debug purposes we decided not to use it.

4.4 Submit a sample

There are two reasons for submitting a sample to the Cuckoo system. The first one is the so called "offline analysis" where we want to check if the sample is a DarkComet, and in positive case we will try to extract the configuration from the memory. The second scenario is when we know that an operator is online and we want to submit his sample to our machines, so that the operator can attack us. Those two scenarios are basically two different versions of a single command: the `cuckoo submit`.

In the case where we want to extract the configuration of a DarkComet we set our "custom" parameter to offline and we also want Cuckoo to do a memory dump. In the options parameter we set human to active, so the cuckoo agent will simulate some "clicks" in the desktop to simulate the presence of a person. Also we enable the deletion of the memory dump if the analysis was successful. Finally we set an analysis time of 50 seconds, the name of the machine we want to use and the path to the sample to be submitted.

```
cuckoo submit --custom offline --memory --options human=1,free=yes,screenshots=0 --timeout 50 --
                                                machine [machine_name] path/to/file
```

For the second scenario then, the so called "online analysis", the logic is quite similar to the other one. Now we set the custom parameter to online so that in the end of the analysis, our cuckoo plugin *pcapparser.py* will try to decrypt the whole network connection. We do not want anymore the human plugin to be active. As with human plugin interaction, the virtual machine becomes quite inoperable. The time for this type of analysis is set then to one hour.

```
cuckoo submit --custom online --options human=0,screenshots=0 --timeout 3600 --machine [machine_name
                                                ] path/to/file
```

4.5 Folder structure

As in the next sections we are going to explain the way how our scripts work, we think that it is important to undestand first our basic folder structure. It is represented as a tree in the next schema:

```
automatic
├── 1_BeingAnalysed
├── 1_VT
├── 2_DarkComet
├── 2_Other
├── 3_ScanningResults
│   ├── activity.csv
│   ├── result.txt
│   └── trash.csv
├── 3_ToScan.csv
├── 4_Analysed.csv
├── 4_Online.csv
├── apscheduler.py
├── results.csv
├── script.nse
└── temp_results.csv
```

The whole structure is set under a folder that we called *automatic*. *1_VT* is the folder where we put the samples which we want to extract the DarkComet configuration. Before the submission, those samples are moved to the *1_BeingAnalysed* folder. If the system detects that a sample is a DarkComet and extracts its configuration, it moves it to the *2_DarkComet* folder, otherwise to the

2_Other folder. As once per hour we perform a scan of all our known operators, in folder *3_ScanningResults* we have some files that take trace of who is online at a certain time.

The other files in the *automatic* directory are *3_ToScan.csv* that contains all operators with an IP or domain name still active, so they are going to be scanned once per hour. *4_Analysed.csv* contains the operators that have been in a live analysis in the last 4 hours, so that we do not scan them too frequently. *4_Online.csv* contains all the operators that are online while finishing a scanning tour. *apscheduler.py* is our automation script described in detail in chapter 4.6.4. *results.csv* contains all the DarkComet configurations extracted with our Cuckoo environment and the *temp_results.csv* contains the configurations just extracted, but we do not know yet if the IP or domain name still exists, so that they can be added to the *3_ToScan.csv* list. Finally, the *script.nse* is the script used with nmap to detect if in the other side, an operator has the DarkComet client (GUI) open. This process is explained in section 4.6.3.

4.6 Scripts implementation

In this section we will present part of the code that we have done for this work. Although, we decided to show only the important and interesting pieces.

4.6.1 Volatility

In order to extract the configuration of a DarkComet, we want to analyse the memory dump of a virtual machine produced by Cuckoo. To do that we need basically to do two things: the first is to write a volatility plugin and place in in the volatility's plugins folder. The other is to modify the Cuckoo script *Memory.py* located under `Cuckoo_Installation_Directory/processing/memory.py`.

Memory.py
In the *memory.py* file, we need first of all to insert our "darkcometconfigdump" into the PLUGIN list. This list is present in the VolatilityManager class. Then we need to add a function to the VolatilityAPI class with the following signature *darkcometconfigdump(self)*. The function is reported here below.

```
    def darkcometconfigdump(self):
        """Volatility sockscan plugin.
            @see volatility/plugins/darkcometconfigdump.py
        """
        results = []
        number = re.search(r"(?<=\/)\d+", self.memdump).group();
        command = self.plugins["darkcometconfigdump"](self.config)
        command.setNumber(number, self.file_path)
        var = command.calculate()
        if "darkcomet" in var:
            if subprocess.call("mv "+self.file_path+" /home/proj2/Desktop/automatic/2_DarkComet/",
                                                                    shell=True) == 0:
                print "move to 2_DarkComet folder"
            else:
                print "Error moving to 2_DarkComet"
        else:
            if subprocess.call("mv " + self.file_path + " /home/proj2/Desktop/automatic/2_Other/",
                                                                    shell=True) == 0:
                print "move to 2_Other folder"
            else:
                print "Error moving to 2_Other"
        return dict(config={}, data=results)
```

This function will basically call the darkcometconfigdump plugin that we developed for volatility and if the plugin returns the string "darkcomet", it means that the configuration has been successfully extracted, the function will move the sample that is being analysed to the folder *2_DarkComet* otherwise it moves the sample to the folder *2_Other*.

darkcometconfigdump.py
Actually, this is the script used to extract the configuration of DarkComet from the memory dump produced by Cuckoo. We do not speak about the whole script because there would be too much details to discuss, although the code is available on github. This script has been initially developed by *dfirn00b* on github [63], and then modified by us.

```
sigs = {
    'dc_config':'rule dc_config{strings: $a = "#BEGIN DARKCOMET DATA --" ascii nocase condition:$a}'
                                                                    ,
    'version':'rule version{strings: $a = /#KCMDDC\d+#-890/ ascii nocase condition: $a}',
}
```

We initially started using the above set of YARA rules to find the configuration in the memory and the version of DarkComet, but then we noticed that with this method we were able to extract only configurations of DarkComet version 5.3.0, so we tried to find a solution and we found out that "MUTEX" should be in all configurations, so we use it in the set of YARA rules below.

```
sigs = {
    'dc_config':'rule dc_config{strings: $a = "MUTEX" ascii nocase condition: $a}',
    'version':'rule version{strings: $a = /#KCMDDC\d+#-890/ ascii nocase condition: $a}',
}
```

For DarkComet versions different than 5.3.1 we are able to find the host and the port but not the password. That is not so a big problem because as we will illustrate in table 7 there are very few samples today that are different than version 5.3.1. Moreover, as illustrated in table 3 most of the operators do not set a password.

4.6.2 Decrypt Network Communication

To decrypt the whole network communication we first need to convert the PCAP file, generated by tcpdump, to a human readable JSON file. For this we use tshark. Then we iterate over all packets to decrypt the commands sent.

TCP-Dump-Conversion
To convert the PCAP file generated by tcpdump to a JSON readable file we use the following line of code. It is important to see that only the packages with source or destination port like the one used by the operator are converted into the JSON file.

```
os.system('tshark -Y  tcp.port==' + str(portAddress) + ' -t ad -T json -x -r ' + pcapPath + ' > ' +
                                                    self.analysis_path + '/resultPcap.json')
```

TCP-Dump-Filtering & RC4 Decryption

At this point we open the JSON file created just above and we prepare ourselves to decrypt and filter the commands that have been done.

```
filter_list = ["KEEPALIVE", "backinfoes", "RefreshSIN"]
```

The "filter_list" contains only the three most occuring auto-generated commands. This commands are going to be filtered out.

More filtering has been done for the analysis and evaluation, at the end of the project. This way, we keep all the data at this point, that could maybe be interesting later.

```
for item in data:
    jsonFrame = None
    if "tcp" in item['_source']['layers']:
        if "tcp.payload" in item['_source']['layers']['tcp']:
            jsonFrame = item['_source']['layers']["tcp"]["tcp.payload"]
    elif "data" in item['_source']['layers']:
        if "data.data" in item['_source']['layers']['data']:
            jsonFrame = item['_source']['layers']['data']['data.data']
    try:
        if jsonFrame is not None:
            hex_data = jsonFrame.replace(':', '')
            command = rc4(hex_data.decode('hex').decode('hex'), password)
            if any(x in command for x in filter_list):
                continue
            dataList.append(command)
        else:
            raise Exception("No data.data or tcp.payload")
        timeList.append(item['_source']['layers']['frame']['frame.time'])
```

For every frame in the JSON file, we check if a "tcp.payload" or "data"-block exist. Those fields are actually containing the interesting information.

Then as those fields contain the hexadecimal representation of a string, that is once more encoded in hexadecimal, we need to call the RC4 function in this way: `rc4(hex_data.decode('hex').decode('hex'), password)`.
The "password" gets taken from the *results.csv* file where we have stored the configurations of all samples. Finally, we compare the extracted command with the "filter_list". If it is not one of the commands to filter, we add the command and the timestamp to the results file for the current analysis.

4.6.3 Scanning-Scripts

Nmap Banner Script

As already mentioned in point 3.6.4, this script is based on the "banner" script available in the Nmap library [60]. It's written in Lua, the language used by Nmap Scripting Engine (NSE). We modified it to the specific needs of the project. Once we receive the banner, we use a regular expression to see if it is a DarkComet. We are therefore looking if it is composed of a hexadecimal string of length 12. If this is the case, we then check if it matches a default password. In the code below, the result of an Nmap scan with this script is visible. The banner output is highlighted as much as possible so that it can be easily reused later.

```
Nmap scan report for 10.0.0.5
Host is up (0.085s latency).

PORT     STATE SERVICE
1344/tcp open  icap
|_script: DarkComet 5.1 use default password, Banner: BF7CAB464EFB
```

Get Operators from Shodan

Every day the new operators found by Shodan are downloaded. Below, it is the crontab line that is used to automatically start the script. With *2>&1* all error messages, as well as the standard output can be written to a file. It is therefore easy to know what happened.

```
45 02 * * * /usr/bin/python /home/scan/Documents/ShodanOperators.py >> /home/scan/Documents/
                                                    logShodan.txt 2>&1
```

Shodan provides a library for python, which greatly simplifies access to the database. This makes it easy to connect and get the results that are wanted. We work here in Coordinated Universal Time (UTC) format, to be on the same time as Shodan. It prevents us from already being the next day and missing results by miscalculating the variable *yesterday*.

```
def get_shodan_results():
    #SHODAN_API_KEY is available in my account on Shodan's website
    api = shodan.Shodan(SHODAN_API_KEY)

    yesterday = (datetime.utcnow()-timedelta(1)).strftime('%d/%m/%Y')

    try:
        results = api.search('product:"DarkComet trojan" after:'+yesterday, limit=300)
        return results
    except shodan.APIError as e:
        print 'AN ERROR OCCURED during the search'
        return []
```

Scan to find new operators

The part of the code that is responsible for starting masscan is shown below. We can see how the port(s) are given as parameters. Of course there's also the list of IP addresses to scan (*-iL*). This has been previously reduced to 60,000 addresses. The *-interactive* parameter displays, in real time, the addresses where a port is open. Unfortunately, when using it in a python script, they are only displayed at the end of the scan. But that doesn't change a lot because we're interested in the output file (*-oL*) where the results are written when masscan is done. If open ports have been discovered these are then passed to Nmap.

```
process = subprocess.Popen("sudo masscan -p " + PORTS + " --interactive -iL \"" + target_file + "\"
                                                    -oL \"" + temp_result_file + "\"",stdout=
                                                   subprocess.PIPE, shell=True)
(output, err) = process.communicate()
print output

if os.stat(temp_result_file).st_size == 0:
 print "--> No results"
 os.remove(temp_result_file)
else:
 print"--> Found results"
 prepare_nmap_target_file(temp_result_file)
```

4.6.4 apscheduler.py

In this section we describe in a quite short mode how our automation script *apscheduler.py* works by showing different pieces of code.

download_vt()

This function has been early designed to make queries on Virtustotal API, download the samples and then decompress the downloaded archive to the right folder: *1_VT*. As we already explained, it did not work this way but we still decided to keep this function so that in a future work it could be easy to implement this functionality. In order to see that it is called we simply print that the function started but did nothing.

```
def dowload_vt():
    """
    The idea was to execute the query to VirusTotal API and download the samples to 1_VT
    """
    print get_date_time_now() + " ==> Download VT Samples started!"
    print get_date_time_now() + " ==> Nothing downloaded"
```

offline()

This function is intended to check if there are new samples in the folder *1_VT*, rename them by computing their sha256 hash, moving them to the folder *1_BeingAnalised* and finally submit them for a offline analyse.

The submission code is the code explained in section 4.4 apart for the selection of the virtual machines names. This logic is done setting a python dictionary that basically is a set of key-values with a number and the corresponding virtual machine name. Each time that a sample is submitted a counter is incremented and when it reaches the maximum number of virtual machines it is reset to one.

Doing this we have a linear distribution of the samples over the 3 virtual machines that we use for this scope.

```
def offline():
    """
    If there are samples in 1_VT, it submits them to cuckoo to extract their configuration from
                                                    memory
    """
    print "Offline analyse started!"
    if os.listdir(BASE_PATH+"1_VT/"):
        global MACHINE_COUNTER_OFFLINE

        #Rename files to sha256 hex digest
        for filename in os.listdir(BASE_PATH+"1_VT/"):
            file_data = open(BASE_PATH+"1_VT/"+filename, 'rb').read()
            hash_value = hashlib.sha256(file_data).hexdigest()
            os.rename(BASE_PATH+"1_VT/"+filename, BASE_PATH+"1_VT/"+hash_value)

        #Submit samples
        for file in os.listdir(BASE_PATH+"1_VT/"):
            if os.path.isfile(BASE_PATH+"1_VT/"+file):
                subprocess.Popen("mv "+ BASE_PATH + "1_VT/" + file + " " + BASE_PATH + "
                                                        1_BeingAnalysed/", shell=True).
                                                        wait()
                subprocess.Popen("cuckoo submit --custom offline --memory --options human=1,free=yes
                                                        ,screenshots=0 --timeout 50 --
                                                        machine "+ machines_offline[
                                                        MACHINE_COUNTER_OFFLINE] + " " +
                                                        BASE_PATH + "1_BeingAnalysed/" +
                                                        file, shell=True).wait()

                if MACHINE_COUNTER_OFFLINE == len(machines_offline):
                    MACHINE_COUNTER_OFFLINE = 1
                else:
                    MACHINE_COUNTER_OFFLINE += 1
```

scan()

The scan function is a quite big function, but its main goal is to get the operators that need to be scanned from the *3_ToScan.csv* file, get the configurations of their samples from *results.csv* and then make a Nmap scan on the network port configured by the operator. Moreover the *script.nse* is used to detect if the operator is active in that given moment. If a network address cannot be resolved, it will be automatically moved to our trash file.

```
    with open(targetFile, 'r') as csvFile:
        targetList = csv.DictReader(csvFile)
        with open(tempFile, 'w') as f:
            wrTemp = csv.writer(f)
            wrTemp.writerow(['HOST', 'PORT', 'FILE HASH'])
            for target in targetList:

                process = subprocess.Popen("sudo nmap -p " + target[
                    'PORT'] + " -n --data-string \"" + messageScan + "\" --script " +
                                                            darkCometScript + " --append-
                                                            output -oN " + resultLog + " "
                                                             +
                                           target['HOST'], stdout=subprocess.PIPE, shell=True)
                (output, err) = process.communicate()
                print output

                if "0 IP addresses" in output:
                    # Means the domain name could not be resolved
                    print "--> Goes to trash"
                    addHeaderToCSVIfNecessery(trashFile)
                    row = [timestampFile, target['HOST'], target['PORT'], target['FILE HASH']]
                    with open(trashFile, 'a') as f:
                        wr = csv.writer(f)
                        wr.writerow(row)
                elif "|_script: DarkComet" in output:
                    # Means the operator is active
                    print "--> Operator is active"
```

online()

With a quite similar logic to the offline function, the online function will reach which operators are online from the *4_Online.csv* file and then submit their samples that at this stage are placed in the folder *2_DarkComet*. The same technique as for offline is used in order to choose the name of the virtual machine to use.

```
def online():
    """
    If there are operators online, it submits their sample to cuckoo to perform a live analysis
    """
    global MACHINE_COUNTER_ONLINE
    print get_date_time_now() + " ==> online analysis started"
    checklist = []
    with open(BASE_PATH + "4_Online.csv", 'r') as online_file:
        online_reader = csv.reader(online_file)
        for row in online_reader:
            if row[0] not in checklist:
                subprocess.Popen("cuckoo submit --custom online --options human=0,screenshots=0 --
                                                                    timeout 3600 --machine " +
                                                                    machines_online[
                                                                    MACHINE_COUNTER_ONLINE] + " " +
                                                                    BASE_PATH + "2_DarkComet/" +row[0]
                                                                    , shell=True)
                checklist.append(row[0])

                if MACHINE_COUNTER_ONLINE == len(machines_online):
                    MACHINE_COUNTER_ONLINE = 1
                else:
                    MACHINE_COUNTER_ONLINE += 1

    #delete all rows
    with open(BASE_PATH + "4_Online.csv", 'w') as online_file:
        online_file.truncate()
```

main

The part of *apscheduler.py* that starts the whole process is the code below. It basically schedules the functions seen in the sub-sections above at a given time or interval. So, it will simulate the download of samples from Virutstotal every day at 02:00 in the morning. It will then submit the downloaded samples, or in our case the samples already present in the folder *1_VT*, to extract the samples configurations, every day at 04:00 in the morning. Moreover, each hour a scan of all operators found is scheduled.

```
    schedule.every().day.at("02:00").do(dowload_vt)
    schedule.every().day.at("04:00").do(offline)
    schedule.every(60).minutes.do(scan)

    try:
        while True:
            schedule.run_pending()
            time.sleep(60)
            print get_date_time_now() + " ==> -----"
```

5 Experiments

5.1 Reverse Engineering

The reverse engineering experiments are principally the improvement of the "*darkcometconfigdump.py*" script so that we could extract and store the samples's configuration. A description of how our volatility plugin works is done on section 4.6.1.

In the next section some statistics over the "*results.csv*" file, that contains the extracted configurations of the samples, are done.

5.1.1 Results

During the whole work 746 DarkComet samples out of 2441 samples have been detected and a configuration has been extracted. In this section we will then analyse the results coming from the configurations of the samples.

Most used password
When it comes to analyse the configurations extracted, one of the most relevant things is to check if operators set passwords on their samples, and if yes what kind of passwords they set. Sorting our results file we found 25 different, unique passwords set by operators. The next table shows the top 15 of them.

Password	# Found	% Percent
NO PASSWORD SET	702	94.1 %
0123456789	14	1.88 %
1234	3	0.4 %
123456	2	0.27 %
123456789	2	0.27 %
2q43refwdfw32refasdqwreqwedqqrqe 32eeqwrqfdasdasdqweqweqweqwr	2	0.27 %
404*]HaCK3D[*	2	0.27 %
123	1	0.13 %
1234321	1	0.13 %
12345	1	0.13 %
12345678	1	0.13 %
134322418e	1	0.13 %
1443813678	1	0.13 %
1453	1	0.13 %
23051999	1	0.13 %

Table 3: 15 Most used password

Analysing table 3, it is interesting to see that 94.1% of the operators did not set any password on their samples in order to "secure" their network connection, because as explained in section 2.9, a default IDTYPE banner is detectable analysing network dumps. Looking at the same table we can obeseve that oper-

ators have not so much fantasy as we tough in the begin. We say that because most of the passwords used are a variant of the pattern "123456789" truncated ad different lenghts. Anyway, in the top 15 results, we can still see six password that are not so common.

- 2q43refwdfw32refasdqwreqwedqqrqe32eeqwrqfdasdasdqweqweqweqwr
- 404*]HaCK3D[*
- 134322418e
- 1443813678
- 1453
- 23051999

The last password of the above list is probably a birth date.

Most used ports
An other important thing that one needs to figure out while analysing the usage of a RAT is which ports are being used by operators. Doing that, we can have a feeling on the post used ports and block them. In the case they need to be open, at leas we know that they need to be well monitored. By analysing this part of the data we noticed that over the 746 samples, there are 199 different ports configured.

Ports	# Found	% Percent
1604	464	62.12 %
81	28	3.75 %
35000	15	2.01 %
1605	14	1.87 %
80	10	1.34 %
25565	9	1.2 %
1337	8	1.07 %
8080	8	1.07 %
1111	6	0.8 %
1177	6	0.8 %
1609	6	0.8 %
1909	6	0.8 %
200	6	0.8 %
443	6	0.8 %
4444	6	0.8 %

Table 4: 15 Most used ports

Checking the results of table 4, we are not surprised to see that the default port of DarkComet (1604) is the most used with a rate of 62.12%. Then we expected to see in this statistic some ports around 1604 as we tough operators would not have fantasy and desire enough in order to completely change the port number.

This is partially confirmed as we see ports like 1605 and 1609 in this ranking. We have to say, although, that we would have expected some more ports around 1604 in this ranking. Other well known ports line 80, 81, 443 and 8080 are also important evidence, as many times those ports are open in firewalls, so the communication should pass without problems. Here our work, and that of people like us, becomes interesting because if we can detect any DarkComet communication on any port, we could block this communication without have to close the port for other legitimate users.

We have then ports like the 1111 and 4444 that are used because of their easy pattern. Although, ports like 35000 or 25565 does not seem to have any special sense to us, but they are still in this top 15 ranking. We can suppose that maybe all samples with the port 35000 and respectively 25565 are only from two distinct operators, but this is just a supposition.

Samples with default port and no password
Roughly speaking we always considered as default sample, a sample that has no password set by the operator and a standard port (1604). So now that we have an overview on both password and port statistics, we can check how many of them are considered for us "default samples".

	# Found	% Percent
No Password & Default Port	450	60.24 %
Not Default config	297	39.76 %

Table 5: Results of default port combined with no password

With the assistance of table 5, we can see that 60.24% of the samples were considered "default samples" and in other hand, 39.76% of the samples had either the password set by the operator or an other port than 1604 or even both conditions.

Process names of DarkComet samples

When we were developing our volatility script so that we could extract the configuration of a sample from a memory dump, we decided to take track of the name of the process that is created on or virtual machines and that contains indeed the DarkComet sample. At the begin it was more as a debug feature, but then we thought that it would be great to know the names of the sample (configurable in the DarkComet builder while configuring the sample) so that in the future we can maybe use some of them in YARA rules. How to use them here is explained in section 2.5. We know that process names like *msdcsc.exe* and

Processes	# Found	% Percent
msdcsc.exe	162	21.66 %
iexplore.exe	132	17.65 %
svchost.exe	64	8.56 %
IMDCSC.exe	52	6.95 %
Gpers.exe	14	1.87 %
micoffice.exe	12	1.6 %
RegSvcs.exe	7	0.94 %
svhost.exe	7	0.94 %
SD.exe	6	0.8 %
vbc.exe	6	0.8 %
msg.exe	5	0.67 %
cvtres.exe	4	0.53 %
notepad.exe	4	0.53 %
yandex.exe	4	0.53 %
taskmgr.exe	3	0.4 %

Table 6: 15 most used process names to hide DarkComet

IMDCSC.exe are standard process names used by DarkComet, because of that we are not surprised when we see in table 6 that they are respectively first and fourth in the top 15 process names ranking. Something we did not expected, but we are glad to see, is the process names related to Office like the *micoffice.exe* that sounds like strange because Microsoft has no product that creates a process with that name. There are then process names that try to emulate processes existing in Windows or with the help of third software could exist, like *svchost.exe*, *RegSvcs.exe*, *taskmgr.exe* or *notepad.exe*. Finally we can also see some samples that seem to be really fake because they do not even have the correct name of the process they try to emulate, like *iexplore.exe* that should be "iexplorer.exe" and *svhost.exe* that should be "svchost.exe".

Versions of DarkComet samples in the wild
We stored for each sample the version of DarkComet, because we needed it to build the encryption key which as reminder is the version key chained with the password set by the operator. Finally it has been useful to understand something we already imagined, version 5.3.0 of DarkComet is the most present in the wild actually.

Version	Version key	# Found	% Percent
v5.3.0	#KCMDDC51#-890	717	96.11 %
v2.0	#KCMDDC2#-890	8	1.07 %
Errors	–	8	1.07 %
v5.0	#KCMDDC5#-890	7	0.94 %
v4.0	#KCMDDC4#-890	6	0.8 %

Table 7: Most common versions of DarkComet in the wild

As expected, looking at table 7 we can see that with a rate of 96.11% DarkComet version 5.3.0 is the most present at the moment. We suppose that is so because version 5.3.0 is the newest version, so it has more functions and a better GUI for operator. It could also be that the other versions were easily detectable and then they started to use the last version. As long as we know, the last version of the DarkComet samples builder is also the most easily findable in the web.

As a surprise we notice that DarkComet version 2.0 is slightly more present than version 5.0 that is newer. We have to say though, that the difference in the numbers is only of one unit.

Countries of DarkComet samples
We decided not to mention the countries of the samples here because this argument is already treated in chapter 5.2.1 and the statistics would come from the same "results" file.

Dynamic DNS exploitation

For operators it is quite comfortable to use dynamic DNS names because once they have configured it in their samples, if for a certain reason they need to change their IP address, they do not need to change it on all their already deployed samples. They just need to have a manner to update some DNS records to point to their new IP address. Here is where dynamic DNS providers come into play. They provide you a software that takes your actual IP address so that they can update dynamically your IP in their DNS entries.

DDNS	# Found	% Percent
Other IP/Domains	266	30 %
Private	226	25 %
ddns.net	206	23.02 %
duckdns.org	64	7.15 %
no-ip	63	7.04 %
hopto.org	41	4.58 %
zapto.org	29	3.24 %

Table 8: Usage of dynamic DNS providers, private addresses and other IP / domains into DarkComet samples

From table 8 we notice that 55% of our samples had no dynamic addresses configured. 25% are private IPs like 192.168.x.x or addresses like that, so they were not useful for us. 30% of the IP addresses were hard-coded, which means that the problem with the IP change described above could happen. What to do in a such case? Well, the operator needs to update all his victim's with a new server configuration, but to do that he needs to have all his victims online. The other way can be to build an other sample and try to re-infect his victims.

In any case we see that 45% of the samples were configured with a dynamic host name, with a major usage of ddns.net.

Firewall bypass

Because we inserted it in our results file, we can also do a small statistical analysis to see how many operators set the firewall bypass to active and how many not.

Firewall Baypass	# Found	% Percent
Inactive (0)	498	66.76 %
Active (1)	135	18.1 %
Not extracted	113	15.15 %

Table 9: Results of Firewall Baypass by sample configuration extraction

With the help of table 9 we see that 66.76% of the samples had firewall bypass inactive and 18.1% had it active. Why such a difference? We are not sure about it, but we suppose they let it inactive so that a little less evidence is left into the system: so less chances to be caught.

Offline keylogger

Following the same logic of firewall bypass, we also did a small statistic on how many samples did have offline keylogger active. The results are quite impressive.

KeyLogger	# Found	% Percent
Inactive (0)	0	0 %
Active (1)	714	95.71 %
Not Extracted	32	4.29 %

Table 10: Results of keylogger active by default

As easy remarkable, 95.71% of the samples had the offline keylogger active and that is already something that speaks for itself. The most relevant acknowledgement is that none of our samples had it inactive. Unfortunately for 4.29% of the samples, we had some errors while extracting the configurations and we cannot say if they have offline keylogger active or inactive.

5.2 Scanning

Un des défis majeurs pour les statistiques liées au scanning est de produire des résultats les plus représentatifs possibles. Un même opérateur peut nous apparaître tantôt avec son adresse IP, tantôt avec son nom de domaine, s'il n'en possède pas même plusieurs. De plus, comme la majorité des adresses IP sont dynamiques, celles-ci vont encore changer en cours de route. Si on comptait comme 5 opérateurs une personne apparaissant sous diverses adresses, celle-ci aurait beaucoup plus de poids dans les résultats et ceux-ci seraient tout simplement faux. Nous avons donc essayé de relier au mieux les diverses adresses d'une même personne. Les adresses ayant une forte ressemblance, malgré nos forts soupçons que soit la même personne qui se dissimule derrière, n'ont pas été reliés. Il serait en effet impossible de définir une frontière nette où on arrête nos suppositions.

Tous les résultats se basent sur le fuseau horaire de la Suisse.

5.2.1 Résultats des samples

Dans cette partie, les résultats liés au samples, provenant de VirusTotal, sont présentés. Ici, l'intérêt n'est pas porté sur le sample en lui-même, mais sur les opérateurs qui se trouvent derrière et qui ont été groupés du mieux possible.

N'ayant d'abord pas réalisé la nécessité de posséder les adresses IPs pour les statistiques, nous avons 19 noms de domaines qui n'ont plus pu être résolus au moment des statistiques. Nous les avons compté chacun comme un opérateur différent. Ils n'apparaissent cependant que dans les statistiques des pays, car ils n'ont jamais été actifs et pour cette raison nous n'avons pas pu retrouver l'information manquante dans nos logs files. Nous possédons donc au final au minimum 314 et au maximum 333 opérateurs uniques.

Pays

Dans le tableau 11, nous pouvons voir la Russie, la Turquie et les Etats-Unis en tête de classement. Ceci n'est pas très étonnant car c'était le même trio dans le paper *To Catch a Ratter: Monitoring the Behavior of Amateur DarkComet RAT Operators in the Wild*" [3].

Pays	# opérateurs	% opérateurs
Russie	66	21.02%
Turquie	58	18.47%
Etats-Unis	27	8.60%
Ukraine	22	7.01%
Allemagne	19	6.05%
France	16	5.10%
Pays-bas	10	3.18%
Royaume-Uni	9	2.87%
Brésil	7	2.23%
Maroc	6	1.91%
Autres pays	74	23.65%

Table 11: Pays des opérateurs

Au total, nous avons recensé des opérateurs de 51 pays. En Suisse 3 opérateurs ont été découverts.

Fournisseurs d'accès à internet

Il est difficile de vraiment faire un classement des fournisseurs d'accès. Parfois des réseaux sont partagés, fusionnés, vendus entre différents opérateurs. Cela crée souvent des noms qui diffèrent légèrement d'un réseau à un autre. Nous avons tout de même essayé de créer le classement de tête. Au final nous comptons 176 ISP. Les classement des fournisseurs d'accès, dans le tableau 12, suit en grande

Fournisseurs d'accès à internet	# opérateurs	% opérateurs
Turk Telekomunikasyon A. S.	38	12.10%
PJSC Rostelecom	20	6.37%
Vodafone GmbH	11	3.50%
JSC ER-Telecom Holding	9	2.87%
OVH SAS	6	1.91%
MTS PJSC	6	1.91%
VODAFONE NET ILETISIM HIZMETLERI A. S.	4	1.27%
Leaseweb USA, Inc.	4	1.27%
TELLCOM ILETISIM HIZMETLERI A.S.	4	1.27%
Orange Côte d'Ivoire	4	1.27%
ONPT (Maroc Telecom) / IAM	4	1.27%
CONTENT DELIVERY NETWORK LTD	4	1.27%
Autres FAI	200	63.69%

Table 12: Fournisseurs d'accès à Internet (FAI) des opérateurs

partie celui des pays. Türk Telekom appartient logiquement à la Turquie. Rostelecom, ER-Telcom et MTS sont tous des fournisseurs d'accès en Russie.
Le plus intéressant dans cette partie est que 2 adresses appartiennent au ministère de la défense du Royaume-Uni. Bien que nous ayons dit dans l'introduction que DarkComet était aussi utilisé par les gouvernements, nous ne pensions pas faire face à cela dans notre travail.

Activité des opérateurs
La phase de scanning des samples a duré du 11 avril au 28 mai. Au début, différentes coupures ont été nécessaires à la mise en place correcte de notre système, cependant ceux-ci ne devraient pas trop influencer les résultats finaux. Durant la période de notre travail, nous avons eu 50 opérateurs qui ont été actifs. Cela ne représente qu'approximativement 15% des opérateurs découverts dans les samples. Ce n'est qu'avec ces personnes-là que nous pouvons potentiellement obtenir des informations sur leurs intentions en effectuant une analyse. Cependant ce nombre a encore de fortes chances de baisser car il ne va sûrement pas y avoir tout le monde qui va se connecter à nos machines.

Dans le tableau 13, il est intéressant de constater comme le classement des pays a été bouleversé. Les Etats-Unis n'y apparaissent plus. L'Ukraine, quant à elle, arrive à la première place, suivie de peu par les Pays-Bas et la Turquie. Au total nous avons eu 23 pays où des opérateurs ont été actifs.

Pays actifs	# opérateurs actifs	% opérateurs actifs
Ukraine	6	12%
Pays-Bas	5	10%
Turquie	5	10%
Russie	4	8%
France	3	6%
Allemagne	3	6%

Table 13: Pays qui comptent le plus d'opérateurs actifs

Dans le graphique 22, nous pouvons voir le nombre d'opérateurs actif par jour. L'évolution de l'arrivée des samples y est d'ailleurs bien visible. Au début, nous n'avions qu'environs 5 opérateurs actifs par jour, à la fin, cela tournait plutôt autour des 12 personnes.

Le pic du 10 mai peut éventuellement s'expliquer par le week-end prolongé de l'Ascension. Il est cependant difficile de trouver une explications aux différents pics. Le trou visible dans le graphique est dû à une panne de nos serveurs du 18 mai au alentours de 13 heures au 22 mai vers les 16 heures.

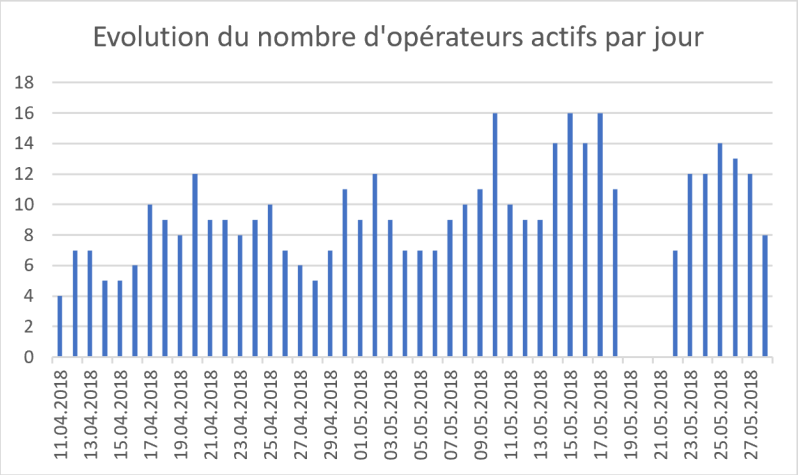

Figure 22: Evolution du nombre d'opérateurs actifs par jour

En regardant l'activité par jour de semaine des opérateurs, dans le graphique 23, nous sommes surpris qu'il n'y ait pas plus d'opérateurs le samedi et dimanche. Nous nous demandons si le résultat a pu être affecté par la panne de serveur de mai. Cependant le vendredi et le lundi ne semblent pas vraiment en avoir été affectés...

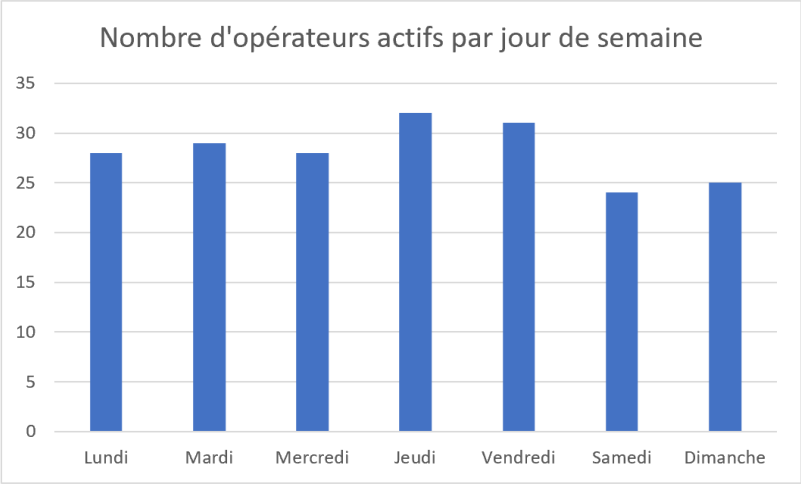

Figure 23: Activité des opérateurs par jour de semaine

Dans le graphique 24, on peut constater que de 2 heures à 6 heures et de 7 heures à 10 heures se sont des heures creuses. Cela pourrait éventuellement être expliqué

dans les faits que les pays comptant le plus d'opérateurs viennent de zones plus ou moins proches de la Suisse. Ils sont dans le même fuseau horaire que nous ou alors avec une heure de décalage.

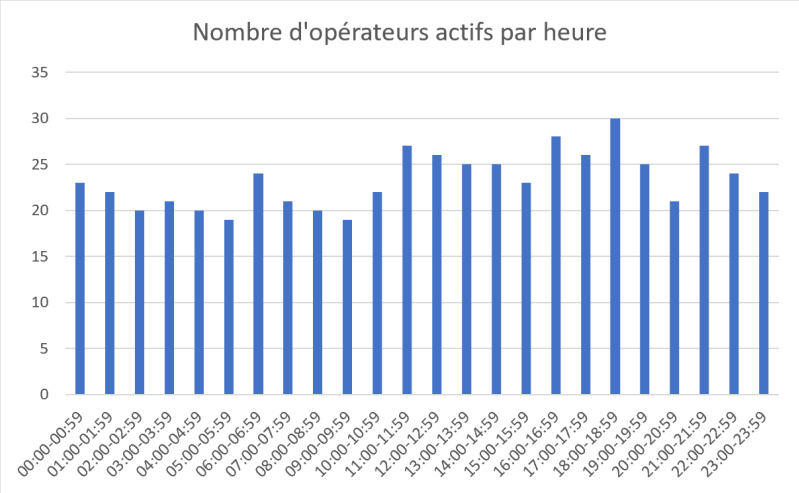

Figure 24: Activité des opérateurs par heure

5.2.2 Résultats obtenus avec Shodan

Nous avons utilisé le même principe qu'au chapitre 5.2.1. Ici, Shodan nous fournit automatiquement les adresses IP et les noms de domaine qu'il possède. Malheureusement, nous perdons beaucoup plus facilement la trace d'un opérateur par le fait que nous ne connaissons pas le nom de domaine employé dans le sample. En effet, comme nous avons été confrontés au même problème avec notre propre réseau, qui est expliqué dans le chapitre 3.6.1, Shodan retrouve très facilement le nom de domaine attribué par le fournisseur d'accès, mais celui-là est lié à l'adresse IP et ne va pas nous permettre de suivre l'opérateur. Nous avons donc très rarement lié un opérateur à plusieurs adresses IP.

En tout, nous avons trouvé 1152 opérateurs entre le 4 décembre 2017 et le 29 mai 2018. Nous avons découvert qu'en quelques heures, 148 adresses différentes ont été utilisées dans un même réseau. Il doit sûrement provenir du ou des mêmes opérateurs. Dans l'évaluation des données, il est donc nécessaire d'en tenir compte, vu l'impact que cela a.

IDTYPE
Comme dans Shodan, il faut définir la banner recherchée, et qu'il n'est pas possible d'utiliser des expressions régulières, nous devons nous contenter des mots de passe par défaut. En plus de cela, quelques mots de passe découverts à travers les samples ont été recherchés. La recherche de produits quand à elle, se limite aussi aux banners par défaut. Les résultats des banners par défaut peuvent cependant nous aider à avoir un aperçu des versions de DarkComet utilisées dans

le monde. Comme on peut le voir dans le tableau 14, c'est évidemment la banner de la version la plus récente de DarkComet (5.1+) que l'on trouve le plus souvent. Logiquement, c'est la version précédent(5.0) qui suit dans le classement. Beaucoup plus étonnant, en 3ème position, c'est une toute ancienne version de DarkComet (2.x + 3.x) qui prend la place. Finalement, il y a encore 2 autres versions, la 4.1 et la 4.2F, qui apparaissent. La version 4.2 quant à elle, n'est jamais apparue.

IDTYPE	# utilisations	% utilisations
BF7CAB464EFB	1123	96.15%
1164805C82EE	22	1.88%
8EA4AB05FA7E	4	0.34%
B47CB892B702	1	0.09%
155CAD31A61F	1	0.09%
Autre IDTYPE	17	1.53%

Table 14: Shodan - IDTYPE trouvés sur Shodan

Ports
Au total, 90 différents ports ont été employés par les opérateurs DarkComet. Le graphique 15 donne un aperçu des ports qui ont été choisis le plus souvent. Les deux qui sont le plus utilisés, sont les mêmes que ceux pour les samples de VirusTotal. Beaucoup d'autres apparaissent dans les deux tableaux. Étant donné qu'ici nous avons presque que des mots de passe par défaut, il n'est pas étonnant que le pourcentage de l'utilisation du port par défaut est plus élevée.

Ports	# utilisations	% utilisations
1604	931	74.60%
81	87	6.97%
9999	16	1.28%
7777	13	1.04%
4444	13	1.04%
82	11	0.88%
80	11	0.88%
8080	11	0.88%
2222	10	0.80%
1177	10	0.80%
Autres ports	135	10.80%

Table 15: Shodan - Ports utilisés par les opérateurs

Pays
Dans les pays, nous trouvons toujours les mêmes qui sont en tête de classement. A cause des 148 hosts du même réseau, il faut cependant relativiser dans ce cas la place des Etats-Unis. En effet sans cela, ils arriveraient en 4ème position, juste derrière la France. Quant à la Suisse, un seul opérateur a été trouvé.

Dans le tableau 16, nous voyons les pays qui arrivent les plus fréquemment, des quelques 75 qu'il y a au total. Il est important de noter que nous avons dû revoir la liste des pays. Shodan donne également comme pays des îles telles que la Martinique ou la Nouvelle Calédonie qui font partie de la Français. Le territoire palestinien, bien qu'il y ait des conflits, fait également partie d'Israël.

Pays	# opérateurs	% opérateurs
Turquie	212	18.40%
Etats-Unis	211	18.32%
Russie	160	13.89%
France	68	5.90%
Ukraine	56	4.86%
Egypte	27	2.34%
Italie	23	2.00%
Grèce	20	1.74%
Maroc	19	1.65%
Pays-Bas	18	1.56%
Allemagne	17	1.48%
Royaume-Uni	17	1.48%
Roumanie	16	1.39%
Autres pays	288	25%

Table 16: Shodan - Pays des opérateurs

Fournisseurs d'accès à internet
Le premier fournisseur d'accès du graphique 17, doit être placé entre parenthèse. En effet c'est là qu'il y a eu les 148 adresses de la même IP-Range. En-dehors de cela, nous trouvons à nouveau les fournisseurs d'accès à internet turcs et russes.

Fournisseurs d'accès à internet	# opérateurs	% opérateurs
DXTL Tseung Kwan O Service	148	12.85%
Turk Telekom	76	6.60%
TTNet A.S.	72	6.25%
PJSC Rostelecom	49	4.25%
TE-AS	26	2.26%
OVH SAS	22	1.91%
JSC ER-Telecom Holding	18	1.56%
Vodafone NET Iletisim Hizmetleri A.S.	18	1.56%
OJSC Vimpelcom	15	1.30%
Autres FAI	708	61.92%

Table 17: Shodan - Fournisseurs d'accès à internet (FAI) des opérateurs

Activité

Au total, nous avons eu 87 opérateurs actifs. Ceux-ci provenaient majoritairement de la France, de la Russie et des Etats-Unis, comme nous pouvons le voir dans le tableau 18. Ils proviennent au total de 32 pays. La Suisse ne compte ici aucun opérateur actif.

Pays actifs	# opérateurs actifs	% opérateurs actifs
France	12	13.79%
Russie	11	12.64%
Etats-Unis	9	10.34%
Roumanie	6	6.90%
Allemagne	5	5.75%
Turquie	5	5.75%

Table 18: Shodan - Pays qui comptent le plus d'opérateurs actifs

L'évolution du nombre d'opérateur est assez constante selon le graphique 25. Cela pourrait s'expliquer dans le fait que les opérateurs changent d'adresse IP et que l'ancienne devienne caduque. Tout de même, quatre opérateurs ont été actifs tous les jours, durant la période où nous avons effectué les scans.

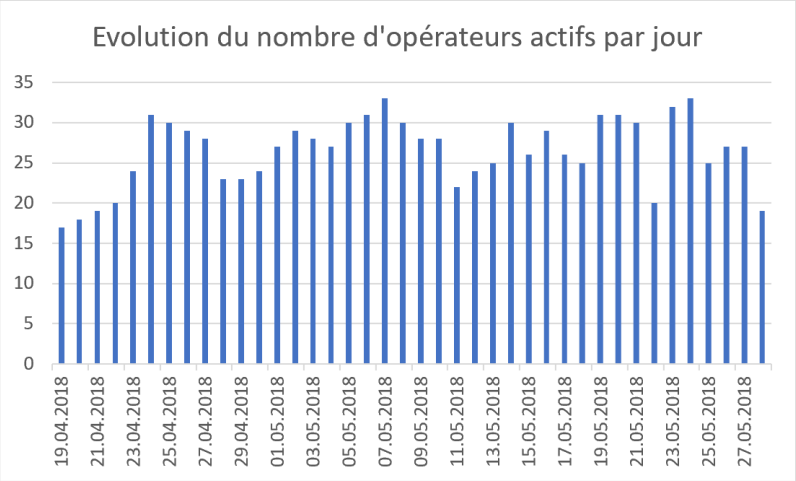

Figure 25: Shodan - Evolution du nombre d'opérateurs distincts actifs par jour

Il est étonnant de voir que le jour où les opérateurs sont le plus présents tombe le lundi. Selon le graphique 26, le vendredi serait le jour où le moins d'opérateurs se connectent. Ces données ont été cumulées durant plusieurs semaines.

C'est durant la nuit, de minuit à 8h du matin que l'activité des opérateurs est la plus faible. En regardant le graphique 27, nous pouvons voir qu'au miminum, il y a eu 49 personnes actives simultanément et au maximum 68.

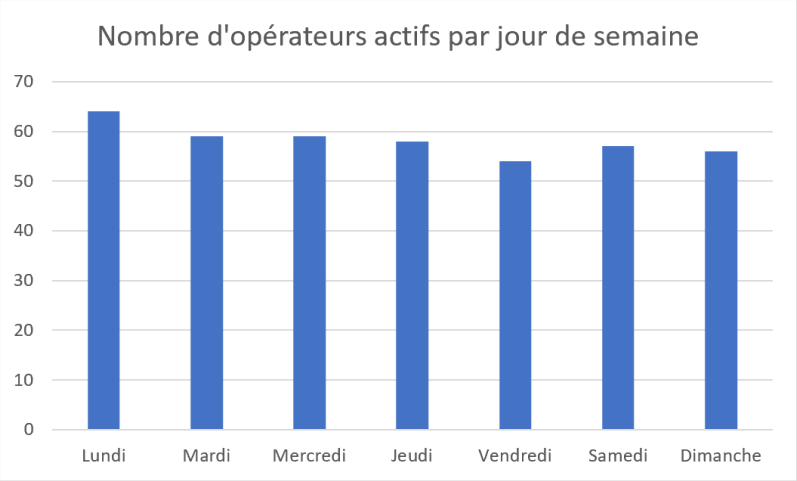

Figure 26: Shodan - Activité des opérateurs par jour de semaine

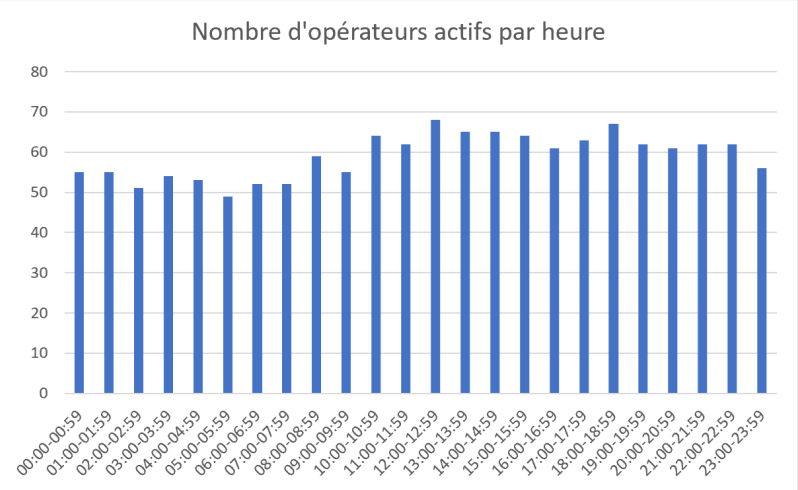

Figure 27: Shodan - Activités des opérateurs par heure

5.2.3 Résultats des scans

L'approche des différents types de scans a été expliquée dans le chapitre 3.6.3. Ici ce sont les résultats obtenus avec chaque démarche qui sont présentés. Les adresses scannées représentent une petite partie des 4'294'967'296 adresses, ou plutôt 3'419'656'832 adresses en enlevant celles qui sont réservées [64]. Cependant, ces différents essais ont permis d'acquérir des connaissances pour des scans qui seraient plus grands.

Scans par ASN

Le scan par Autonomous System Number a été utilisé en tout premier lieu pour la Turquie. Malgrès les restrictions, la plus grande partie du pays a été scannée [64], soit 9'995'262 adresses. Pour le pays, étant placé en tête de liste, nous avons été déçus de ne trouver qu'un opérateur. Celui-ci avait d'ailleurs déjà été trouvé par Shodan.

Par le même principe, nous avons continué par scanner la Grèce (2'807'552) et l'Arménie (253'952), où nous n'avons trouvé aucun opérateur. Finalement, nous avons découvert 2 opérateurs en Russie, en scannant 16'627'794 adresses. Ceux-ci se trouvaient aussi déjà sur Shodan.

Scans par pays

Dans le but de trouver des opérateurs n'étant pas dans la base de donnée de Shodan, le champ du scan fût élargi à un pays. Nous avons choisi l'Espagne, qui compte moins d'adresses que l'Allemagne ou la France. Nous aurions aussi pu prendre la Suisse pour des statistiques plus locales. Nous avons aussi augmenté nos chances en ajoutant un port, le 81, qui est le deuxième plus employé.

Après avoir démarré le scan des 33'020'807 adresses, nous avons dû très vite constater que l'ajout de ce port a considérablement augmenté le temps nécessaire. Il n'a pas simplement doublé, le port 81 est bien plus fréquemment employé que le 1604. Nmap prenait donc un temps considérable pour déterminer s'il s'agissait d'un opérateur DarkComet. Le plus frustrant ne fut finalement pas la période d'attente, mais le fait qu'il n'y ait pas un seul opérateur qui soit découvert à travers tout l'Espagne.

Scans par hosting provider

Il fallait donc quand même plus cibler la recherche dans l'espoir de découvrir des opérateurs. Shodan trouvait pas mal d'opérateurs utilisant OVH, un hosting provider français. Nous avons donc scanné leur réseau, soit 2'727'424 adresses. Nous nous sommes par contre à nouveau limité au port 1604. C'est là que nous sommes arrivés le plus rapidement à 3 nouveaux opérateurs.

Fort de cet exploit, nous avons encore essayé d'autre hosting providers: partiellement Akamai (2'629'376) et Amazon (32'843'792), GoDaddy (1'012'992) et Hostey (14'336). Malheureusement sur c'est hosting provider, nous n'avons trouvé aucun opérateur.

Dans tous les scans, c'est OVH qui a été le plus intéressant au niveau du nombre d'opérateur et du fait qu'il n'était pas découvert par Shodan. Cependant, par la suite, ils ont tous été ajoutés dans la base de données de Shodan.

5.3 Online-Analysen

Die Online-Analysen erfolgten in zwei Zeitabschnitten. Der erste Abschnitt dauerte 3.5 Wochen, vom 2.April bis am 26.April. In dieser Zeit wurde auf dem Desktop-Computer gearbeitet. Der zweite Abschnitt dauerte dann 4 Wochen, vom 30.April bis am 28.Mai und spielte sich auf dem leistungsstärkeren Server ab. Genauere Informationen zum Design des Host- und des Gast-Systems in den Kapitel 3.1 und 3.2. Die zwei Analyse-Perioden werden als 1.Experiment und 2.Experiment gekennzeichnet.

Auf dem Desktop-Computer, im 1.Experiment, haben wir mit drei virtuellen Maschinen gearbeitet. Diese wurden für die Analysen jeweils 15 Minuten betrieben.
Auf dem Server, im 2.Experiment, haben wir mit sechs virtuellen Maschinen gearbeitet. Diese wurden für die Analyse jeweils 60 Minuten betrieben.

Es werden Resultate von beiden Experimenten präsentiert. Jedoch wird dann nur auf die Daten des 2.Experiments vertieft eingegangen. Aus folgendem Grund: Da wir die Analysen des 1.Experiments, auf nur 15 Minuten designt haben, ist davon auszugehen, dass viele Operatoren mitten im Prozess unterbrochen wurden. Zudem waren die virtuellen Maschinen noch nicht so ausgereift und differenziert wie die finale Version, geschaffen nach der Vorlage aus dem Opfer-Design im Kapitel 3.2.1.

5.3.1 Übersicht

Insgesamt wurden 741 RAT-Samples in unseren virtuellen Maschinen einer Online-Analyse unterzogen. Davon sind 132 Analysen erfolgreich gewesen. Das heisst, dass bei insgesamt 17.8% der Analysen erfolgreich eine Verbindung zum Operator hergestellt werden konnte. Anders gesagt, in diesen 17.8% hat ein Operator eine unserer virtuellen Maschinen übernommen. Wie im Table 19 zu sehen ist, haben wir im 2.Experiment um gute 3% erfolgreicher abgeschlossen. Im 1.Experiment haben wir total, im Vergleich zum 2.Experimen, lediglich 14% soviele Samples analysieren können. Dies beeinflusst die Resultate natürlich ebenfalls. Wir gehen aber stark davon aus, dass das neue Design bezüglich den virtuellen Maschinen, sowie auch die angepassten Analyse-Zyklen, einen positiven Einfluss hatten.
Insgesamt wurde wärend 40'320 Minuten oder anders gesagt wärend 672 Stu-

	1.Experiment	2.Experiment
tot. Anz. Analysen	92	649
erfolgr. Analysen	14	118
erfolgr. Analysen [%]	15.2	18.2

Table 19: Anzahl durchgeführter Analysen, bez. ihrer Erfolgsrate

den online analysiert. Als erfolgreich erwiesen sich davon 7'290 Minuten, sprich 121.5 Stunden. Als erfolgreich gezählt werden die Analysen, in welchen ein Operator aktiv wird. Es wurde Operator-Aktivität aufgezeichnet für insgesamt 932.3 Minuten oder 15.5 Stunden. Betrachtet man die Dauer der Operator-Aktivität,

im Table 20, relativ zur Dauer der erfolgreichen Online-Analysen, so ergibt das einen Wert von 8.5% für das 1.Experiment. Im 2.Experiment steigerte sich dieser Wert um 4.4% auf 12.9%. Auch dies ist ein weiteres Indiz, das unser revidiertes Design in eine gute Richtung weist.

	1.Experiment	2.Experiment
tot. Dauer Online-Analysen [min]	1380	38940
erfolgr. Online-Analysen [min]	210	7080
Operator Aktivität [min]	17.8	914.5
rel. Operator Aktivität [%]	8.5	12.9

Table 20: Dauer der Analysen, bez. Erfolgsrate und Operator-Aktivität

Ab Table 21 habe wir dann nur noch auf das 2.Experiment Bezug genommen. Es zeigt sich, dass unser automatisiertes System, die RAT-Samples gleichmässig auf die sechs virtuellen Maschinen verteilt hat. Die leichte Diskrepanz lässt sich dadurch erklären, dass gewisse Operatoren zum Zeitpunkt der Analyse bereits nicht mehr online waren. Ebenfalls die Abweichung bezüglich den erfolgreichen Analysen, sagt noch nichts über die Qualität der jeweiligen Maschine aus. Es hängt viel mehr davon ab, ob ein Operator überhaupt online ist und sich verbindet.

	tot. Anz. Analysen	erfolgr. Analysen	erfolgr. Analysen [%]
Maschine 1	111	21	18.9
Maschine 2	116	20	17.2
Maschine 3	106	17	16.0
Maschine 4	105	17	16.2
Maschine 5	106	20	18.9
Maschine 6	105	23	21.9

Table 21: Verteilung der Samples und Erfolgsrate der virt. Maschinen

5.3.2 Aktivität der Operatoren

Im der Figure 28 sind sämtliche Analysen des 2.Experiments abgebildet. Das Diagramm zeigt die Analysen, nach Aktivität geordnet. Die Aktivität ergibt sich aus der Anzahl Befehlen, welche vom Operator benutzt wurden und der Zeitspanne, in welcher der Operator aktiv war. Diese Zeitspanne ist ein Delta-Wert, angefangen beim ersten, bis zum letzten Befehl, welcher vom Operator ausgeht. Sehr viel Netzwerkverkehr oder eben Befehle, werden automatisch generiert und wurden für die Auswertung herausgefiltert. Es ist eine Streuung zu sehen, wobei der Schwerpunkt im unteren Bereich liegt und scheinbar gegen Null zieht. In der

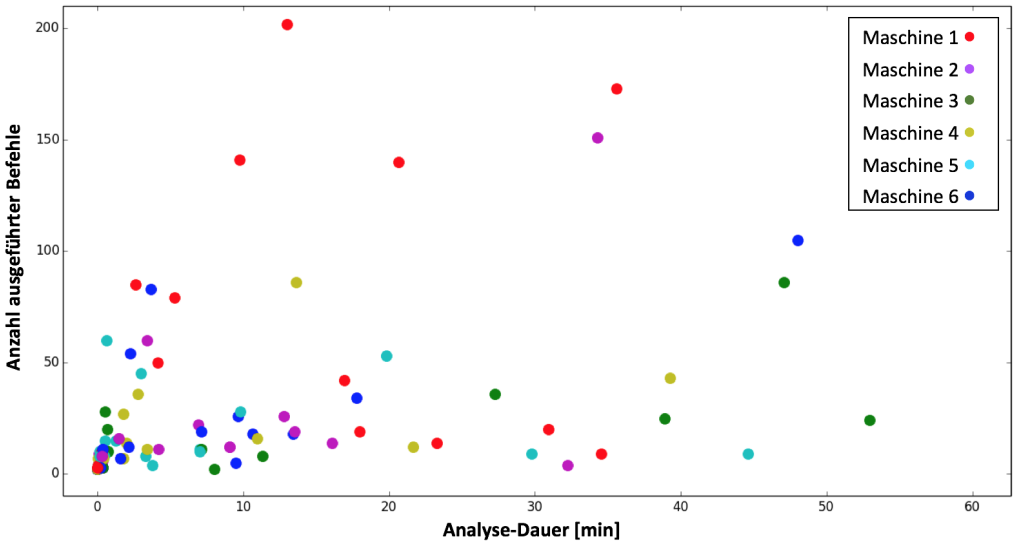

Figure 28: Aktivität der Operator

Figure 29 wurde genau dieser Ausschnitt herangezoomt. Es handelt sich um die Analysen, welche nur sehr kurz andauerten (unter einer Minute) und eher wenige Befehle aufwiesen. In diesem Ausschnitt sind dennoch knapp über 30 Analysen zu finden, welche in Figure 28 kaum zur Geltung kommen. Das ist ein guter Viertel, der gesamten Analysen, welche sich in diesem kleinen Bereich auffinden.

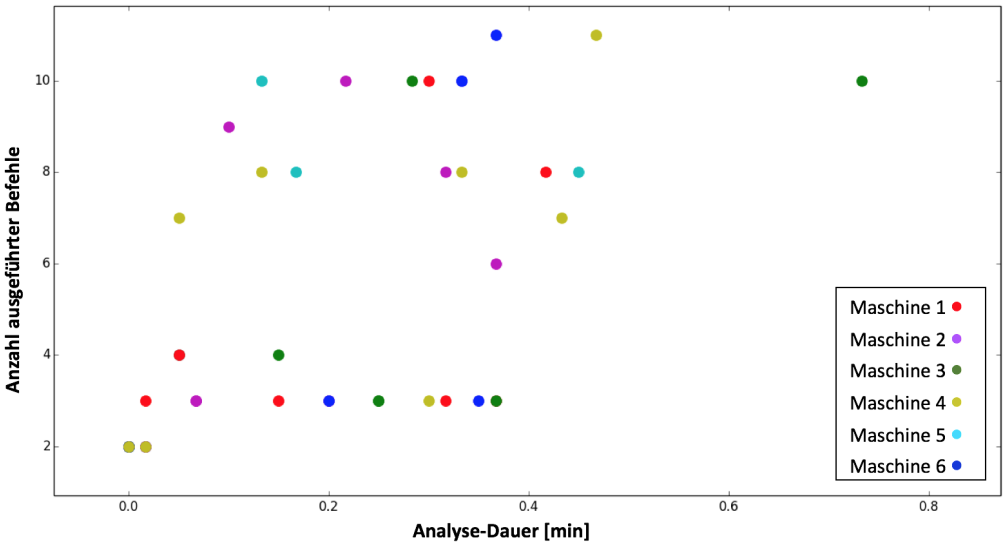

Figure 29: Ausschnitt der Aktivität der Operator

Wir haben versucht diese nach Aktivität geordneten Analysen, grob zu kategorisieren. Dies erlaubt direkt einen Rückschluss auf das Verhalten der Operatoren. In der Figure 30 werden diese dargestellt. Im Bereich 1. finden sich die kurzzeitig-passiven Operatoren. Es wurden in bis zu 20 Minuten, maximal 30 Befehle ausgeführt. Über 73% der Analysen fallen auf diesen Ausschnitt. Im Bereich 2. lassen sich die langzeitig-passiven Operatoren finden. Von 20 bis zu 60 Minuten, mit ebenfalls maximal 30 Befehlen. Nicht ganz 8% weisen ein solches Verhalten auf. Der Bereich 3. umfasst alle kurzzeitig-aktiven Operatoren, welche mindestens 30 Befehle, in bis zu 20 Minuten ausführten. Knapp 13% aller Analysen befinden sich in diesem Ausschnitt. Der letzte und 4. Bereich zielt auf die langzeit-aktiven Operatoren ab. Mindestens 30 ausgeführte Befehle, in 20 bis 60 Minuten. Dieser Ausschnitt weist lediglich knappe 6% auf. Die meisten Operatoren sind also eher kurzzeitig-passiv, wobei passiv hier ein relativer Begriff ist, welchen wir selber definiert haben.

Es stellte sich die Frage, ob irgendwelche äusseren Einwirkungen den Schwerpunkt in Figure 30 auf diesen Bereich 1. definieren? Zwei, aus unserer Sicht massgebenden Faktoren, wurden analysiert.

1. Der erste Faktor ist die Anzahl Befehle, welche hierbei für unsere Auswertung und die Graphiken berücksichtigt wurden. Insgesamt wurden 2717 Befehle registriert, exklusive den automatisch generierten. Von diesen 2717 Befehlen sind jedoch immer noch sehr viele das Resultat von automatischen Antwort-Befehlen oder Befehls-Ketten, welche jeweils generiert werden. So wird bei Daten-Abfragen normalerweise mit einem "Dataflux-XXX" geantwortet, bevor die eigentlichen Daten übertragen werden. Bei Bot-Befehlen oder Server-Actions, sowie auch vielen Fun-Befehlen, wird lediglich einseitig kommuniziert. Ebenfalls gibt es gewisse Befehle, welche in einer

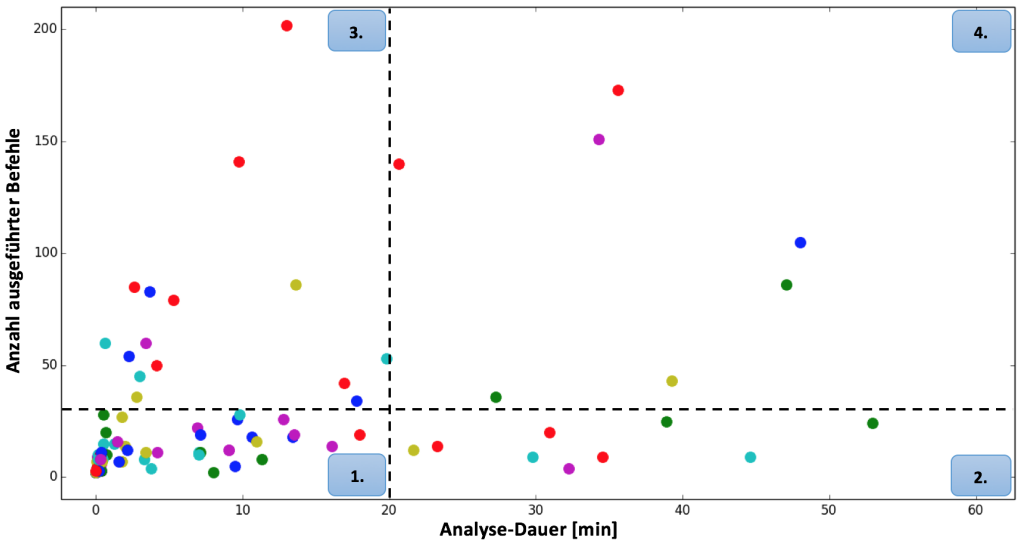

Figure 30: Differenzierung der Operator nach Aktivität

ganzer Kette daherkommen. Das heisst, dass für eine einzige Aktion bis zu vier Befehle generiert werden. Zwei vom Operator zum Opfer und zwei in umgekehrter Richtung. Im Table 22 wird in der Spalte "Anz. Befehle", der gefilterte Realwert präsentiert. Somit kommt man insgesamt auf nur noch 1274 Befehle, was nicht ganz 47% der ursprünglichen Anzahl entspricht. In der Spalte "rel. Anz. Befehle" wurde noch einmal vereinfacht und lediglich die relative Anzahl der Befehle gezählt. Das heisst, dass jeder in einer Analyse vorkommende Befehl, maximal einmal gezählt wird. Es gab diverse Analysen, wo die gleichen Befehle teilweise bis zu 50 mal verwendet wurden. Mit dieser Auswertung erhält man nun eine Übersicht, bezüglich der relativen Verwendug der Befehle, pro Analyse. Somit werden total nur noch 509 Befehle gezählt. Dies entsprich dann noch 19% der ursprünglichen Anzahl. Die Kategorie der Befehle kann bereits einen grosse Einfluss haben, bezüglich der Position auf dem Aktivitäts-Diagramm Figure 30. Dies bewirkt jedoch eher eine noch grössere Verstreuung und nicht eine Schwerpunktverschiebung. Dieser Faktor fällt also weg. Man kann aber davon ausgehen, dass sich der Schwerpunkt noch verstärkter im Bereich 1. zusammenziehen würde. Die meisten Analysen würden auf der Graphik aber beinahe proportional zueinander verschieben. Dies, da bei den meisten Analysen jeweils viele verschiedene Befehlen eingesetzt wurden. Schlussendlich würde die Graphik sehr ähnlich bleiben, mit einer deutlich reduzierten Y-Achse.

2. Der zweite Faktor könnte das Zeit-Delta zwischen Analyse-Start und erstmaligem reagieren des Operators sein. So kann es durchaus vorkommen, dass ein Operator erst nach 50 Minuten laufender Analyse selber tatsächlich aktiv wird und ins Geschehen eingreift. Somit würde die Analyse nach weiteren 10 Minuten, da die Analysen auf 60 Minuten designt und limitiert wurden, abbrechen. Dies geschieht ganz gleich, ob der Operator noch aktiv

ist oder bereits nicht mehr. Dieser Faktor konnte jedoch wiederlegt werden. Gerade mal bei 7 Analysen wurde der Operator erst nach über 40 Minuten laufender Analyse aktiv. Dass heisst, das nur bei diesen 7 Analysen, falls die jeweiligen Operatoren länger als 20 Minuten aktiv gewesen wären, die Verbindung früher abgebrochen hätte. Bei 94% hatte dieses Zeit-Delta also keinen Einfluss und das jeweilige Abbild bezüglich der X-Achse ist korrekt. Das Design mit einer Analyse-Dauer von 60 Minuten hat sich erneut bewiesen.

5.3.3 Vergleich der virtuellen Maschinen

Im Weiteren werden die virtuellen Maschinen, alias Opfer, bezüglich der Auswertung in Figure 30 untersucht. Im direkten Vergleich schneiden die sechs virtuellen Maschinen relativ ähnlich ab. Maschine01 sticht etwas hervor, da sie als einzige mit nicht ganz 50% in Bereich 1. am wenigsten vertreten ist. Dafür ist sie in Bereich 3. und 4. relativ stark vertreten. In der Figure 29 lässt sich bezüglich Bereich 1. mehr erkennen. Maschine01 und Maschine03 sind mit sechs und vier Analysen, in den Langzeit-Bereichen 2. und 4. am häufigsten vertreten. Bis auf die Maschine01 sind alle Maschinen mit über 70% vor allem im Bereich 1. vertreten. Die Maschinen01 bis 05 waren ja beinahe identisch. Lediglich unterschiedliche Desktopaktivität und Desktophintergrund unterscheidet sie. Die Diskrepanzen in den Resultaten, sind aller Wahrscheinlichkeit statistischen Ausreissern zuzuschreiben. Unsere Annahme ist, dass sich diese Effekte, bei einer grösseren Anzahl von Analysen verschwinden würde.

Maschine06, mit der virtuellen Webcam, wäre die einzige gewesen, welche noch mehr geboten hätte. Diese ist aber mit über 80% vor allem in Bereich 1. vertreten. In den Langzeit-Bereichen ist diese nicht stark vertreten. Wir schlussfolgern daraus, dass die tatsächliche Möglichkeit, sich mit einer Webcam zu verbinden, nicht all zu grossen Einfluss hat, auf die Aktivität eines Operators. Oder es besteht eine einfache Möglichkeit, die Webcam als virtuell zu detektieren und somit ist die Wirkung viel mehr abschreckend. Im Idealfall würde eine echte Webcam eingesetzt werden, mit welcher sich die Operatoren verbinden können. In unserem Design gab es zwei Problme, welche dies verhinderten.

1. Wir haben durch das Machine-Hardening, erklärt im Kapitel 3.2.3, die I/O-Komponenten, welche der Hypervisor benutzen würde, um eine Webcam vom Host-System der virtuellen Maschine zur Verfügung zu stellen deinstalliert und gelöscht. Somit konnte eine USB-Webcam nicht mehr weitergeleitet werden in die virtuelle Umgebung.

2. Selbst wenn wir Punk 1. umgangen wären, hätten wir für jede der sechs virtuellen Maschinen, eine eigene Webcam zur Verfügung stellen müssen. Eine Hardware-Ressource kann normalerweise nur einmal zugeteilt werden.

5.3.4 Befehle & Aktionen

Kategorie & Befehl-Name	Anz. Befehle	in %	rel. Anz. Befehle	in %	pass.	akt.	aggr.
Spy Functions							
Sound-Capture	11	0.9	6	1.2		X	
List Cams	80	6.3	37	7.3	X		
Cam-Capture	3	0.2	2	0.4		X	
List Monitors	78	6.1	58	11.4	X		
Remote-Desktop	88	6.9	56	11.0		X	
Desktop Snapshot	89	7.0	54	10.6		X	
I/O-Control	81	6.4	56	11.0			X
KeyLogger	145	11.4	43	8.4		X	
Automatic							
Bot-Functions	17	1.3	17	3.3			X
Uninstall	3	0.2	3	0.6			X
File Manager							
File Explorer	296	23.2	37	7.3	X		
Upload	2	0.2	1	0.2			X
Download	21	1.6	5	1.0		X	
Fun Functions							
Chat	10	0.8	4	0.8			X
Other	94	7.4	40	7.9			X
MISC							
Clipboard	3	0.2	3	0.6		X	
Network Functions							
DDOS-UDP-Flood	12	0.9	5	1.0			X
Data-Theft							
Password	158	12.4	49	9.6		X	
uTorrent-Data	8	0.6	2	0.4		X	
Server Actions							
Upload & Execute	62	4.9	21	4.1			X
Update	6	0.5	3	0.6			X
System Functions							
Process Manager	2	0.2	2	0.4		X	
Power OFF	1	0.1	1	0.2			X
System Infos	4	0.3	4	0.8	X		
Total	1274		509				

Table 22: Auflistung benutzer Befehle

Im Table 22 wurden die bereits gefilterten Befehle nach Kategorien geordnet. Im Detail erklärt werden die Befehle und Befehls-Ketten in der DarkComet-Dokumentation, aufzufinden in Apendix A. Die Befehle wurden zusätzlich als "passiv", "aktiv" oder "aggressiv" differenziert und gekennzeichnet. Passive Befehle bleiben von der Opfer-Seite unbemerkt. Es werden lediglich Basis-Informationen abgegriffen und noch nicht direkt auf die eigentlichen Daten-Quellen zugegriffen.

So werden zB. die verfügbaren Monitore für eine Remote-Desktop Verbindung angezeig oder der Operator arbeitet sich durch das File-System. Er bekommt noch keine direkte Einsicht in die Dateien. Lediglich Verzeichnisse, Ordner und Datei-Namen sind sichtbar. Unter einem aktiven Befehl versteht man zB. den Zugriff auf eine Datei, sprich den Download einer Datei. Alle übertragenen Audio- und Video-Daten, sowie Remote-Desktop Aktionen, gehen ebenfalls unter aktiv. Aggressive Befehle gehen noch einen Schritt weiter. Hier macht sich der Operator bemerkbar oder/und manipuliert die Opfer-Maschine. So kann ein Remote-Desktop-Zugriff zuerst nur aktiv sein. Sobald aber die I/O-Control eingeschalten wurde, wird Maus und Tastatur übernommen und der Operator macht sich sichtbar, was dann unter aggressiv einzuordnen ist. Sämtliche Fun-Befehle gehen ebenfalls unter aggressiv, da diese genau das Ziel haben, sich beim Opfer bemerkbar zu machen. Uploads, Updates oder Fern-Installationen sind ebenfalls als aggressiv einzustufen, da diese die Opfer-Maschinen verändern und beeinflussen.

Die relative Anzahl der Befehle, im Table 22, beleuchtet die Statistik nochmals von einer anderen Seite. Beispielsweise werden in den häufigsten Fällen, pro Analyse maximal eine Remote-Desktop Verbindung aufgebaut. Dagegen wird der File-Explorer natürlich nicht nur ein einziges Mal, sondern vermehrt eingesetzt, wenn das File-System durchforstet wird. Beim File-System macht das einen Unterschied von über 87.5%, während beim Remote-Desktop lediglich ein Unterschied von 36% besteht. Um also Befehle bezüglich ihrem relativen Vorkommen pro Analyse zu bewerten, orientieren wir uns der relativen Anzahl.

Am häufigsten vertreten sind die Remote-Desktop Befehle. Mit insgesamt 44% decken sie wohl das grösste Interesse oder die grösste Aktivität ab. Die Zahlen zeigen, dass beinahe jede Remote-Desktop-Verbindung mit einem Auflisten der verfügbaren Monitore beginnt, dann tatsächlich eine Verbindung aufbaut und die I/O-Control einschaltet. Ebenfalls werden auch fast immer Snapshots vom Desktop genommen. Mit 9.6% ist der Password-Theft der zweithäufigste Befehl. Der dritthäufigste Befehl ist dann der KeyLogger mit 8.4%. Das Interesse an diesen zwei letzteren Befehlen ist wohl dasselbe und zielt insbesondere auf die Passwort-Daten der Opfer ab.

Vereinfacht man noch weiter nach Befehlsgruppen, so ergibt sich die Figure 31. So fallen unter Remote-Desktop dann alle oben erwähnten Befehle. Diese werden auch noch einmal relativiert. List Monitors, Desktop Snapshot und I/O-Control fallen weg und man zählt insgesamt noch 56 Remote-Desktop Verbindungen. Es wird jeweils also noch eine einzige, zusammenhängende Aktion gezählt. So ergibt sich dann in Figure 31 Graphik bezüglich den Befehlsgruppen. Remote Desktop ist mit 16% nach wie vor die häufigste vorkommende Aktion.

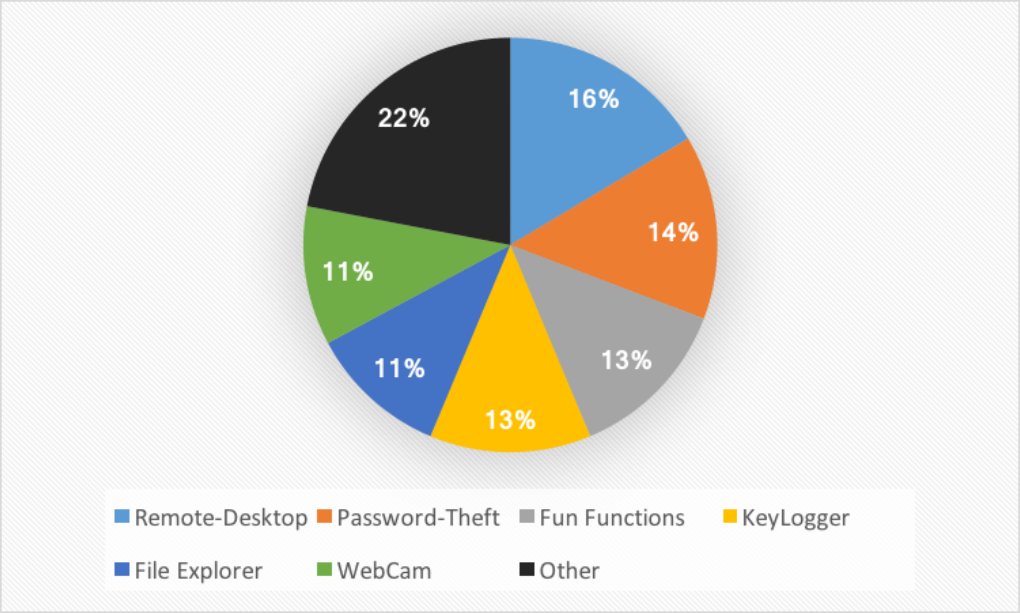

Figure 31: Befehlsgruppen

In Figure 32 haben wir den Ablauf bezüglich initialen und finalen Aktionen vereinfacht dargestellt. Alle rot gefärbten Verbindungen führen zum ersten und gleich auch letzten Befehl. Das heisst, die Operator tätigten eine einzige Aktion. Die blau gefärbten Verbindungen zeigen Remote-Desktop Aktivität. Die jeweiligen Zahlen kennzeichnen die Operatoren, welche in diesem Stadium einer spezifischen Aktion nachgekommen sind. Die Verbindungen bilden ab, wieviele Operatoren von einer Aktion zur nächsten übergehen.
Spitzenreiter der initialen Aktionen ist Password-Theft, wobei bei einem Drittel dies auch gleich die finale Aktion ist. Zweithäufigste initiale Aktion ist Remote Desktop. Nicht ganz die Hälfte der Benutzer von Remote Desktop starten mit dieser Aktion gleich zu Beginn. Die Server Actions, genauer gesagt hauptsächlich Upload & Execute, machen die dritthäufigste initiale Aktion aus. Bei 65% ist dies auch die letzte. Bei einigen Operatoren wird initial, auch gleich nach einer Webcam gesucht. Einige wenige beginnen direkt mit der Erforschung des File-Systems. Interessant ist, dass sehr viele Operatoren, im Verlauf der Analyse noch auf Remote Desktop wechseln, wenn sie es nicht schon initial getan haben. 84% der Remote-Desktop Benutzer stoppen ihre Session als letzte Aktion auch wieder. Die Restlichen brechen die Verbindung zum Opfer einfach so ab oder wurden durch die maximalen 60 Minuten Analyse-Dauer unterbrochen.
Password-Theft ist die zweithäufigste finale Aktion. Vor Beendigung der Session wird vielfach nocheinmal versucht, die Passwörter abzugreifen. Die Server Actions machen die dritthäufigste finale Aktion aus. Dies möglicherweise um noch eine weitere Malware zu installieren und die Opfer-Maschine auf Dauer, in einem weiteren Kontext zu kompromittieren.
Es gab vier Distributed Denial of Service (DDoS) Attacken, welche gleich als finale Befehle auftraten. Und schlussendlich haben drei Operator den Bot benutzt, um am Ende den RAT wieder zu deinstallieren.

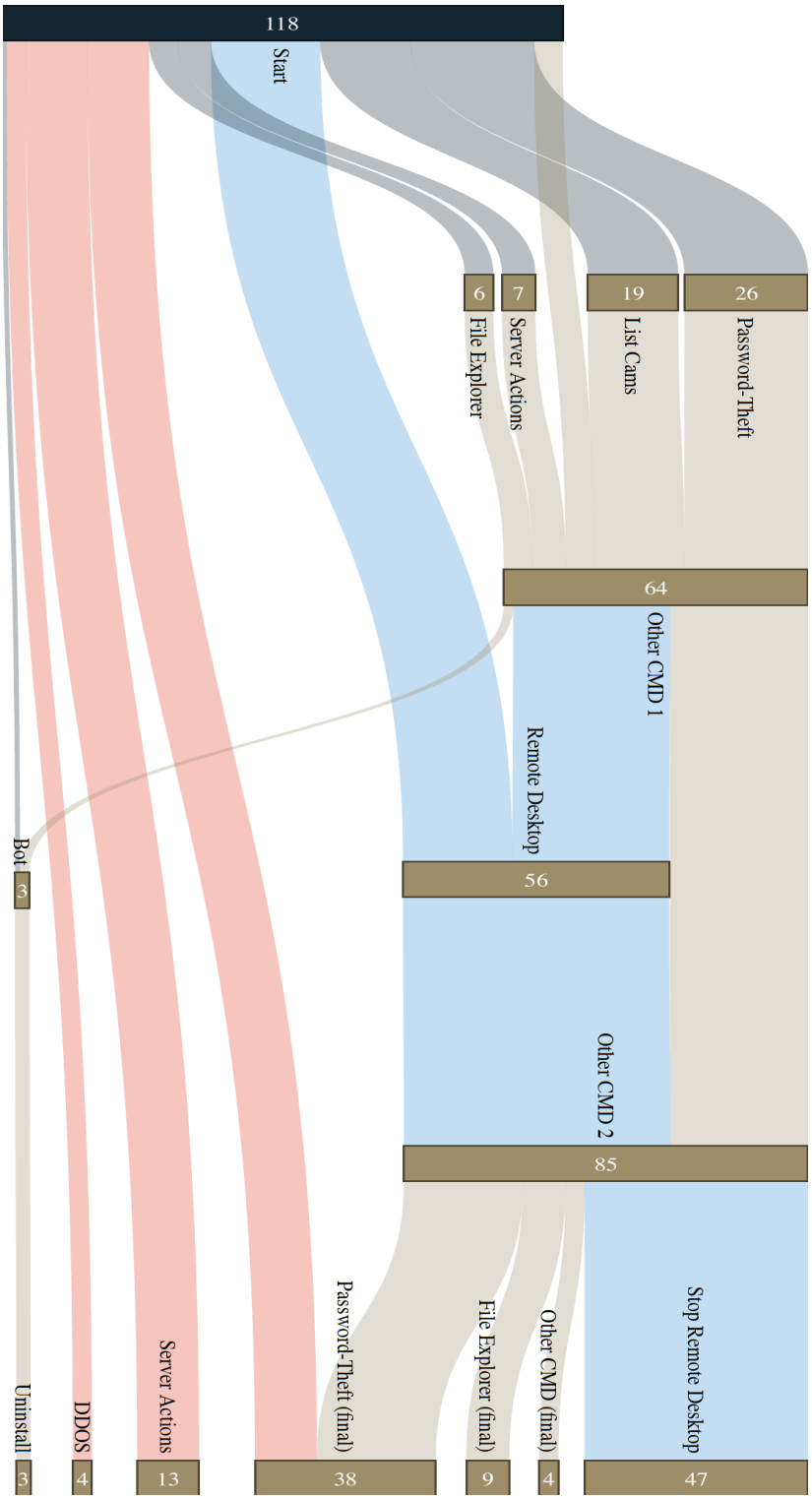

Figure 32: Sankey-Diagramm bezüglich den Befehlen

5.3.5 Befehls- & Operatorkategorien

Wir haben überprüft, inwiefern die Operatoren auf die drei Kategorien bezüglich den Befehlen verteilt sind. Im Table 22 sind diese in passiv, aktiv oder aggressiv eingestuft. In der Figure 33 werden diese bezüglich der Verteilung auf die Analysen abgebiltet. 81% der Analysen beinhalten passive Befehle, 51% aktive Befehle und in 75% der Analysen wurden auch aggressive Befehle benutzt.

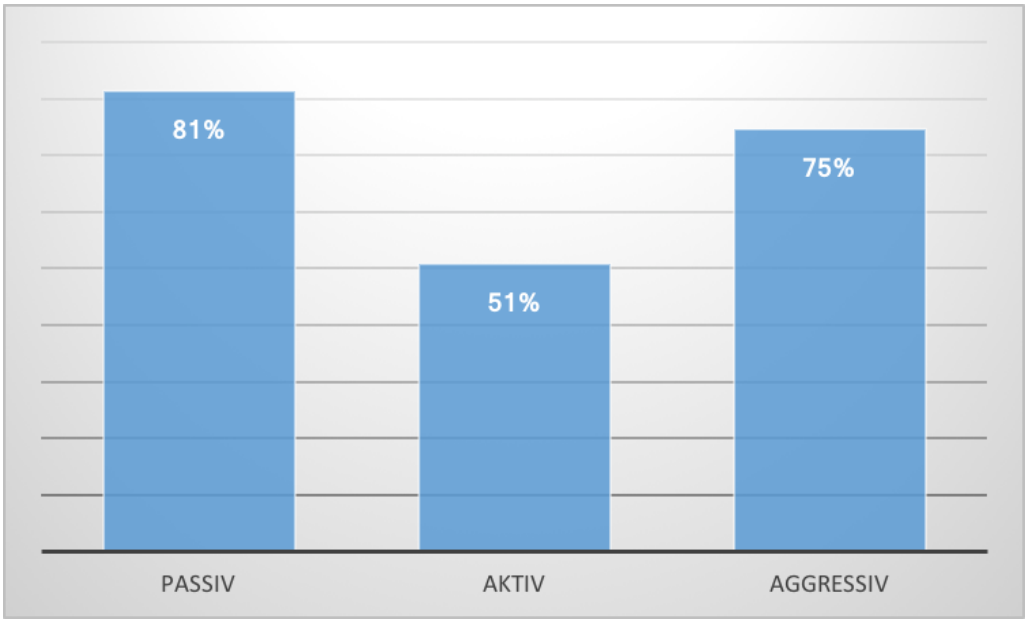

Figure 33: Befehlskategorien

In der Figure 34 wird diese Verteilung nochmals von einem anderen Standpunkt aufgezeigt. Gerade mal 2% der Analysen bestehen aus rein passiven Befehlen. 23% der Analysen bestehen aus passiven und aktiven Befehlen, mit mindestens einer aktiven Aktion. Zusammengefasst ergeben das lediglich 25% der Analysen, in welchen die Operatoren sich bedeckt halten. Bei den restlichen 75% macht sich der Operator mit einer aggressiven Aktion bemerkbar.

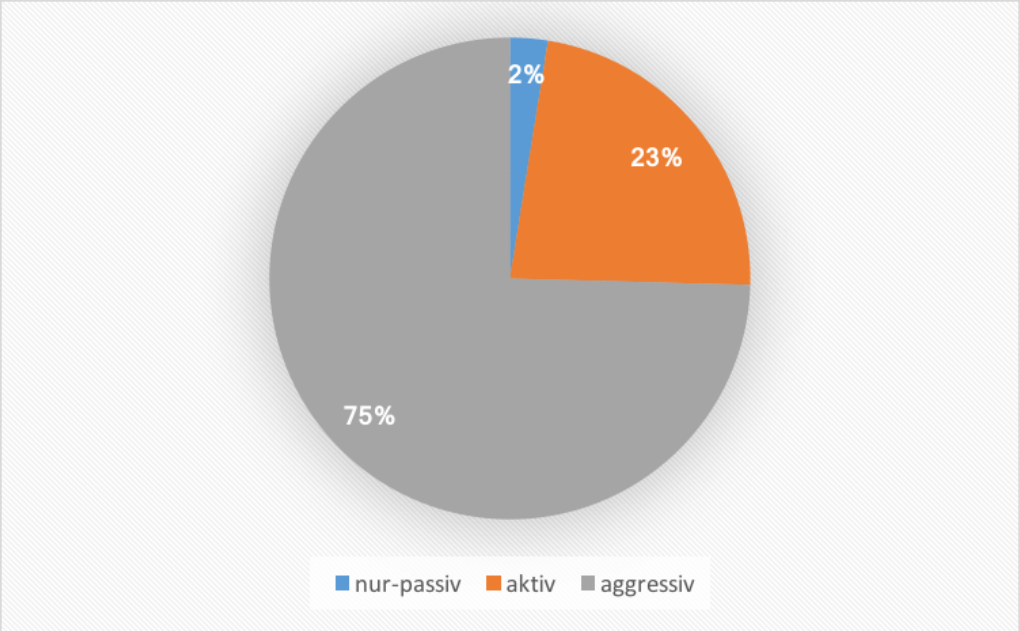

Figure 34: Operator-Verteilung bezüglich Befehlskategorien

In der Figure 35 wird eine Kategorisierung bezüglich den Operatoren abgebildet. Es wurden drei zusammenfassende Kategorien definiert.
"Spion" beinhaltet alle Operatoren, welche während den Analysen Befehle benutzten, um an Informationen und Daten zu kommen. Dazu zählen alle Spy Functions, exklusive I/O-Control. Weiter gehört dieser Kategorie der File Explorer an, sowie sämtliche Downloads von Daten und Data-Theft. I/O-Control ist zwar ein beinahe fester Bestandteil von Remote-Desktop Verbindungen, jedoch zählen wir diesen Befehl zur Kategorie "Spass & Störung". Dies daher, dass bei einer seriösen Informationsbeschaffung grundsätzlich kein I/O-Control nötig sein würde. Man könnte alle Daten via File Explorer ausfindig machen und herunterladen. Remote-Desktop kann beispielsweise für die Aufzeichnung des Verhaltens vom Opfer gebraucht werden und geht somit unter Spionage/Informationsbeschaffung. I/O-Control dagegen würde hier von keinem ersichtlichen Nutzen sein. "Spass & Störung" umfasst alle Fun Functions, I/O-Control, DDOS-Attacken, sowie weitere Befehle, welche in erster Linie die Opfer-Maschine beeinflussen und beschädigen. Diese Kategorie ist grundsätzlich nicht von konstruktivem Charakter und dient einem Operator, in unserem Kontext, nicht für weitere Aktionen. Wäre das Opfer nicht ein normaler User, wie in unserem Fall, sondern eine Schlüsselkomponente, zB. ein wichtiger Server, so könnten DDOS- oder PowerOff-Attacken wiederum sinnvoll sein. Dies hängt vom jeweiligen Kontext ab. "Eindringling" beschreibt die Kategorie von Operatoren, welche Daten hochladen. Dafür wird Upload im File Manager und Upload&Execute von den Server Actions benutzt. Diese Aktionen können verschiedene Hintergründe haben. Die Platzierung von Daten, um später zu erpressen, wäre eine Möglichkeit. Am häufigsten wird vermutlich weitere Malware hochgeladen und installiert, für zukünftige Angriffe und Abgriffe von Daten.

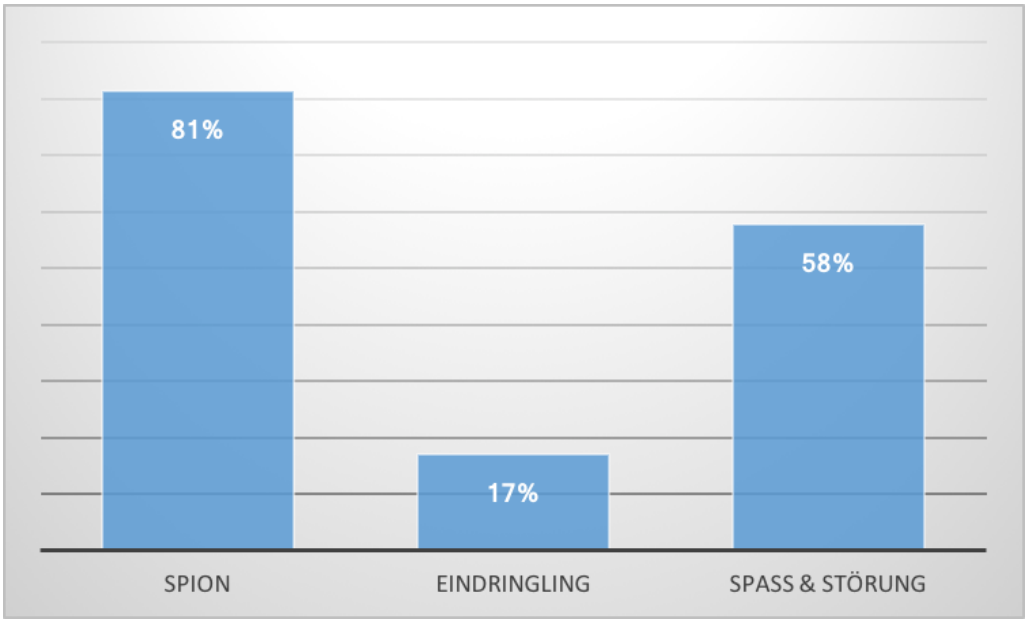

Figure 35: Operatorkategorien

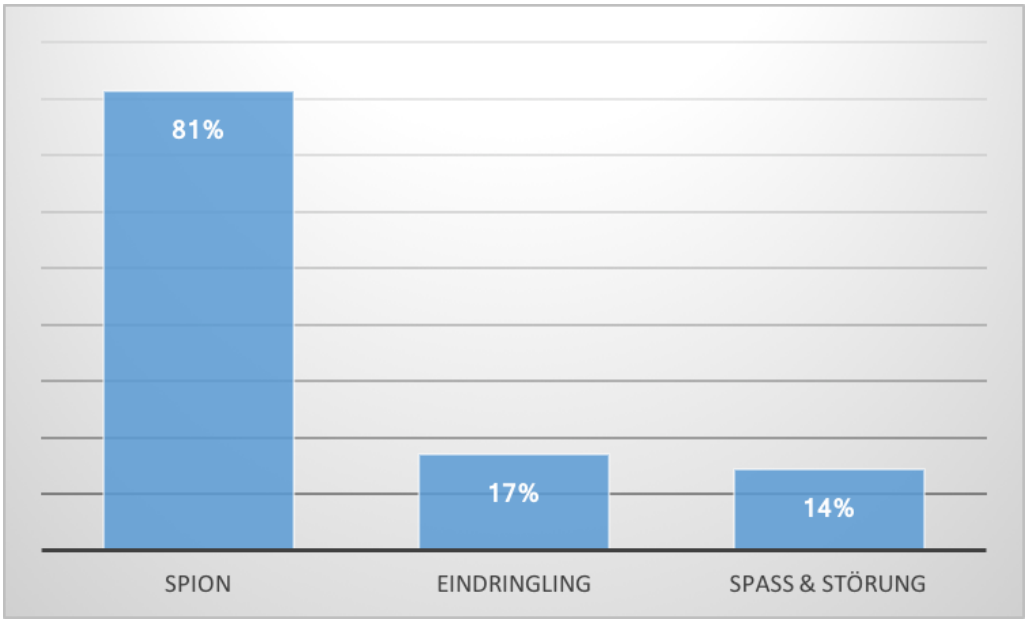

Figure 36: Operatorkategorien (ohne I/O-Control)

In Figure 35 wird abgebildet, dass mit 81% die meisten Operatoren im Bereich Informationsbeschaffung oder eben Spionage aktiv sind. 58% der Operatoren sind in unserem Kontext und nach unserem Verständnis als Störungsfaktoren aktiv, ohne mit diesen Aktionen ein konkretes Ziel zu verfolgen. Gerade mal 17% der Aktionen werden den Eindringlingen zugeschrieben, welche hingegen sehrwahrscheinlich noch weitere Ziele verfolgen.

Wir haben zum Vergleich noch eine weitere Graphik erstellt, wobei I/O-Control nicht zu "Spass & Störung" gezählt wird. In Figure 36 ist zu sehen, wie diese Kategorie, ohne I/O-Control nur noch den kleinsten Anteil von insgesamt 14% ausmacht. Das bedeutet, dass 44% der Analysen lediglich durch I/O-Control in diese Kategorie gerutscht sind. So bestehen diese 14% fast hauptsächlich aus Fun Functions. Welche Graphik nun als richtungsweisender verstanden werden sollte, kann nicht abschliessend definiert werden.

5.3.6 Mehrfachanalyse von Operatoren

Weiter wurde analysiert, wie häufig ein Operator, also ein RAT-Sample erfolgreich einer Online-Analyse unterzogen wurde. Dabei hat sich herausgestellt, wie in Figure 37 zu sehen ist, dass ein grosser Unterschied besteht. 14 Operatoren wurden genau einmal erfolgreich analysiert. 10 Operatoren wurden zwei- bis dreimal und 9 Operatoren vier- bis siebenmal erfolgreich analysiert. Die zwei statistischen Aussreisser wurden 12 und 14 Mal einer erfolgreichen Analyse unterzogen. Klar ist, dass einige wenige Operatoren, einen relativ grossen Einfluss auf unsere Statistik haben.

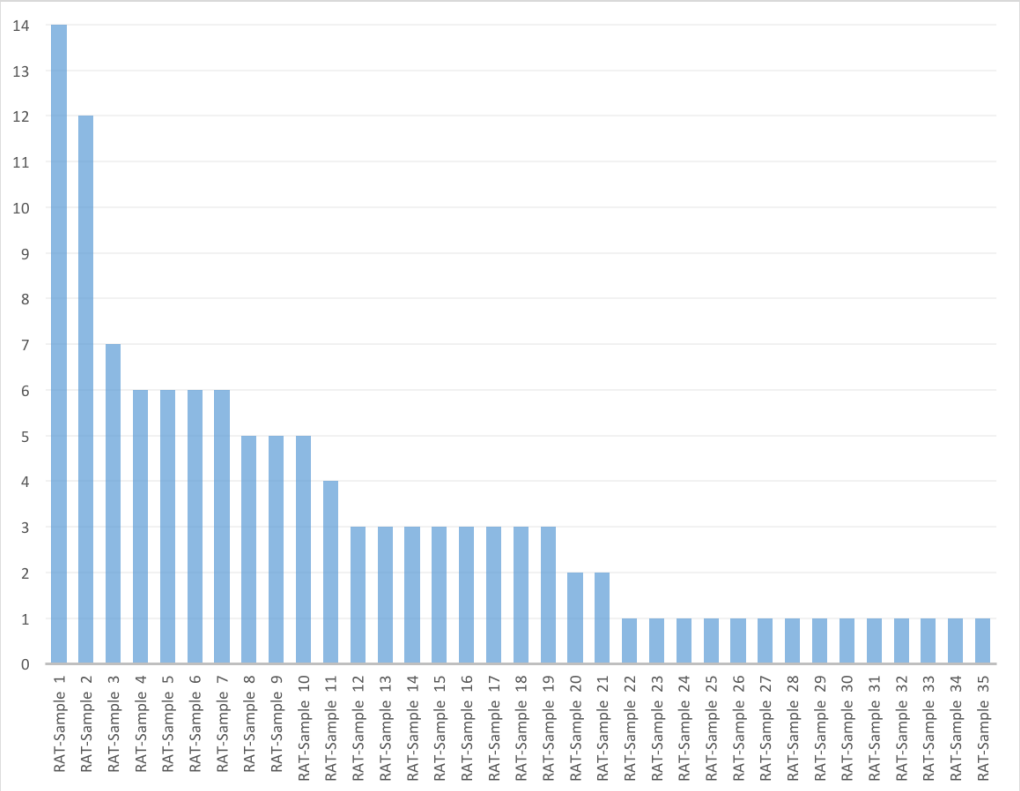

Figure 37: Anzahl erfolgreicher Analysen pro Sample

Im Table 23 werden die jeweiligen Analyse-Daten, der fünf am häufigsten analysierten Operatoren aufgezeigt. Es lies sich kein eindeutiges Muster erkennen. Jeweils bestehen einige, bis realtiv viele Analysen, in welchen der Operator kaum aktiv war. In jeweils drei bis vier Analysen, ist mehr Aktivität über längere Zeit zu sehen. Es kann also nicht definitiv gesagt werden, dass die Opfer-Maschinen bereits nach ein- oder zweimaligem Erkunden uninteressant geworden sind. Ebenso lässt sich bei den Operatoren kein wirklich systematisches Verhalten feststellen. Beim RAT-Sample 1 könnte man interpretieren, dass bei den ersten fünf Analysen lediglich die Basis-Informationsbeschaffung im Vordergrund stand und dann in den drei langzeitig-aktiven Analysen, die tatsächliche Infiltration oder der Daten-Klau stattfand.

RAT-Sample 1

Zeitintervall:	02.05 bis 25.05
Analyse-Dauer [min:sec]	Anz. Befehle
00:15	3
00:15	3
00:19	3
00:27	8
00:09	3
01:47	27
00:09	4
01:27	16
00:22	3
34:17	151
38:55	25
00:26	7
00:22	6
52:56	24
Total: 02:12:06	

RAT-Sample 2

Zeitintervall:	02.05 bis 28.05
Analyse-Dauer [min:sec]	Anz. Befehle
09:04	12
00:04	3
09:04	12
00:04	3
00:28	11
02:09	12
00:19	8
00:08	8
00:10	8
00:08	10
10:58	16
30:55	20
Total: 01:03:31	

RAT-Sample 3

Zeitintervall:	17.05 bis 26.05
Analyse-Dauer [min:sec]	Anz. Befehle
00:00	2
00:00	2
10:38	18
16:57	42
00:00	2
12:48	26
09:48	28
Total: 00:50:11	

RAT-Sample 4

Zeitintervall:	30.04 bis 16.05
Analyse-Dauer [min:sec]	Anz. Befehle
16:07	14
21:39	12
00:01	3
07:03	10
23:17	14
00:22	3
Total: 01:08:29	

RAT-Sample 5

Zeitintervall:	30.04 bis 10.05
Analyse-Dauer [min:sec]	Anz. Befehle
05:18	79
01:47	7
09:29	5
09:39	26
00:21	3
00:18	3
Total: 00:26:52	

Table 23: Operator-Verhalten über Zeit

6 Discussion

The summary and discussion of the important results.

6.1 Reverse-Engineering

During this part of our thesis, we extracted the configurations of 746 samples. We noticed that 94.1% of the operators did not set a password on their samples. In 62.12% of the cases, samples had the DarkComet's default port (1604) in use. Moreover, common ports are 81, 35000, 1605, 80, and some times simple port patterns like 1111 or 4444 have been used. The most used version of DarkComet in the wild is for sure v5.3.0, as since 2012 no more official versions are available. Finally, 95.71% of the samples had the offline keylogger active.

These information are quite important to us, so we understand now what kind of operators are still present in the wild, and this was one of the main goals of this work. Somehow, we already forecasted some of those results such like the most used network port or that the majority of operatord did not set a password.

A big question that remains open in this work is, if we would have had the possibility to use the VirusTotal hunting version, would the results be the same, or at least similar? Probable we will never know, but it could be a good starting point for a new work in the future.

6.2 Scanning

The operators we analysed come from 80 different countries. We were surprised to see, as mentioned in the point 5.2, that the countries with the most operators found are not those that are actually the most active during our scans. Moreover, it is interesting to see that whatever the day or the hour, the number of active operators does not vary so much.

In this work, 16 operators discovered in the samples, were also found by Shodan. So Shodan discovered 32 percent of the active operators. All active operators having used default passwords, Shodan would have been able to recognize them all with the current Shodan script.

The question arises about the usefulness of own scans to discover new operators. Shodan provided them all, sooner or later. It would be interesting to see if a high speed scan brings anything. Would the traffic generated by the scans really be offset by the results discovered?

It could also be considered to keep scans only for specific networks, such as OVH where there are many active operators on a limited number of addresses.

When we see the number of different ports, which we obtained through the samples and Shodan, it is obvious that it is not possible to scan all of them. An operator who chooses a good port and password can therefore remain invisible.

6.3 Live Analysis

During our experiment, we asserted that with 73% of all analyses, the area where most of the operators appear is area 1. in figure 30. Accordingly, the most operators act in a 20 minutes time-frame, using up to 30 commands. However, it was important that we designed our analysis for 60 minutes. So, we achieved also the recording of the interesting 14% of the analyses, with long-time action.

As seen in figure 34, only 2% of the Operators act passive-only. 23% explore the victim and its data with active commands. The 75% of all the operators act aggressive and make themselves noticeable to the victim. We expected a more covered behaviour from the majority.

Showed in figure 32, the most used initial and second most used final command is password-theft. Many operators just came for that single action. Showed in figure 31, with 16% remote desktop forms the most occurred action. So we can confirm, that the RAT DarkComet is primary used for remote desktop connection.

We can also confirm, that this RAT is primary used by operators who like to spy on the victims. Showed in figure 35, with 81% of all analyses, they form the biggest category over all.

We showed in figure 37, that the influence by the operators on the statistics, differs immensely. Many operators do not want to connect more than one time, while others do this up to 14 times. The more analyses are included to the evaluation, the more irrelevant this factor becomes.

The design of our analysis environment, implemented with the Cuckoo Sandbox, worked fine and is well-proven.

7 Conclusion

The DarkComet Tracker is a project to catch RAT users, the operators. During our Bachelor Thesis, we learned much about this remote access trojan, the analysis environment Cuckoo Sandbox and diverse scanning tools. The tracker worked as expected. However, there is still much room for improvement. For istance, one of the things we thought that was nice to implement, was a database. This idea came to avoid the use of CSV based strcture, but then we had to choose to implement it or to pass from computer based environment to server based environment, and the choiche was clear. The biggest part of the project was to build the basis of this tracker and then automating it. Reaching this point of the project and later analysing our results showed us that it still has a lot of potential.

For more accurate statistical results, this project should be done on a larger scale. Therefore more physical power and capacity would be needed to do future analyses. Much more RAT samples of quality would be needed, like those provided by VirusTotal hunting. Also more virtual machines would be needed, which could run much longer for each online analysis. As an extra, for the interesting operators, there is a long-term analysis running for hours up to weeks. The virtual machines, our victims, could be even more realistic. In the future they should act like a real victim during an analysis and could perform human like actions. Additionally, the webcam problem should be reviewed. So, in the future a full functional webcam should be supported by every victim.

If we keep scanning to find new operators, see chapter 6.2, a better connection would certainly be necessary. The script could also be improved by parallelizing Masscan and Nmap scans. This implies that Masscan sometimes has to wait to give Nmap time to follow.

8 Appendix

- **Appendix A - DarkComet Documenation**
  Documentation of the DarkComet commands and functionalities
- **Appendix B - Breakdown of the Work**
  Explanation who wrote which part of the work.

Glossary

Banner - Banner - Bannière Information about the running service on a given port. 89

Bot A software application that runs automated tasks. 89

Capture - Erfassung - Captage Capture of data e.g.. 89

Command - Befehl - Commande . 89

Data-Theft - Datenklau - Vol de Données . 89

Encoding - Codierung - Codage Encoding transforms data into another format using a scheme that is publicly available so that it can easily be reversed. 89

Encryption - Verschlüsselung - Cryptage Encryption transforms data into another format in such a way that only specific individual(s) can reverse the transformation. 89

Guest - Gast - Client Computer system or software that is using a host-provided service. 89

Host Computer system that is providing some service. 89

I/O-Control - E/A-Kontrolle - Contrôle-I/O Control over mouse and keyboard. 89

KeyLogger A method of capturing and recording computer users' keystrokes. 89

List - Auflistung - Listage . 89

Network Traffic - Netzwerkverkehr - Traffic Réseau Data that is transmitted over the Network. 89

Operator - Operator - Opérateur Person who controls the victim's computer remotely. 89

Password - Passwort - Mot de Passe . 89

RAT Remote Access Trojan, virus that gives the possibility to control the victim's computer remotely. 89

Remote - Fern - Lointain Remote access e.g.. 89

Sample A piece of code - in our context a alleged piece of malware. 89

Acronyms

API Application Programming Interface. 14, 89

ASN Autonomous System Number. 39, 83, 89

BFH Berner Fachhochschule. 35, 89

CPU Central Processing Unit. 16, 89

CSV Comma-separated Values. 29, 89

CV Curriculum Vitae. 25, 89

DDoS Distributed Denial of Service. 69, 89

DHCP Dynamic Host Configuration Protocol. 36, 89

DNS Domain Name System. 5, 89

DoS Denial of Service. 3, 89

FAI Fournisseur d'accès à Internet. 10, 77, 82, 89

GUI Graphical User Interface. 12, 89

HDD Hard Disk Drive. 89

HP Hewlett-Packard. 26, 89

HTTP Hypertext Transfer Protocol. 14, 89

HTTPS Hypertext Transfer Protocol Secure. 89

I/O Input & Output. 89

IDS Intrusion Detection System. 5, 89

IOC Indicator of Compromise. 5, 89

IP Internet Protocol. 2, 89

ISP Internet Service Provider. 21, 31, 89

IT Information Technology. 5, 89

JSON JavaScript Object Notation. 31, 89

MAC Media Access Control. 26, 89

NAT Network Address Translation. 44, 89

NSE Nmap Scripting Engine. 50, 89

PCAP Packet CAPture. 31, 89

RAM Random Access Memory. 13, 89

RAT Remote Access Trojan / Tool. 1, 4, 89

RC4 Rivest Cipher 4. 8, 89

REST REpresentational State Transfer. 24, 89

RIPE Réseaux IP Européens. 31, 89

RISIS Research Institute for Security in the Information Society. 36, 89

SSD Solid State Drive. 23, 42, 89

TCP Transmission Control Protocol. 11, 89

UDP User Datagram Protocol. 19, 89

URL Uniform Resource Locator. 14, 89

UTC Coordinated Universal Time. 50, 89

VM Virtual Machine. 9, 27, 89

VPN Virtual Private Network. 35, 89

WEP Wired Equivalent Privacy. 8, 89

References

[1] Fred Cohen. "Computer Viruses". PhD thesis. University of Southern California, 1986. URL: http://www.profsandhu.com/cs5323_s17/cohen-1987.pdf.

[2] *Operating system usage rate from 2015 to 2018.* URL: http://gs.statcounter.com/os-market-share/desktop/worldwide/#yearly-2015-2018.

[3] Brown Farinholt et al. "To Catch a Ratter: Monitoring the Behavior of Amateur DarkComet RAT Operators in the Wild". In: May 2017. URL: http://ieeexplore.ieee.org/document/7958609/.

[4] *K. Breen. Rat decoders.* URL: https://github.com/kevthehermit/RATDecoders.

[5] *Sunrise AG.* URL: https://www.sunrise.ch.

[6] *Swisscom AG.* URL: https://www.swisscom.ch.

[7] *Virustotal.* URL: https://www.virustotal.com.

[8] *Behind the Syrian conflict's digital front lines.* URL: https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-behind-the-syria-conflict.pdf.

[9] *How the Boy Next Door Accidentally Built a Syrian Spy Tool.* URL: https://www.wired.com/2012/07/dark-comet-syrian-spy-tool/.

[10] *Indicator of Compromise (IOC) definition.* URL: https://searchsecurity.techtarget.com/definition/Indicators-of-Compromise-IOC.

[11] *Pyramid of Pain for IOCs.* URL: http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html.

[12] *UPX Packer.* URL: https://upx.github.io/.

[13] *MPRESS Packer.* URL: http://www.matcode.com/mpress.htm.

[14] *RC4 (Rivest Cipher 4).* URL: https://en.wikipedia.org/wiki/RC4.

[15] *WEP (Wired Equivalent Privacy).* URL: https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy.

[16] Nikita Borisov, Ian Goldberg, and David Wagner. "Intercepting Mobile Communications: The Insecurity of 802.11". In: MobiCom '01. 2001. URL: http://doi.acm.org/10.1145/381677.381695.

[17] *Wireshark Network Sniffer.* URL: https://www.wireshark.org/.

[18] *Tcpdump Network Sniffer.* URL: https://www.tcpdump.org/.

[19] *TCP Three Way Handshake.* URL: https://www.geeksforgeeks.org/computer-network-tcp-3-way-handshake-process/.

[20] *Volatility.* URL: http://www.volatilityfoundation.org.

[21] *Winpmem.* URL: https://github.com/google/rekall/tree/master/tools/windows/winpmem.

[22] *Dumpit.* URL: https://my.comae.io/login.

[23] *Regular Expression (regex)*. URL: https://en.wikipedia.org/wiki/Regular_expression.

[24] *YARA: The pattern matching swiss knife for malware researchers*. URL: http://virustotal.github.io/yara/.

[25] *Cuckoo Sandbox*. URL: https://cuckoosandbox.org.

[26] *Hypervisor – Virtual Machine Monitor*. URL: https://www.searchdatacenter.de/definition/Hypervisor-Virtual-Machine-Monitor-VMM.

[27] *A Virtual Honeypot Framework*. URL: http://static.usenix.org/event/sec04/tech/full_papers/provos/provos_html/.

[28] *Nmap Documentation*. URL: https://nmap.org/docs.html.

[29] *Nmap Book*. URL: https://nmap.org/book/.

[30] *Nmap Scripts*. URL: https://nmap.org/nsedoc/index.html.

[31] *Masscan GitHub*. URL: https://github.com/robertdavidgraham/masscan.

[32] *Zmap*. URL: https://zmap.io/.

[33] *Zmap Research*. URL: https://zmap.io/research.

[34] *Shodan*. URL: https://www.shodan.io/.

[35] John Matherly. *Academic Upgrade*. Email. Nov. 30, 2017. URL: jmath@shodan.io.

[36] Zakir Durumeric et al. "A Search Engine Backed by Internet-Wide Scanning". In: *22nd ACM Conference on Computer and Communications Security*. Oct. 2015.

[37] *Shodan and Censys: the ominous guides through the Internet of Things*. URL: https://www.kaspersky.com/blog/shodan-censys/11430/.

[38] *Research Access to Censys Data*. URL: https://support.censys.io/getting-started/research-access-to-censys-data.

[39] *BigQuery Pricing*. URL: https://cloud.google.com/bigquery/pricing.

[40] *BigQuery Introduction — Censys Help Center*. URL: https://support.censys.io/google-bigquery/bigquery-introduction.

[41] *Looking at the Sky for a DarkComet*. URL: https://www.fidelissecurity.com/sites/default/files/FTA_1018_looking_at_the_sky_for_a_dark_comet.pdf.

[42] *Who's Scanning Your Network? (A: Everyone)*. URL: https://krebsonsecurity.com/2015/05/whos-scanning-your-network-a-everyone/#more-30752.

[43] *ZMap: Fast Internet-Wide Scanning and its Security Applications*. URL: https://zmap.io/paper.pdf.

[44] Swisscom (Schweiz) AG Mario Micheletti Group Security. *AW: Benutzung DSL fuer Security Scans durch Berner Fachhochschule*. Email. Apr. 26, 2018. URL: abuse@swisscom.com.

[45] *Swiss Criminal Code 143bis1*. URL: https://www.admin.ch/opc/en/classified-compilation/19370083/index.html#a143bis.

[46] *Cuckoo Documentation*. URL: https://cuckoo.sh/docs/index.html.

[47] *Hardening Windows7 x64 on VirtualBox for Malware Analysis*. URL: http://byte-atlas.blogspot.com/2017/02/hardening-vbox-win7x64.html.

[48] *Pafish*. URL: https://github.com/a0rtega/pafish.

[49] *Tshark: Wireshark dumper and analyser*. URL: https://www.wireshark.org/docs/man-pages/tshark.html.

[50] *ipinfo.io : IP Address API and Data Solutions*. URL: https://ipinfo.io/.

[51] *RIPE Network Coordination Centre*. URL: https://www.ripe.net/.

[52] *JS sequence diagrams: Turns text into UML sequence diagrams*. URL: https://bramp.github.io/js-sequence-diagrams/.

[53] *Wix*. URL: https://fr.wix.com/.

[54] *no-ip*. URL: https://www.noip.com/.

[55] *TeamViewer*. URL: https://www.teamviewer.com.

[56] *ExpressVPN*. URL: https://www.expressvpn.com/.

[57] *How to Do IP Address Geolocation Lookups on Linux*. URL: https://www.maketecheasier.com/ip-address-geolocation-lookups-linux/.

[58] *GeoLite2 Free Downloadable Databases*. URL: https://dev.maxmind.com/geoip/geoip2/geolite2/.

[59] *U.S. based Cloud Hosting providers contribute 44% of Malware distribution*. URL: https://thehackernews.com/2014/01/us-based-cloud-hosting-providers.html.

[60] *Nmap File Banner*. URL: https://nmap.org/nsedoc/scripts/banner.html.

[61] *Firewall/IDS Evasion and Spoofing*. URL: https://nmap.org/book/man-bypass-firewalls-ids.html.

[62] *Supervisord*. URL: http://supervisord.org/.

[63] *Base script of our darkcomeconfigdump.py, by dfirn00b on github*. URL: https://github.com/dfirn00b/volatility_plugins.

[64] *List of countries by IPv4 address allocation*. URL: https://en.wikipedia.org/wiki/List_of_countries_by_IPv4_address_allocation.

[65] *Cuckoo Installation*. URL: https://cuckoo.sh/docs/installation/index.html.