

**AUTUMN TERM 2022**

# **LAB Assignment 1**

**Group 14**

**Jesper Blomqvist,  
jebi6563 Lekhaz  
Adapa, lead3201  
Puja Poudel,  
pupo3339**

## **Table of contents**

[Jesper Blomqvist, jebi6563 Lekhaz Adapa, lead3201 Puja Poudel, pupo3339](#)

[Exercise 1 Code Injection](#)

[1.1 File properties](#)

[1.1.1 General tab](#)

[1.1.2 Compatibility tab](#)

[1.1.3 Security tab](#)

[1.1.4 Details tab](#)

[Previous versions tab](#)

[1.2 Double click injected\\_calc.EXE](#)

[1.3 Checksum hash tools](#)

[1.3.1 Microsoft File Checksum Integrity Verifier \(fciv\)](#)

[1.3.2 Certutil.exe](#)

[1.3.3 Windows Powershell](#)

[1.3.4 Questions exercise 1](#)

[1.3.4. What implications does this have for security?](#)

[1.3.4.2 In what circumstances might an attacker \(or malicious software\) modify an executable as part of an attack, or utilize modified executables in an attack?](#)

[1.3.4.3 What could an executable be modified to do?](#)

[1.3.4.4 How might one be exposed to such threats, and how can one protect oneself from them?](#)

## [Exercise 2. Windows Registry](#)

[2.1 Registry Exercise A](#)

[2.2 Registry Exercise B](#)

[2.3 Registry Exercise C](#)

## [Exercise 3. Spoofing - Bypassing the Login Screen](#)

## [Exercise 4. Monitoring Keyboard Strokes](#)

## [Exercise 5. Disclosing masked passwords](#)

## [Exercise 6. NetBus - Take control over another computer](#)

## [Part 2. Laboratory Assignments for Linux](#)

### [Exercise 1. Utilising port and vulnerability scanners](#)

#### [1.1 Windows Machine](#)

[1.1.1 Quick scan:](#)

[1.1.2 Intense scan](#)

[1.1.3 Intense + UDP Scan](#)

[1.1.4 NC.EXE Backdoor](#)

#### [1.3 Linux\\_Ubuntu Machine:](#)

[1.3 Firewall](#)

### [Exercise 2 Sniffing Tools](#)

[2.1 Greenbone Vulnerability Management \(GVM\) \[Formerly OpenVAS\]](#)

[2.2 URLSNARF](#)

[2.3 WebspY](#)

[2.4 dsniff](#)

[2.4.1 FTP](#)

[2.4.2 Telnet](#)

[Dsniff registers not only telnet commands. All inputs, like for example "list" are registered. 2.4.3 MITM](#)

[2.4.4 Ettercap](#)

### [Exercise 3. Analysing network traffic](#)

[3.1 tcpdump](#)

[3.2 Wireshark](#)

[3.3 SNORT](#)

### [Exercise 4 Metasploit](#)

# Part 1. Laboratory Assignments for Windows

## Exercise 1 Code Injection

### 1.1 File properties

Two files, `cal.exe` and `injected_calc.exe` were compared by examining their respective file properties.

#### 1.1.1 General tab

Different values found for *size:* and *size on disk:*

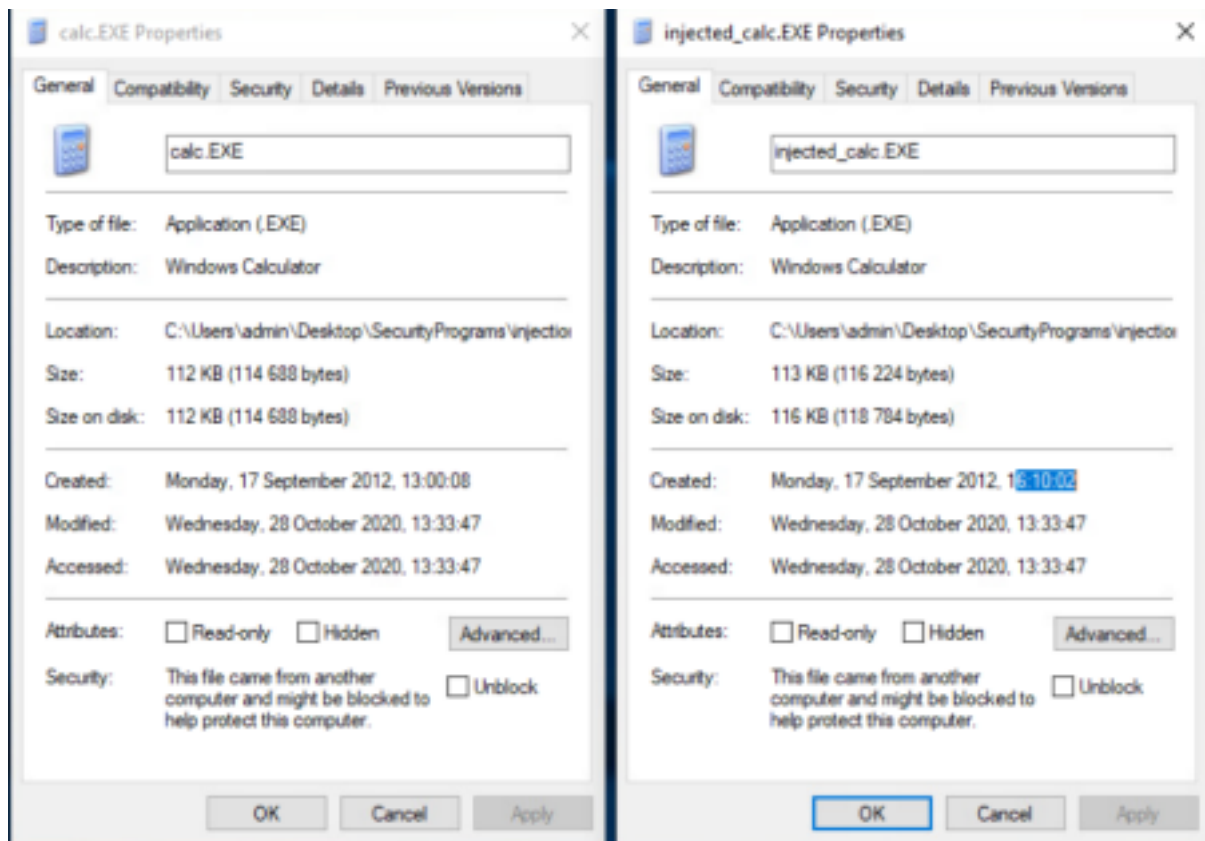
Here we can see that the compiled sizes differ. This is a good indicator that the files are **not** identical, However, we cannot see in which way they differ.

The two files have different timestamps found on *Created:*

*Modified* and *Accessed* dates and the other timestamps are identical.

This could be a flag indicating something may not be as it appears.

There may be legitimate reasons for different timestamps, and it does not look like changing something superficial like the file name affects the time stamps on the general tab. We cannot explain why it is the timestamp on *Created* rather than *Modified* that differs unless someone spoofed that date/timestamp. But if they are able to do that, why leave one timestamp out?



Of interest:

We noticed that there was a message under the general tab that read “Security: This file came from another computer and might be blocked to help protect the computer.”

This indicates that the OS does security checks on files introduced into the OS file system.

### 1.1.2 Compatibility tab

No discernible difference.

### 1.1.3 Security tab

No discernible differences

Both files have the same User/Group Permissions and Owner.

Perhaps a flag could be if one file has a higher level permission and/or Owner than the other.

### 1.1.4 Details tab

Again, a size difference, 112kb vs 113kb, was detected

Here we can see that the compiled sizes differ. This is a good indicator that the files are **not** identical, However, we cannot see in which way they differ.

The files could have different versions but still be legit. If they have the same version, at least one of the file is likely not what it appears to be. Given the information about alleged identical file versions under the details tab and the different timestamp under the general tab we may suspect that one (or both) files are not what they appear to be.

## Previous versions tab

No discernible differences

## 1.2 Double click injected\_calc.EXE

When running the file injected\_calc.exe, a pop up window appears:



When clicking OK the “normal” calculator GUI appears.

Other than that, both calculators behave identically as far as we can tell. Basic operations (+, -, /, \*) were tested in standard layout and then again after switching to scientific layout where we additionally checked n!, x^y, x^2, x^3 and 1/x.

## 1.3 Checksum hash tools

### 1.3.1 Microsoft File Checksum Integrity Verifier (fciv)

First, the files were checked with fciv which produced two different MD5

hashes:

```
Select Command Prompt

C:\Users\admin\Desktop\SecurityPrograms\fciv>fciv c:\Users\admin\Desktop\
SecurityPrograms\injection\injected_calc.EXE
//
// File Checksum Integrity Verifier version 2.05.
//
e1fe13a125bdcf9d2bc8651d93e9bdf8 c:\users\admin\desktop\securityprograms\
injection\injected_calc.exe

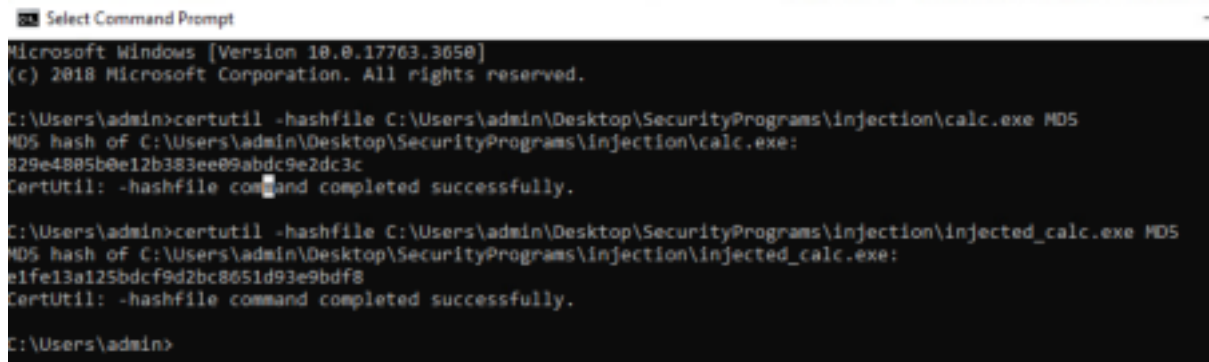
C:\Users\admin\Desktop\SecurityPrograms\fciv>fciv c:\Users\admin\Desktop\
SecurityPrograms\injection\calc.EXE
//
// File Checksum Integrity Verifier version 2.05.
//
829e4805b0e12b383ee09abdc9e2dc3c c:\users\admin\desktop\securityprograms\
injection\calc.exe

C:\Users\admin\Desktop\SecurityPrograms\fciv>
```

### 1.3.2 Certutil.exe

A new command prompt was opened and the following commands were used: certutil  
-hashfile C:\Users\admin\Desktop\SecurityPrograms\injection\calc.exe MD5 certutil  
-hashfile  
C:\Users\admin\Desktop\SecurityPrograms\injection\injected\_calc.exe MD5

We got the following results:



```
Select Command Prompt
Microsoft Windows [Version 10.0.17763.3650]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\admin>certutil -hashfile C:\Users\admin\Desktop\SecurityPrograms\injection\calc.exe MD5
MD5 hash of C:\Users\admin\Desktop\SecurityPrograms\injection\calc.exe:
829E4885B0E12B383EE09ABDC9E2DC3C
CertUtil: -hashfile command completed successfully.

C:\Users\admin>certutil -hashfile C:\Users\admin\Desktop\SecurityPrograms\injection\injected_calc.exe MD5
MD5 hash of C:\Users\admin\Desktop\SecurityPrograms\injection\injected_calc.exe:
E1FE13A125BDCF9D2BC8651D93E9BDF8
CertUtil: -hashfile command completed successfully.

C:\Users\admin>
```

cs2

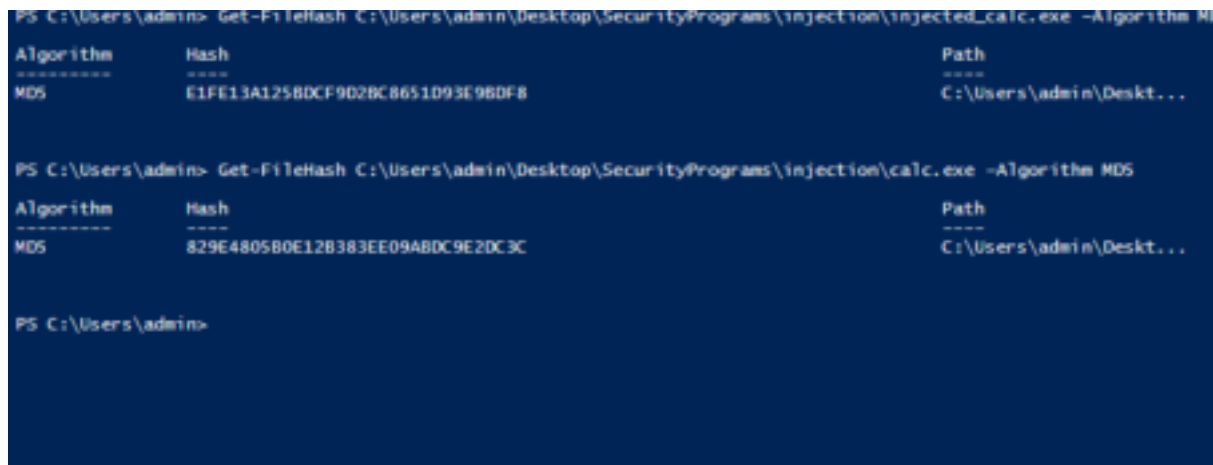
The same MD 5 hashes as in the previous step were produced for each file.

### 1.3.3 Windows Powershell

Finally, we used MS PowerShell and used the following commands:

```
Get-FileHash
C:\Users\admin\Desktop\SecurityPrograms\injection\injected_calc.exe -Algorithm
MD5
Get-FileHash C:\Users\admin\Desktop\SecurityPrograms\injection\calc.exe -Algorithm
MD5
```

The results were as follows:



```
PS C:\Users\admin> Get-FileHash C:\Users\admin\Desktop\SecurityPrograms\injection\injected_calc.exe -Algorithm MD5

Algorithm      Hash                                     Path
-----
MD5             E1FE13A125BDCF9D2BC8651D93E9BDF8      C:\Users\admin\Desktop\...

PS C:\Users\admin> Get-FileHash C:\Users\admin\Desktop\SecurityPrograms\injection\calc.exe -Algorithm MD5

Algorithm      Hash                                     Path
-----
MD5             829E4805B0E12B383EE09ABDC9E2DC3C      C:\Users\admin\Desktop\...

PS C:\Users\admin>
```

Again, MD5 hash checksums were produced for each file and they matched the previous steps. This time they were presented in CAPS, however.

### 1.3.4 Questions exercise 1

#### 1.3.4. What implications does this have for security?

If a program does not behave as expected it cannot be trusted, and renders the whole system unreliable and at risk of further exploits. App developers often provide proof of authenticity in the form of MD5 checksums published alongside download links for the app. Of course, this does not provide a 100% failproof solution as a more elaborate attack can trick users into downloading an altered version of the program together with a checksum that matches the malicious file provided by the attacker.

#### 1.3.4.2 In what circumstances might an attacker (or malicious software) modify an executable as part of an attack, or utilize modified executables in an attack?

One way of describing such an attack could be: a) Getting the file onto the intended target b) ascertain the right security privileges and c) execute the file to achieve the intended effect. If the user is tricked into transferring the file to the target system it can then often also be tricked into executing it with admin privileges since windows, even though it has user access control, often lets users.

For businesses, the IT department often has various policies in place to restrict both access and privileges to computers, services, files and other shared network devices. These policies are enforced using management tools, which in turn can have vulnerabilities and are targets as well.

Many files, especially Windows OS files are seldom updated manually and/or may not have published checksum or other means of proof of authenticity. This means that if a system vulnerability is exploited, it can lead to multiple files being “updated” to malicious versions without the user becoming aware of it.

#### 1.3.4.3 What could an executable be modified to do?

As far as we understand it, pretty much anything. If given the right privileges. It could change the behavior of the original program to collect and disseminate personal information without the user being aware. It could delete or encrypt files, or otherwise make important information inaccessible and cause loss of time, money and information for the individual or whole organizations if the infected machine is part of a larger network with shared authentication services.

#### 1.3.4.4 How might one be exposed to such threats, and how can one protect oneself from them?

This question is similar to 1.3.4.2, but a few examples are:

1. A user being actively tricked into downloading or copying a tampered version of the file from a physical device onto the system.  
This could be achieved via fake emails, by users deliberately using pirated programs or media files, since executables are often altered to bypass digital rights management tools used by the IP owner to limit use of their software to verified owners and may therefore be expected to not be identical to the original files and checksums
2. Various exploits leading to backdoors being used by attackers to insert or replace existing files that then are executed without checking their authenticity.
- 3.

Ways to protect the system is to keep it updated and never install anything from untrusted sources. Even double checking “trusted” sources may be a good idea. One could also use scanning tools and services from trusted providers to continually check for vulnerabilities and security risks.

Restricting users’ privileges during everyday tasks may also be an option to prevent consequences of accidental execution of malicious software.

Another measure to be taken is to switch to less widely used platforms/OS:s since there is less general gain from infecting those if the purpose of the attack is to reach as many potential victims as possible (which may not be the case if the attack is targeted at a specific device/person/organization and therefore more bespoke methods are used, like social engineering or direct physical access).

## **Exercise 2. Windows Registry**

### **2.1 Registry Exercise A**

Changed the following registry entry to “1”:

Logged out and in to observe the change and then reverted it back to “0”.

### **2.2 Registry Exercise B**

Tested this with both values 4 and 12.

Both c and d drives disappeared

### **2.3 Registry Exercise C**

Added a DWORD “NoRun” with value “1”

Logged in and out and got the following message when trying to use “Run”





app:

Restored it by simply running the registry editor directly (easiest way to access is opening start menu - then type “reg” and pick the suggested app. Could have accessed it in the C:\windows\system32\ folder or via command prompt too.

## Exercise 3. Spoofing - Bypassing the Login Screen

After transferring ownership from TrustedInstaller to Admin and changing the Admin permissions to “Full Control” we were able to rename the sethc.exe to sethc\_old.exe. Then we made a copy of cmd.exe and renamed that to sethc.exe. By doing so we could trigger the cmd.exe from the login screen.

We subsequently restored the files and permissions by following the same steps in reverse. One peculiar artefact of this procedure was that the restored sethc.exe still had the icon for cmd.exe. Revisiting the file a few days later, the original file icon was once again present, however.

**Note:** We somehow lost the ability to assign read/write privileges from user Admin during the process, so we changed ownership to TrustedInstaller and assigned Read & Execute + Read.

We can perform any operation and we can be any part of any role if we have the same privileges which are assigned to the admin.

The security level must be increased so that OS can find the one who has right to access it and even if the user has the role of the admin the OS should crosscheck who's having the control by implementing the UAC (User Account Control) so that the admin role might not get misused by chance of anyone.

Accessing a cmd command line with admin permissions gives an attacker access a plethora of functions and can be used to replace, delete or introduce files for example.

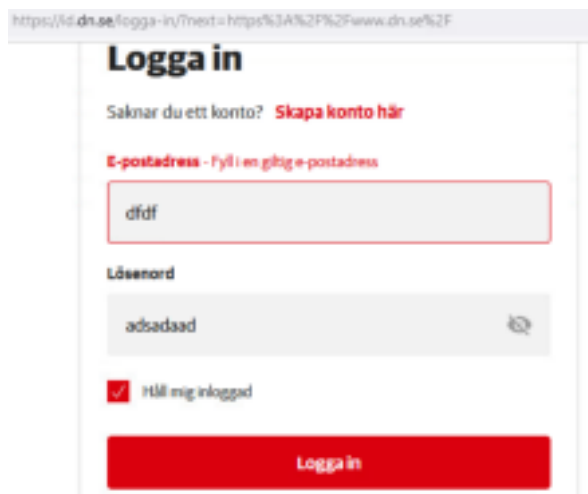
## Exercise 4. Monitoring Keyboard Strokes

We followed the instructions given, the klogger.txt contains all keystrokes that we entered. The keyboard strokes were logged more than once due to multiple instances of klogger.exe since there is no visual feedback given when activating the executable. We had 3 instances of klogger.exe running.

## Exercise 5. Disclosing masked passwords

We followed the instructions given, the result says that Google Chrome has access to unmask the password by pressing and holding the Cntrl key while in the Firefox browser nothing happens when we push and hold the Cntrl button because there is no preinstalled

extension in the Firefox browser. However, some browsers/sites provide this functionality as a feature :



https://id.dn.se/login?next=https%3A%2F%2Fwww.dn.se%2F

## Logga in

Saknar du ett konto? [Skapa konto här](#)

E-postadress - Fyll i en giltig e-postadress

Lösenord

☒ Håll mig inloggad

Logga in

You can also access and view stored passwords in the browser by similar functionality.

## Exercise 6. NetBus - Take control over another computer

We installed Netbus with the Patch.exe on our machine.

We then ran Netbus.exe on our own machine and observed the results of some of the functions. We did not find a group to collaborate with, but in this case it was not necessary since we could install Netbus and run the client on the same machine. This could have been done with two Windows VMs as well.

## Part 2. Laboratory Assignments for Linux

### Exercise 1. Utilising port and vulnerability scanners

We ran 3 scans on different levels for Windows and Linux (Ubuntu) Virtual Machines: Ping Scan, Quick Scan and Intense Scan.

For the Windows machine we also ran an Intense scan with UDP. Finally we scanned a network firewall.

We got the following results:

## 1.1 Windows Machine

### 1.1.1 Quick scan:

#### Port Service

80/TCP http

135/TCP msrpc

139/TCP netbios-ssn

445/TCP microsoft-ds? files/printer sharing srv

+ Mac address

+ 96 closed ports

### 1.1.2 Intense scan

80/TCP http

135/TCP msrpc

139/TCP netbios-ssn

445/TCP microsoft-ds?

12345/TCP NETBUS trojan

+ Mac address

+ 995 closed ports

+ **Info about Netbus installation**

+ Info about OS/NETBIOS/TRACEROUTE

The quick scan did not reveal the netbus trojan on the windows machine, but it appeared on the intensive scan.

### 1.1.3 Intense + UDP Scan

We also tried using the Intense + UDP profile, which gave the same results as Intense for TCP ports, plus 21 UDP ports:

```
137/udp open netbios-ns Microsoft Windows netbios-ns (workgroup: INTROSEC)
| nbns-interfaces:
|   hostname: INTROSEC-WIN14
|   interfaces:
|     10.11.202.78
|_
138/udp open|filtered netbios-dgm
500/udp open|filtered isakmp
1900/udp open|filtered upnp
2002/udp open|filtered globe
3457/udp open|filtered vat-control
3702/udp open|filtered ws-discovery
4500/udp open|filtered nat-t-ike
5050/udp open|filtered mmcc
5353/udp open|filtered zeroconf
5355/udp open|filtered llmnr
16573/udp open|filtered unknown
16700/udp open|filtered unknown
19687/udp open|filtered unknown
21649/udp open|filtered unknown
22739/udp open|filtered unknown
39632/udp open|filtered unknown
49161/udp open|filtered unknown
49195/udp open|filtered unknown
51905/udp open|filtered unknown
```

The OS could not be detected in this scan, but it was detected with the regular intense scan.

### 1.1.4 NC.EXE Backdoor

We started the NC.EXE on the windows machine and ran the intensive scan again and observed that a new port, 100/TCP with the service newacct? had opened We also received information that this service could not be recognized “1 service unrecognized despite returning data[...]

```
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-Port100-TCP:V=7.92%I=7%D=11/25%Time=63809370AP=x86_64-pc-linux-gnu(r(NU
SF:LL,97,"Microsoft\x20Windows\x20\[Version\x2010\,0\,17763\,3650\]\r\n(c
SF:)\x202018\x20Microsoft\x20Corporation\,.\x20All\x20rights\x20reserved\,
SF:r\n\r\nC:\\Users\\admin\\Desktop\\SecurityPrograms\\hxdelf100r>")\r(Gen
SF:erLines,167,"Microsoft\x20Windows\x20\[Version\x2010\,0\,17763\,3650\
SF:)\r\n(c)\x202018\x20Microsoft\x20Corporation\,.\x20All\x20rights\x20re
SF:served\,.\r\n\r\nC:\\Users\\admin\\Desktop\\SecurityPrograms\\hxdelf100r>
SF:r\nC:\\Users\\admin\\Desktop\\SecurityPrograms\\hxdelf100r>\r\nC:\\User
SF:s\\admin\\Desktop\\SecurityPrograms\\hxdelf100r>\r\nC:\\Users\\admin\\De
SF:sktop\\SecurityPrograms\\hxdelf100r>\r\nC:\\Users\\admin\\Desktop\\Secur
SF:ityPrograms\\hxdelf100r>");
```

## 1.3 Linux\_Ubuntu Machine:

### Quick scan

- 0 ports
- 100 closed

### Intense sca

- 0 ports
- 1000 closed (all scanned ports are in ignored states).
- No OS info
- TRACEROUTE info

## 1.3 Firewall

### Quick scan

- 53/tcp domain
- 80/tcp http
- 443/tcp https

97 filtered ports

### Intense scan:

- 53/tcp domain
- 80/tcp http
- 443/tcp http
- 666/tcp darkstat network analyzer httpd 3.0.719
- + Info about OS (guess)
- 996 ports filtered

## Exercise 2 Sniffing Tools

### 2.1 Greenbone Vulnerability Management (GVM) [Formerly OpenVAS]

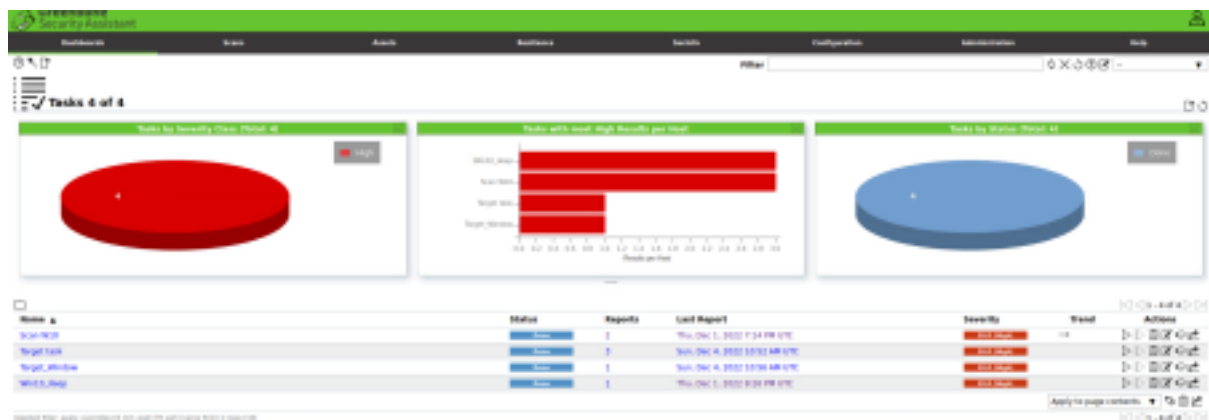
We used this Greenbone Vulnerability Management tool in order to test how secure the system is. We first opened the pre-installed GVM scanning server on our Kali VM.

We began by setting up the configuration that included updating databases with the latest vulnerability information with the command `sudo gvm-setup`.

The `sudo gvm-check-setup` checked if everything was correctly configured or not. After running this command we got that the GVM installation was ok.

And, finally the GVM was run with the command `sudo gvm-start`.

We also checked the vulnerability scanning in our windows and Kali VMs. We created new tasks for each VM and under the host section manual, we assigned the ip address for each of our VMs and scanned the vulnerability of each VMs.



Looking at the reports obtained from scan and configuration, we found that a generic check for HTTP directory traversal vulnerabilities had the highest base vulnerability scoring. The quality of detection (QoD) i.e. the remote vulnerability was also highest (99%). The impact created by this vulnerability would allow attackers to access paths and directories that should normally not be accessible by a user. This could result in disclosure of confidential information.

Observing the reports for each targets, we found that the windows scan had its operating system Microsoft Windows, however in the other VM tasks no operating system was shown. We also tried scanning with different targets, for example creating a task for kali VM and setting target scan for windows and observing the results.

Also, the instructions are confusing. They read: “Note: webspy seems **not** to have stopped working with modern browsers, however it is a plus if you do manage to get it working, or if you find an equivalent tool that provides similar functionality”

We assume it was meant to be "Note: webspy seems to have stopped working with modern browsers [...]"

## 2.4 dsniff

### 2.4.1 FTP

When logging in to ftp on 10.11.202.20 we received the following info, all in plain text:

```
12/01/22 11:42:55 tcp 10.11.202.186.4314 → 10.11.202.20.21 (ftp)
```

The format seems to be mm/dd/yy hh:mm:ss tcp client-IP.port → target-IP.port (ftp)

```
USER:msfadmin
```

```
PASS:msfadmin
```

### 2.4.2 Telnet

```
12/01/22 11:42:55 tcp 10.11.202.186.48136 → 10.11.202.20.23 (telnet)
```

First the user and password are shown:

```
msfadmin
```

```
msfadmin
```

This is followed by a list of commands

```
dir
```

```
list
```

```
ls
```

```
cd vulnerable
```

```
ls
```

```
[...]
```

The client port seems to vary from connection to connection, but the target port is always the same (default) for each protocol. 21 for ftp and 23 for telnet.

**Dsnuff registers not only telnet commands. All inputs, like for example "list"**

**are registered.**

### 2.4.3 MITM

First we had problems running `sudo webmitm`. We got an error saying

```
webmitm: bind address already in use.
```

We then restarted the machine and switched to root access with `sudo su`.

Webmitm started with the following message:

```
webmitm: relaying transparently
```

We created an outlook account on <https://www.outlook.com> and tried using that, but nothing happened. We then tried a github account, but still nothing happened.

Then we started getting the following message:

```
webmitm: No virtual host in request
```

We tried creating a new certificate by renaming the existing webmitm.crt, and we tried specifying the host, to no avail.

We decided to continue with the rest of the assignment and come back to this part at a later stage.

#### **2.4.4 Ettercap**

We began using ettercap by typing the command `sudo ettercap -G`. This lead us to the graphical user interface where we could see other options.

The ettercap showed similar function as that of the dsniff-the man in the middle. We tried using the ftp and telnet with the ettercap but somehow the telnet was not working. In the GUI we could also observe that we had other menus like delete a host, see the host lists, sniff or unsnif it.

## **Exercise 3. Analysing network traffic**

### **3.1 tcpdump**

We used the command `sudo tcpdump -i eth0 -tcpdump_ass1` to write the traffic log into a file.

We tried opening it with text editor but it was not UTF 8 formatted.

We then opened the file in wireshark instead.

### **3.2 Wireshark**

We Started the capture by double clicking the interface line.

Traffic information was captured, e.g source IP, destination IP, ports, protocols used and much more.

### **3.3 SNORT**

We followed the `sudo -vde -i ethN` command and there we observed the packet information in detail.

We were also able to save the information into the snortlab.txt file through the command `sudo snort -vdi ethN > ~/Desktop/snortlab.txt`

When trying to use the more advanced command, with results split into different subdirectories based on IP addresses, we got the following message:

```
WARNING: No preprocessors configured for policy 0
```

We could not find an immediate solution to this.



We then compared the snortlab.txt to the tcpdump\_ass1 file created previously on the Kali VM.

Since we could not read the `tcpdump_ass1` file directly, we had to open it with Wireshark. We therefore feel that we cannot compare the two directly.

Both files contain traffic information including source and destination IP, ports used, protocols used et cetera. However, we do not have experience with either tool to discern major differences.

## Exercise 4 Metasploit

Simple Webserver was verified running on the WinVM and accessible from KaliVM with Firefox.

A Simple Webserver Directory Traversal Vulnerability was detected by scanning our WinVM with GVM from the KaliVM. However, this particular vulnerability was associated with version 2.3 rc1 and not 2.2 rc2.

A second scan, this time set to using a full deep ultimate profile, was conducted.

A metasploit framework was initiated by setting the database with the command “sudo msfdb init” then “sudo armitage”. The armitage command prompted us to its GUI to start “Metasploit’s RPC Server.

In Armitage GUI, after exploring the tree menu list and entering the rhost ip address, we launched the instruction to access the target. We observed the log of the exploit that has been being run. However we got the error message aborting the process saying that the target could not be reached. Because of this the operation was not successful and we could not run further instructions.

