Execution

In the execution phase of an attack, the attacker **runs malicious code on a system to achieve their objectives**, which may include establishing persistence, gaining privileges, or preparing for lateral movement. Execution techniques range from using various shells and interpreters to leveraging scheduled tasks or system management tools like WMI on Windows systems.

1. Shells & Interpreters (e.g., PowerShell, Python, JavaScript)

- Definition: **Attackers use shells and interpreters to execute scripts, commands, and code snippets directly on the target machine**. These tools offer flexibility and control over a compromised system.
- Commonly Used Shells and Interpreters:
  - **PowerShell**: A powerful shell and scripting language on Windows, PowerShell is frequently used in attacks because it allows deep access to system resources, file manipulation, and network communication. Attackers can execute commands stealthily, often avoiding detection.
  - **Python**: A cross-platform scripting language with numerous libraries, Python is used in both Windows and Unix environments to run code that performs tasks such as downloading payloads, controlling devices, or interacting with APIs.
  - **JavaScript**: While JavaScript runs primarily in web browsers, attackers may use it in specific attacks (e.g., cross-site scripting) or in environments where JavaScript engines are present. In some cases, JavaScript is used to execute payloads from malicious websites.
- Security Implications: Shells and interpreters allow attackers to execute arbitrary commands on a system, making it easy to download and run additional malware or to control the system remotely. Since tools like PowerShell are legitimate and often necessary, they can be challenging to block without impacting normal operations.

2. Scheduled Tasks and Windows Management Instrumentation (WMI)

- **Scheduled Tasks**:
  - Definition: Attackers create or manipulate scheduled tasks on the target system to **run malicious code at specific times or intervals**. On Windows, the schtasks command allows users to schedule tasks, which can be set to persist across reboots.
  - Purpose: Scheduled tasks can execute code repeatedly, ensuring that malware runs at designated times or during specific system states, like startup.
  - Example: An attacker may schedule a task to download and execute a payload every time a user logs in or periodically every few hours.
  - Security Implications: Scheduled tasks provide attackers with a way to **automate malicious activity and establish persistence without needing direct interaction**. They can also be set to run under different user privileges, making them versatile for privilege escalation.
- **Windows Management Instrumentation (WMI)**:
  - Definition: WMI is a framework that **allows users to query and control various Windows system components and configurations**. Attackers use WMI to **execute code, gather system information, and manage processes remotely**.
  - Usage in Attacks:
    - **Remote Code Execution**: WMI can be used to execute scripts or commands on remote systems without requiring an interactive shell.

        - **Process Creation**: Attackers use WMI to start processes or services, often to create persistence or run scripts.
    - Security Implications: WMI is often used in legitimate Windows administration, making it challenging to detect malicious use. Attackers often use WMI to evade network detection tools, as it enables remote command execution without traditional remote access methods like SSH or RDP.

Summary

- Shells and Interpreters (e.g., PowerShell, Python) allow attackers to **run flexible and powerful scripts**, often evading detection since these tools are legitimate and widely used.
- Scheduled Tasks enable attackers to **automate malware execution, ensuring persistence across reboots and at designated times**.
- WMI provides attackers with **remote code execution capabilities and control over system processes, offering stealth and flexibility in Windows environments**.

Understanding these execution tactics helps defenders recognize and mitigate the risks associated with shells, interpreters, scheduled tasks, and WMI, which are often used in sophisticated attacks. Monitoring for unusual command use, script execution, and new scheduled tasks can help detect and prevent unauthorized execution activities.