See what it's like to have a partner in the fight.

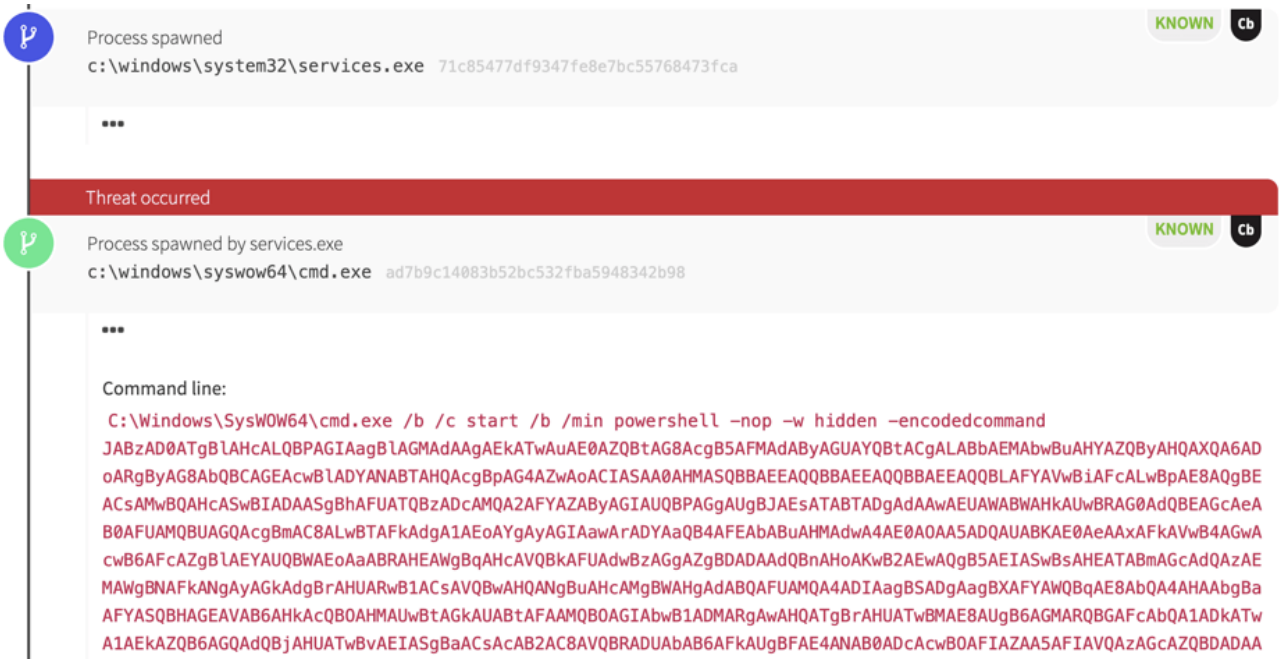
redcanary.com/blog/frameworkpos-and-the-adequate-persistent-threat/

All too often the information security community focuses on a category of adversary known as an Advanced Persistent Threat (APT). Marketing and sales departments all around this industry have focused their messaging on the premise that their product or service will protect you against APT-level adversaries, and this sometimes causes us to lose sight of a simpler truth. Most adversaries do not need to be advanced or sophisticated to execute code or persist in an organization. More often than not, they can simply settle to be an adequate persistent threat, using techniques and artifacts that anyone can find within their organization.

In this threat detection post, we'll look at FrameworkPOS, a point-of-sale malware family that has been tied to an organized APT group in the past. In doing so, we'll show that an adversary doesn't need advanced techniques to execute and persist; they simply need to be good enough.

Point of sale and compromise

Our first indication of trouble for one particular endpoint was the execution of an encoded PowerShell command spawning from the Windows Service Controller, `services.exe`.

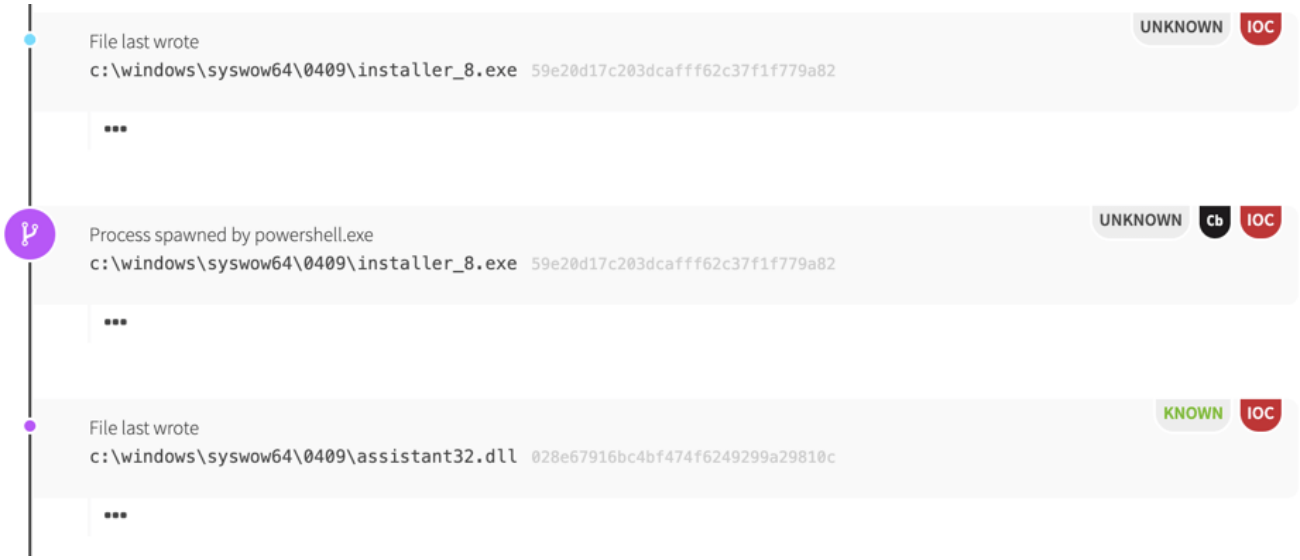


This kind of event typically suggests that there is a Windows Service (T1035, T1050) on the host that will issue this `cmd.exe` and `powershell.exe` command when started. We sometimes see this functionality used legitimately in the maintenance of systems, but those instances almost always use scripts with names and paths. PowerShell is almost always evil when we see it encoded in this context.

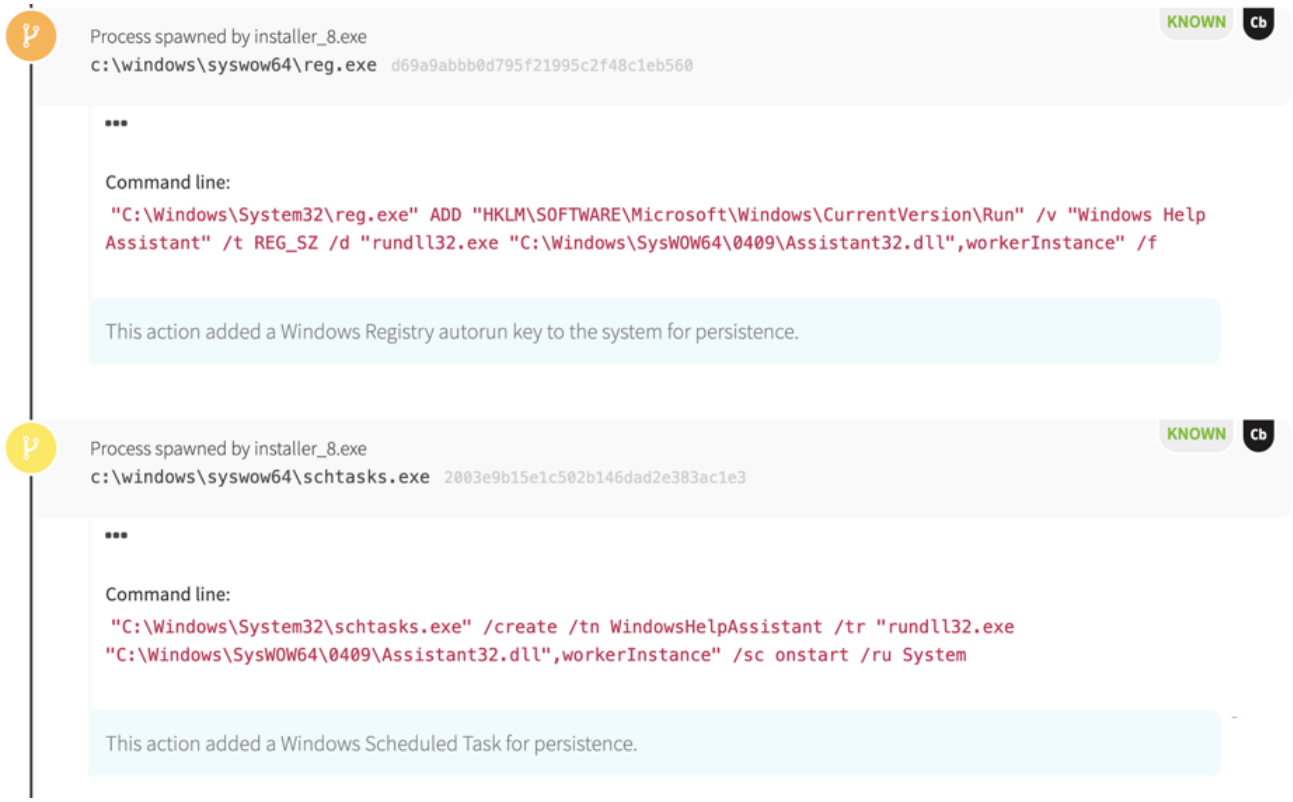
In this case, the malicious PowerShell code deobfuscated partially to reveal some telling strings:

```
$s=New-Object IO.MemoryStream(,[Convert]::FromBase64String("H4sI
```

The base64 encoded PowerShell decoded into the above, showing that a second payload had been compressed using gzip and encoded using base64. We can make this assertion based partially on the FromBase64String function, and partially on the H4sI value at the start of the second payload. Whenever we see this base64 value in the wild, it indicates that there is a gzipped payload within the encoding.



The PowerShell code went on to download and execute `installer_8.exe`, a malicious payload that hadn't been observed by antivirus (AV) vendors at that point. This is an important distinction because it shows how adversaries can evade AV detection just by using tools that are new. In fact, a dynamic-link library (DLL) written by this binary also evaded detection by AV due to a misclassification. AV tools thought the DLL was not malware, which turned out to be false.



`Installer_8.exe` proceeded to establish persistence using two methods:

- a Windows Registry autorun key

- a Scheduled Task

Neither of these methods are particularly sophisticated; the adversary used a well-known autorun key when they could've used far more obscure ones. The task is also relatively simple: it's just trying to blend in under the guise of a Windows Help tool.

Something to note: both of these persistence mechanisms are relatively easy to enumerate and hunt at scale across an enterprise using several solutions.

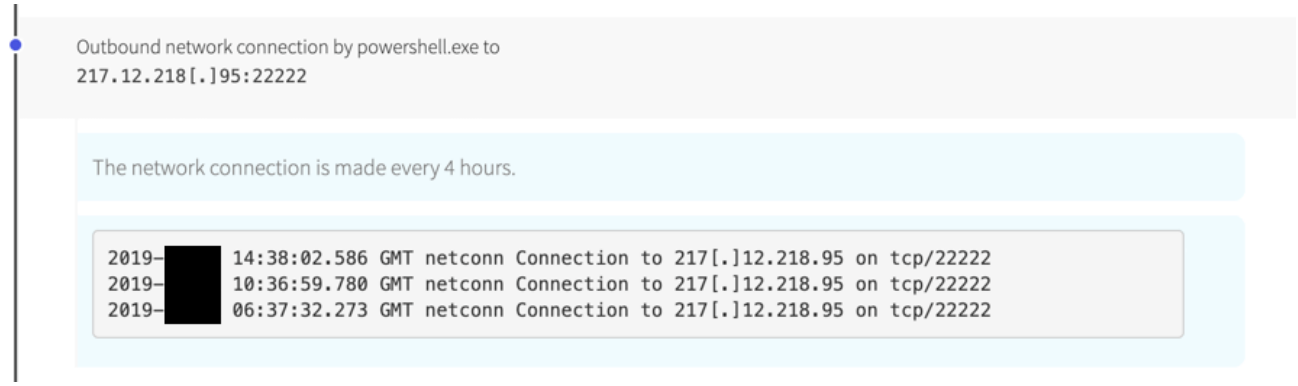
To execute its final payload, `installer_8.exe` spawned `rundll32.exe` to load the `workerInstance` function from `assistant32.dll`. Once this ran, it created a file named `btid.dat`. When the persistence mechanisms executed, we observed the same `rundll32.exe` behavior again.

At this point, we can tell the behaviors observed are most likely malicious due to PowerShell activity, but we don't have a lot of information around this particular DLL. Some quick Google searches unearthed this research from Morphisec, suggesting that we apparently found FrameworkPOS malware!

Finding patient zero

A concerning part of this detection within the Red Canary Cyber Incident Response Team (CIRT) was that the earlier PowerShell communication did not establish an external network connection. Instead we observed a network connection to another host on the internal network. This spawned a lateral movement hypothesis for us: there was likely another host on the network that was patient zero.

After hopping around hosts to find the original source of activity, we found another PowerShell command executing as a service that did establish an external network connection.



After consulting the Morphisec research again, we found the external network connection was consistent with the same campaign they observed in the wild.

Adequate adversaries still exist

Looking into Morphisec's research, there is a possibility that this campaign could be tied to cybercriminal group known commonly as FIN6. This group has targeted POS systems in the past, and recent reports indicate they may be involved with ransomware attacks. FrameworkPOS is very much a tool used by groups considered to be APTs. Even with this qualification, we can see a tendency to use "good enough" persistence and execution.

This provides a good starting point for defenders. We can start simple and grow to have more complex detection capabilities as we mature. To begin hunting for malicious persistence mechanisms, we can search for Registry key values that shouldn't exist in `Microsoft\Windows\CurrentVersion\Run` before hunting for every key referenced in the infamous "Beyond good ol' Run key" blog series.

In addition to the Registry key, we can start simple with event logging by focusing on just events for Scheduled Task execution and new service creation. By the time we become proficient at collecting and hunting through these artifacts, we'll be ready to tackle more advanced techniques.

Behaviors from this detection

Here are some of the search queries that contributed to this post.

FrameworkPOS DLL execution

High confidence

Process is 'rundll32.exe` AND command line contains 'workerInstance'

Shell execution as a service

Medium confidence, needs tuning for your products and admin tools

Parent process is ‘services.exe’ AND process is ‘cmd.exe’ or ‘powershell.exe’

Privileged Scheduled Task execution

Low/medium confidence, tune out administrative activity.

Parent process is ‘taskeng.exe’ AND username is ‘SYSTEM’

Windows Registry Autorun Key modification

Low/medium confidence, tune out new installations and software updates

Modification to Registry Key ‘Microsoft\Windows\CurrentVersion\Run’

Conclusion

Don’t be intimidated by adversaries. More often than not, you’ll find they don’t have to use sophisticated techniques during attacks. If you start simple and work toward becoming more mature as you grow, you’ll be ready for the small stuff and able to hunt down the more complex challenges when they come to you.

Related Articles

Detection and response

ChromeLoader: a pushy malvertiser

Detection and response

Intelligence Insights: May 2022

Detection and response

The Goot cause: Detecting Gootloader and its follow-on activity

Detection and response

Marshmallows & Kerberoasting