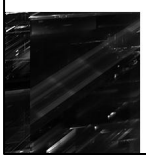
It's ~~Maps~~ Gaps All the Way Down

Why detection and incident response are hard and what can be done to make them easier



red canary

You're looking at a portion of a detection timeline. It shows an explorer.exe process that spawned a conhost.exe process, which in turn spawned a Windows command shell. For any detection and incident response folks in the audience, what about this timeline concerns you?

What additional data, if any, do you want to look at for your investigation?

Suggestions:

- Logs
  - Security event log
  - Application
  - System
- Memory dump
- Full disk image
- Packet capture

Overview

Introductions

- Who are you?
- Who am I?

Why are Detection and Security Incident Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

THING ONE

Introductions

red canary

Who are you?

- Raise your hand if you do detection
- Raise your hand if you do security incident response
- Keep your hand up if you think your job is hard

Go Blue Team!

red canary

Raise your hand if your job responsibilities include

- detecting bad things on endpoints or in networks?
- responding to these events?
- you're a pen tester or red teamer
- you're an expert chess player

Follow-up questions targeting individuals who raised their hands:

Do you do detection or IR or both?
How many years?
Do you find your job difficult? The right answer is yes.
What makes it difficult?

It’s not just you

“… security is quite possibly the most intellectually challenging profession on the planet… for two reasons… complexity… and rate of change [are] your enemy.”

-- Dan Geer

red canary

You may not think your job is all that difficult.

Perhaps you’re a wizard or you have better tools than the rest of us.

But Dan Geer, an industry giant, thinks we have what is possibly the most intellectually challenging job on the planet.

Geer doesn’t even mention that we’re actively working against intelligent adversaries who are working to make our lives difficult without getting caught.

Who am I?

Dave Hull

Detection Engineer
RED CANARY

@davehull

- Detection engineering at Red Canary
- Former technical lead for IR in O365
- Former SANS instructor, blogger, editor
- Creator of Kansa -- a framework for collecting and analyzing endpoint data
- I pick up pennies in front of steam rollers

red canary

I'm a detection engineer at Red Canary. I work on improving Red Canary's detection capabilities in various ways

- building and tuning detection analytics,
- writing code to improve detection capabilities,
- writing code to improve our suppression capabilities.

Prior to Red Canary I was

- senior director for endpoint detection and response at Tanium
- created and delivered product demos,
- created and delivered trainings,
- wrote code and
- worked with software engineers and PMs to deliver features that our customers wanted.

Before Tanium, I was at Microsoft where I was

- technical lead for security incident response in O365, a distributed environment comprised of roughly half a million systems running in a dozen or so data centers around the world.
- I've given talks based on some of my experiences there both at SecKC and the SANS Digital Forensics and Incident Response Summit.

While at MSFT

- I spent several months worth of nights and weekends developing Kansa, which is an open source framework for collecting and analyzing data from endpoints that may be of use during security incident response, hunters or simply to baseline an environment.

From 2007 to 2012 I was

- an instructor with the SANS Institute and taught what was then
    - Introduction to Windows Forensics (408),
    - Advanced Digital Forensics and Incident Response (508) and
    - Reverse Engineering Malware (610).
- I was the managing editor and a leading contributor to the SANS Digital Forensics Blog.

Overview

~~Introductions~~

- ~~Who are you?~~
- ~~Who am I?~~

Why are Detection and Security Incident Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

The idea for this talk came to me during my most recent round of funemployment last summer. I was listening to the Farnam Street podcast when the host was talking about how the map is not the territory and I had been re-reading Daniel Kahneman's book Thinking Fast and Slow. I have a pinned Tweet on this topic on my Twitter feed, if you want the abbreviated, possibly more coherent version of this presentation.

https://twitter.com/davehull/status/1143161775195602946

Every pen tester's favorite topic

What's out of scope?

Some hard problems and some easy ones.

I'm going to gloss over lots of problems that are difficult on their own

- collecting data from endpoints at scale,
- analyzing that data in a scalable and efficient manner, etc.

Standing up

- a homegrown solution,
- a commercial SIEM,
- deploying and maintaining commercial EDR platforms, etc.

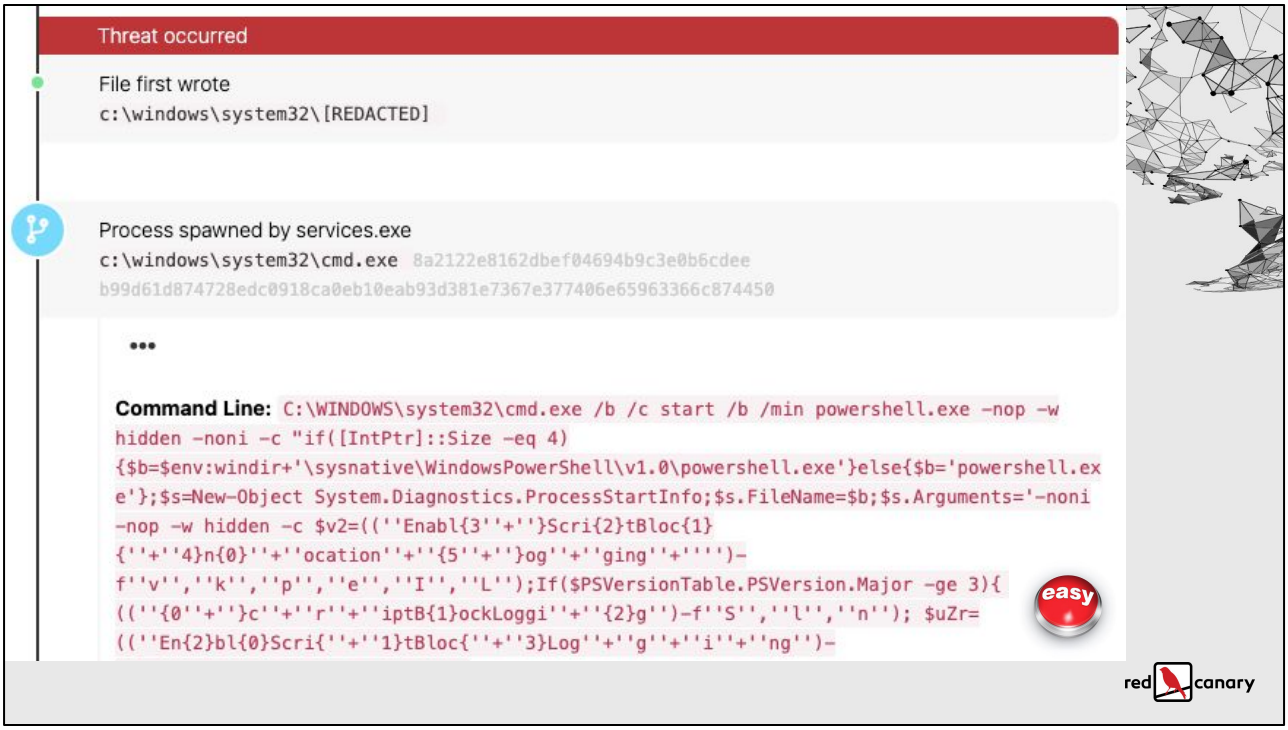
These may be difficult problems on their own. For our purposes, we're operating under the assumption that we've already solved that problem. Our journey begins with the endpoint telemetry.

I'm postulating that detection and incident response are difficult.

Why?

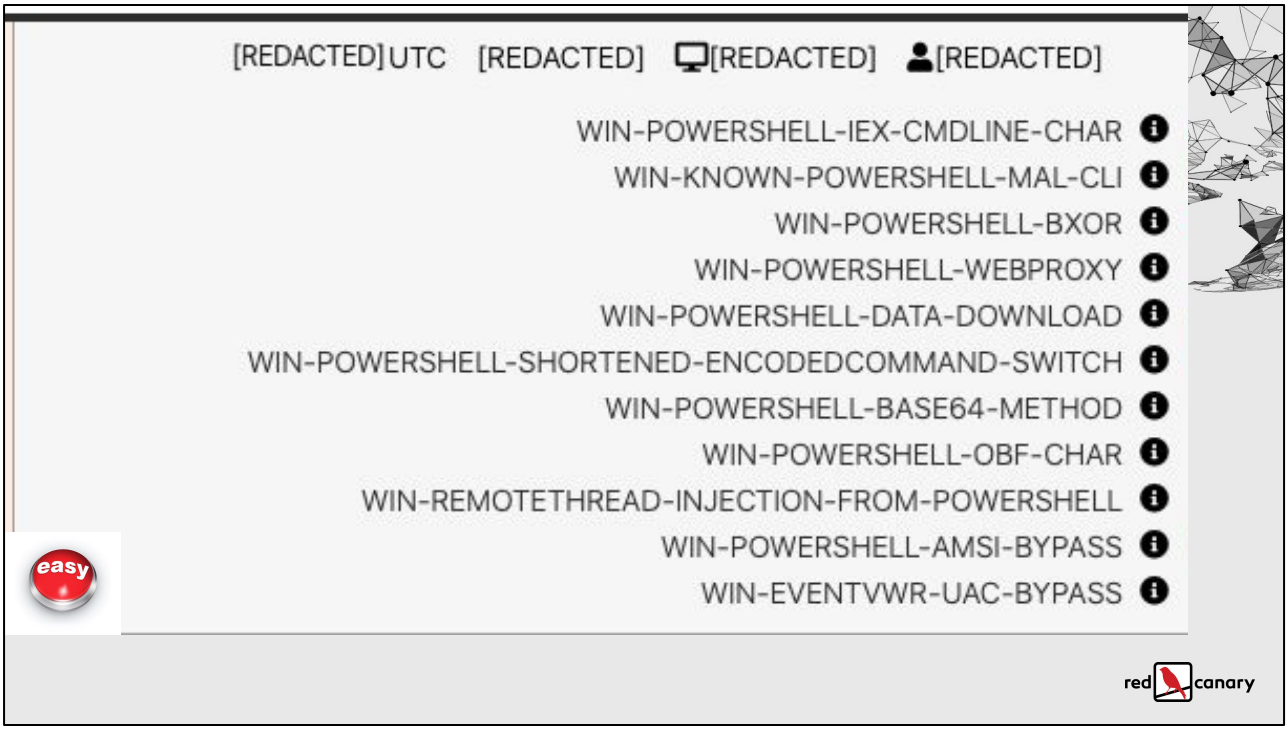
There's no single answer and it's not universally true, easy detections are easy, but the hard ones are hard.

Let's take a look at some of the easy ones.



I'm not talking about easy detections like the one above.

The irony attacks like the one above, which is obfuscated to avoid automated detection, is that the obfuscation makes them easier for humans to recognize as obviously malicious.



Here's another easy one -- Red Canary has thousands of detection analytics. This one event came through our queue recently triggering 11 different analytics. This is not a difficult detection and therefore not the subject of this presentation.

Let's talk detections

A "quick" tour of the problem space

red canary

With a quick look at some easy stuff out of the way, let's dive into more difficult territory.

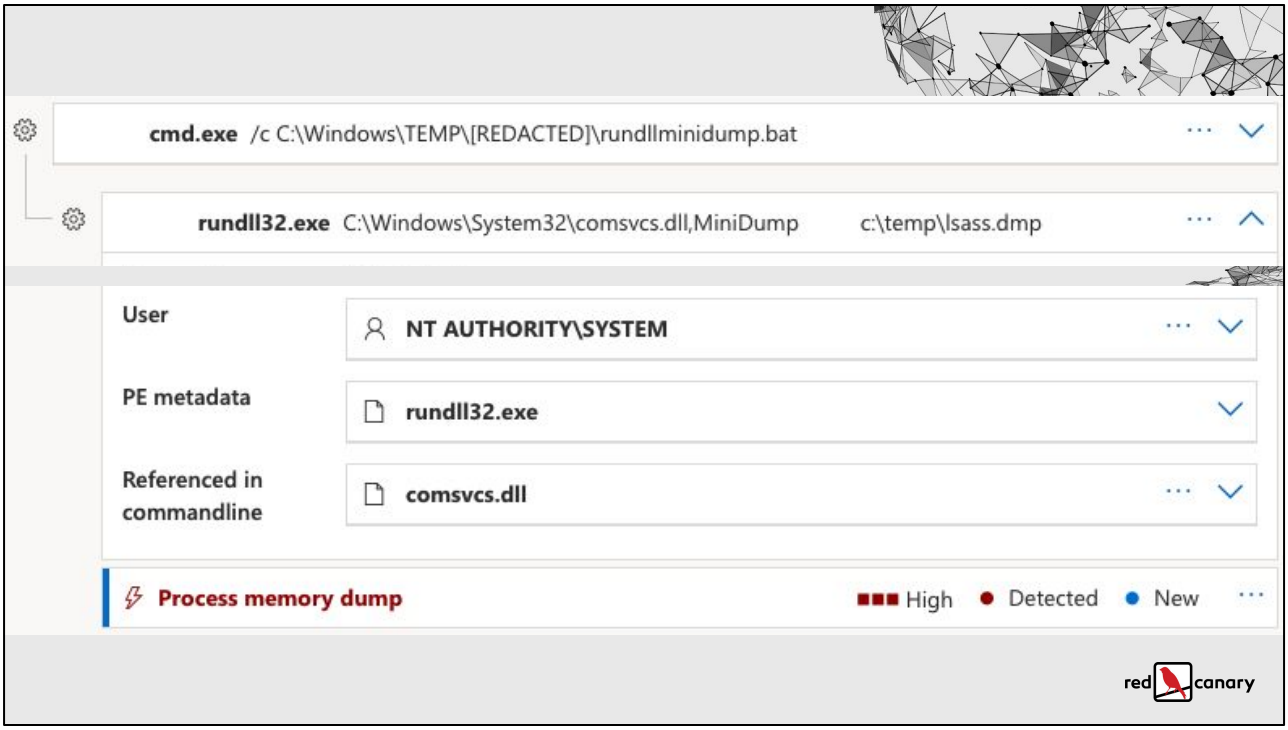
Let's talk when something bad happens

False positives and false negatives

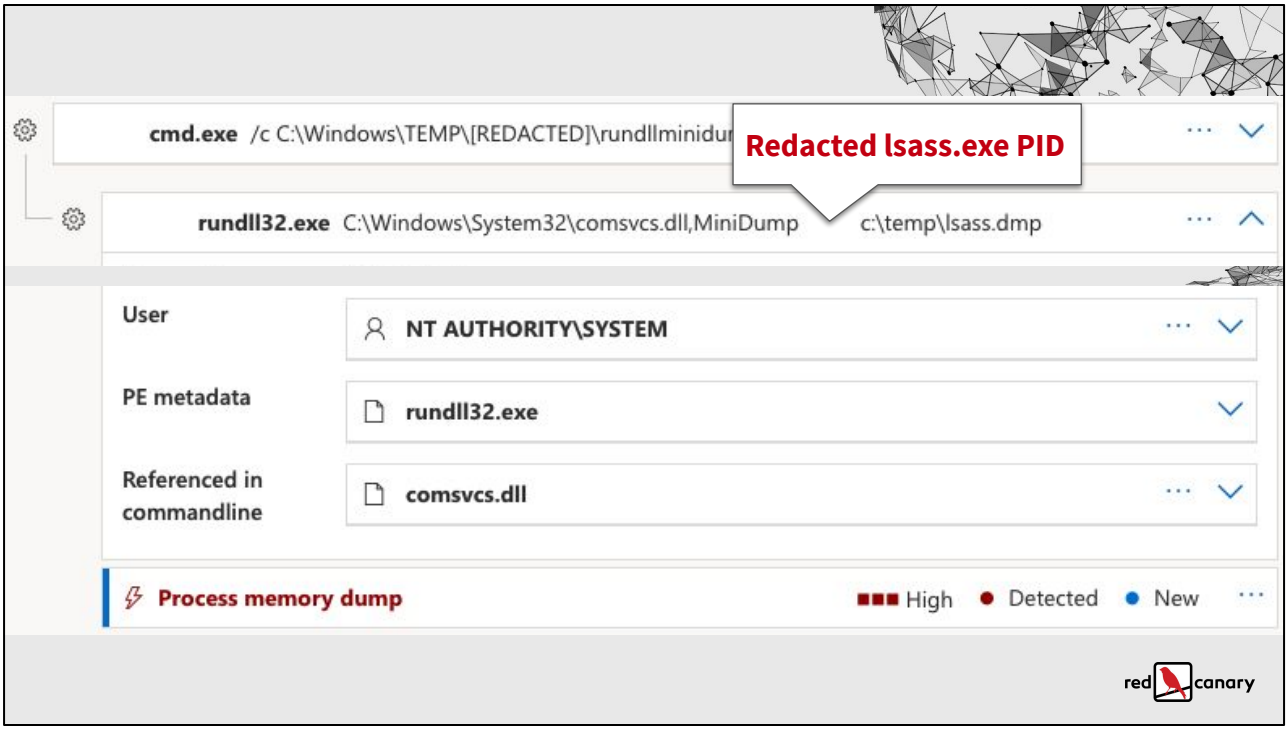
Why are you so negative about false positives?

red canary

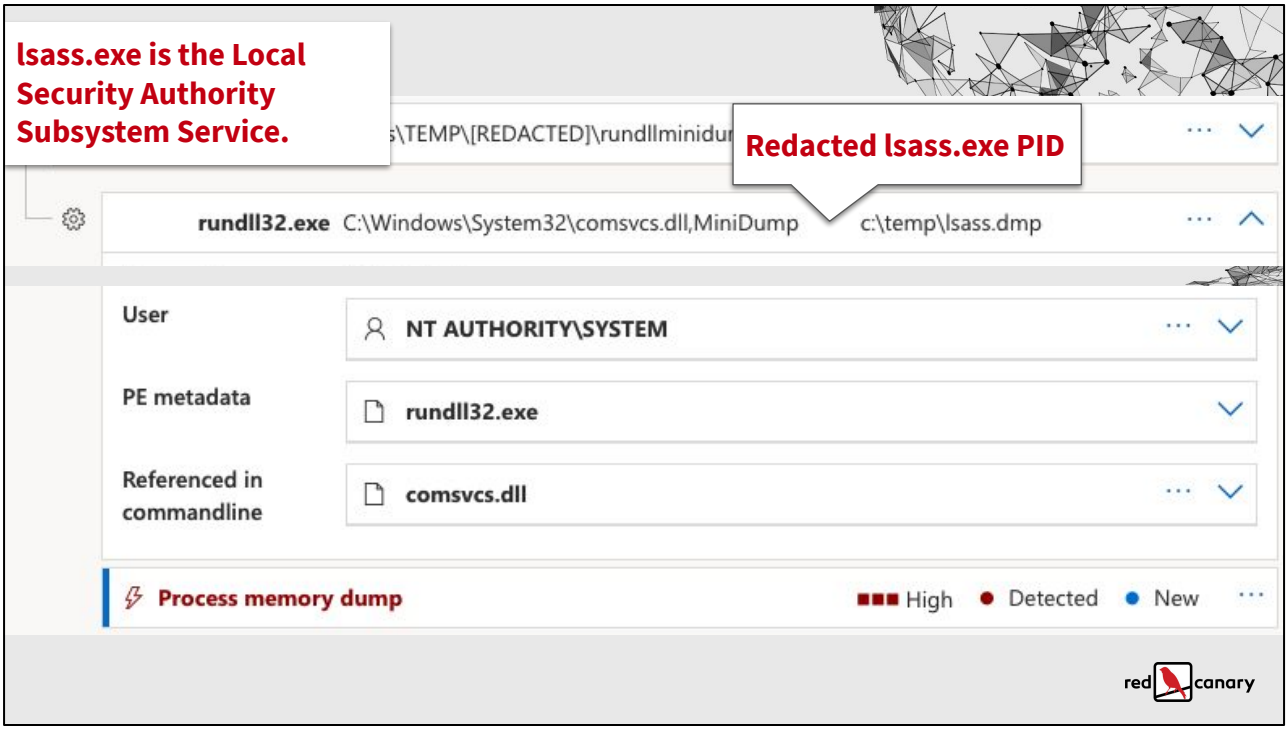
I really don't intend to pick on any single EDR platform, or any of them at all. That's not the point. These are great tools that have moved the needle in the right direction to make detection and incident response easier. These tools have a difficult job and there are adversaries actively trying to circumvent them.



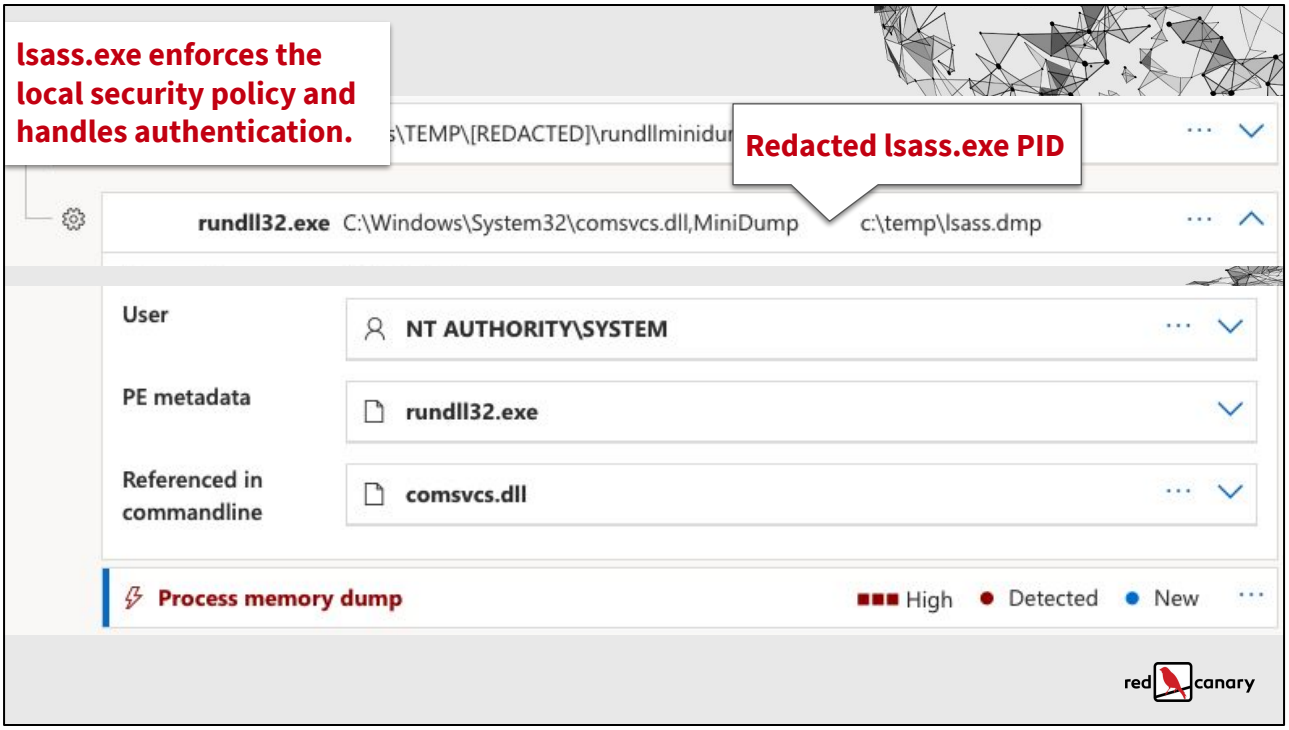
Above we have a partial view of a timeline from an EDR platform.



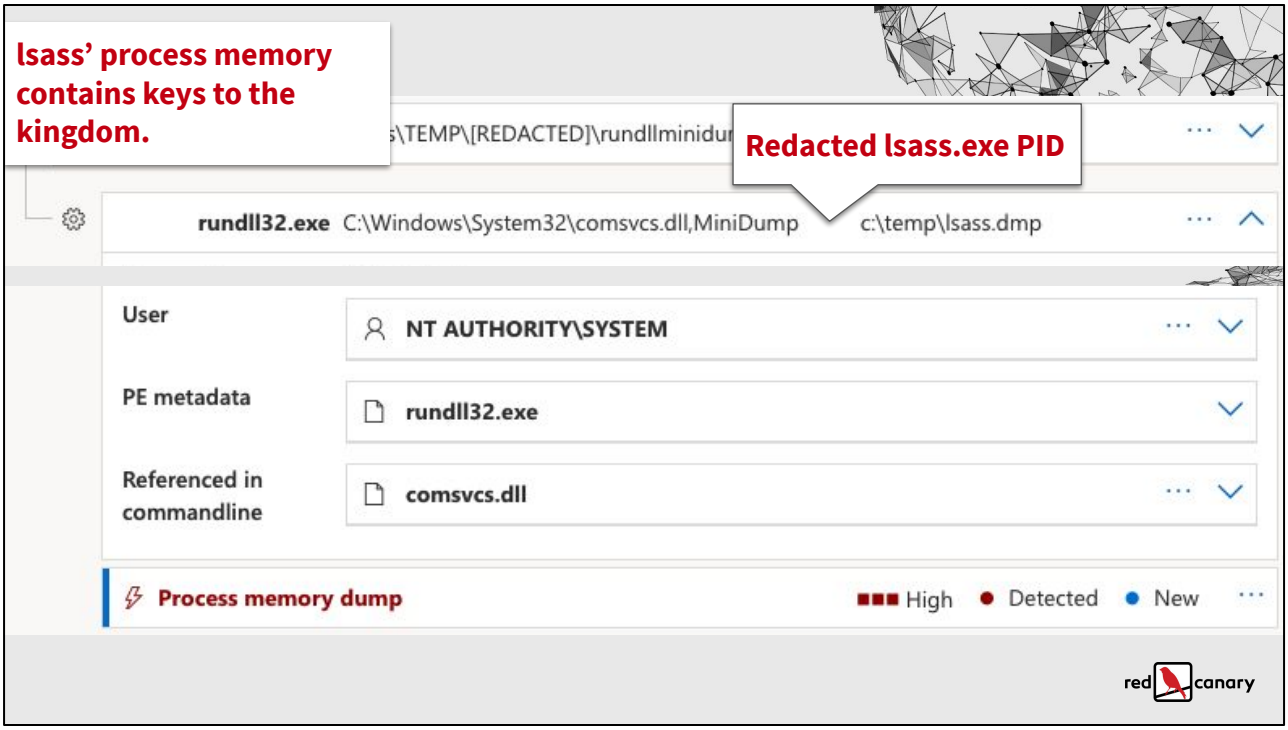
There's a process ID for lsass redacted from the command-line in this image.



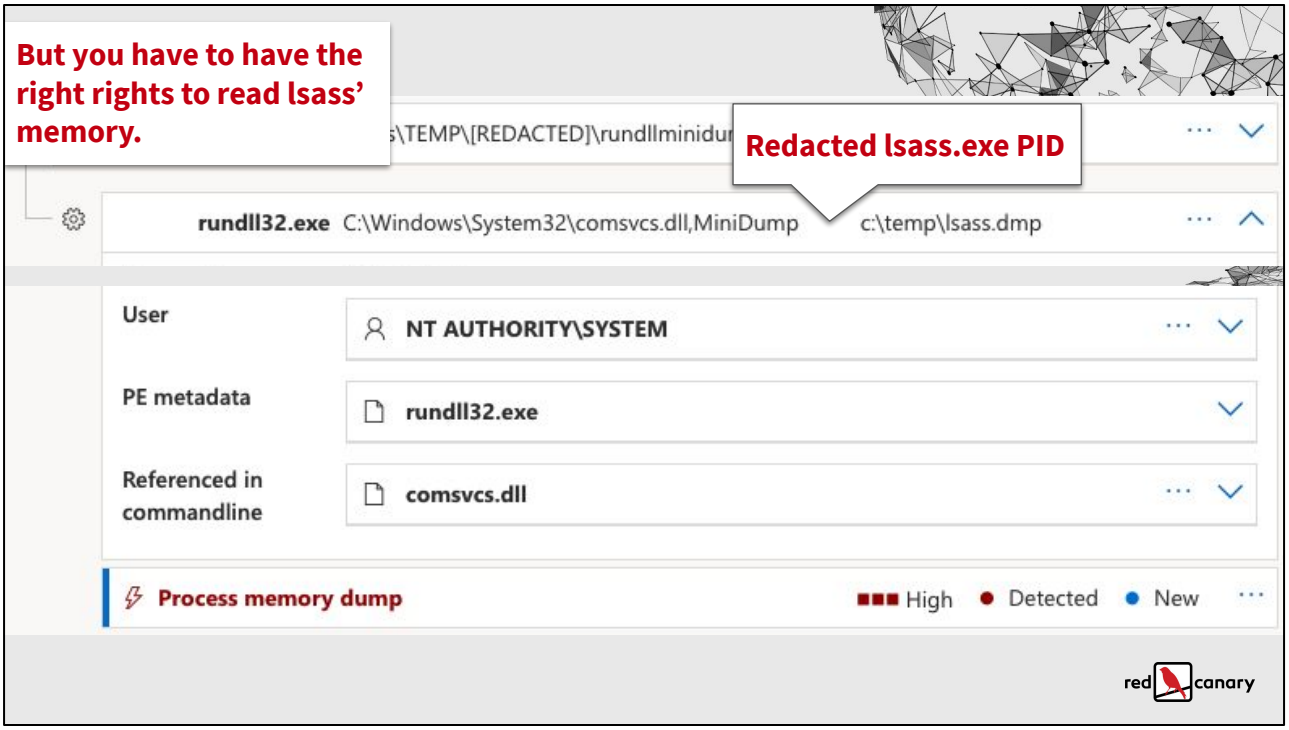
In case you're not familiar with lsass, it is the Local Security Authority Subsystem Service.



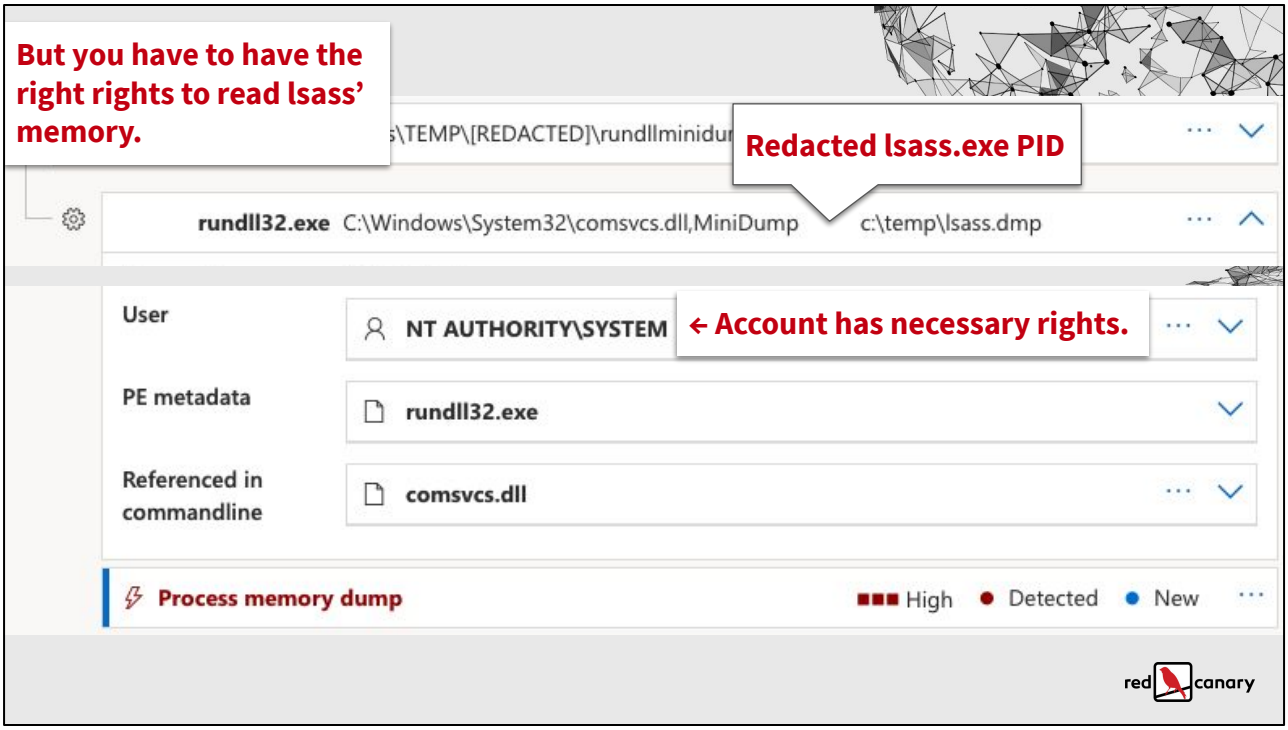
lsass enforces the local security policy and handles authentication on the local machine, to the network, etc.



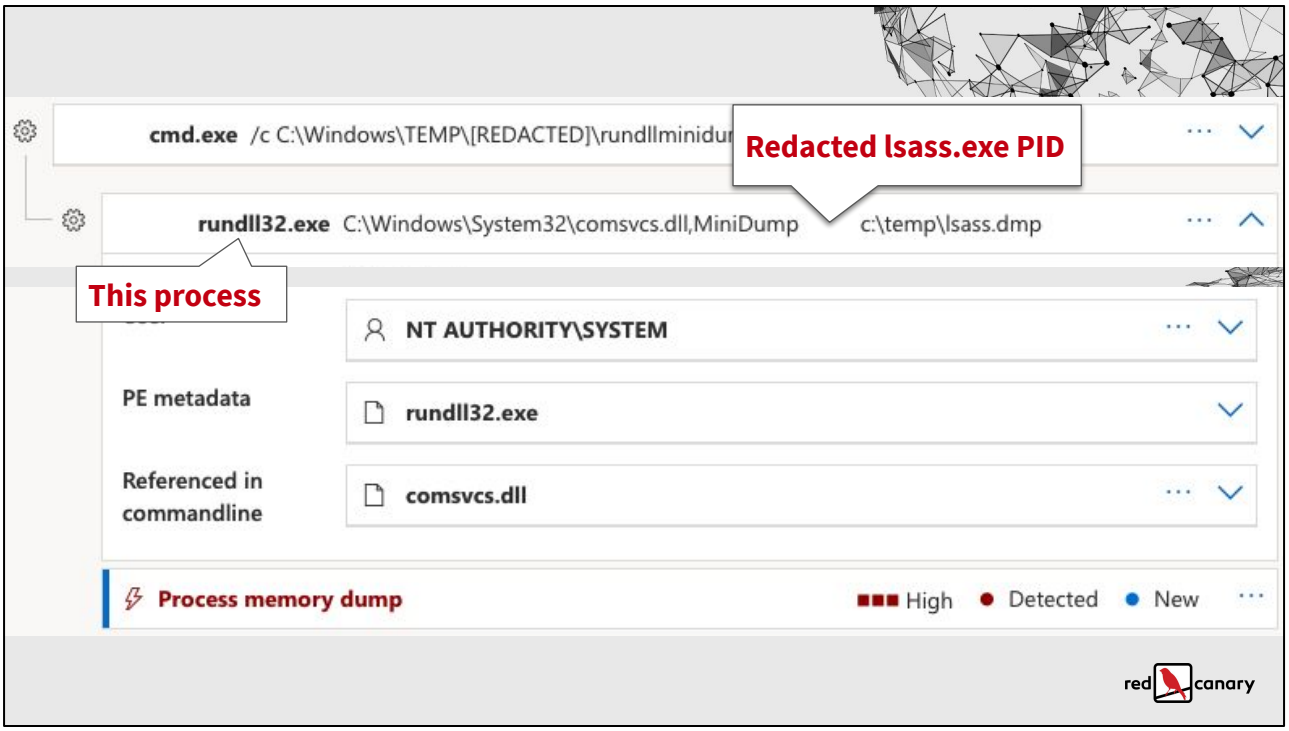
lsass' process memory is a target because of the data it contains. On some systems, it may contain credentials in the clear or it may contain hashes that can be cracked offline.



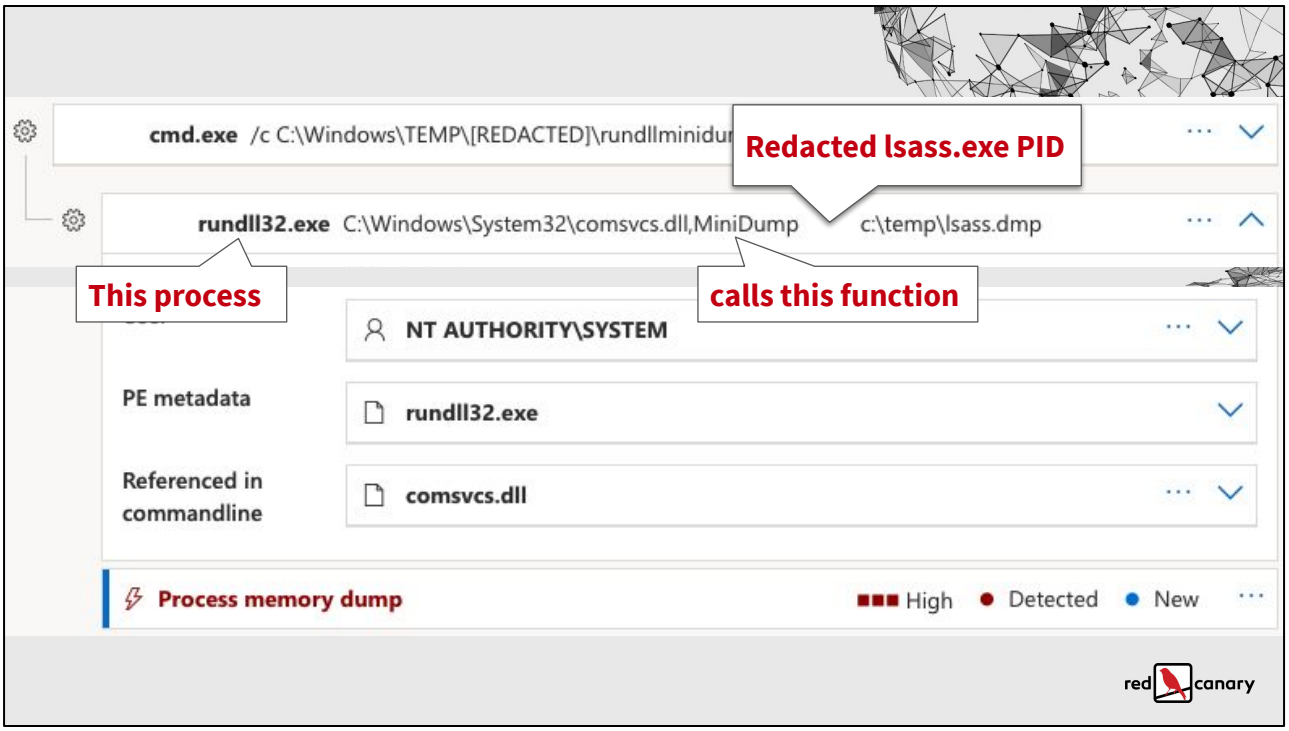
Because lsass process memory contains valuable information, not just any user can access its process memory. You have to have the right level of permissions on the endpoint.



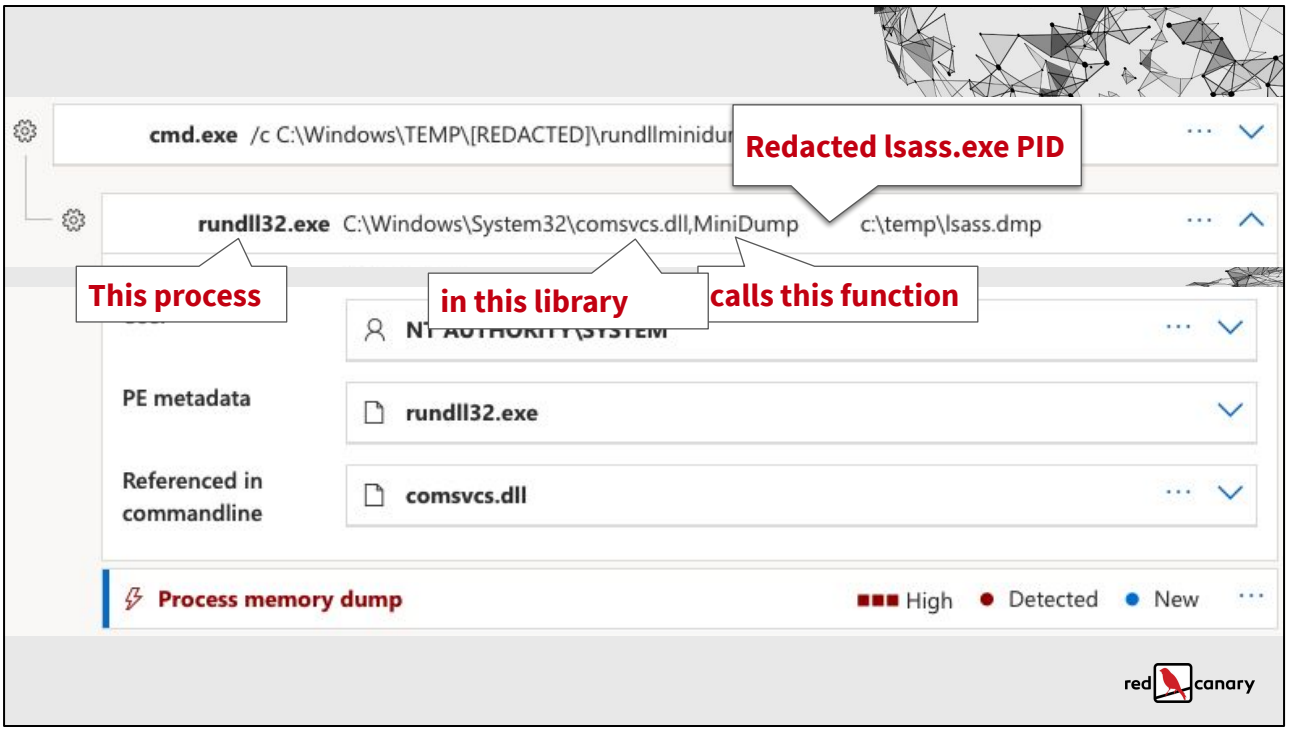
NT AUTHORITY\SYSTEM can do anything on the endpoint. It has the necessary permissions to read lsass' memory.



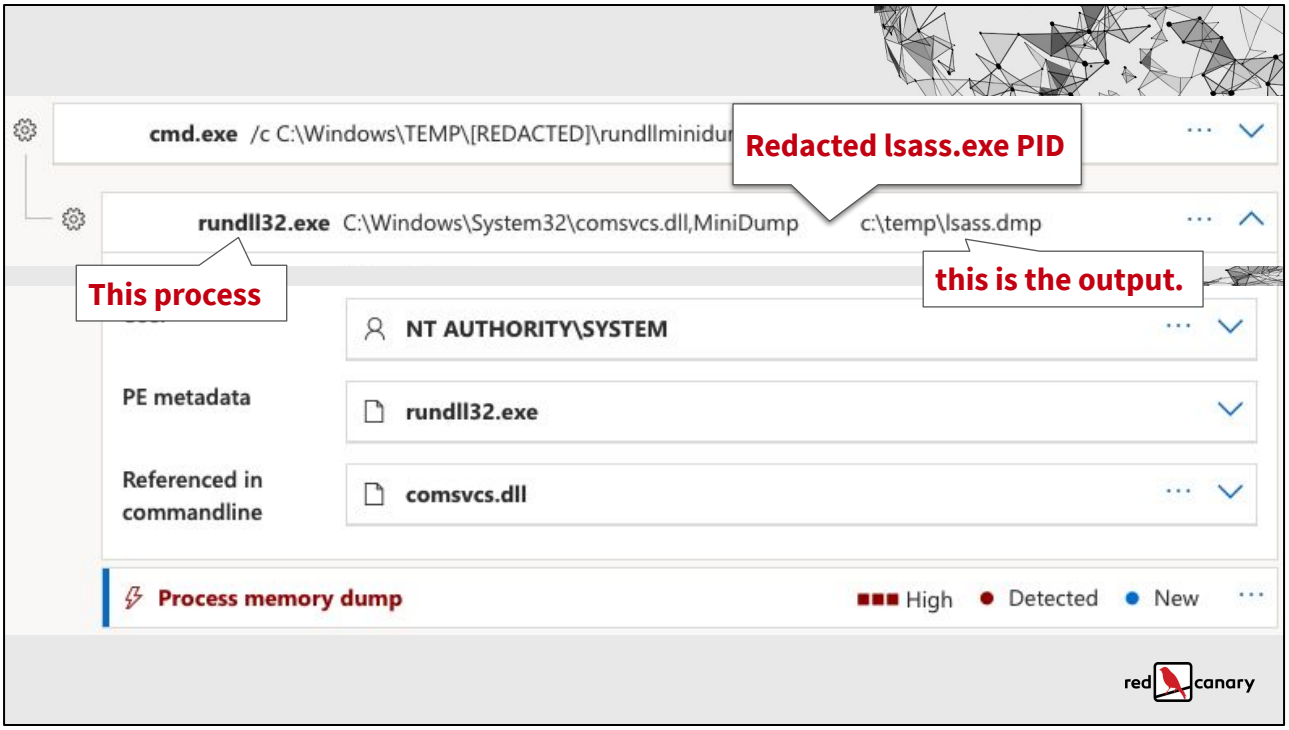
So the attack here starts with rundll32.exe.



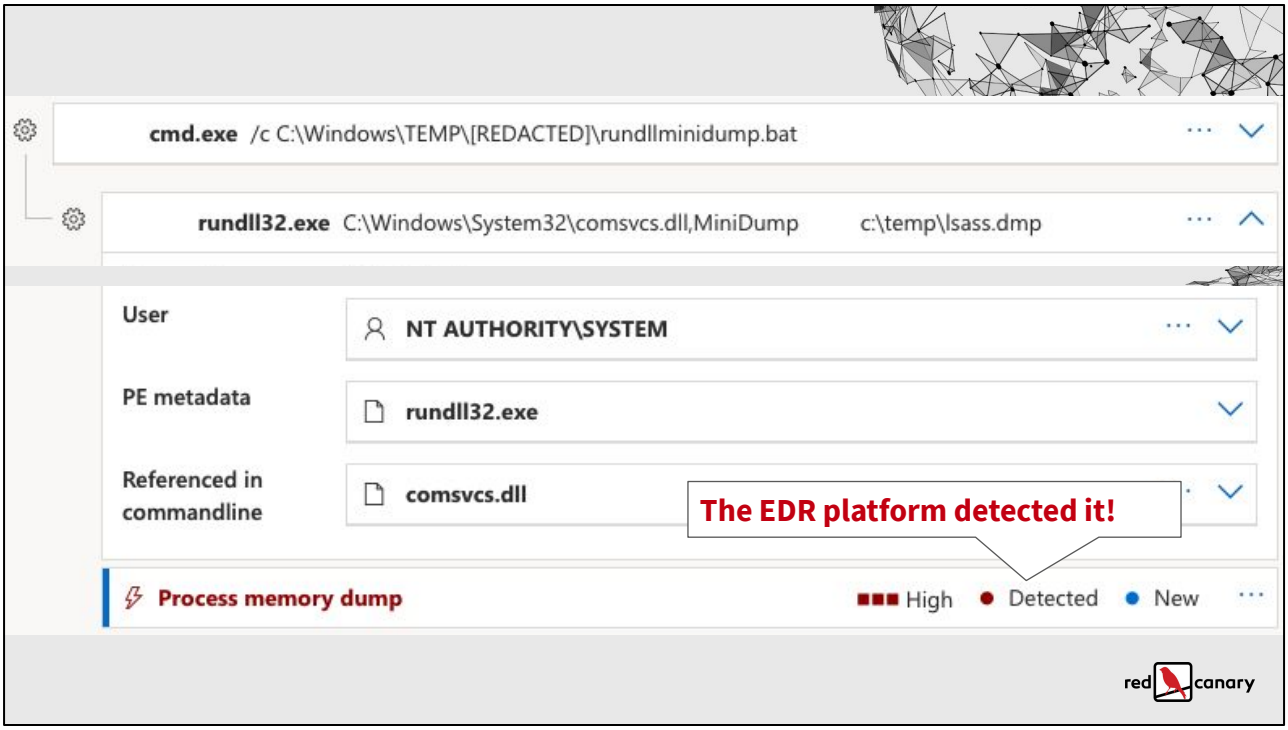
So the attack here starts with rundll32.exe calling the MiniDump function.



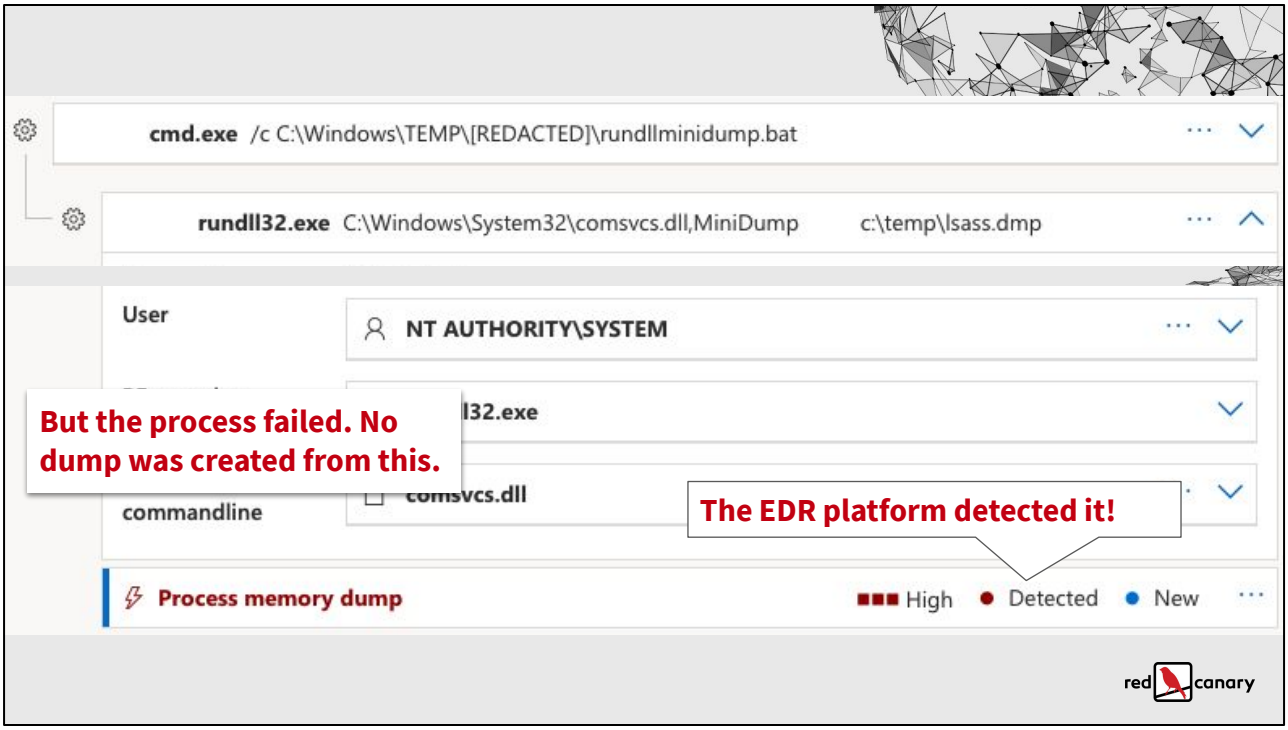
So the attack here starts with rundll32.exe calling the MiniDump function from comsvcs.dll, a native Windows dll for interfacing and COM objects.



So rundll32.exe is calling the MiniDump function exported from the comsvcs library, passing it a redacted process ID (lsass' process id) and telling the function to write its output to C:\temp\lsass.dmp.

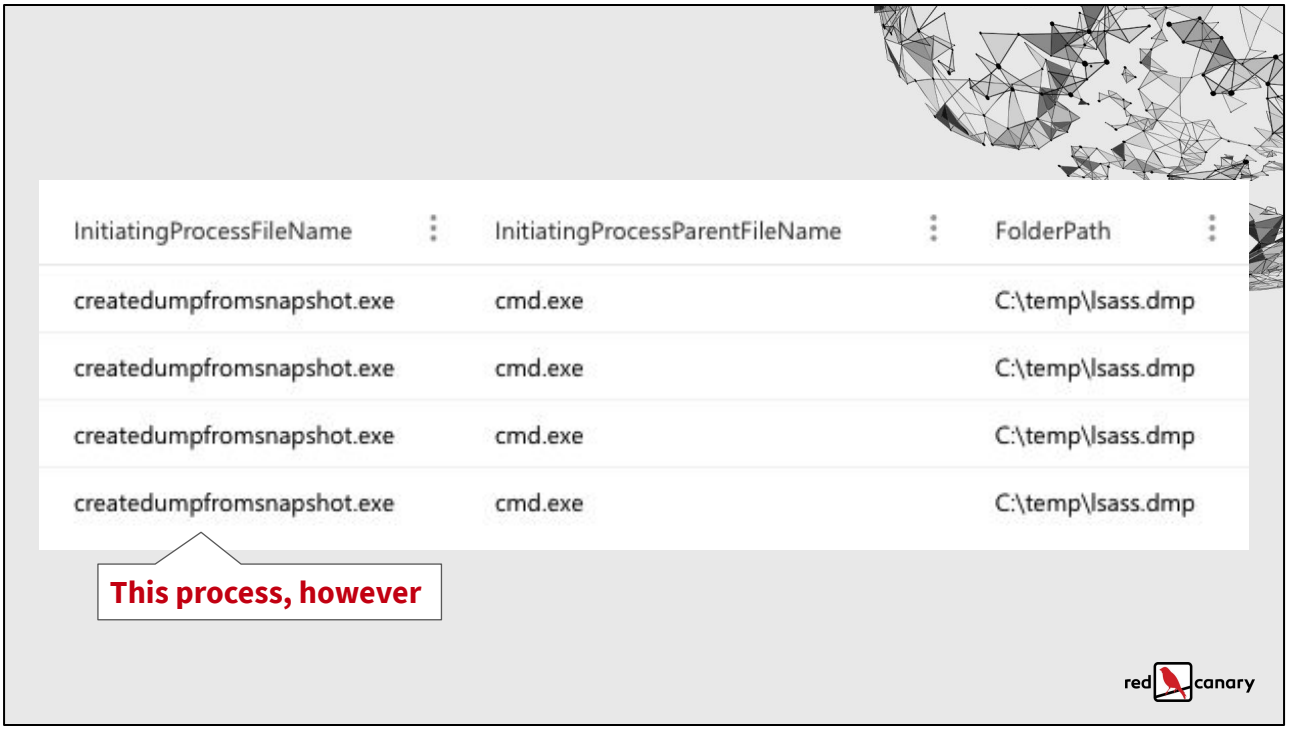


And the EDR platform detects this activity and marks it as “High”.

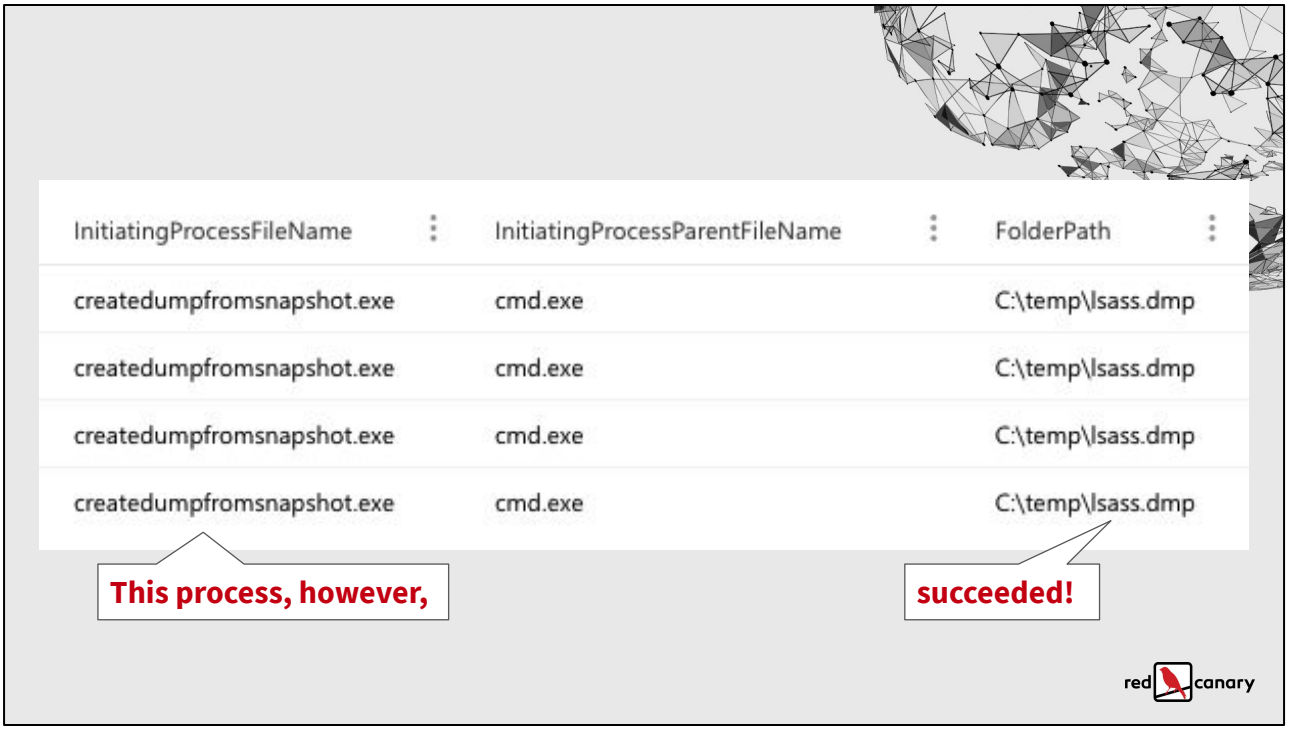


There's only one problem. That rundll32.exe command is incorrect and won't produce the output the attacker wants. In fact, it doesn't produce any output file.

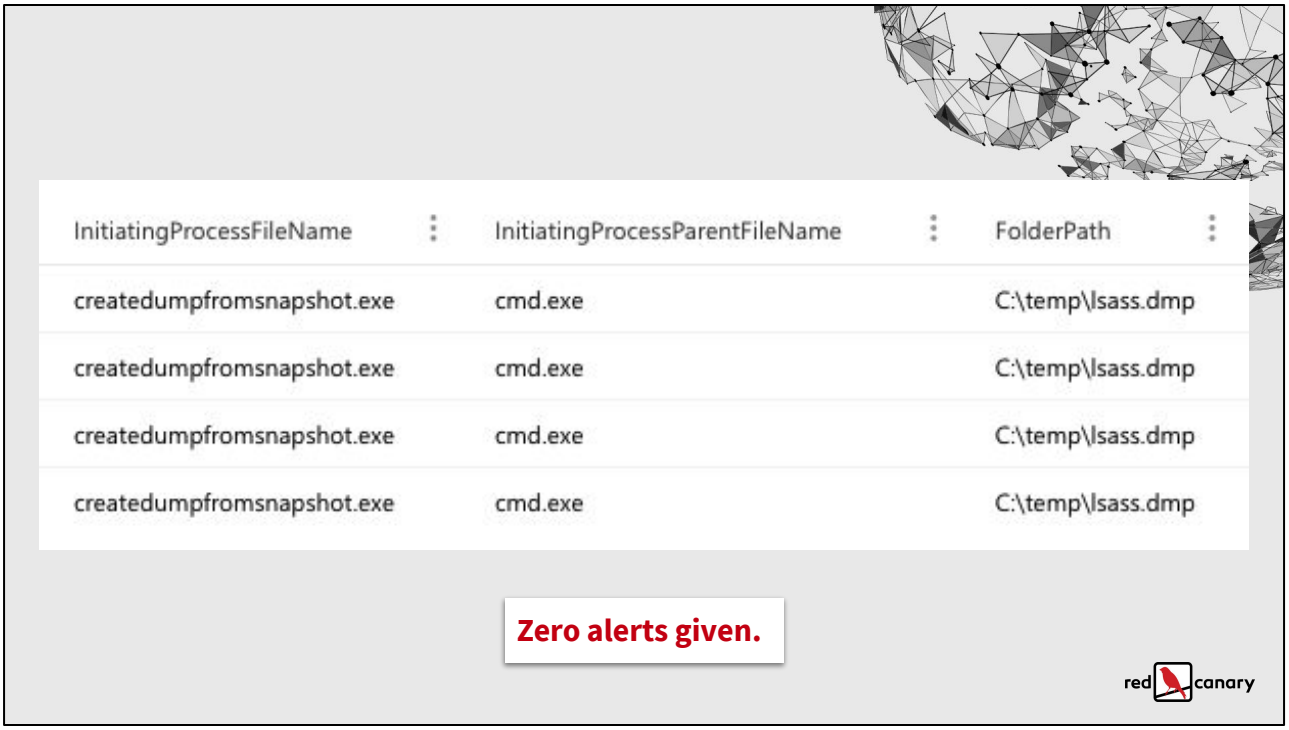
Of course, we want the EDR platform to alert us on things like this, even if they fail, but an analyst is going to look at this and have to spend time searching for the output file and that means writing a customer query, which is going to take time and remember we're in a time crunch here.



Here's another event from the same EDR platform. In this case, we have a different command, createdumpfromsnapshot.exe being executed.

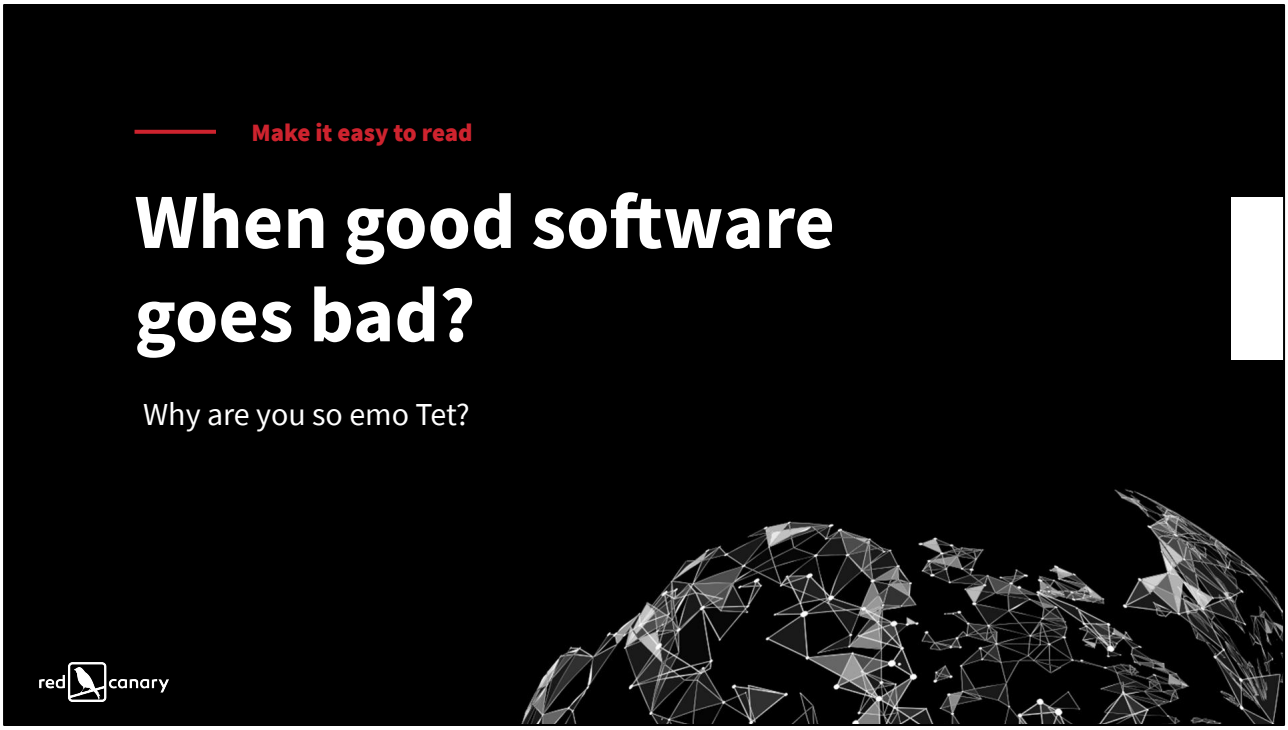


Createdumpfromsnapshot.exe succeeds where the previous command did not.

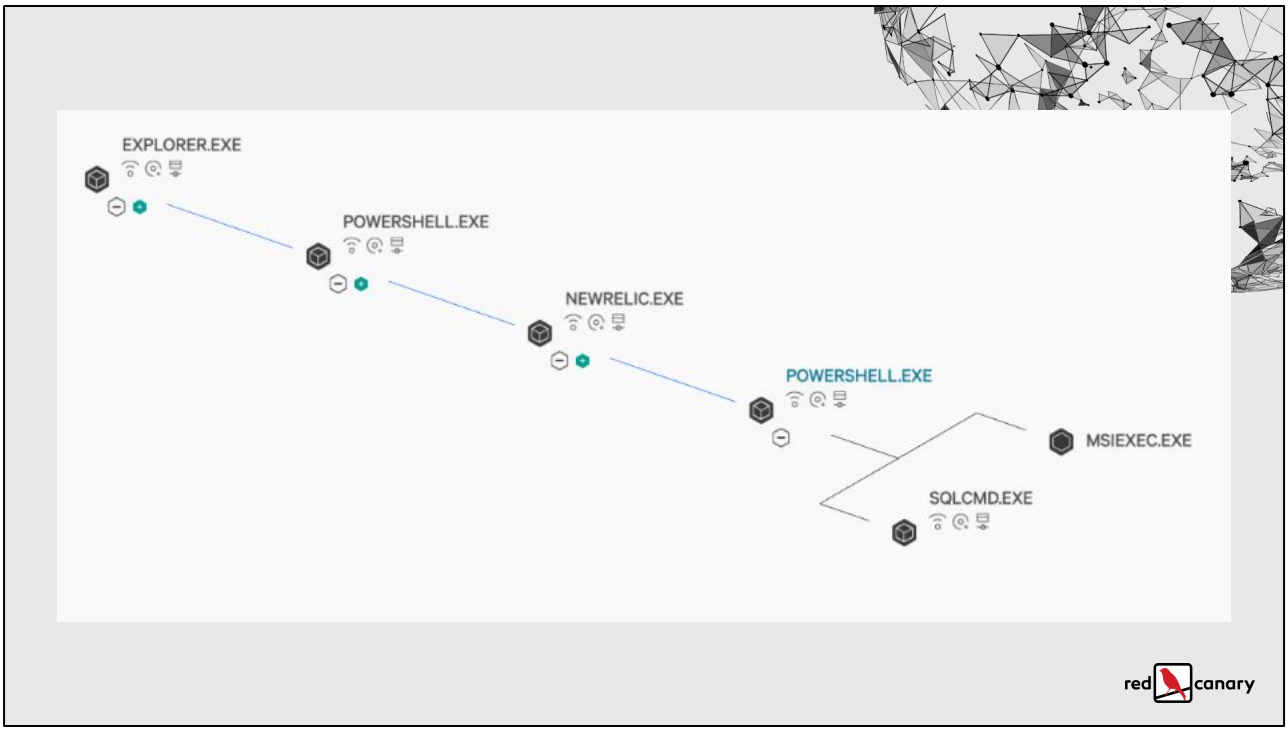


The problem in this instance is that the platform didn't trigger any alerts.

So we have a false positive (sort of) and a false negative.



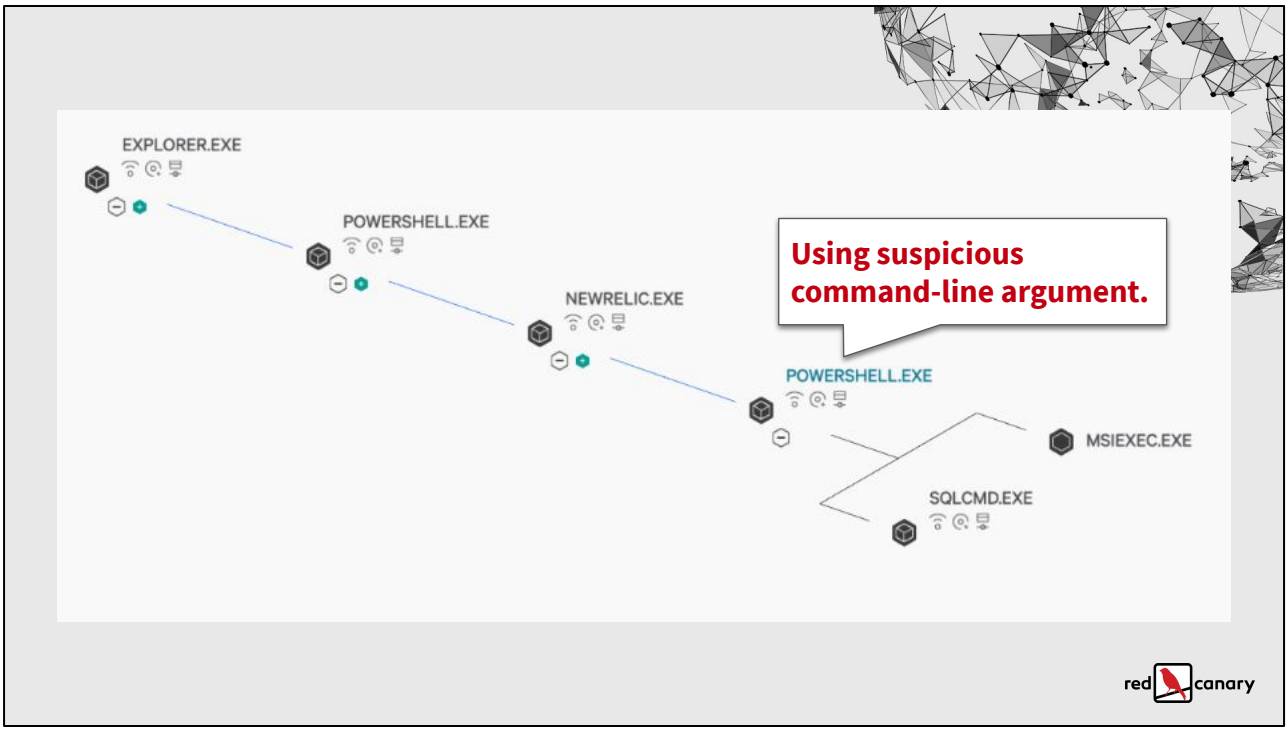
I really don’t intend to pick on any single EDR platform, or any of them at all. That’s not the point. These are great tools that have moved the needle in the right direction to make detection and incident response easier. These tools have a difficult job and there are adversaries actively trying to circumvent them.



Of course our engine is not perfect. We get benign events in our queue and we have occasional misses, but we're continuously improving.

Above is a process tree from another EDR platform. Telemetry from this platform sent an event to our queue for review because PowerShell was see executing with command-line arguments commonly seen in attack platforms.

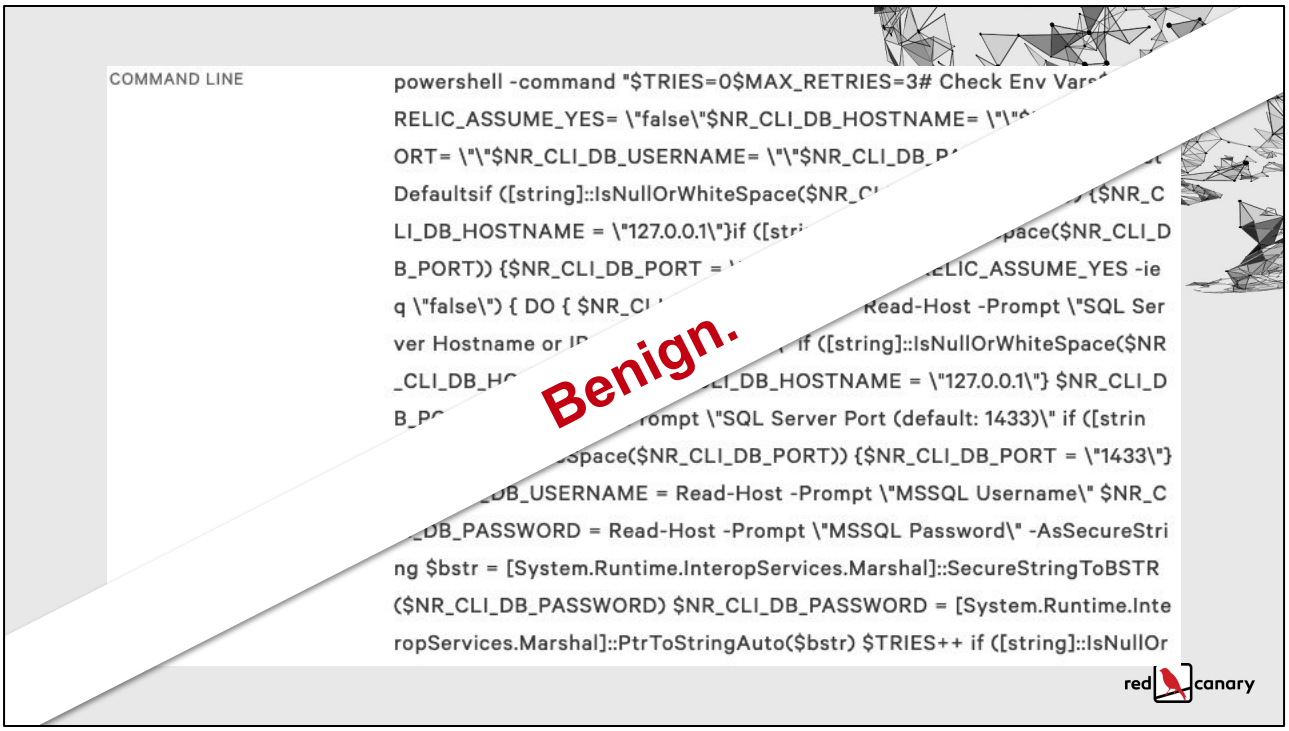
From the above process tree, it's hard to say if anything bad happened or not.



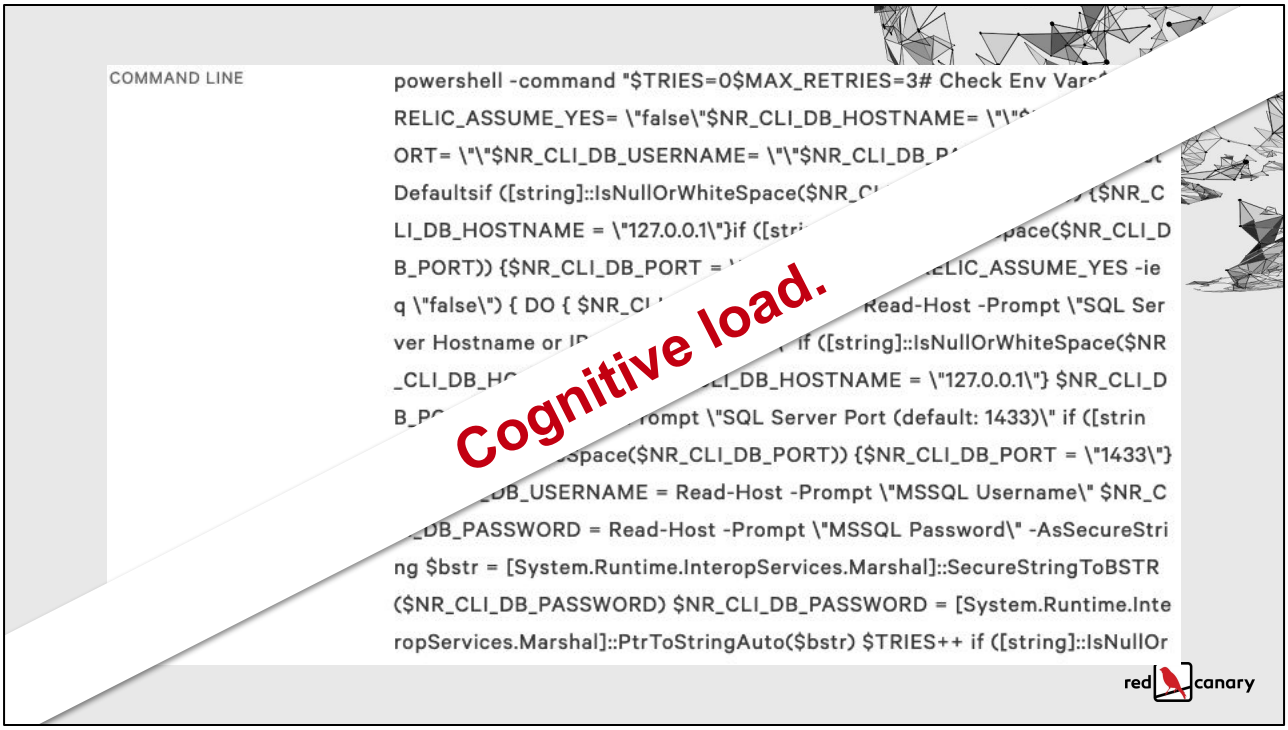
Red Canary's platform flagged this PowerShell process for using a suspicious command-line argument.

Here's part of that command-line. It would be helpful if the platform applied any formatting to this code, but it doesn't.

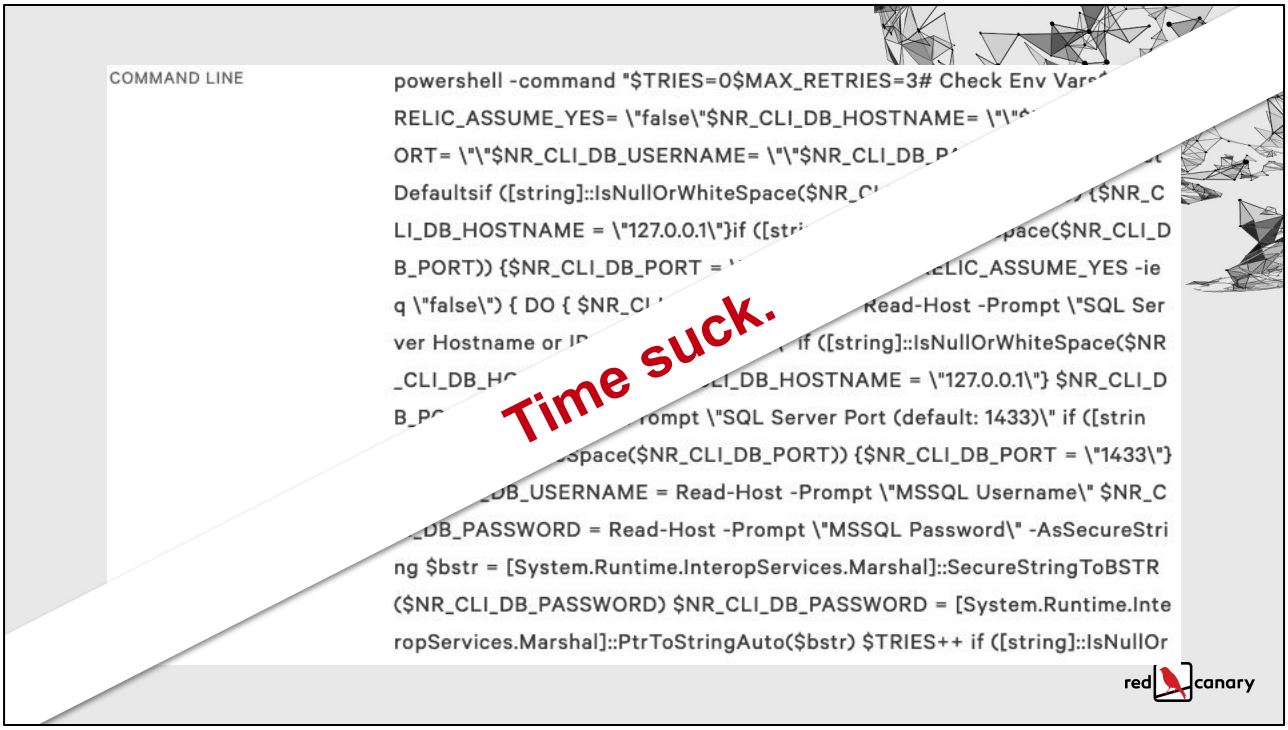
The System.Runtime.InteropServices.Marshal bit is used in Emotet and PowerShell attack frameworks, but in this context, what we're seeing looks benign. We could argue about whether or not having an installer that works this way is ideal or not, but in this case, it's not malicious.



The System.Runtime.InteropServices.Marshal bit is used in Emotet and PowerShell attack frameworks, but in this context, what we're seeing looks benign. We could argue about whether or not having an installer that works this way is ideal or not, but in this case, it's not malicious.



The System.Runtime.InteropServices.Marshal bit is used in Emotet and PowerShell attack frameworks, but in this context, what we're seeing looks benign. We could argue about whether or not having an installer that works this way is ideal or not, but in this case, it's not malicious.



The System.Runtime.InteropServices.Marshal bit is used in Emotet and PowerShell attack frameworks, but in this context, what we're seeing looks benign. We could argue about whether or not having an installer that works this way is ideal or not, but in this case, it's not malicious.

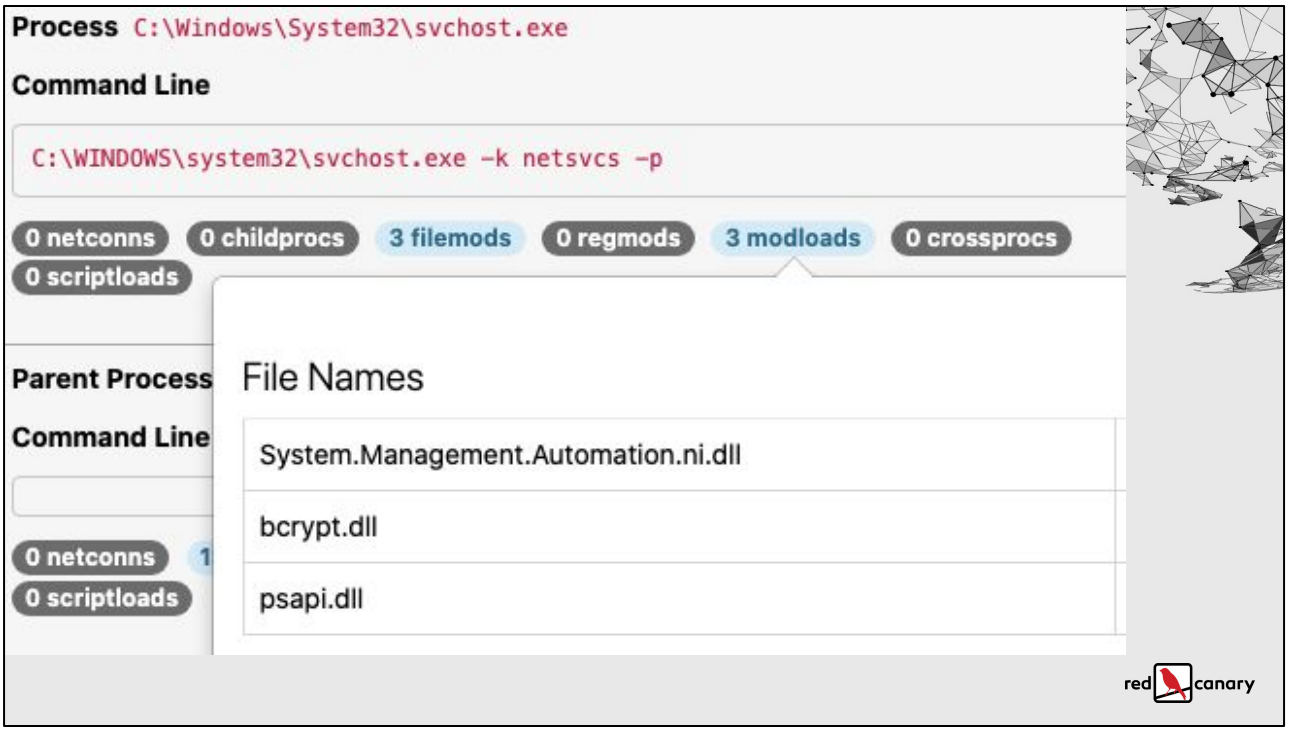
When it’s 12:30 AM UTC what time is it?

Catch me (online), if you can

Can I use your PID?

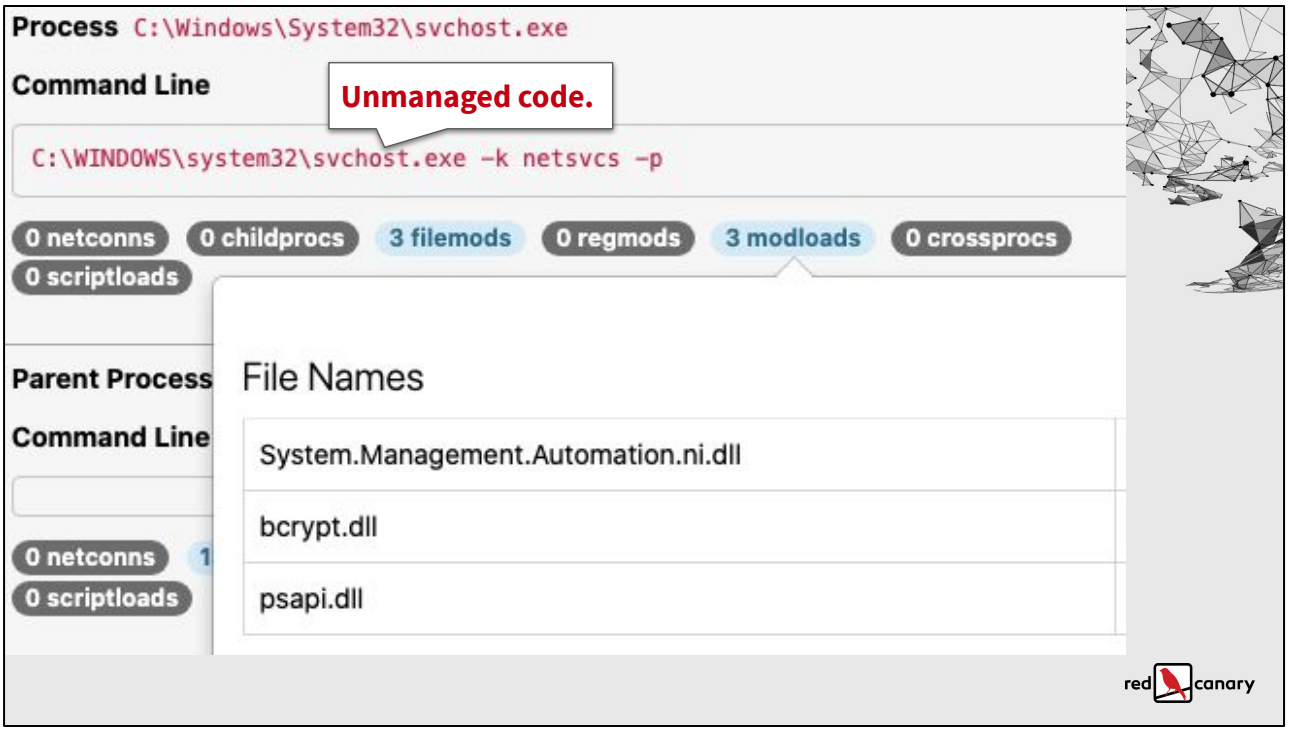
red canary

I really don’t intend to pick on any single EDR platform, or any of them at all. That’s not the point. These are great tools that have moved the needle in the right direction to make detection and incident response easier. These tools have a difficult job and there are adversaries actively trying to circumvent them.

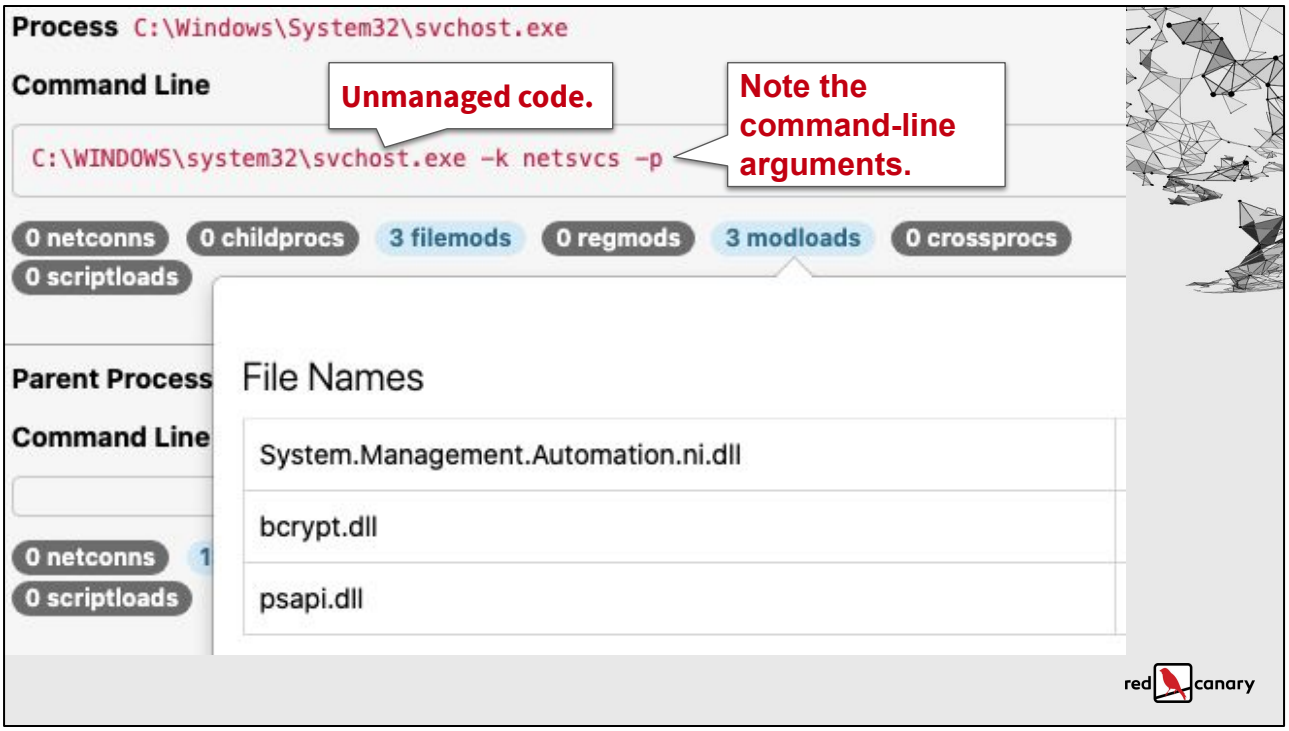


Here's an event that hit our queue at Red Canary.

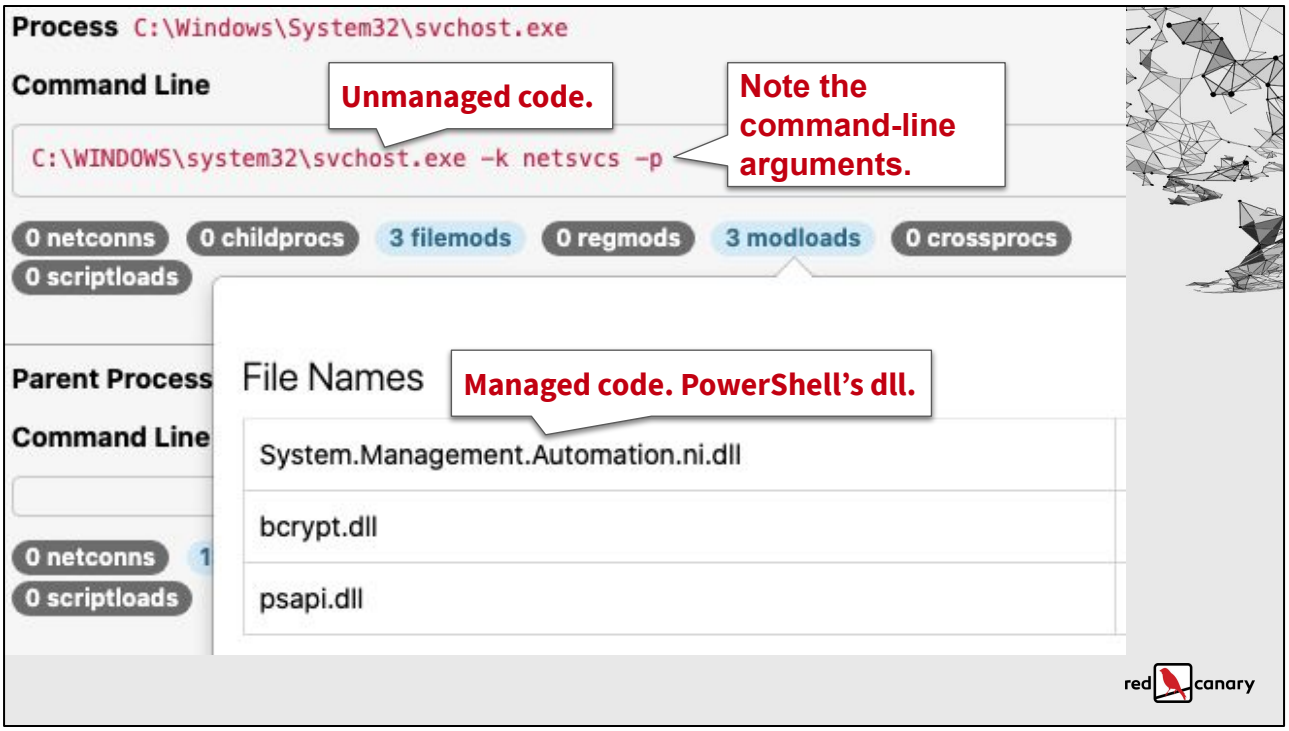
In the case above, we don't really see anything else of note happening -- no child processes, no network connections, harmless filemods, but let's take a look at the EDR platform just for good measure.



The event hit the Red Canary queue because svchost.exe apparently imported the System.Management.Automation.ni.dll. svchost.exe is unmanaged code, meaning it's written in C or C++ and interacts directly with the hardware. It does it's own memory management, etc.

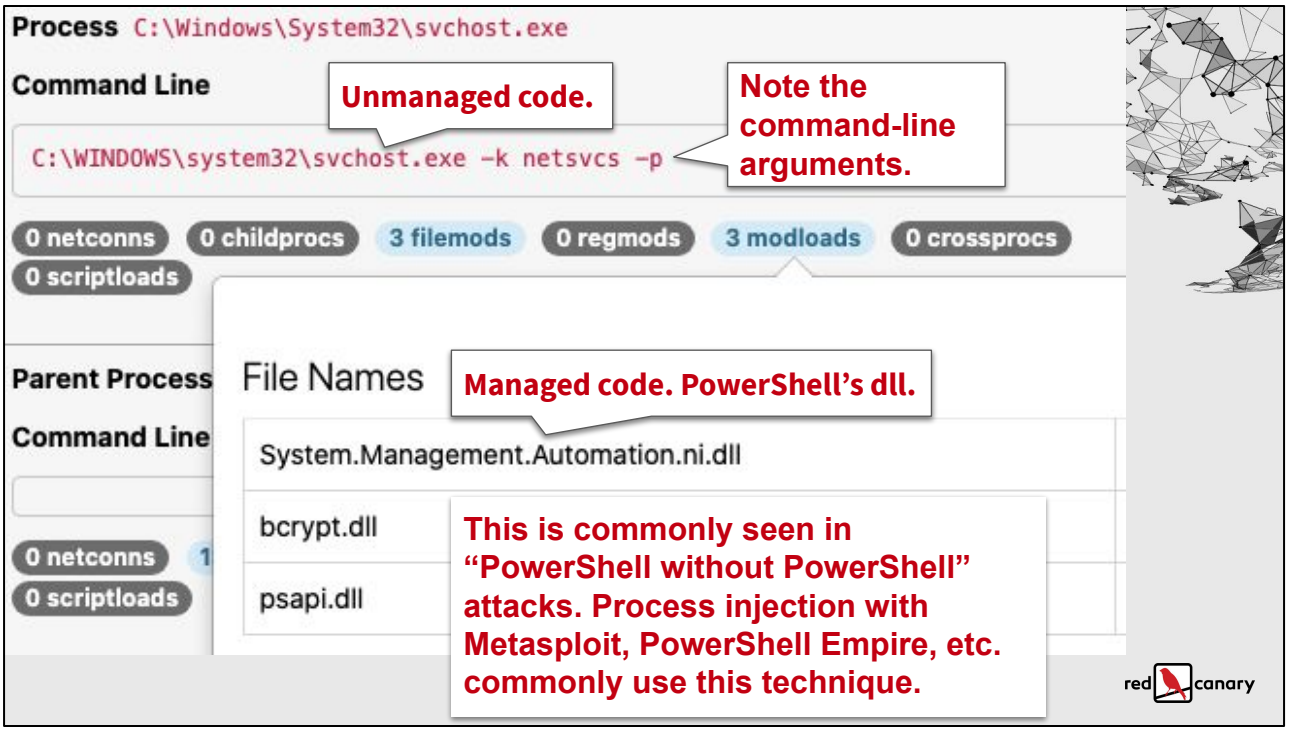


The event hit the Red Canary queue because svchost.exe apparently imported the System.Management.Automation.ni.dll. svchost.exe is unmanaged code, meaning it's written in C or C++ and interacts directly with the hardware. It does it's own memory management, etc.



System.Management.Automation.ni.dll is PowerShell's dll. It's the guts of PowerShell. It's managed code, meaning it runs in a virtual machine (think Java virtual machine, not VMware). The vm interacts with the hardware so the managed code doesn't have to do its own memory management, etc.

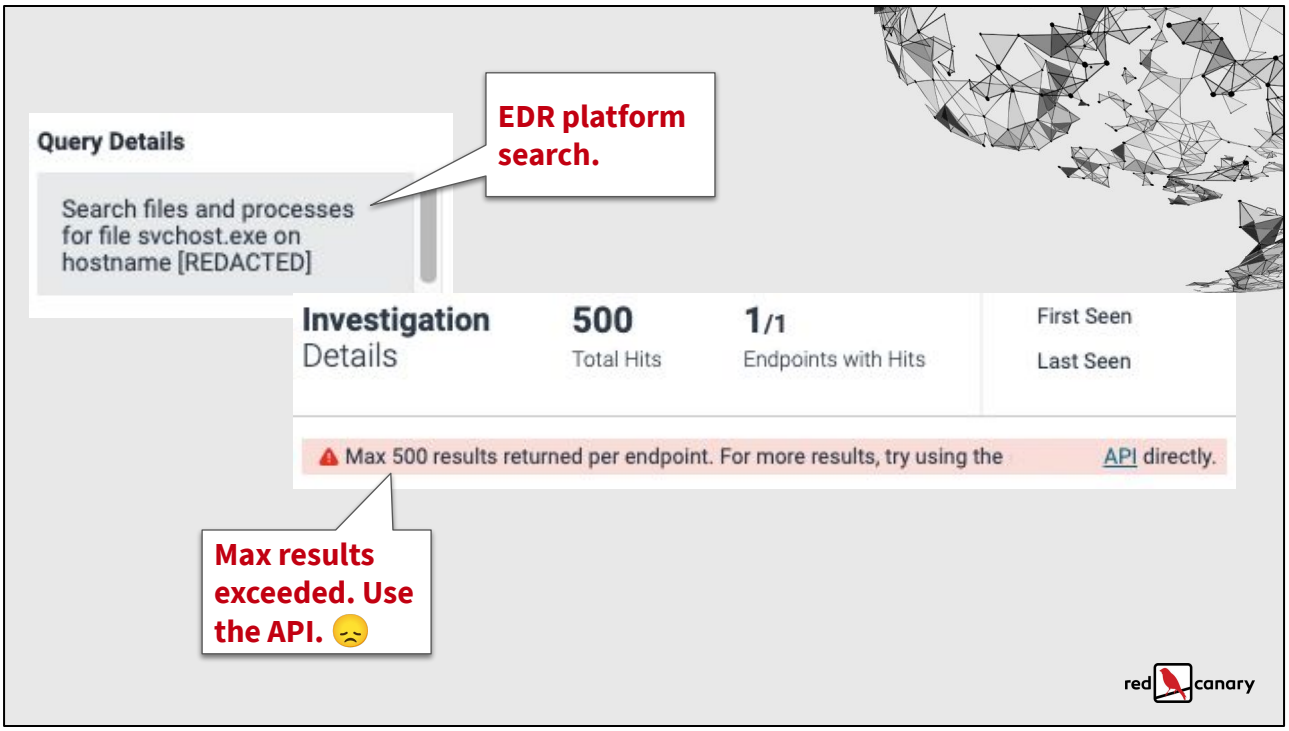
It's very unusual and generally a bad sign to see a process written in unmanaged code importing a managed code dll.



System.Management.Automation.ni.dll is PowerShell's dll. It's the guts of PowerShell. It's managed code, meaning it runs in a virtual machine (think Java virtual machine, not VMware). The vm interacts with the hardware so the managed code doesn't have to do its own memory management, etc.

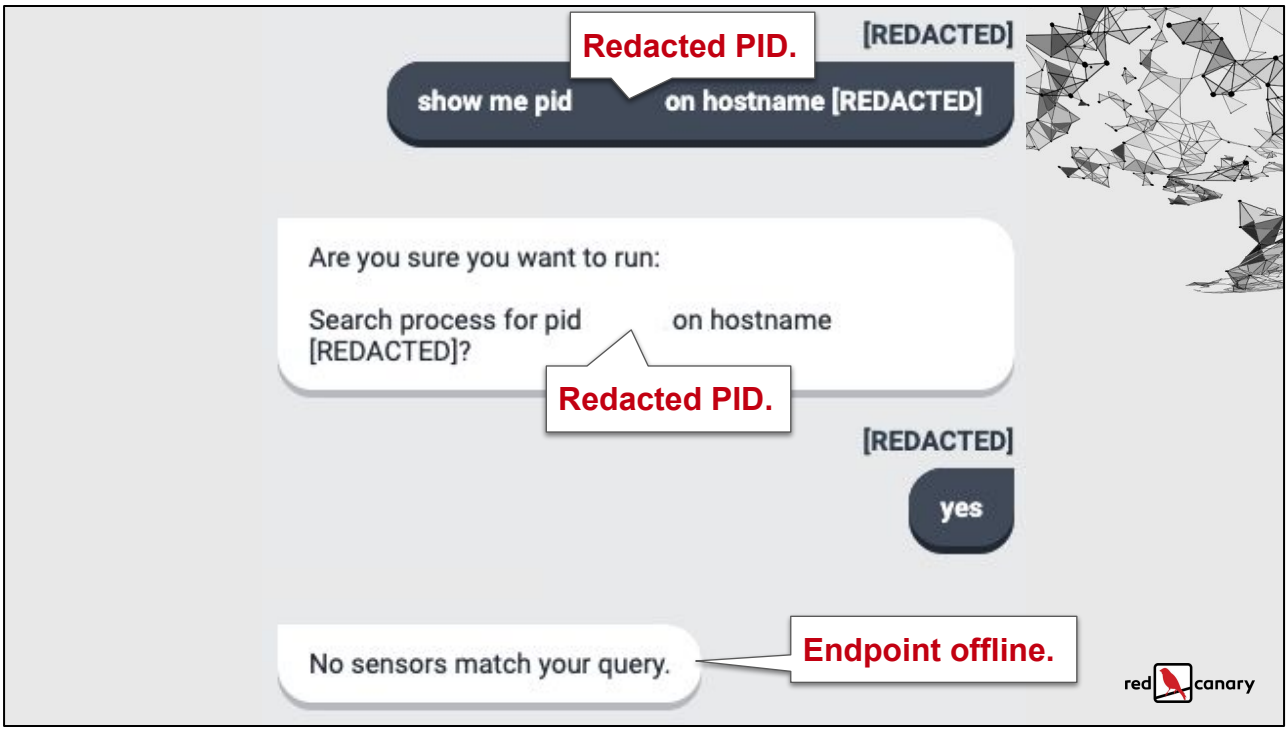
It's very unusual and generally a bad sign to see a process written in unmanaged code importing a managed code dll.

Here's a blog post I wrote during my funemployed summer of 2020 related to this topic:
https://trustedsignal.blogspot.com/2020/08/hunting-injected-processes-by-modules.html

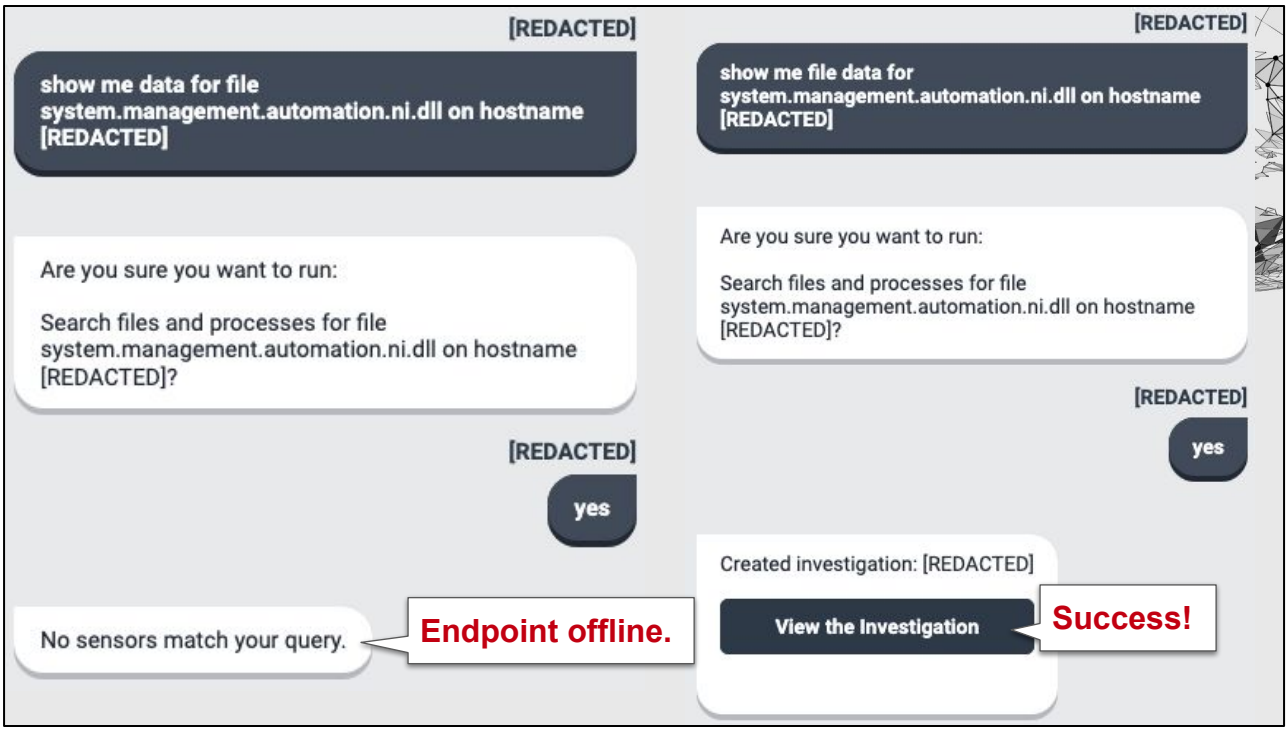


This particular platform has a natural language parser, meaning you can type in a query and it will parse it and if it can make sense of what you're asking, it will run a query and serve up the results. In this case, I looked for svchost.exe processes on the host in question. As you can see, there were more than 500 results, too many for the platform to return, so it helpfully recommends that I use the API.

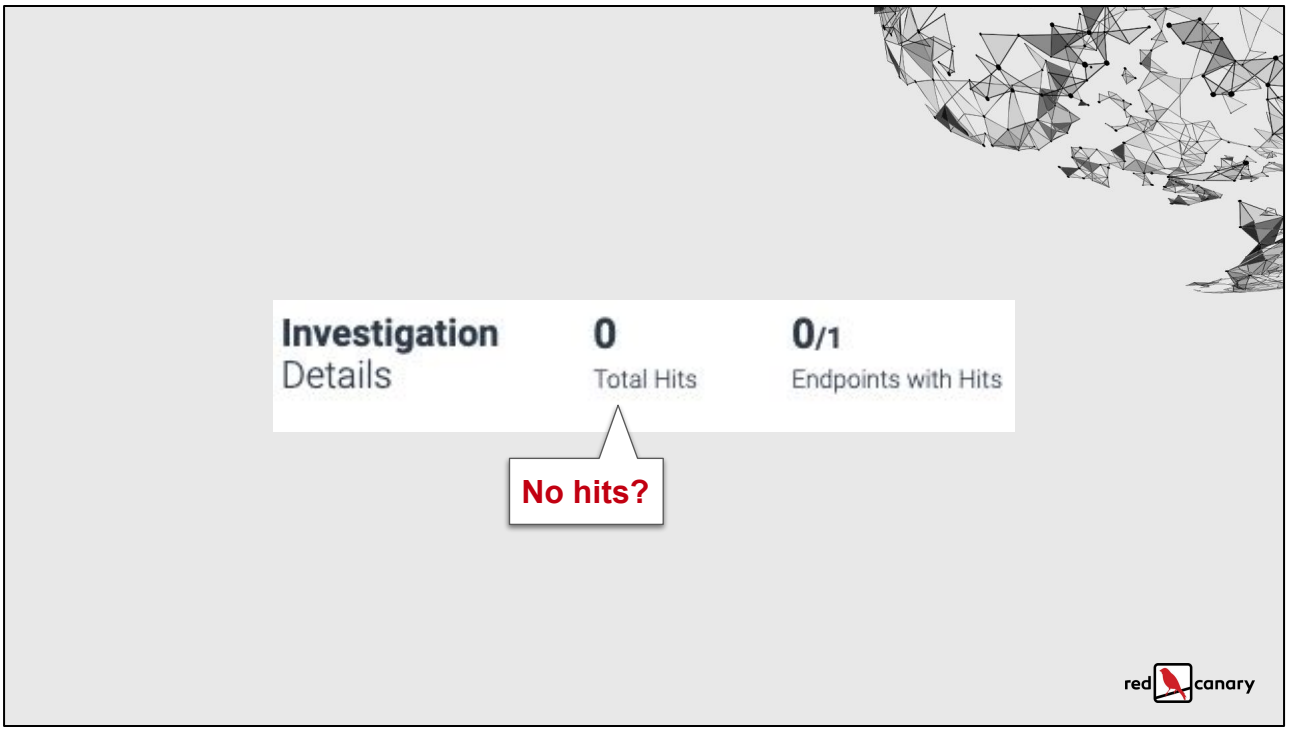
This platform has an incredibly powerful API, but for this presentation, I didn't have time to dig into it enough to use it for this query, so I tried another approach. I went back to the data we had in Red Canary and pulled out the pid of the specific svchost.exe process in our event. This should narrow the scope.



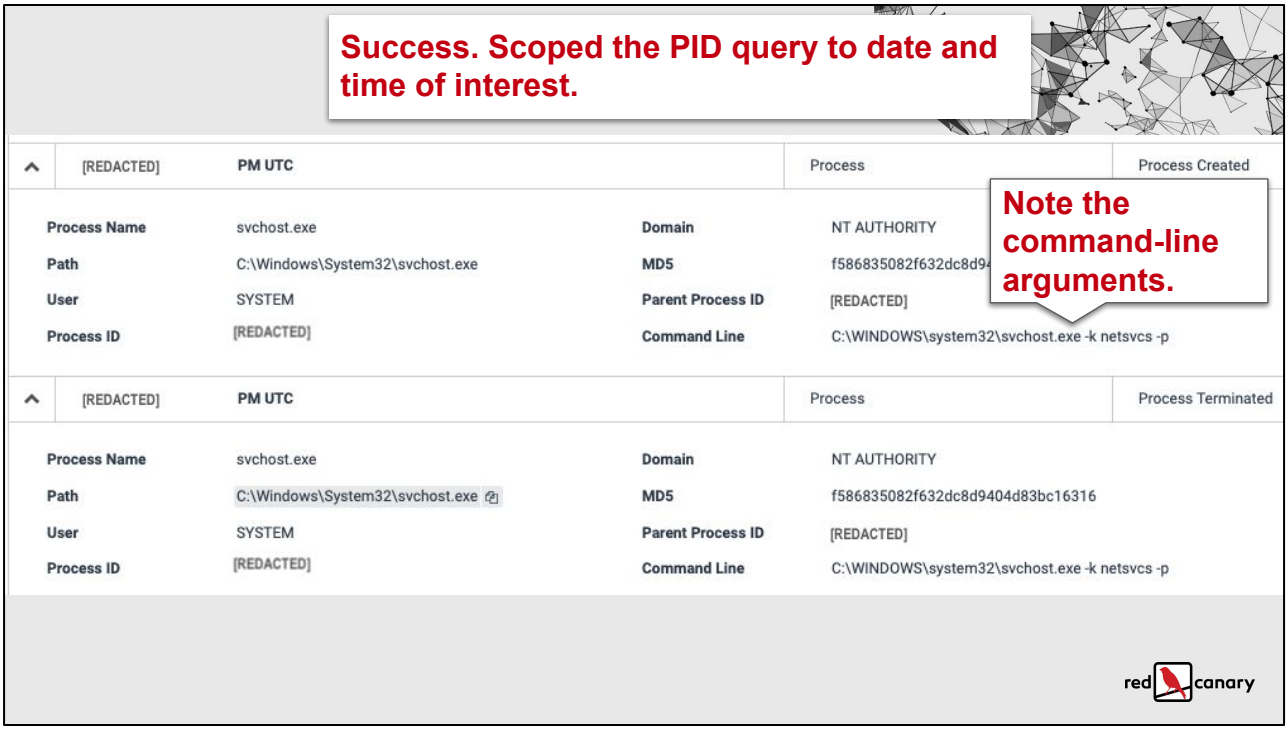
As you can see, my attempt to search on pid failed with “No sensors match your query,” indicating the endpoint in question was offline as this particular platform, at least in this configuration, does not centralize endpoint telemetry.



Undaunted by the device being offline, I kept trying other queries as the device may come back online at any moment. Finally one of my jobs went through and the platform responded with a link to investigation data.

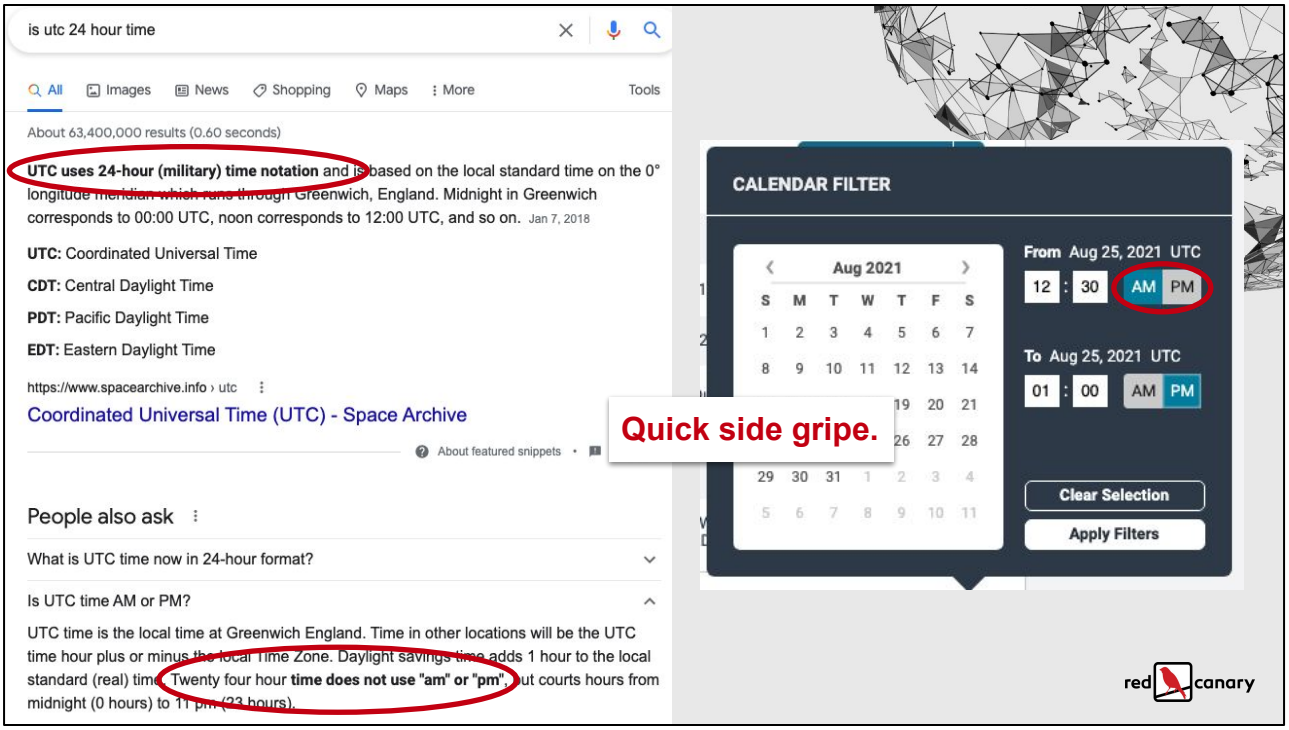

Investigation
Details
0
Total Hits
0/1
Endpoints with Hits
No hits?
red canary

No hits.



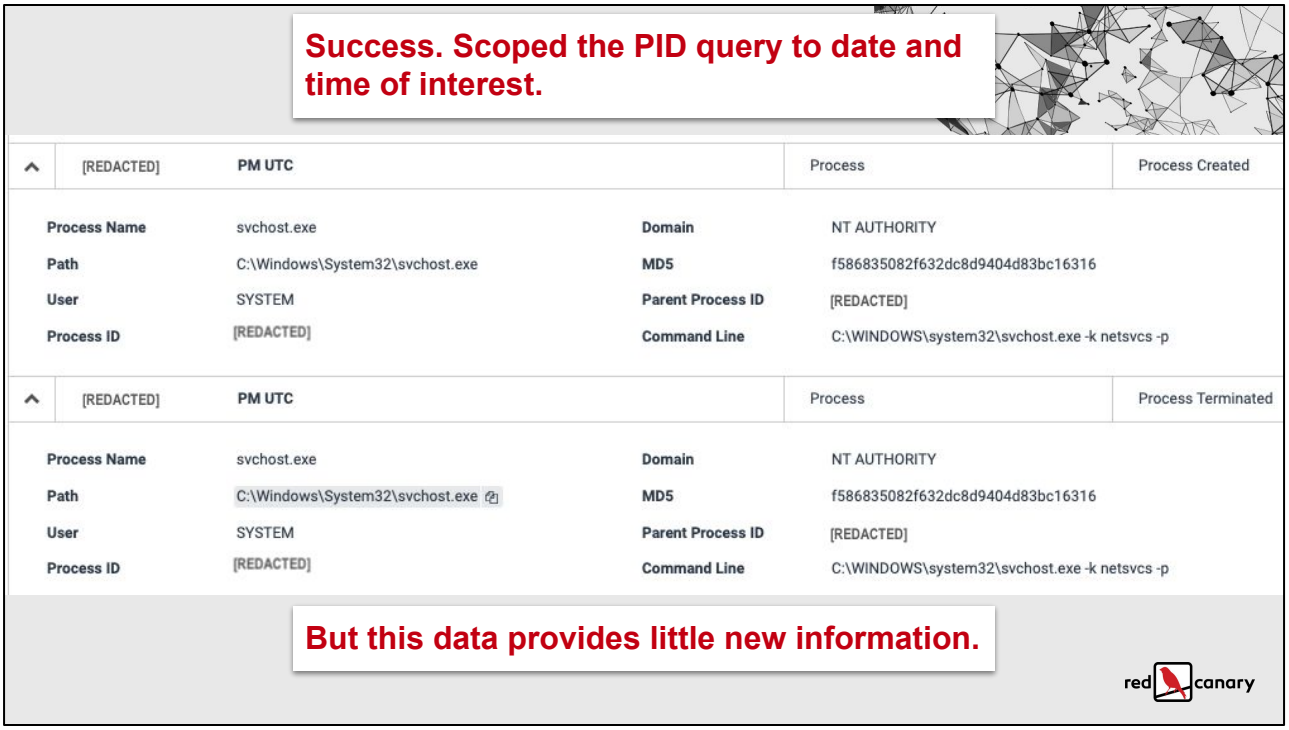
I did finally have some success. Persistence pays off. In the interface I was able to narrow the scope of my process ID search to a specific date and time.

Above you can see our svchost.exe process with the correct pid, the command-line matches what we saw in the detection portal and we can see the termination event for the process, it was quite short-lived. There's no way through the web UI that I could find to dig up the modloads for this process. Reading through the API documentation, modloads are recorded and they are passed to Red Canary's detection engine, so they must be available via the API.

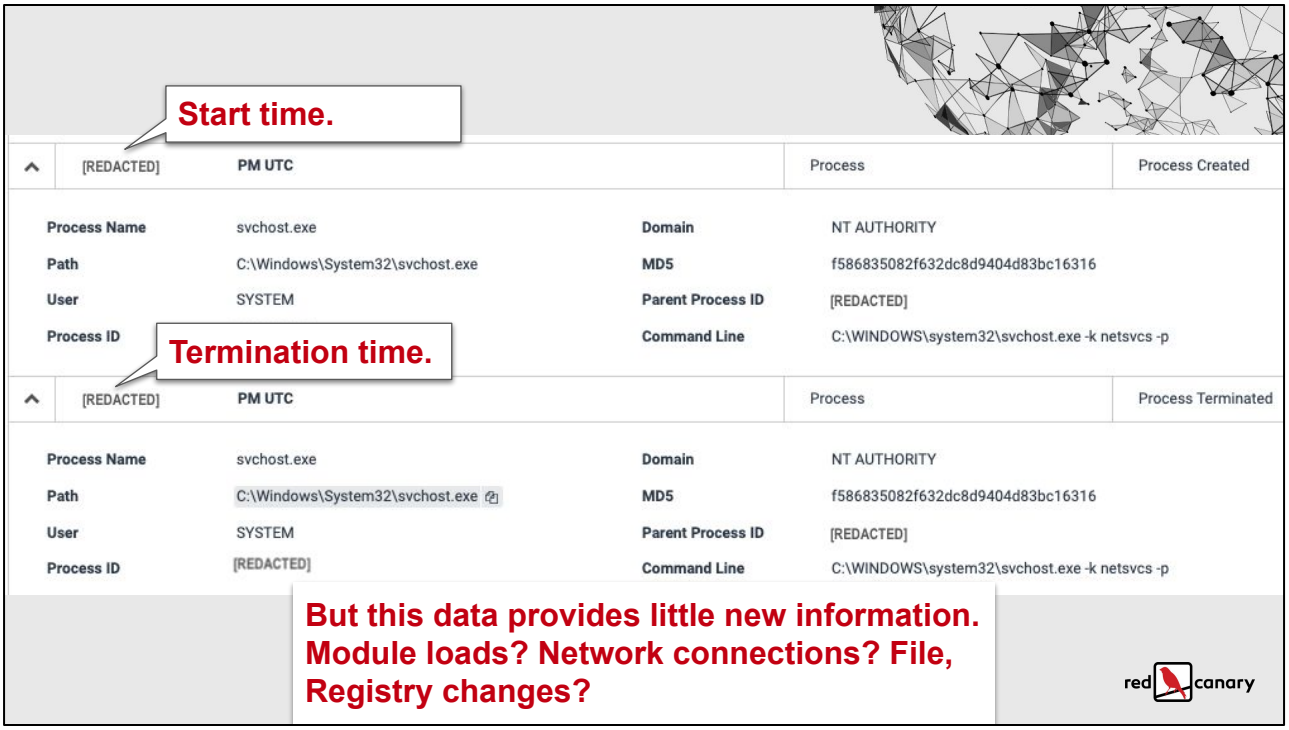


A quick side gripe… The interface that let me narrow down the scope of my search had the date/time picker that you see here. Note that all times in the interface are reportedly UTC, except if you do a quick Google search about UTC, you'll see it specifically uses 24-hour time notation and does not use "am" or "pm".

Why would I mention such a minor thing? Because when you're dealing with a security incident, time is of the essence and there are many different demands on the analyst's attention, cognitively load is high. If data is stored in different systems using different time formats or if they are both using UTC, but one is using a broken UTC implementation that requires the analyst to convert from am to pm, it's an **unnecessary cognitive load** or task for the analyst to deal with.

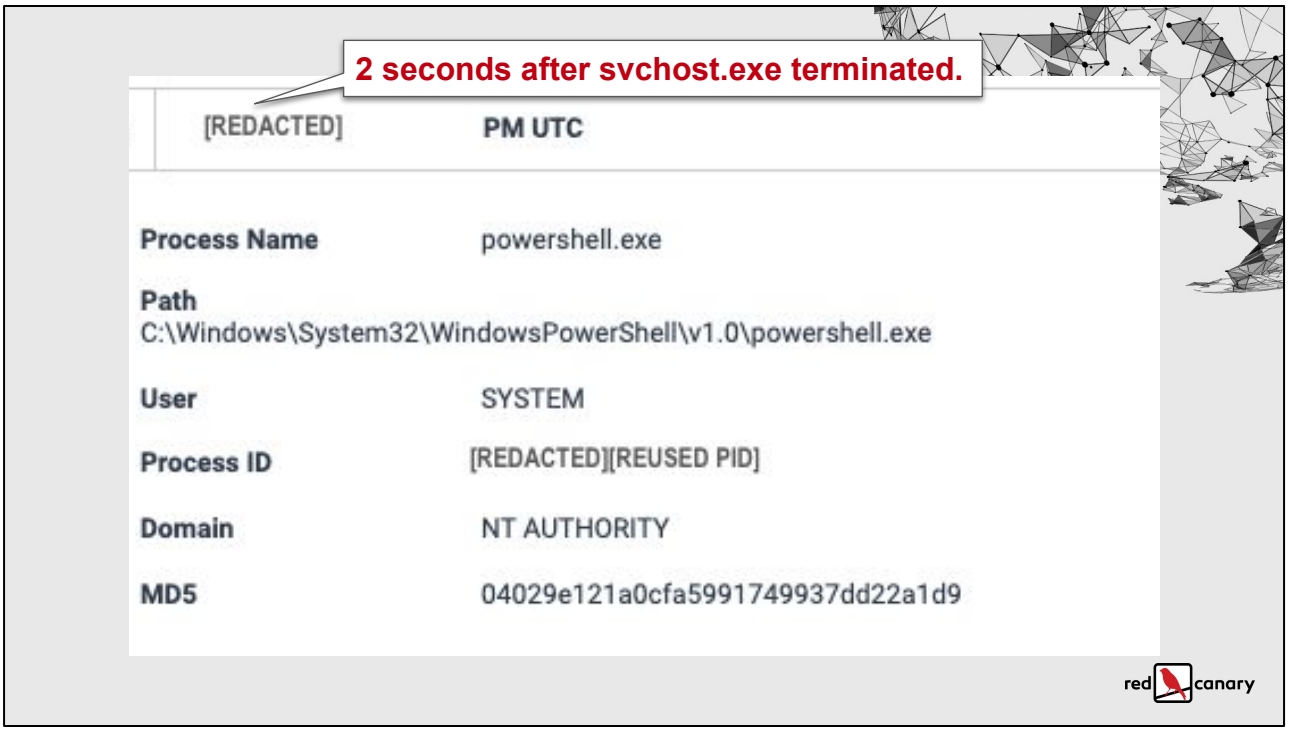


Above you can see our svchost.exe process with the correct pid, the command-line matches what we saw in the detection portal and we can see the termination event for the process, it was quite short-lived. There's no way through the web UI that I could find to dig up the modloads for this process. Reading through the API documentation, modloads are recorded and they are passed to Red Canary's detection engine, so they must be available via the API.

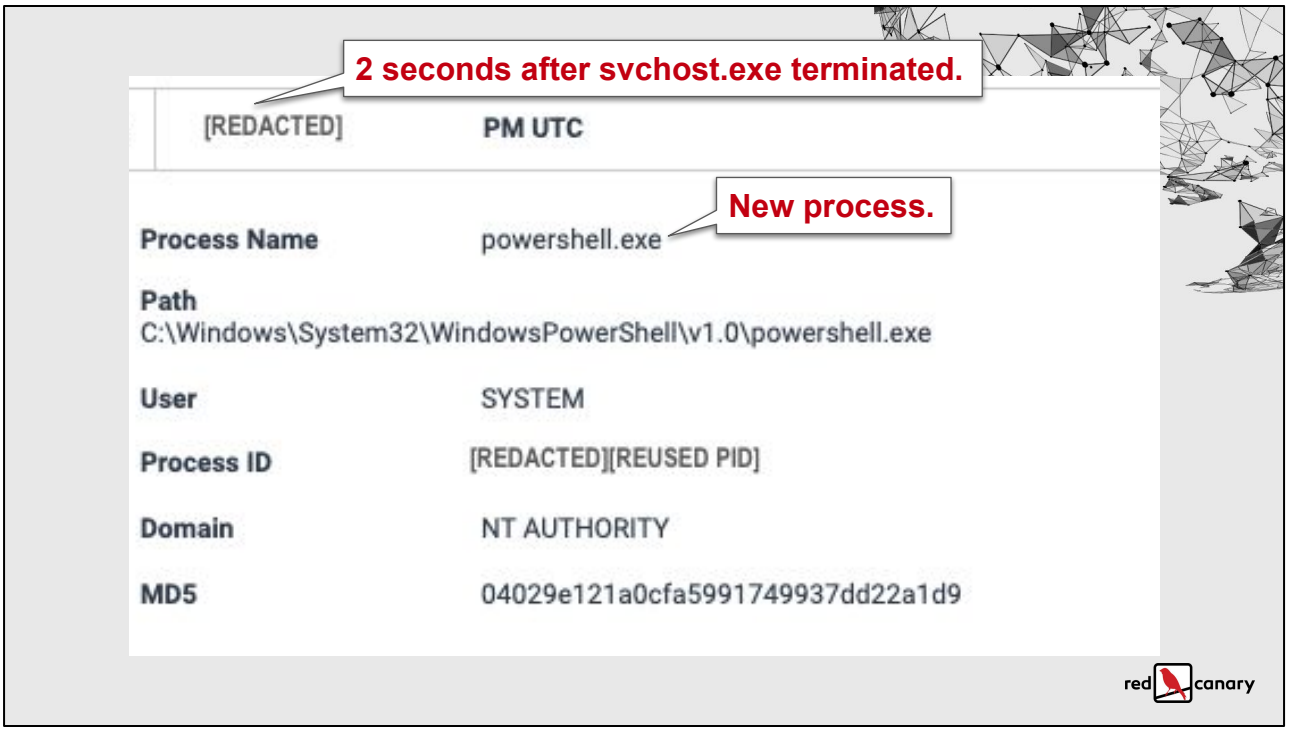


I did finally have some success. Persistence pays off. In the interface I was able to narrow the scope of my initial search to a specific date and time.

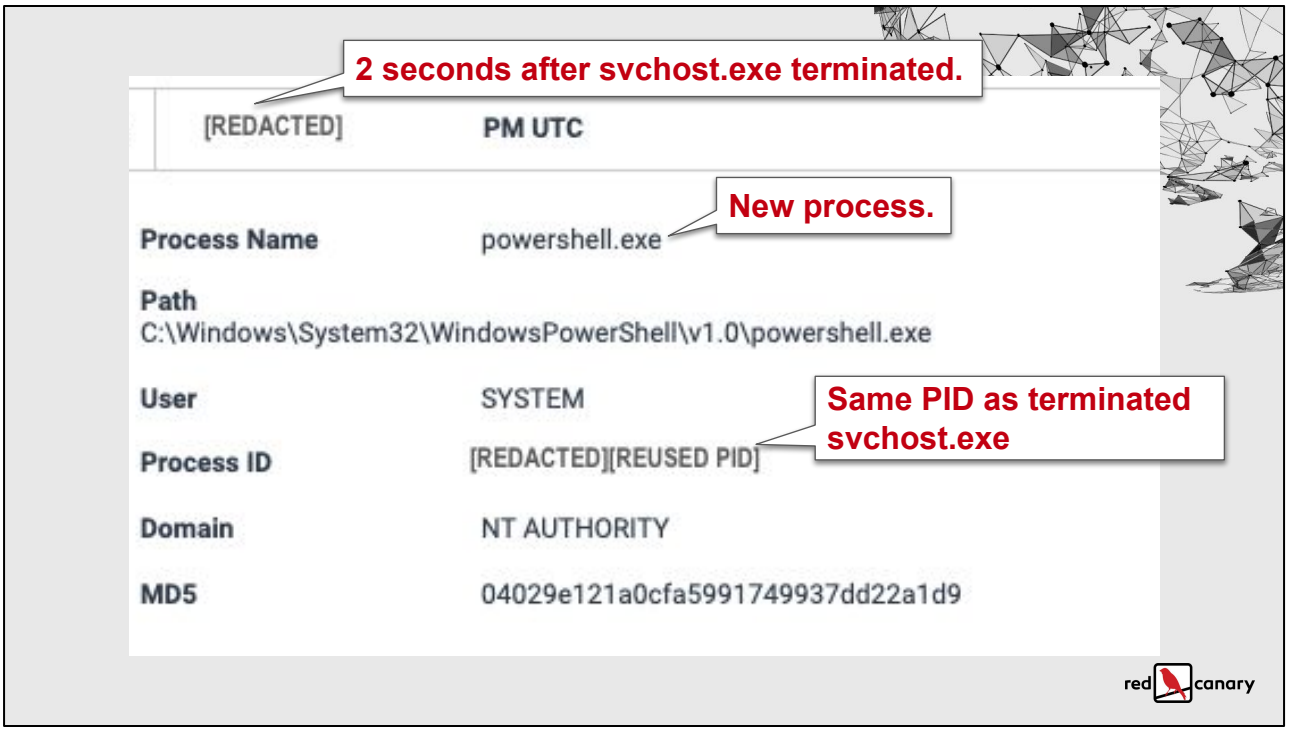
Above you can see our svchost.exe process with the correct pid, the command-line matches what we saw in the detection portal and we can see the termination event for the process, it was quite short-lived. There's no way through the web UI that I could find to dig up the modloads for this process. Reading through the API documentation, modloads are recorded and they are passed to Red Canary's detection engine, so they must be available via the API.



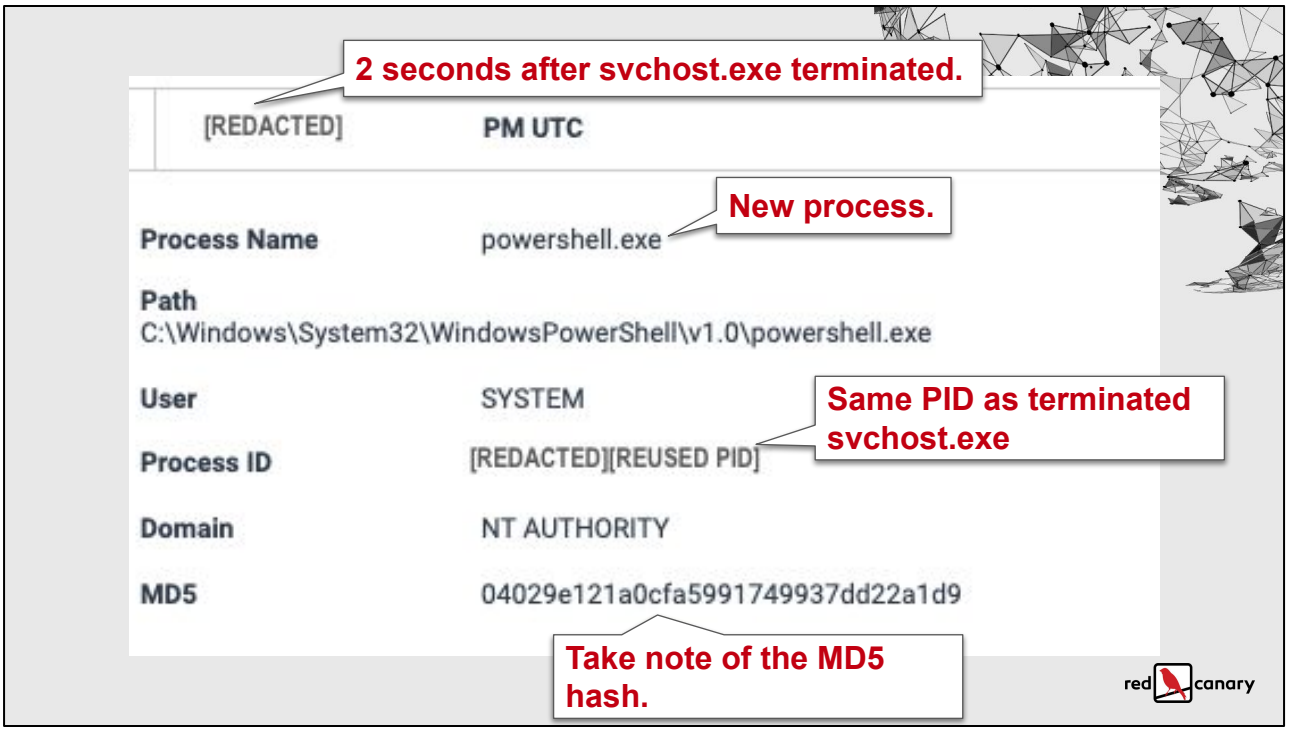
Recall that the data we're looking at here was returned for the process ID of interest.



And this new process that started two seconds after our svchost.exe process exited, is PowerShell.exe. I've redacted the command-line arguments from the screen, but this PowerShell process was executing a script.

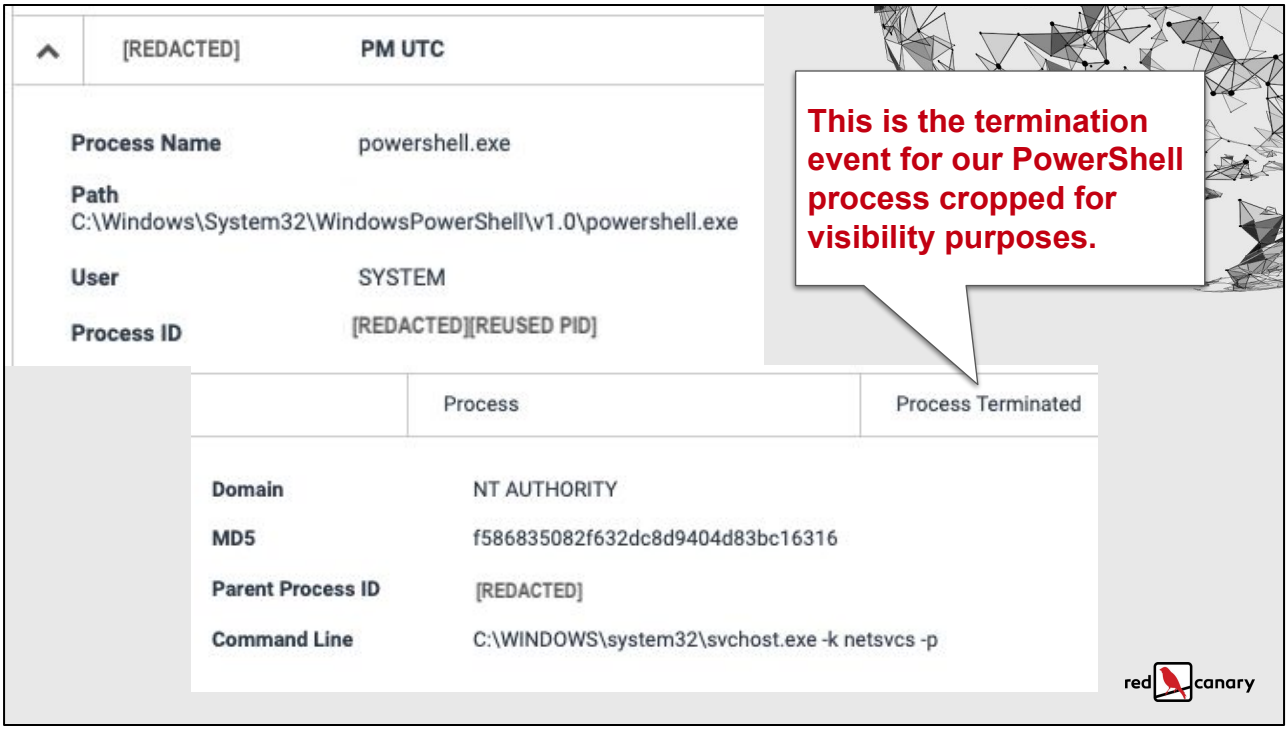


Again, this search was done by process ID, so it makes sense that the redacted process ID here matches the process ID of svchost.exe, but you'll have to take my word for it, since it's redacted.



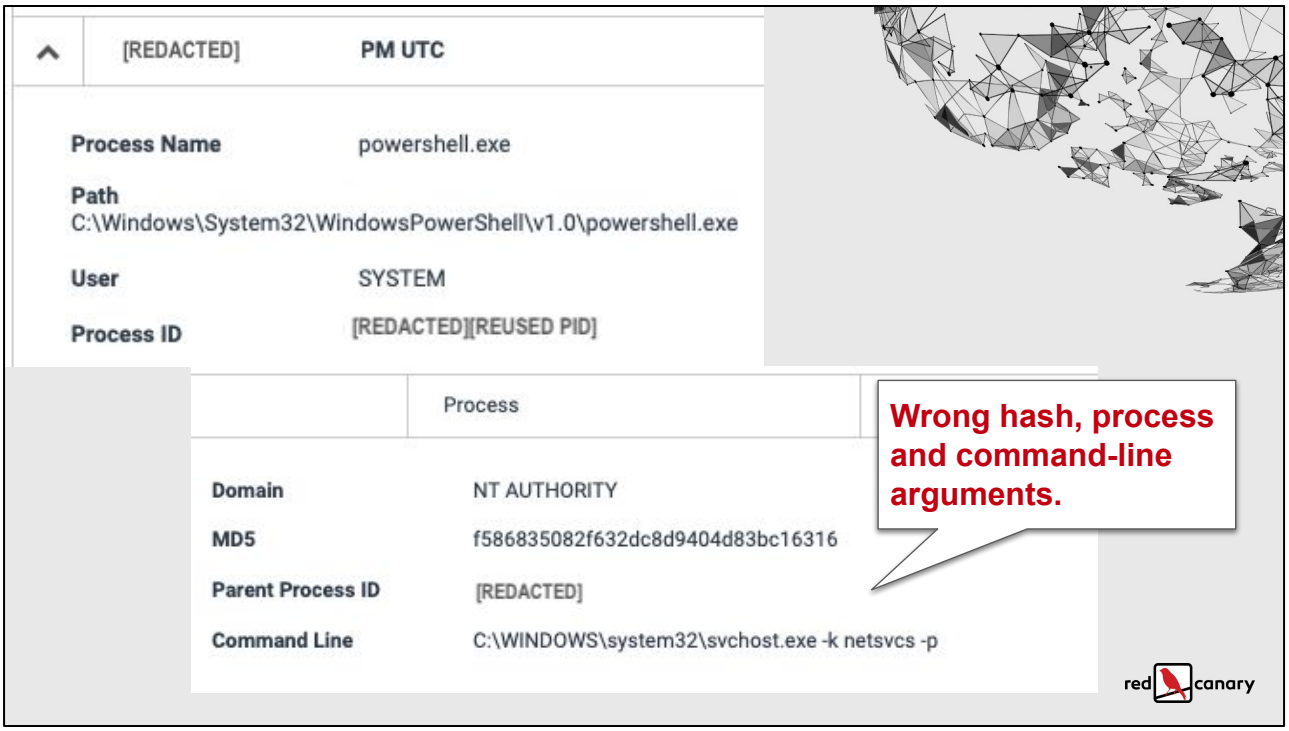
Again, this search was done by process ID, so it makes sense that the redacted process ID here matches the process ID of svchost.exe, but you'll have to take my word for it, since it's redacted.

Note the MD5 hash, which corresponds to a version of PowerShell.exe.



Here's the process termination event according to this EDR platform. **Note the command-line?!**

To be fair, I reached out to a friend of mine who is well know for her memory analysis work and I asked her if she'd ever seen any strange cross-contamination of process memory data structures in Windows systems where pids were reused in a short period of time. While she didn't confirm that she'd seen something like this, she didn't rule it out entirely either. So this may not be a bug with the platform, it's possible it's an artifact of Windows not overwriting this data somewhere. Honestly, I don't know, but if the EDR platform or the operating system can't keep things like this straight, the analyst needs to remain vigilant.



Here's the process termination event according to this EDR platform. **Note the command-line?!**

To be fair, I reached out to a friend of mine who is well know for her memory analysis work and I asked her if she'd ever seen any strange cross-contamination of process memory data structures in Windows systems where pids were reused in a short period of time. While she didn't confirm that she'd seen something like this, she didn't rule it out entirely either. So this may not be a bug with the platform, it's possible it's an artifact of Windows not overwriting this data somewhere. Honestly, I don't know, but if the EDR platform or the operating system can't keep things like this straight, the analyst needs to remain vigilant.

Getting bogged down

Plenty of data, not enough information

Does this forest have any trees?

red canary

Now let’s look at another EDR platform that’s going to provide us with lots of data, but not enough information.

Let's go back to the detection from the start of this talk. Above we can see that the Console Windows Host (conhost.exe) is a child process of explorer.exe, which in and of itself is unusual. Generally conhost is a child of cmd, powershell or other Microsoft Windows console processes. Furthermore, conhost.exe nearly always runs with command-line arguments, but in this case there are none. Next, our conhost.exe process has spawned an instance of cmd.exe, the Windows Command Shell.

If you received an alert like this, what additional data would you want and why?
Pause for audience response, prompt again as necessary.

Examples:

- Modloads?
- Netconns?
- Crossprocs?
- Childprocs?
- Regmods?
- Memory dump?
- Packet capture?

Quick quiz

svchost.exe spawned explorer.exe

- Common or uncommon?

red canary

Quick quiz

svchost.exe spawned explorer.exe

- Common or uncommon?
- Uncommon.

red canary

Quick quiz

svchost.exe spawned explorer.exe

- Common or uncommon?
- Uncommon.

explorer.exe spawned conhost.exe

- Common or uncommon?

red canary

Quick quiz

svchost.exe spawned explorer.exe

- Common or uncommon?
- Uncommon.

explorer.exe spawned conhost.exe

- Common or uncommon?
- Uncommon.

red canary

Quick quiz

svchost.exe spawned explorer.exe

- Common or uncommon?
- Uncommon.

explorer.exe spawned conhost.exe

- Common or uncommon?
- Uncommon.

conhost.exe spawned cmd.exe

- Common or uncommon?

red canary

Quick quiz

svchost.exe spawned explorer.exe

- Common or uncommon?
- Uncommon.

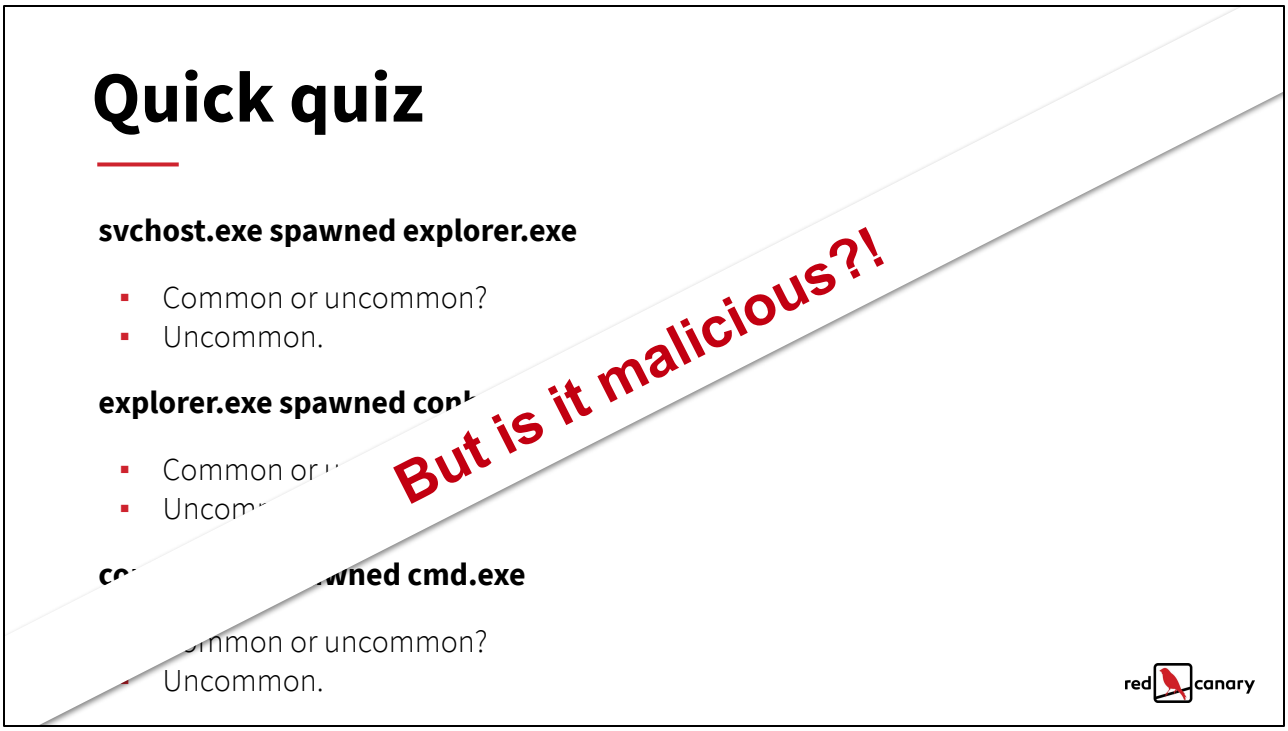
explorer.exe spawned conhost.exe

- Common or uncommon?
- Uncommon.

conhost.exe spawned cmd.exe

- Common or uncommon?
- Uncommon.

red canary



Yes, all of these things are uncommon, but is the overall behavior malicious?

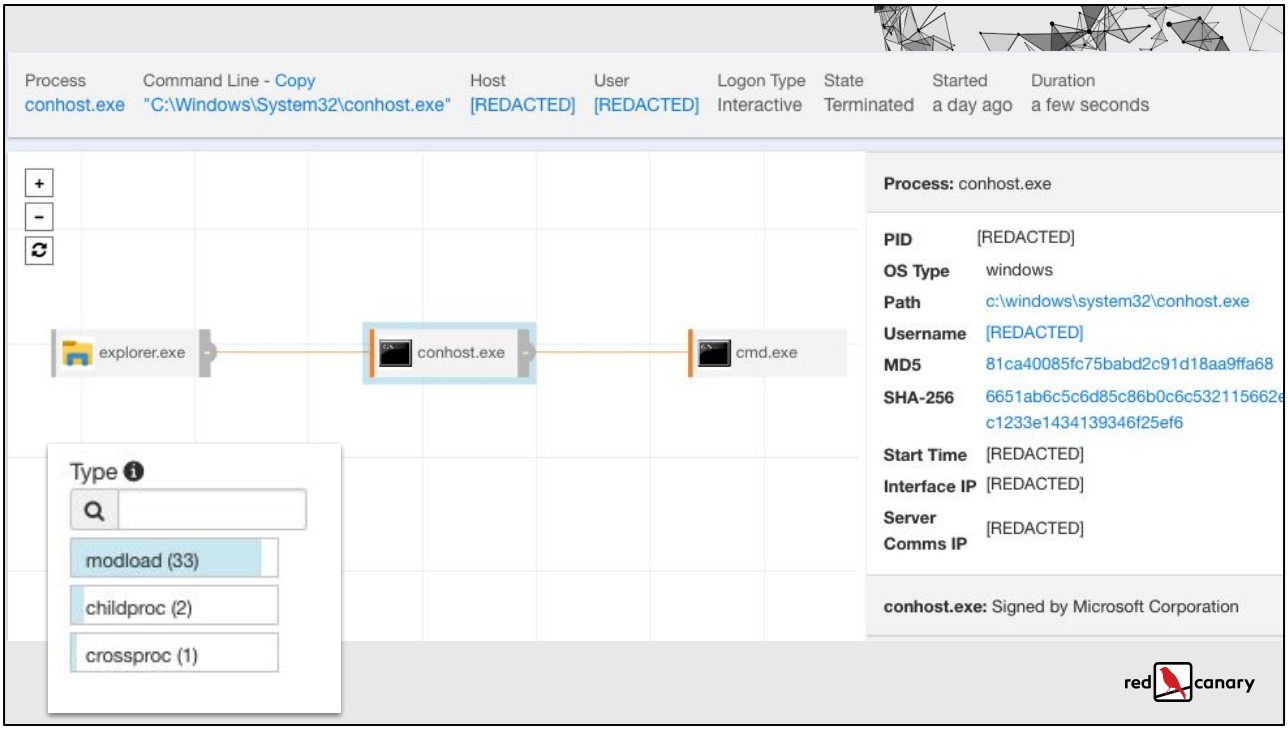
At this point we have more speculation than concrete evidence, maybe looking at the child and cross process events will help.

Here's a bit more detail from the EDR platform that recorded this event.

Notable new information:

- PID (redacted),
- process start time (redacted),
- Signer,

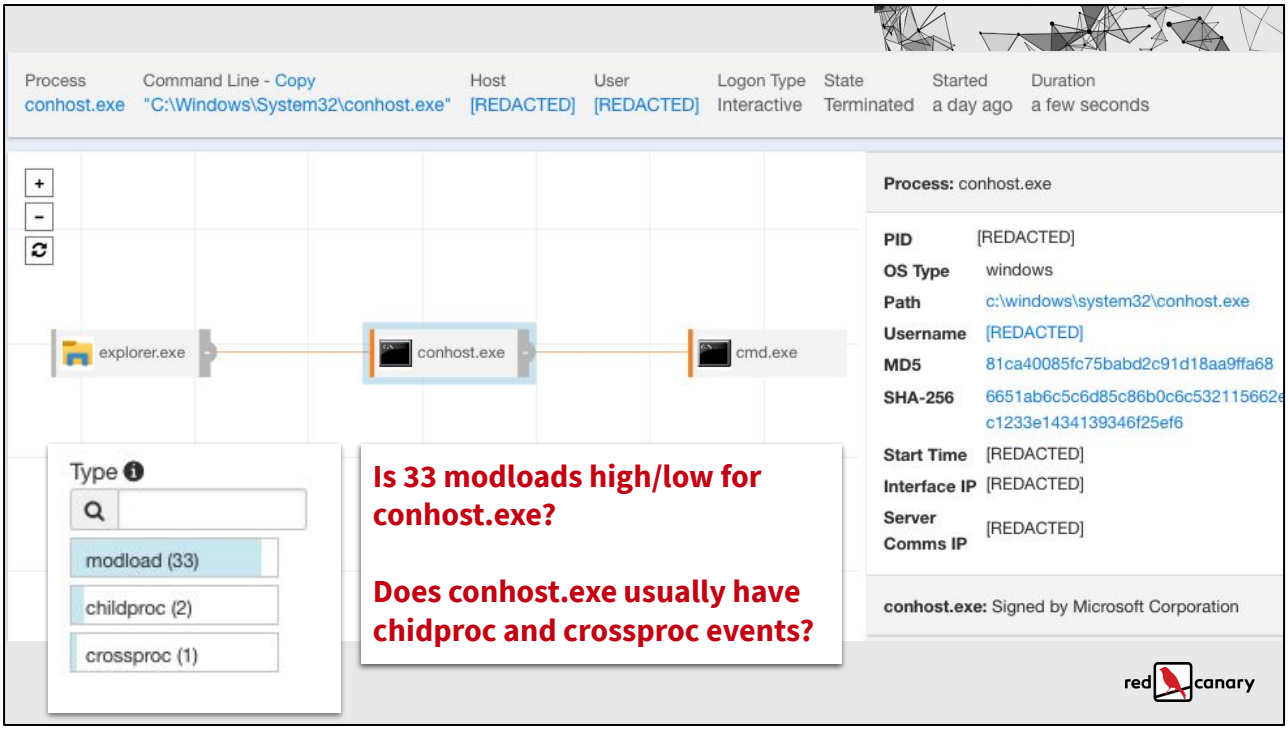
What else do we want to know?



This EDR platform will give us more details:

- Modloads,
- Childprocs,
- Crossprocs

What else do we want to know?



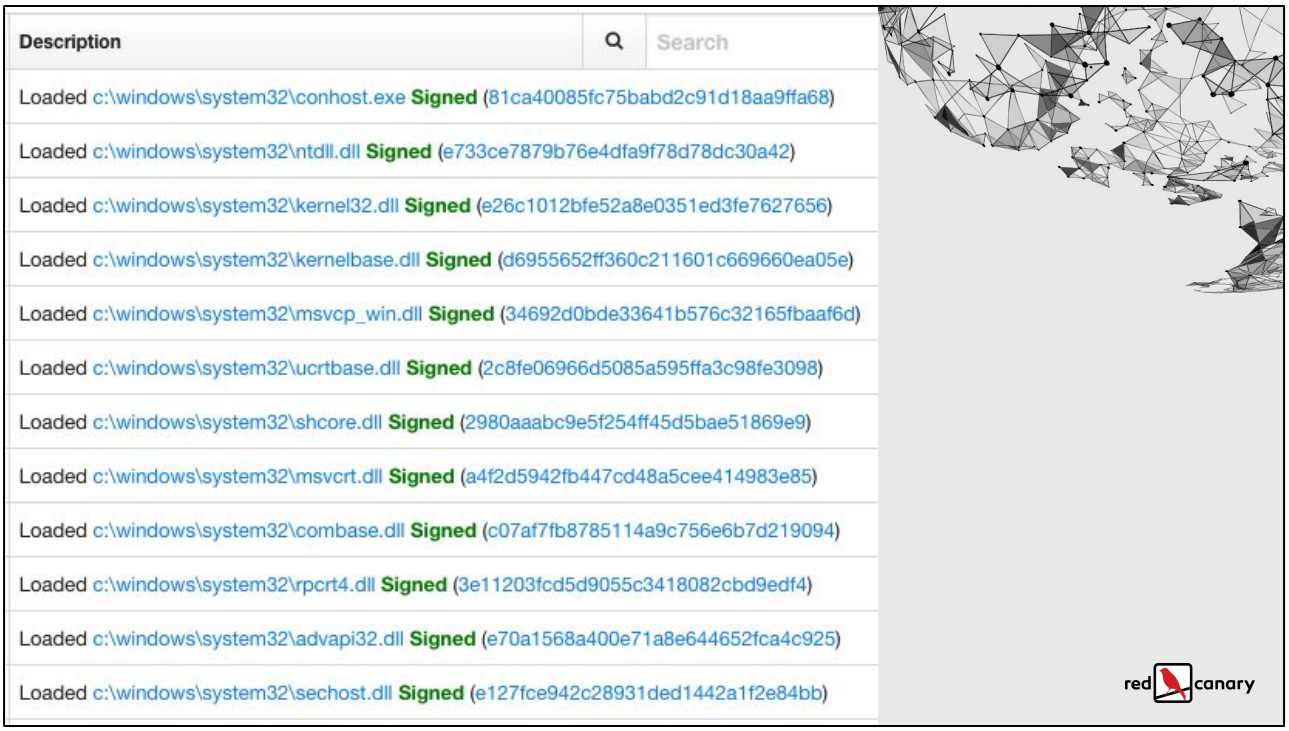
How helpful is this information?

Is 33 modloads normal for conhost.exe?

Does conhost.exe usually have childproc and crossproc events?

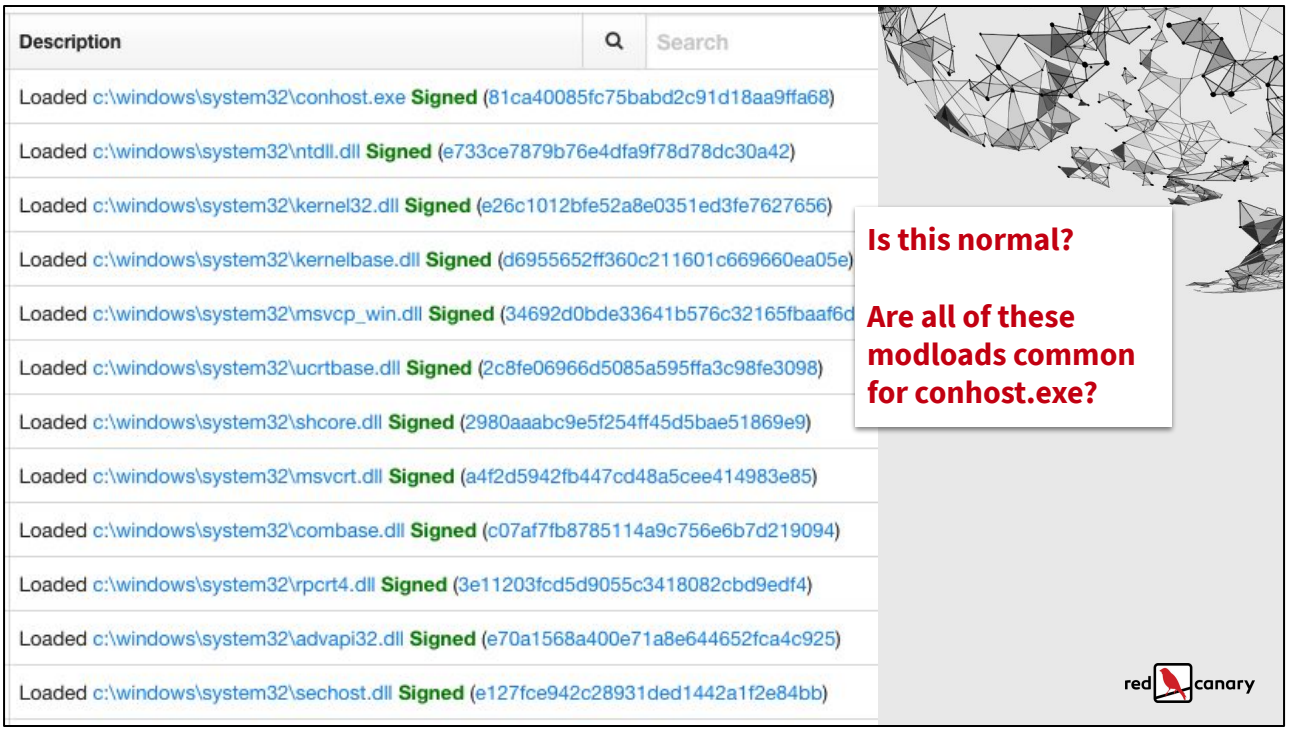
Is there a term for the kinds of information we're after here?

This platform will let us pivot into the module loads. Let's do that.

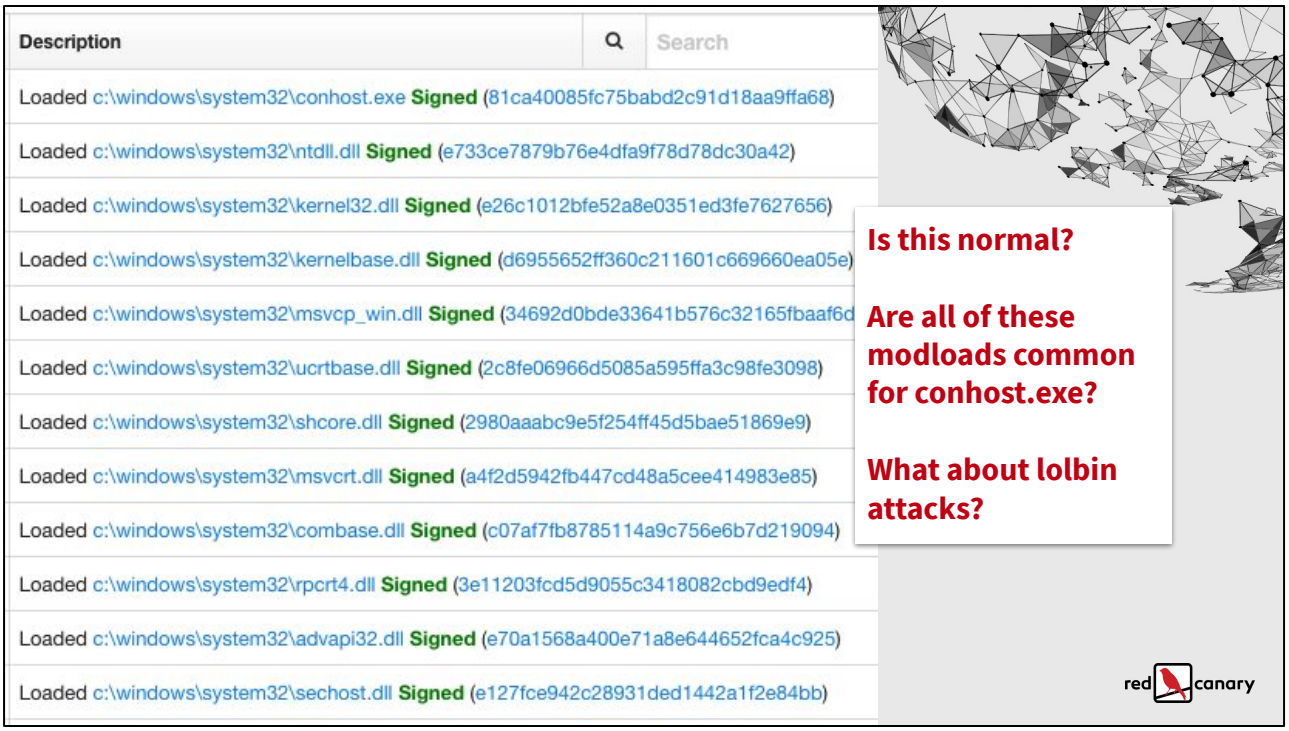


Here's a sampling of the 33 modload events for our conhost.exe process?

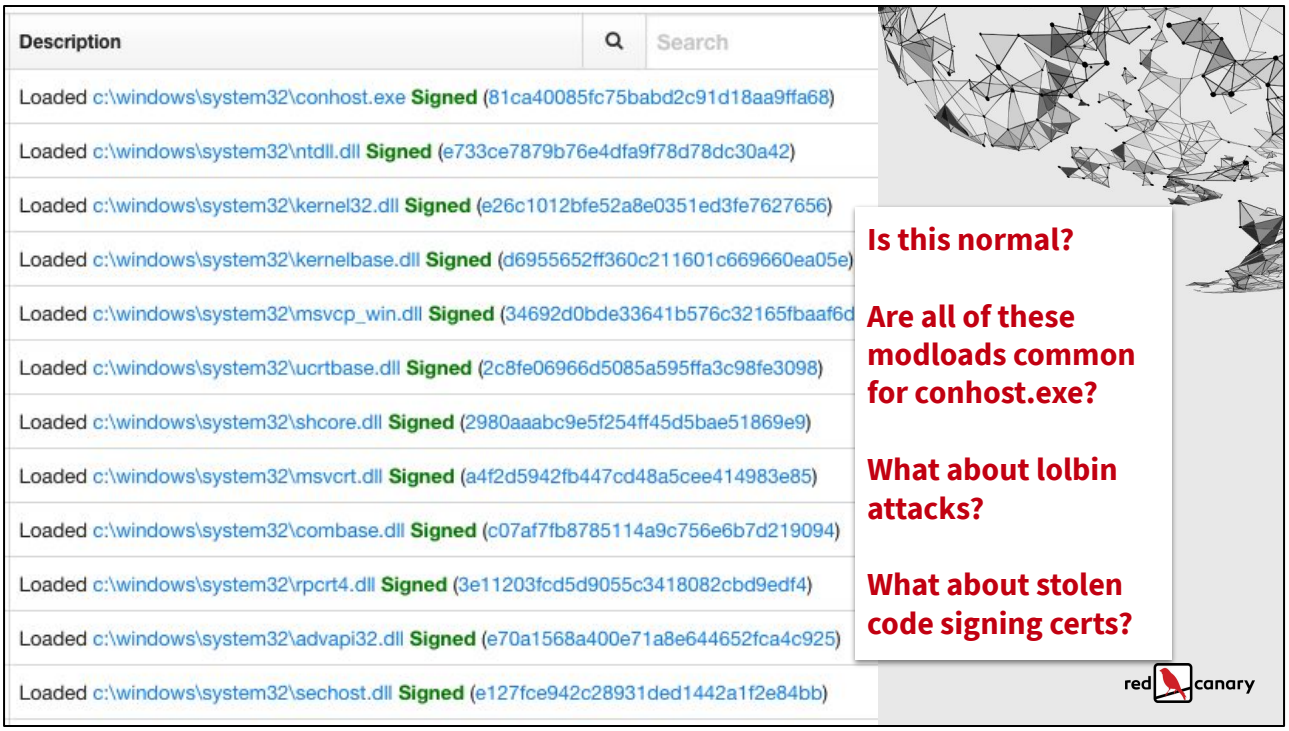
All of them show they are signed, so that's good, right?



A review of all 33 modloads in this EDR platform shows that they are all signed, but what we don't know is they are all commonly loaded into conhost.



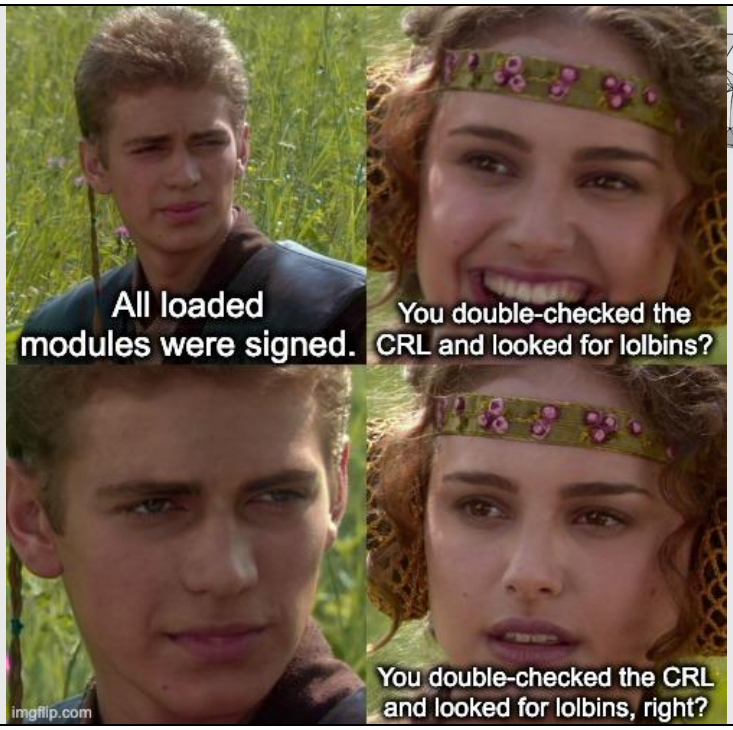
Given living off the land attacks (PowerShell without PowerShell), we can't say with certainty.

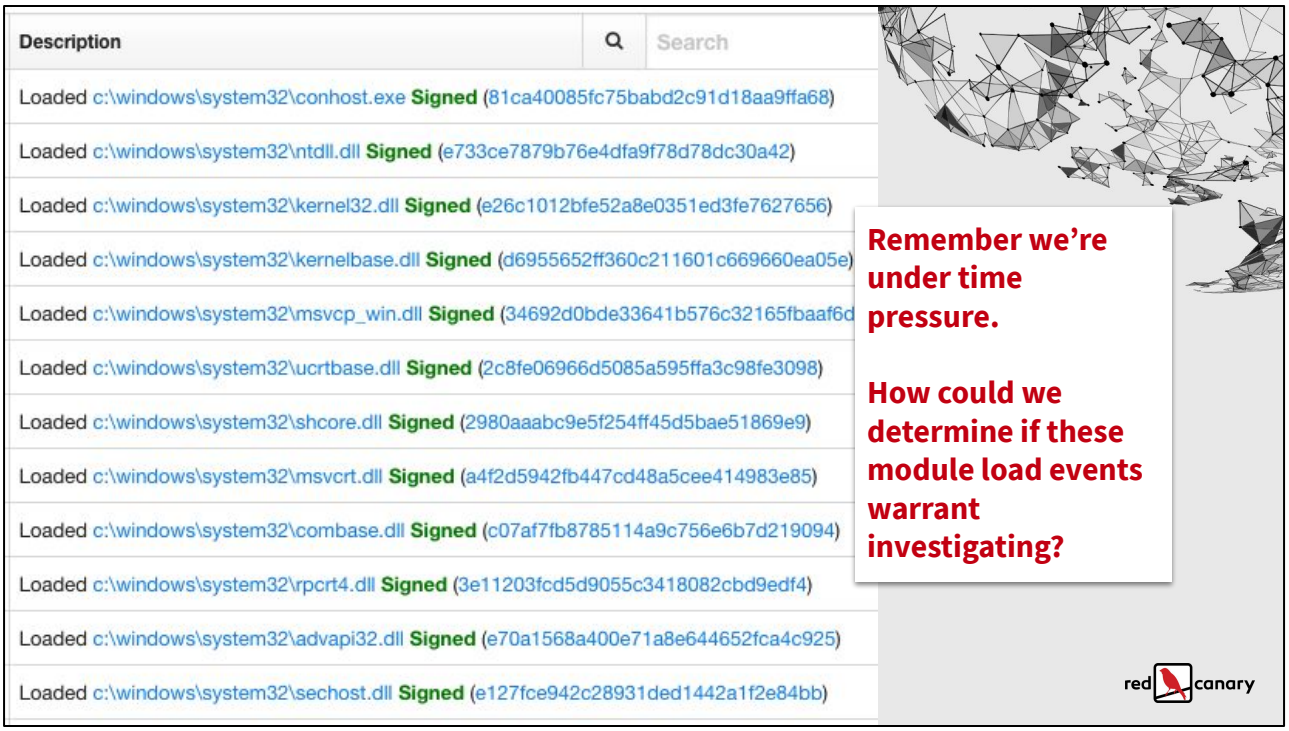


And code signing certificates have been stolen and used nefariously.

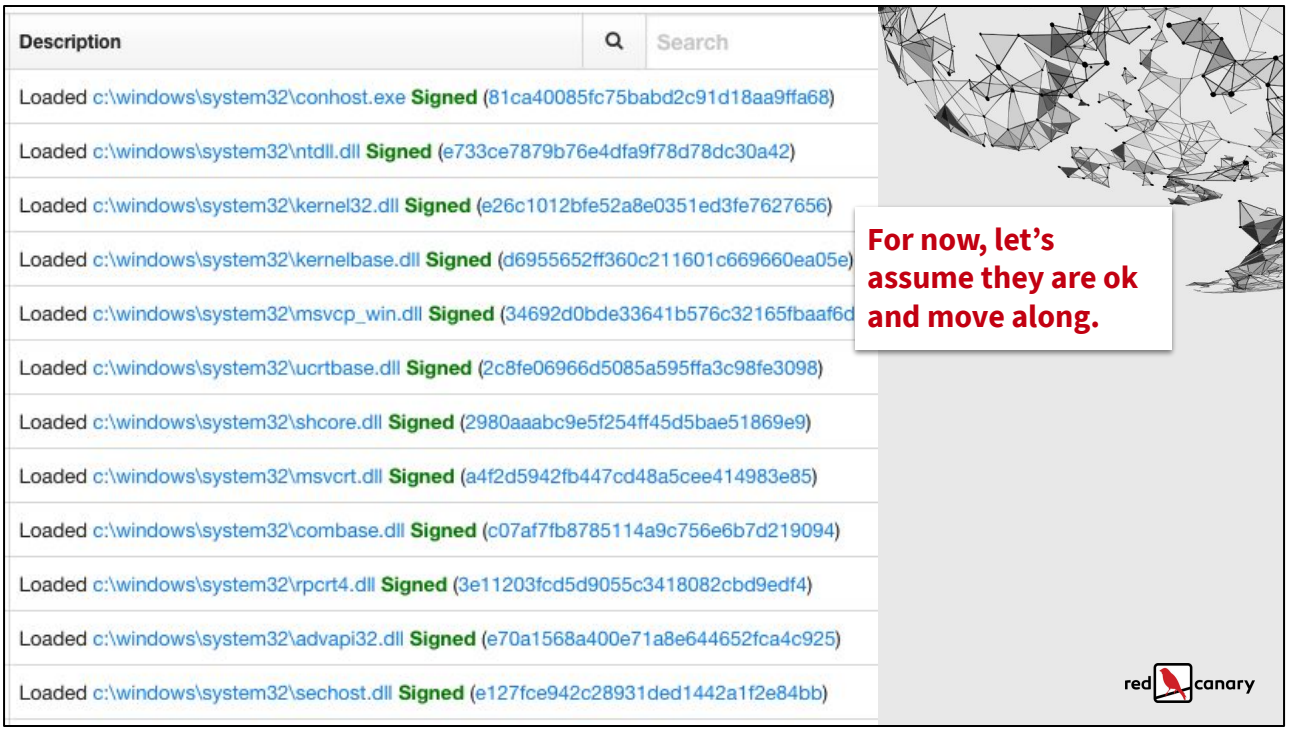
Do we need to go down the rabbit hole of verifying every signing certificate?

Would it be helpful if the EDR platform flagged uncommon module loads?

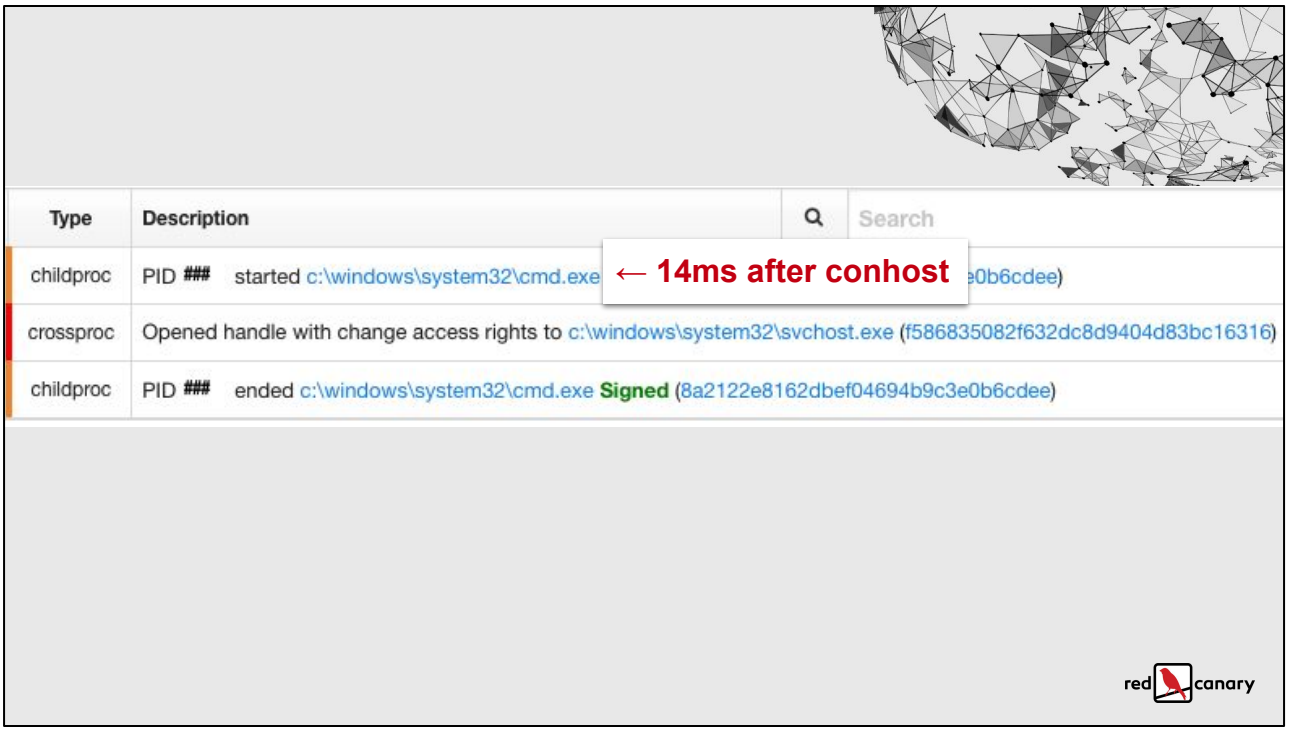

All loaded modules were signed.
You double-checked the CRL and looked for lolbins?
You double-checked the CRL and looked for lolbins, right?
imgflip.com



As users of this platform, performing this analysis isn't easy, but the platform itself has all the data and could run the calculations for us, even if it was a nightly batch job and the data was updated instantly, it would still be valuable to know what the **base rates** are for how often a given process loads a given module.

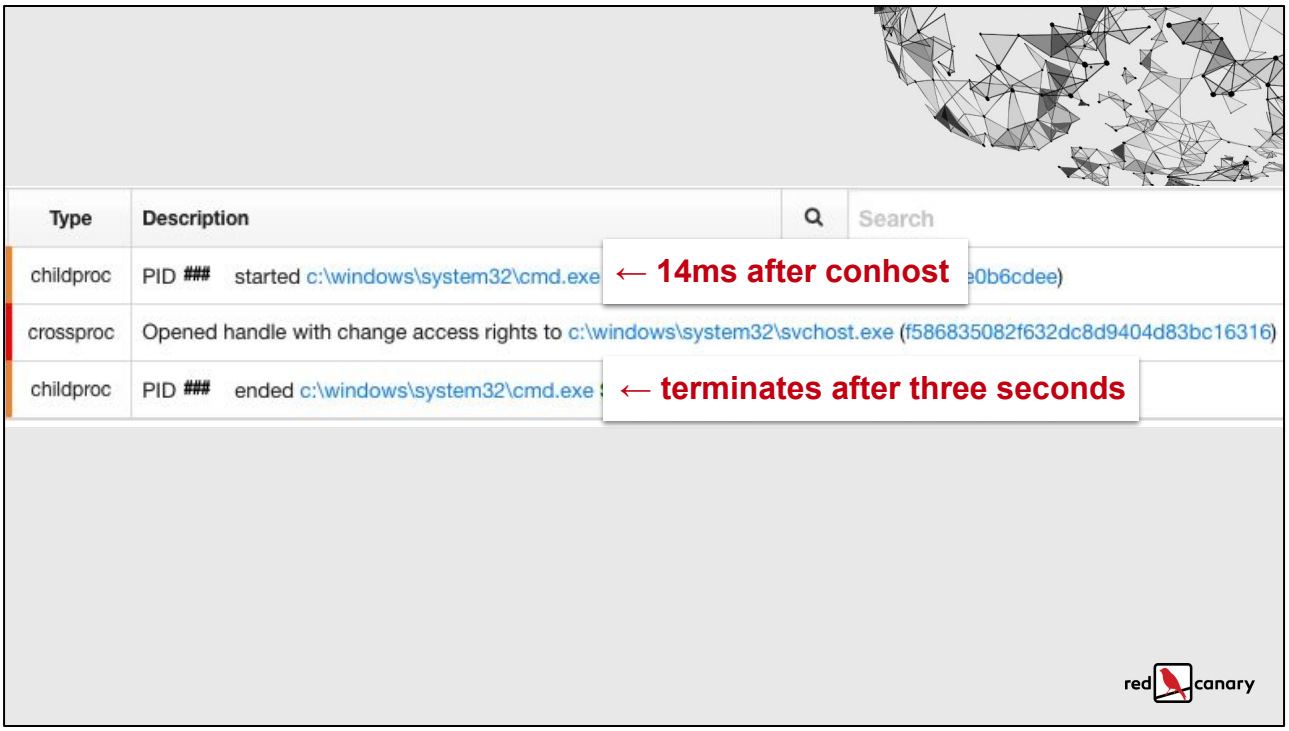


For the sake of time, we'll skip the rabbit hole and move along, but it is worth thinking about how we could quickly determine whether or not the module loads were seeing are benign or warrant further investigation.



What about childprocs and crossprocs?

14 milliseconds after conhost.exe starts, it spawns a cmd.exe child process.



What about childprocs and crossprocs?

14 milliseconds after conhost.exe starts, it spawns a cmd.exe child process, which terminates after three seconds.

What about childprocs and crossprocs?

14 milliseconds after conhost.exe starts, it spawns a cmd.exe child process, which terminates after three seconds.

The command shell has no command-line arguments, which would normally indicate that it's an interactive shell. You may be wondering what an attacker could do in three seconds via an interactive shell.

What about childprocs and crossprocs?

14 milliseconds after conhost.exe starts, it spawns a cmd.exe child process, which terminates after three seconds.

The command shell has no command-line arguments, which would normally indicate that it's an interactive shell. You may be wondering what an attacker could do in three seconds via an interactive shell without spawning any child processes.

What about childprocs and crossprocs?

14 milliseconds after conhost.exe starts, it spawns a cmd.exe child process, which terminates after three seconds.

The command shell has no command-line arguments, which would normally indicate that it's an interactive shell. You may be wondering what an attacker could do in three seconds via an interactive shell without spawning any child processes, without any file or registry modifications.

What about childprocs and crossprocs?

14 milliseconds after conhost.exe starts, it spawns a cmd.exe child process, which terminates after three seconds.

The command shell has no command-line arguments, which would normally indicate that it's an interactive shell. You may be wondering what an attacker could do in three seconds via an interactive shell without spawning any child processes, without any file or registry modifications, without any network connecitons.

What about childprocs and crossprocs?

14 milliseconds after conhost.exe starts, it spawns a cmd.exe child process, which terminates after three seconds.

The command shell has no command-line arguments, which would normally indicate that it's an interactive shell. You may be wondering what an attacker could do in three seconds via an interactive shell without spawning any child processes, without any file or registry modifications, without any network connecitons and with a single cross process event to csrss.exe.

We'll come back to this cross process event.

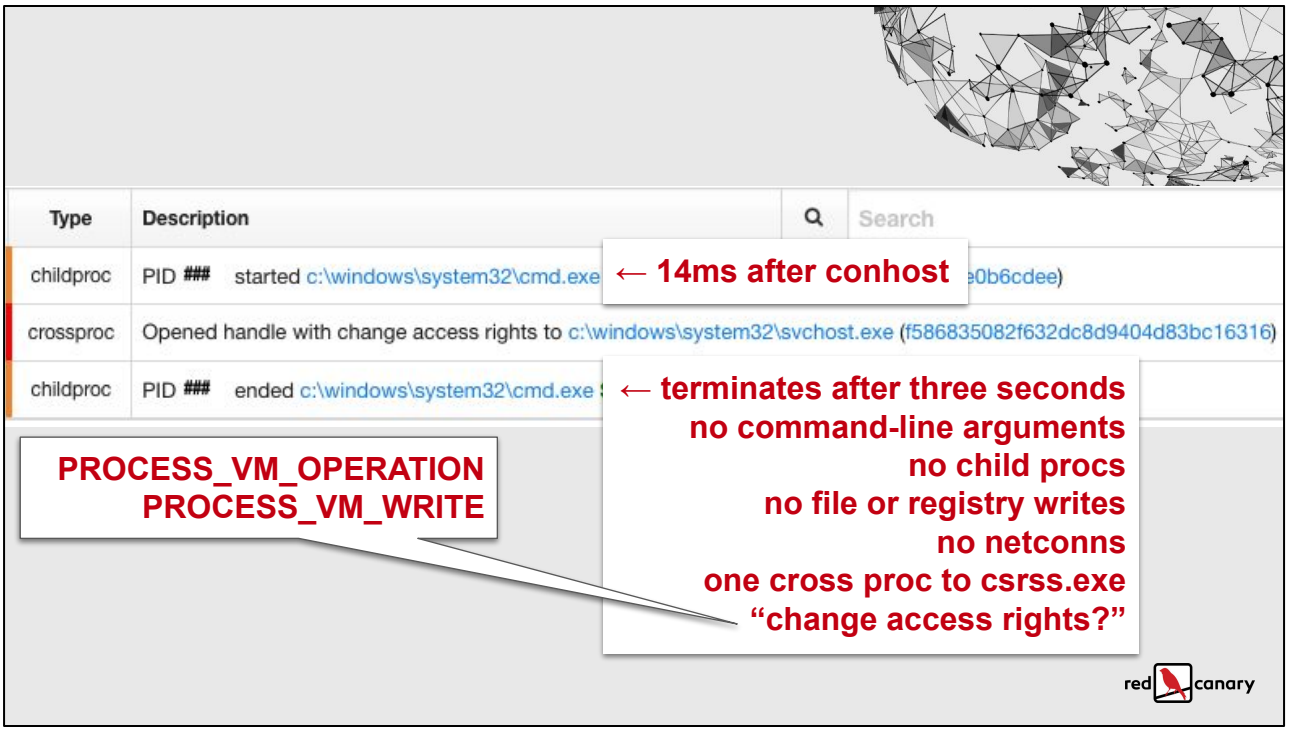
What about childprocs and crossprocs?

14 milliseconds after conhost.exe starts, it spawns a cmd.exe child process, which terminates after three seconds.

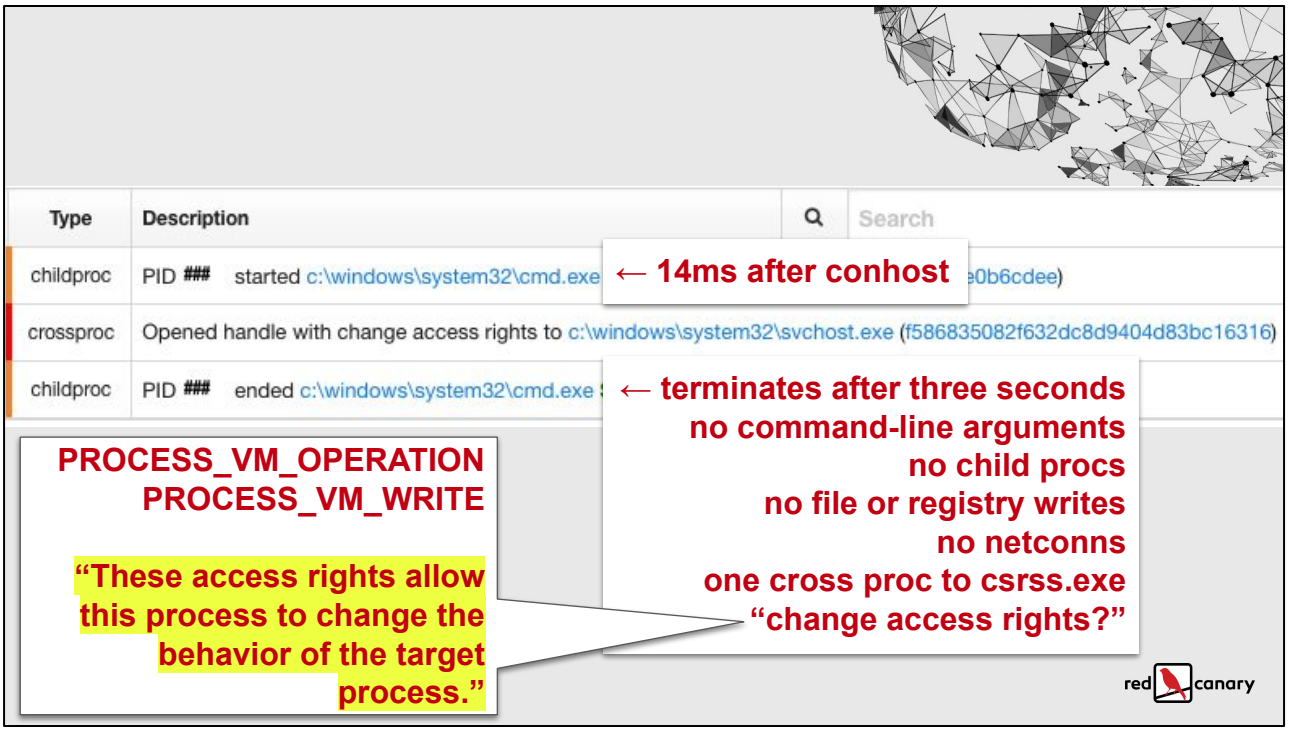
The command shell has no command-line arguments, which would normally indicate that it’s an interactive shell. You may be wondering what an attacker could do in three seconds via an interactive shell without spawning any child processes, without any file or registry modifications, without any network connecitons and with a single cross process event to csrss.exe.

The cross process event from cmd.exe to csrss.exe opens a handle with “change access rights.”

What does this mean?

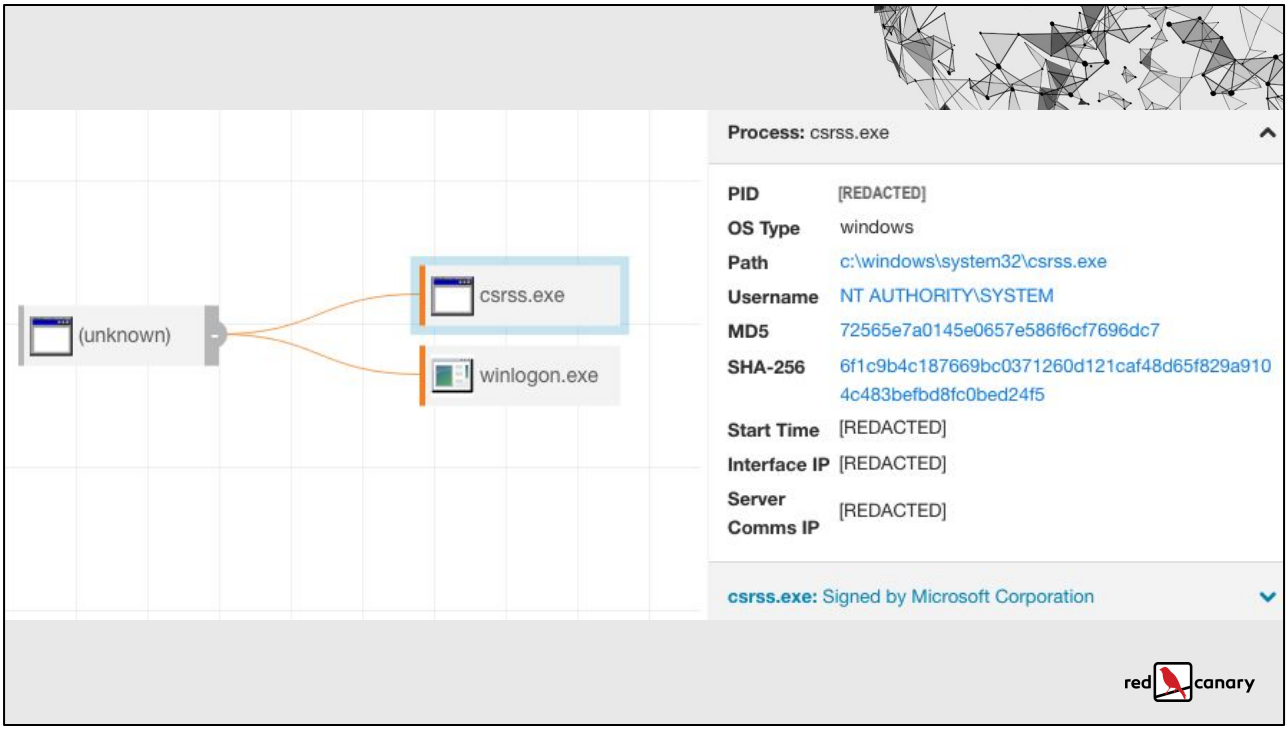


The EDR platform helpfully tells us the specific API calls that are used.



And if you dig a little deeper in the platform it tells you that "these rights allow the process to change the behavior of the target process."

This sounds like it could be interesting, we'd better check it out.

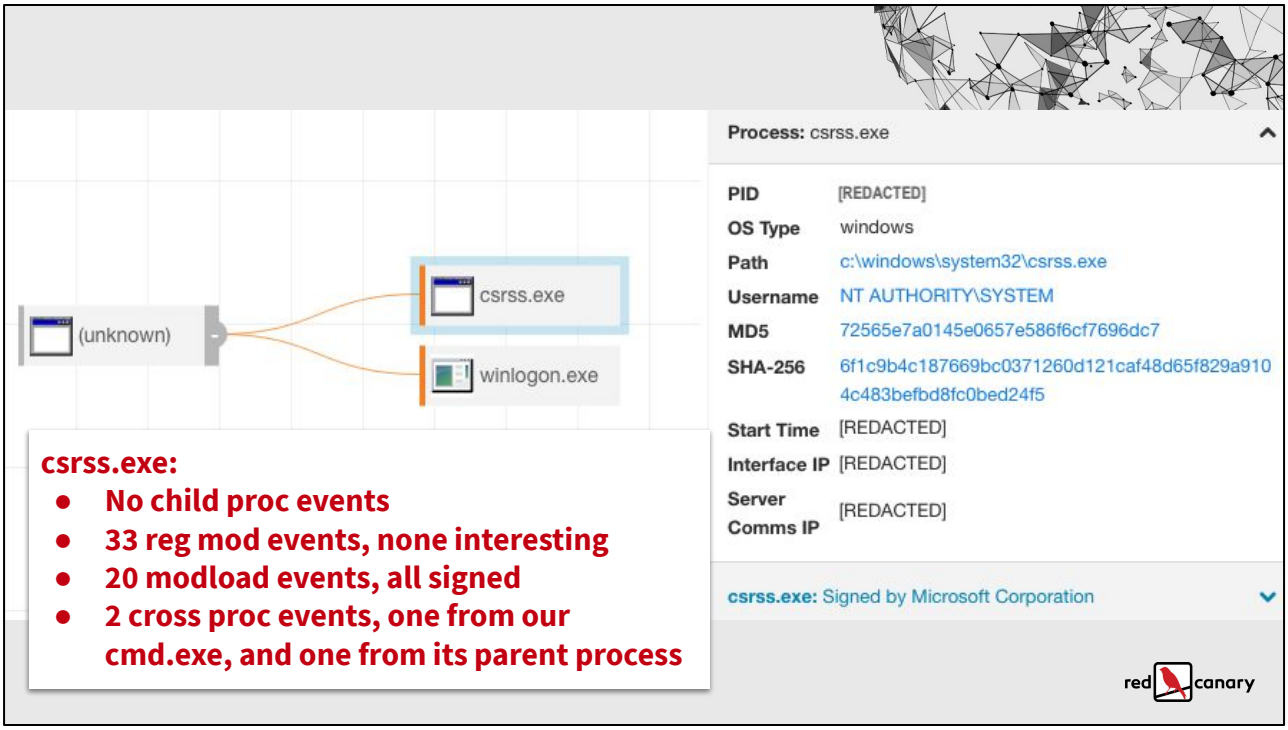


Here's the process tree view in the EDR platform.

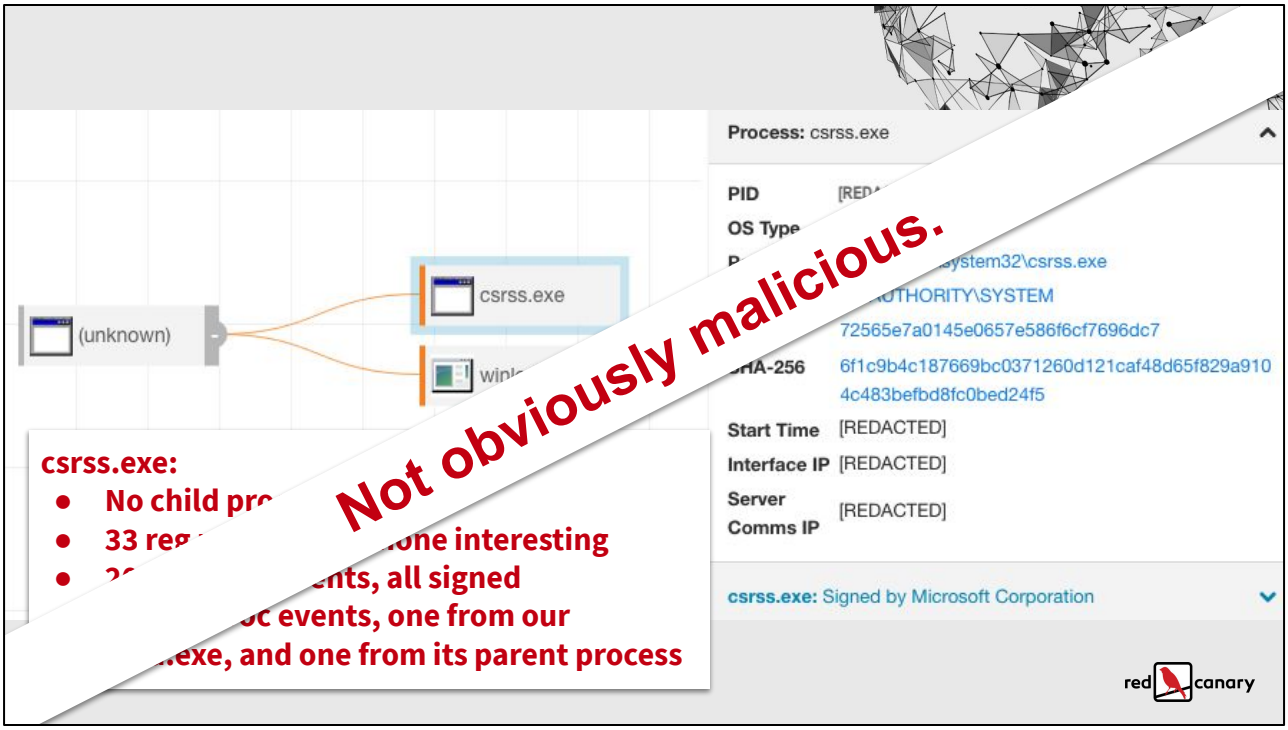
According to SANS, "**csrss is the Client/Server Run-Time Subsystem and is the user-mode process for the Windows subsystem. It manages processes and threads and imports many of the DLLs that provide the Windows API…**"

It makes sense that cmd.exe and csrss.exe would have cross process events.

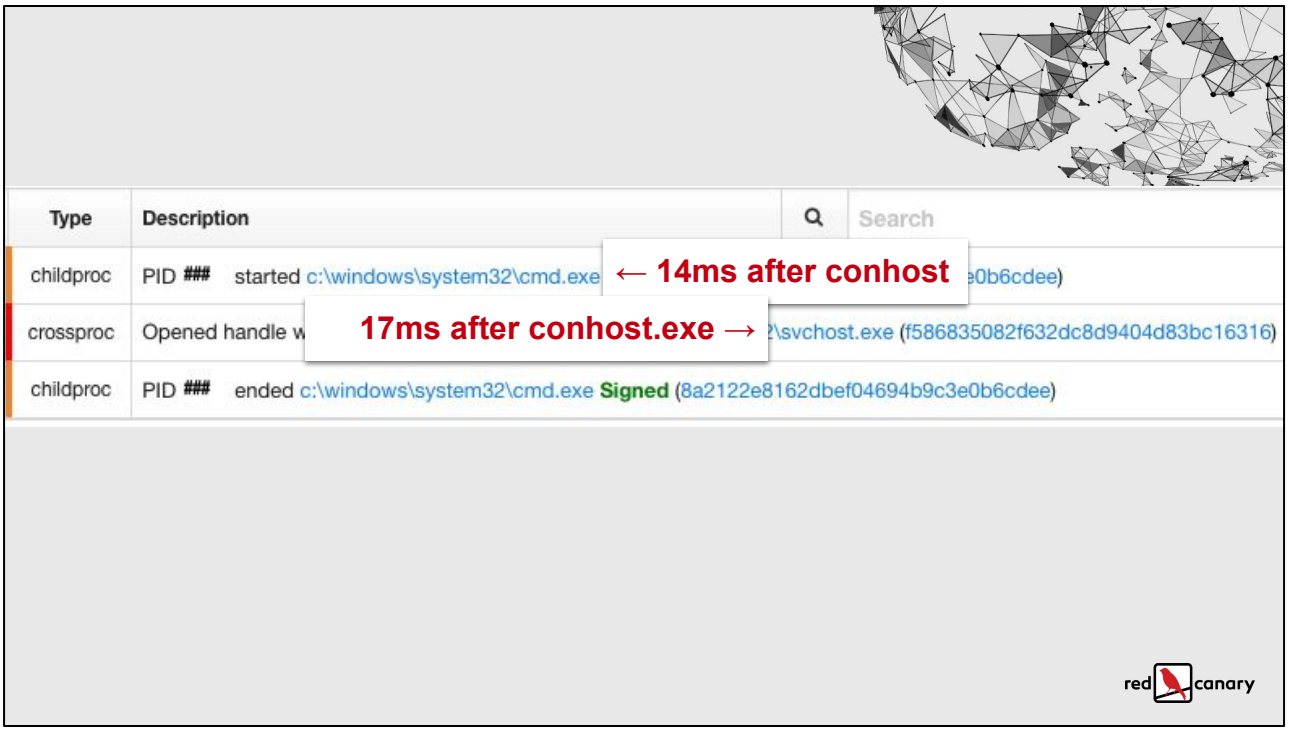
Source: https://www.sans.org/posters/?focus-area=digital-forensics



Again we're presented with many different threads that we could pull on for this investigation, but given that we're under time pressure, it probably makes sense to engage on this at the 10 thousand foot view, is csrss.exe doing anything that appears suspicious?



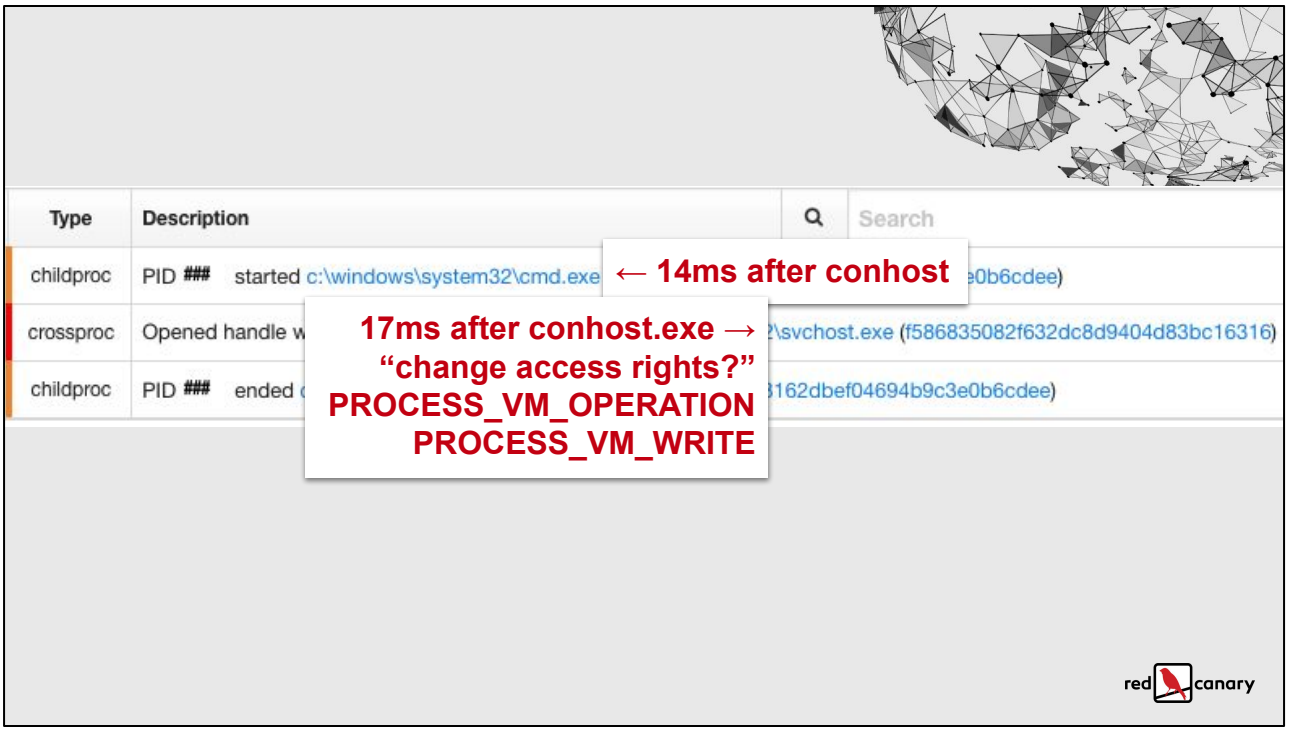
No, there's nothing obvious, so we should move on.



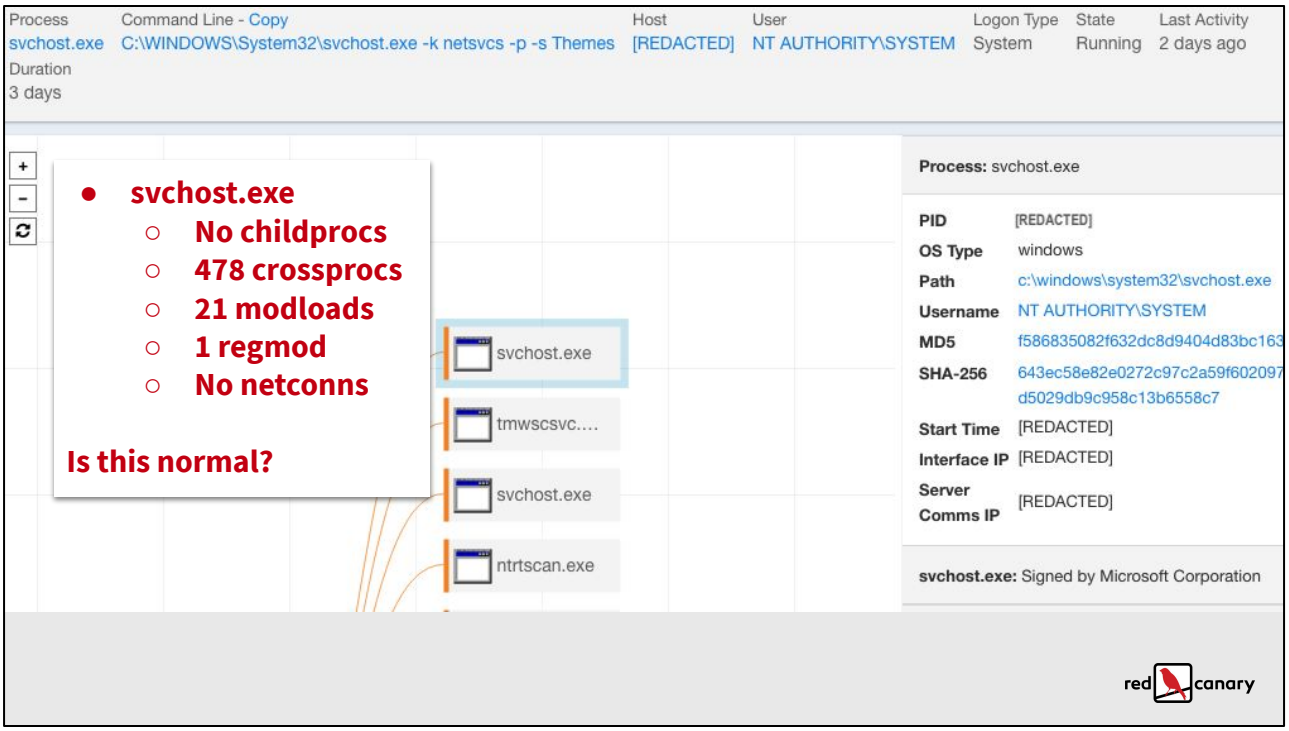
Our cmd.exe child process doesn't appear to be interesting. Let's take a look at the cross process event to that opens a handle to svchost.exe.

Seventeen milliseconds after the conhost.exe process started, it opens a handle with "change access rights" to a svchost.exe process.

Again with the “change access rights.”



And digging in the platform reveals the PROCESS_VM_OPERATION and PROCESS_VM_WRITE APIs were invoked again.



Here's our svchost.exe that conhost has a cross process event to.

This is a longer running process, hosting netsvcs.

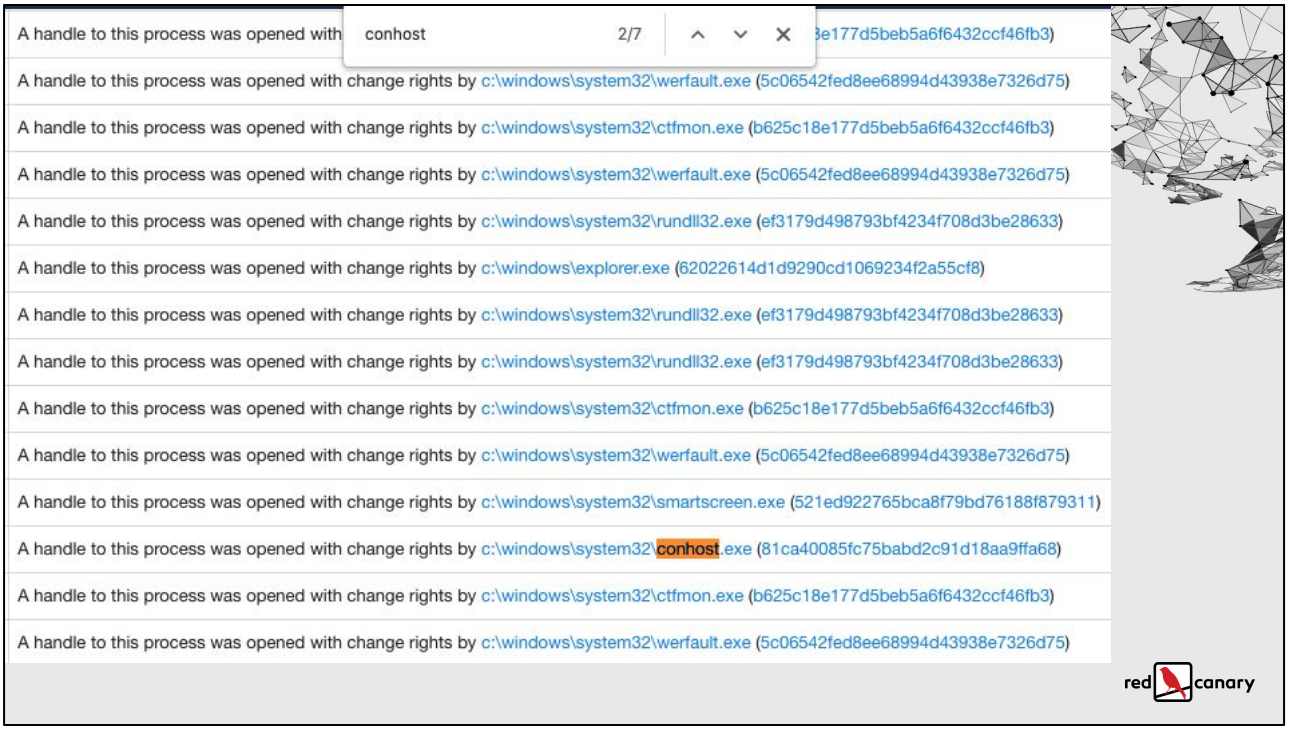
It has

- no child processes,
- 478 crossprocs,
- 21 modloads and
- one regmod.

But is this normal or abnormal? We can't tell from the data given.

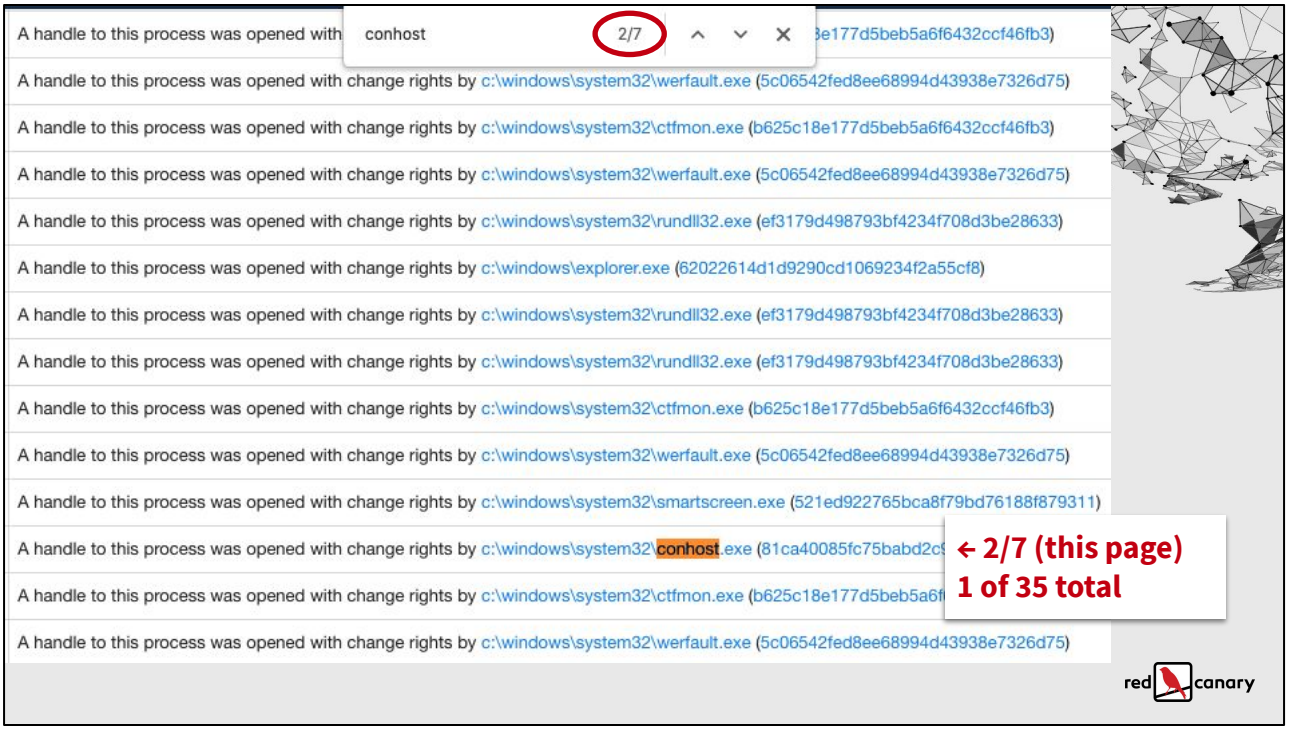
For our purposes, it's probably best to focus on things that happened around the time of our conhost crossproc event.

Useful resource for svchost flags, child procs, etc.: https://docs.google.com/spreadsheets/u/0/d/1MAbln97YNDyR2Ar1PftxLeMpHVVGeZmQRVMybDaOmZ8/htmlview <- Sorry, no idea who made this or I would give them credit.

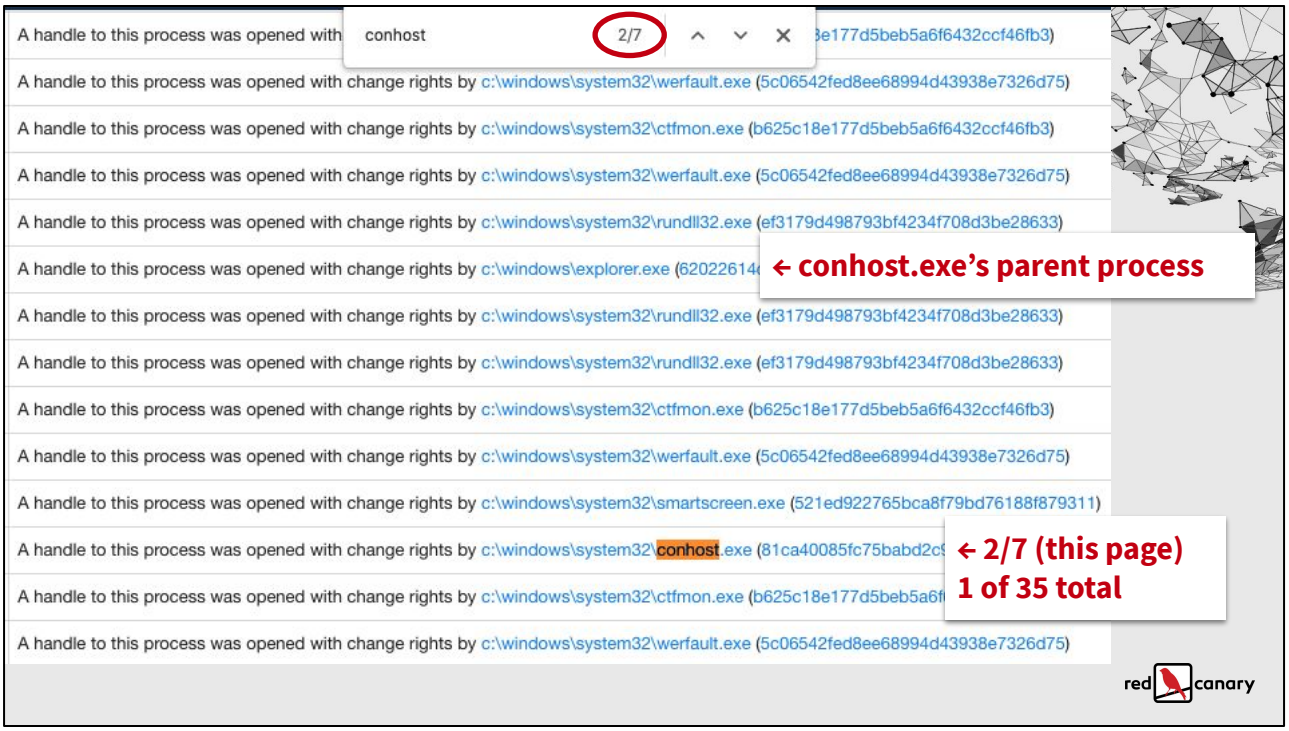


Here we're looking at a partial list of the 478 crossprocs into svchost.exe

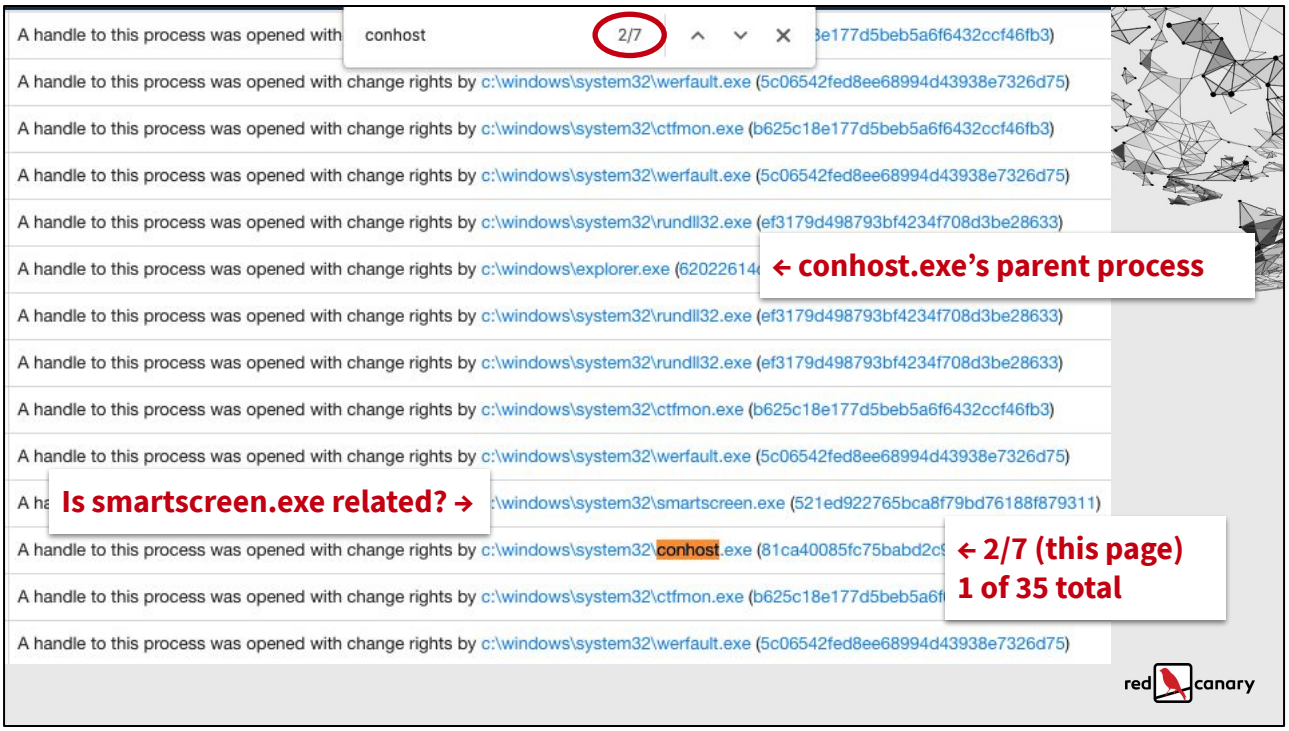
Our conhost.exe process is highlighted.



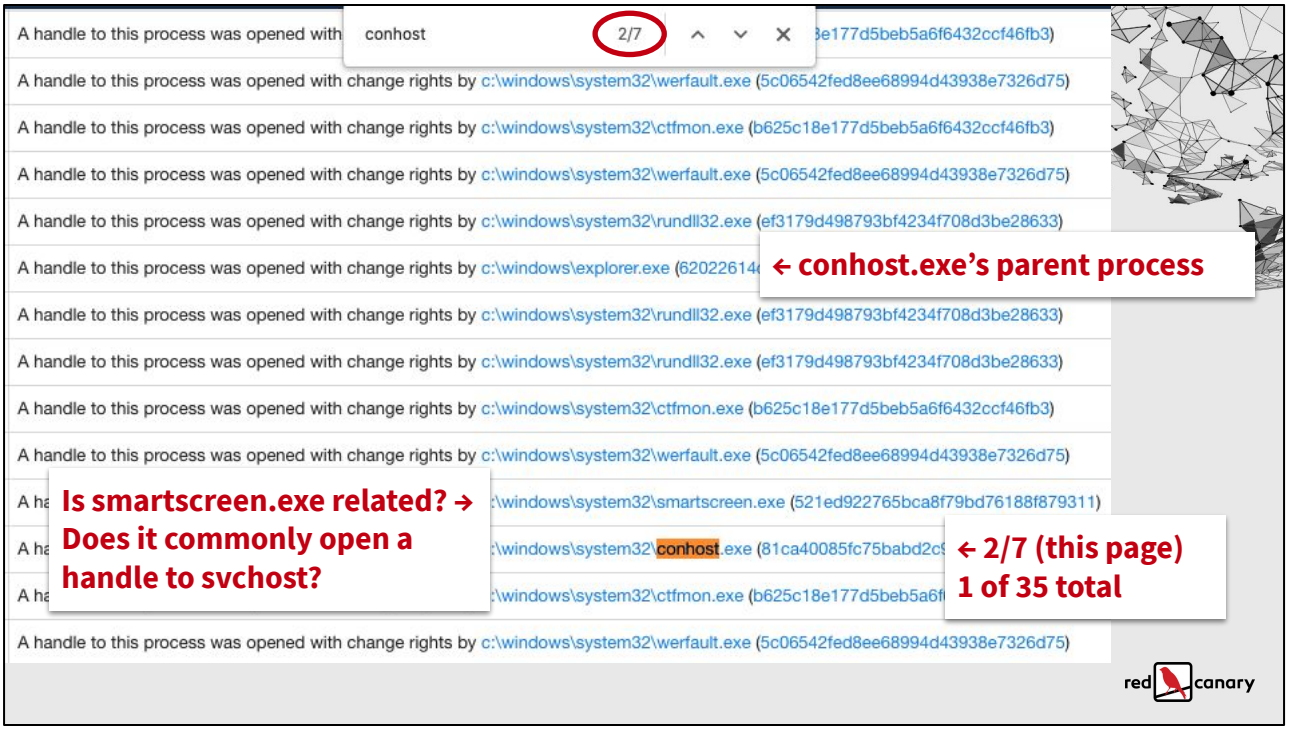
The search dialog at the top indicates this is 2 of 7 conhost.exe cross process events into svchost.exe on this page, I counted 35 in all, so it seems somewhat common, about 7.5% of all the cross process events into this svchost.exe process.



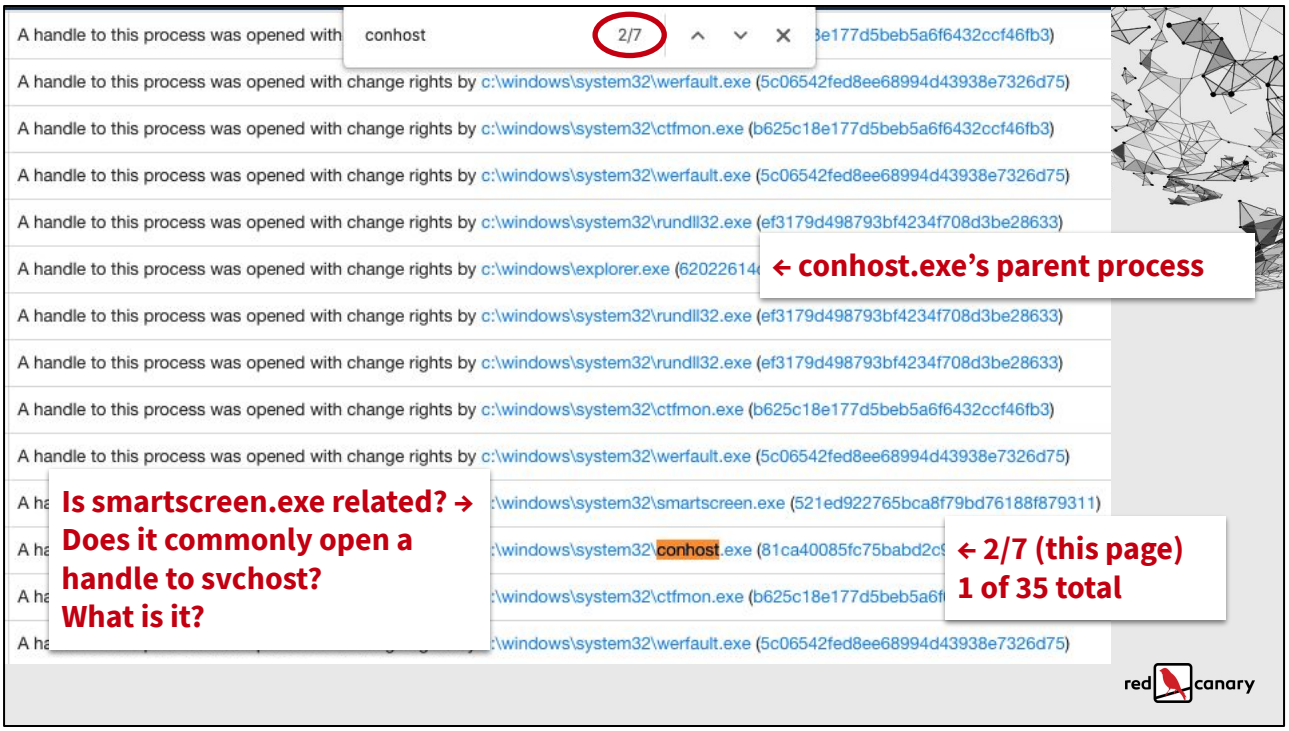
The parent process of conhost.exe, explorer.exe also has a handle open to this svchost.exe instance.



The line right above our conhost.exe process shows that smartscreen.exe has a handle open to this svchost.exe process.



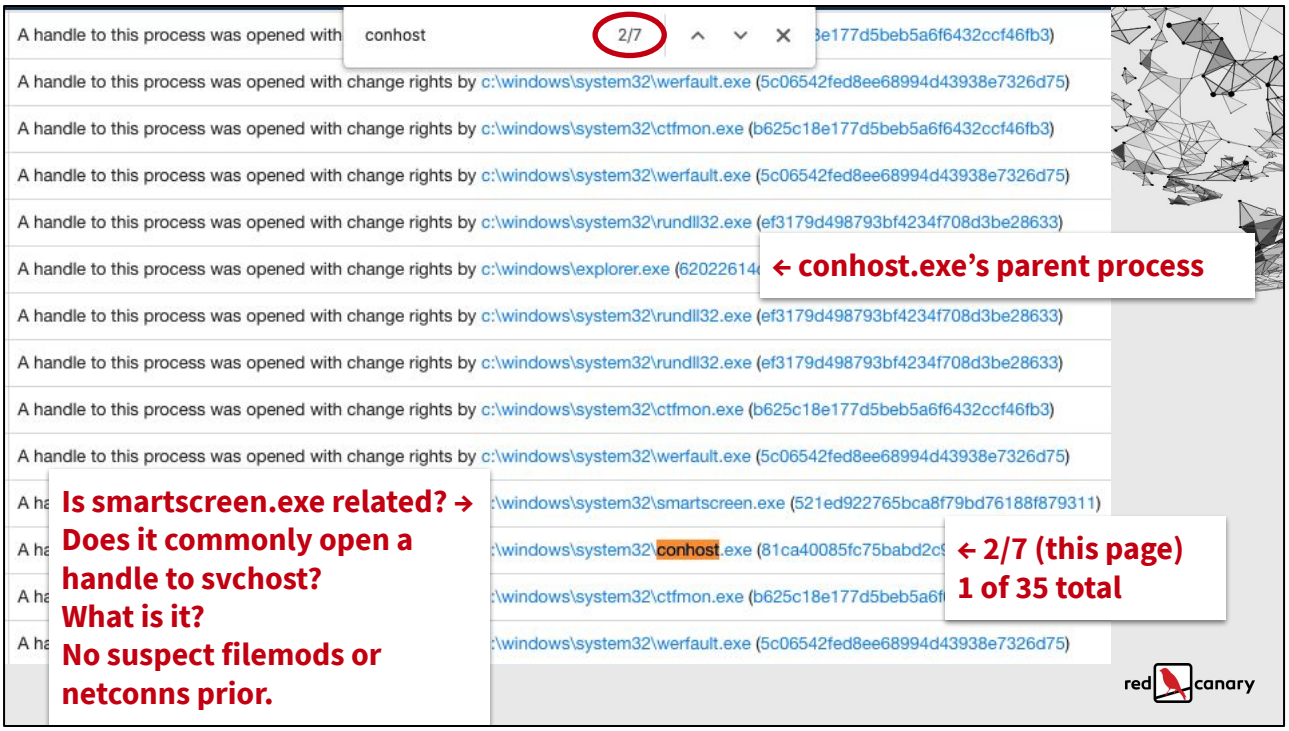
Is this a common occurrence? The platform doesn't tell us, but this was a unique cross process.



What is smartscreen.exe?

It's a process that presents the user with a popup when they have downloaded active content or visited a website that Microsoft thinks could be dangerous. It notifies them that what they are doing is risky and asks them if they want to continue. If this is related to our conhost.exe thread, this could be the first real indicator that something suspicious is happening.

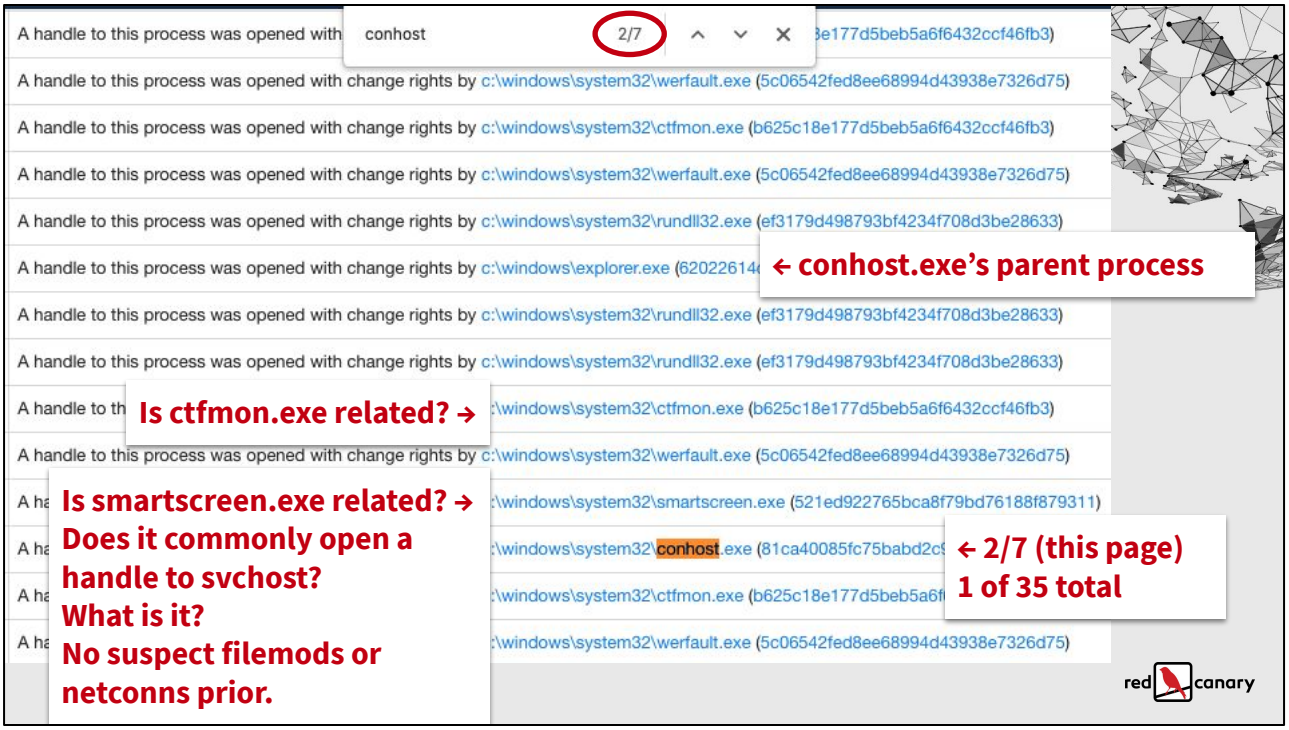
I chased down this lead in the EDR platform, looking for any interesting network connections or file writes in the minutes preceding smartscreen's execution and I found nothing of concern.



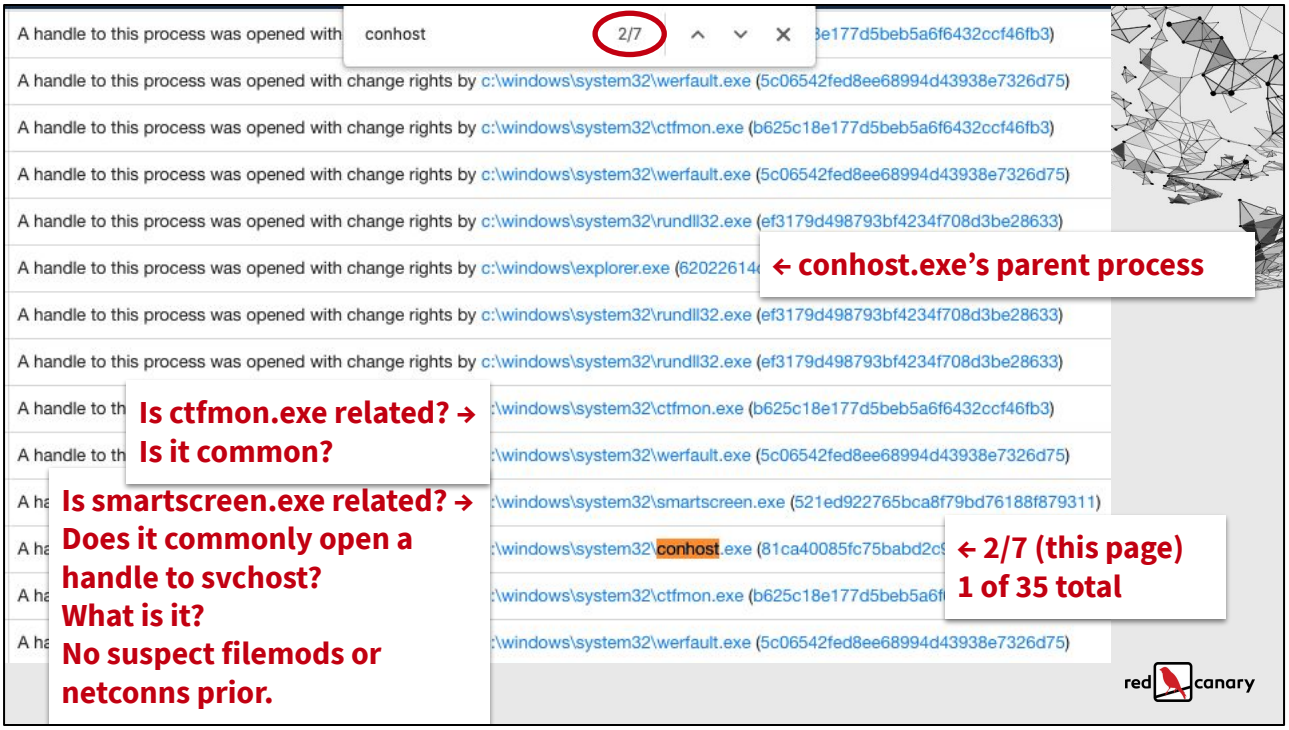
What is smartscreen.exe?

It's a process that presents the user with a popup when they have downloaded active content or visited a website that Microsoft thinks could be dangerous. It notifies them that what they are doing is risky and asks them if they want to continue. If this is related to our conhost.exe thread, this could be the first real indicator that something suspicious is happening.

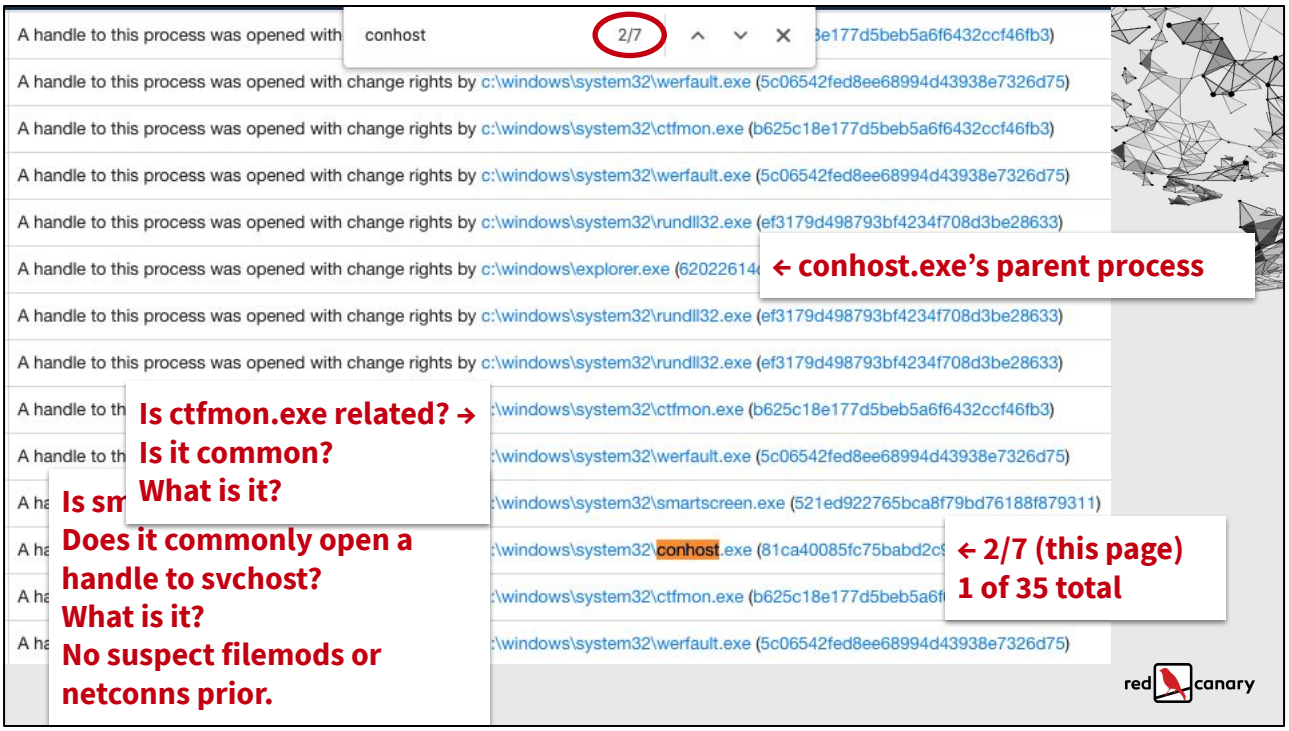
I chased down this lead in the EDR platform, looking for any interesting network connections or file writes in the minutes preceding smartscreen's execution and I found nothing of concern.



What about ctfmon?

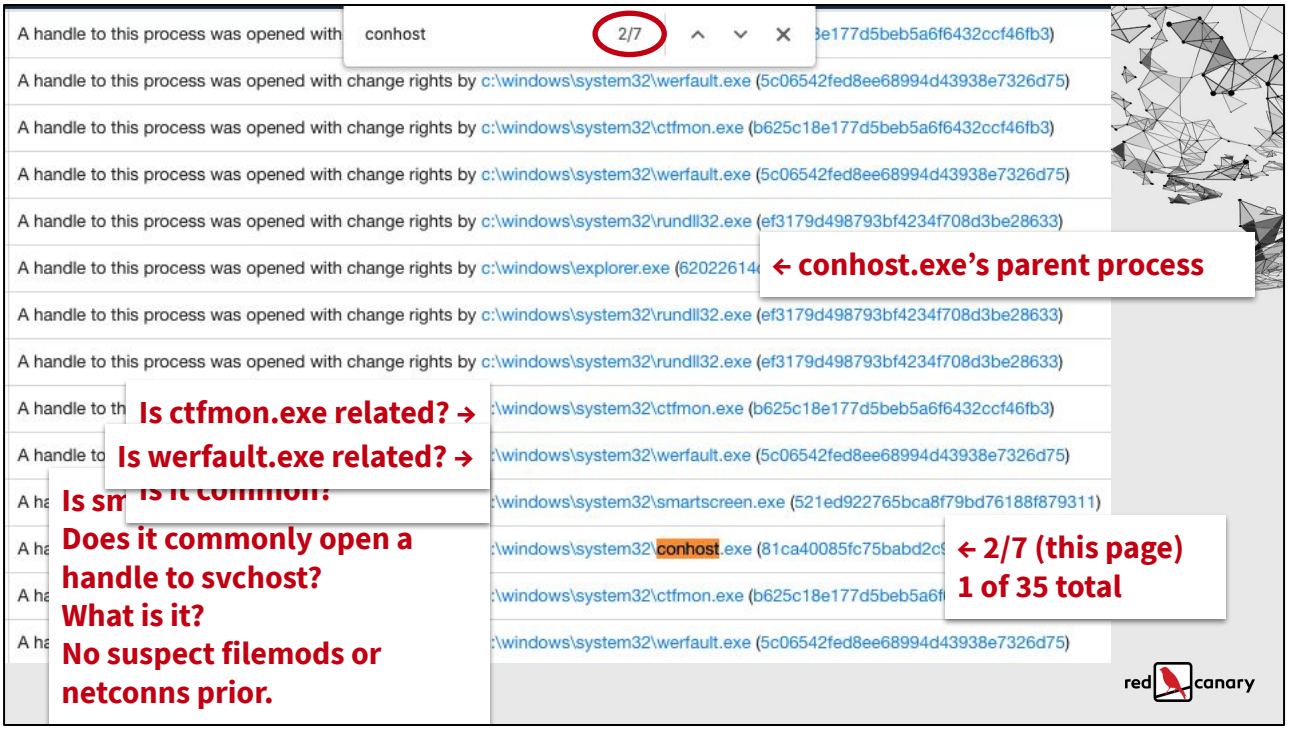


Is it common for ctfmon.exe to have cross process events into svchost.exe? If this screen is any indication, yes.

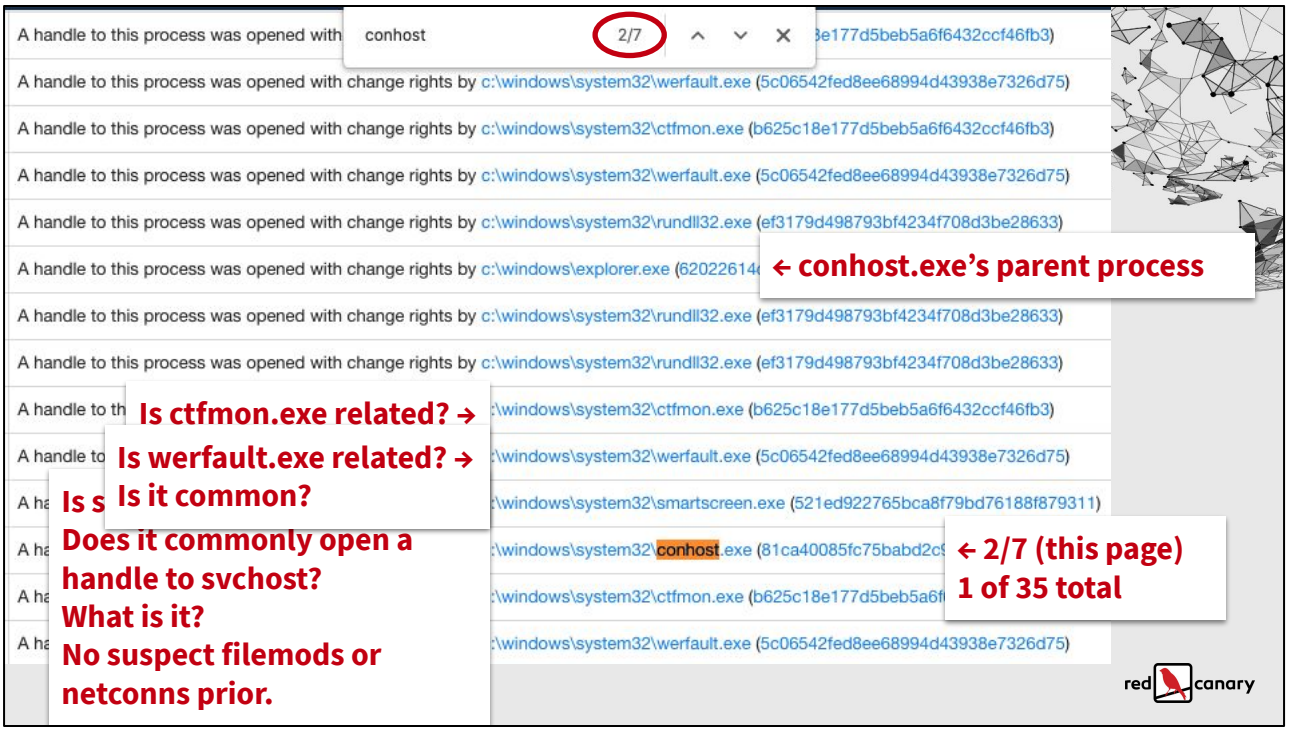


But what is ctfmon.exe?

According to Google results... "Ctfmon is **the Microsoft process that controls Alternative User Input and the Office Language bar**. It's how you can control the computer via speech or a pen tablet, or using the onscreen keyboard inputs for asian languages"

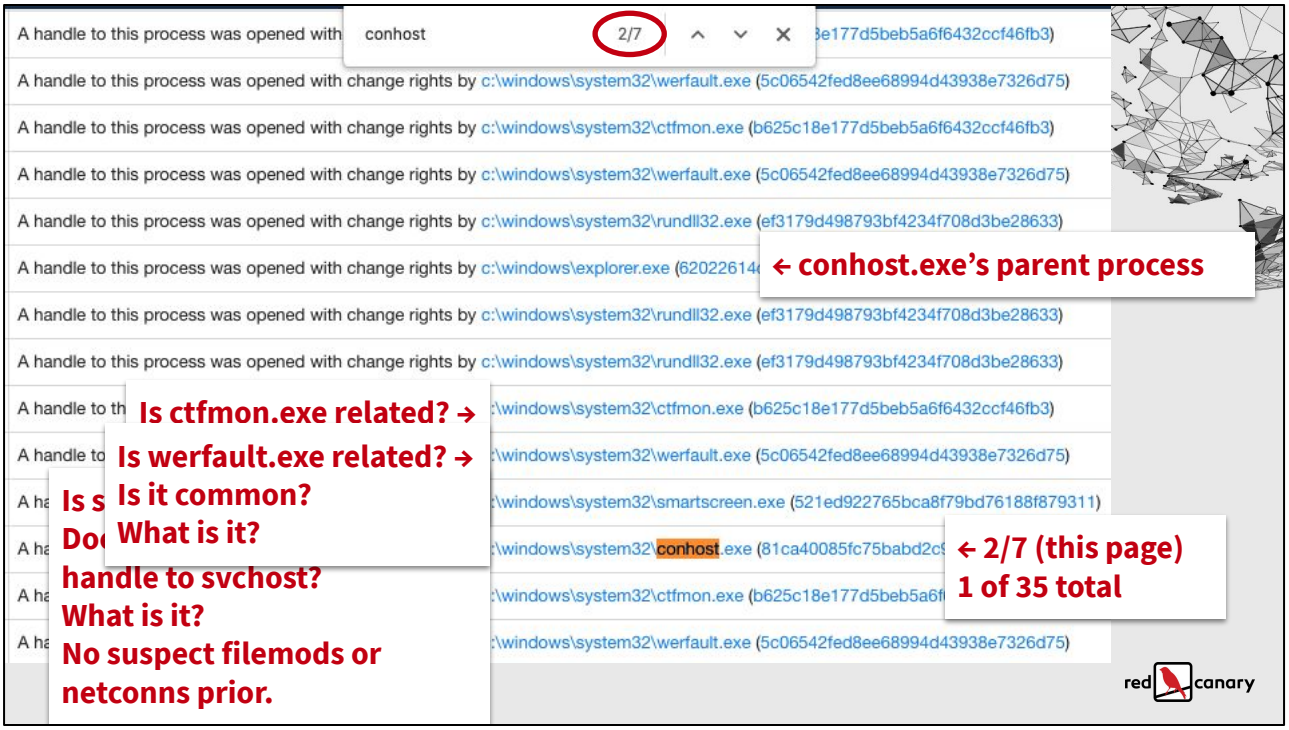


What about werfault.exe?

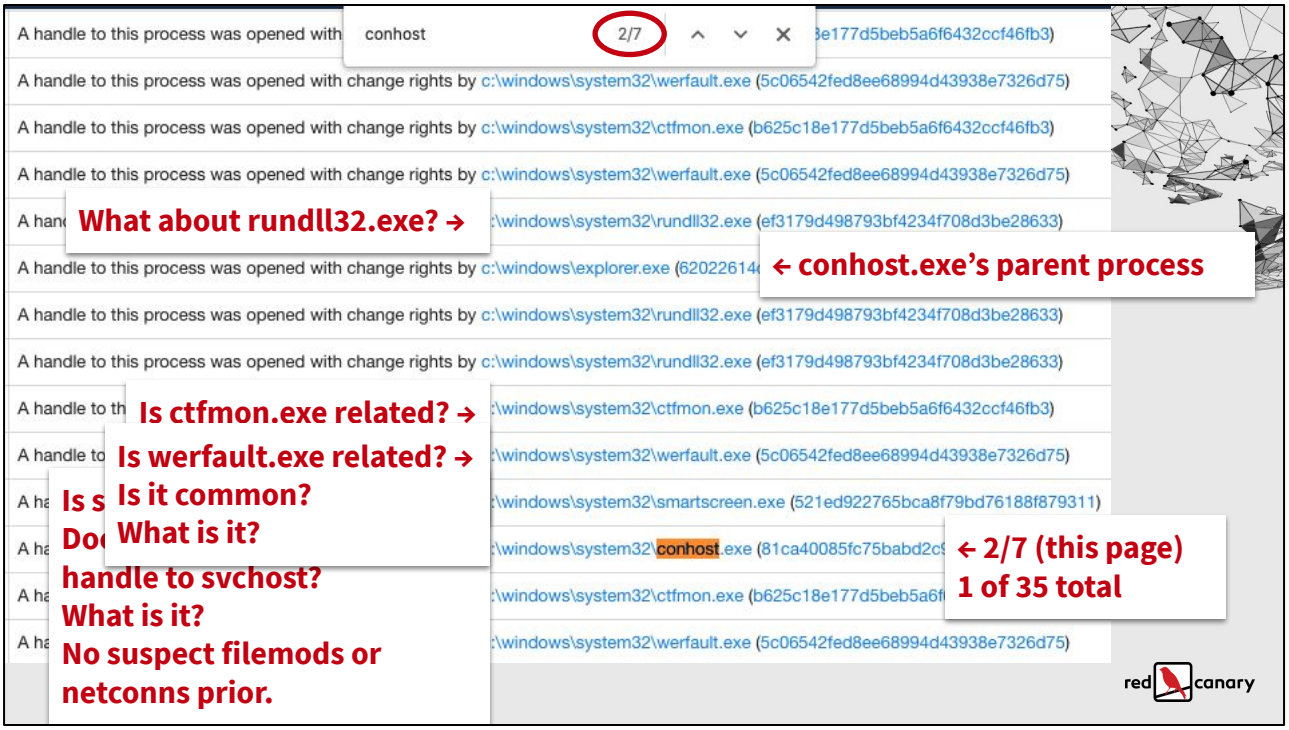


What is the base rate for werfault.exe cross process events into svchost.exe?

Again, the platform doesn't provide that information.



werfault.exe is the Windows Error Reporting service, it typically runs when something crashes and collects information about what went wrong and depending on the host configuration, may send that data off to Microsoft. It also appears to be a very common cross process event into this svchost.exe process. In fact, ctfmon.exe and werfault.exe have cross process events into this svchost.exe event about every five seconds.



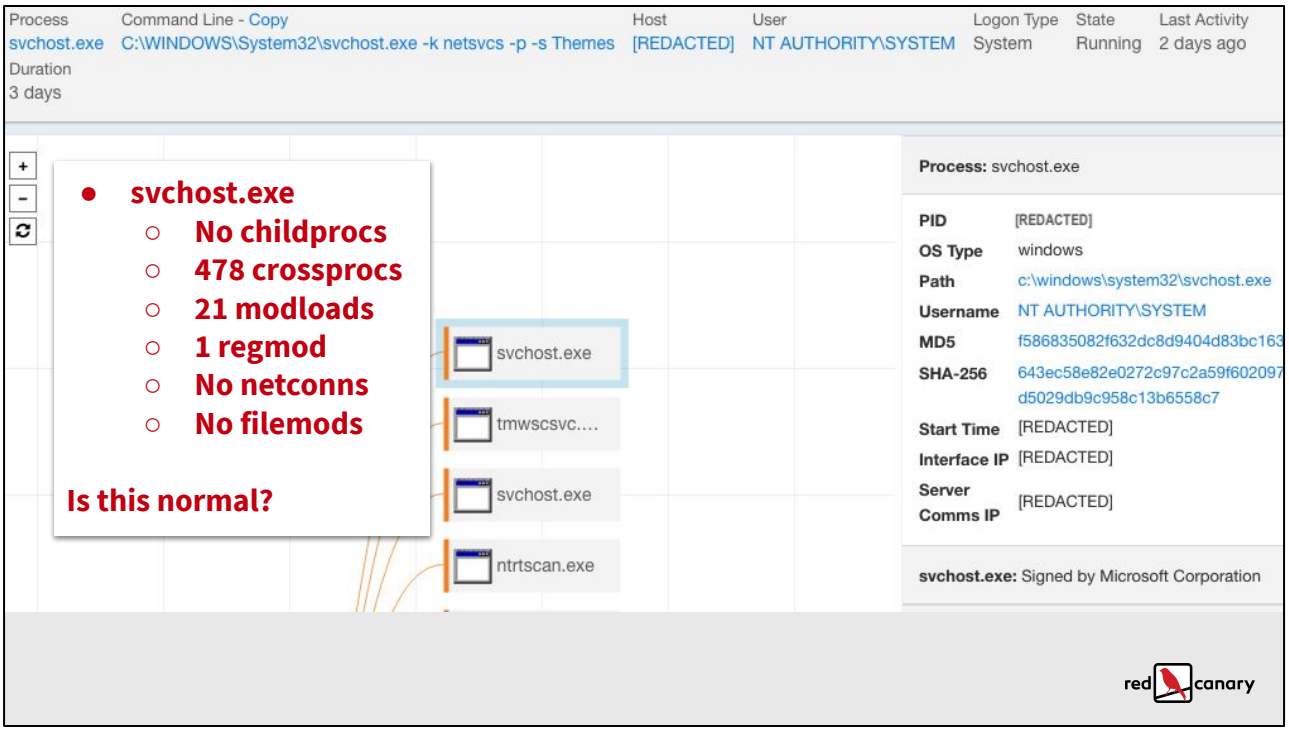
What about rundll32.exe? Does it play a role here in what is happening?

It's hard to know. The EDR platform doesn't really tie together any of these events, except to show that they all relate to this instance of svchost.exe.

There's a gap in the storyline. Our map is missing critical information. Bigger picture, we know the conhost.exe process opened a handle to this svchost.exe process with access rights and API calls that could have allowed it to modify the behavior of this svchost.exe process, but we can't tell what behavior changed, if any. Furthermore, there are 478 cross process events, nearly all of which used the same API calls, which could have also modified the behavior of svchost.exe.

We could go deep down this rabbit hole, investigating every cross process event into and out of svchost.exe, but it would probably be a waste of time and remember, we're under time pressure.

It's probably best to step back and take a 10 thousand foot view of the situation. Do we see svchost.exe doing anything that appears to be malicious?



Let's pop back up the stack a bit. Based on the high-level behaviors of svchost.exe, does it appear to be doing anything obviously malicious?

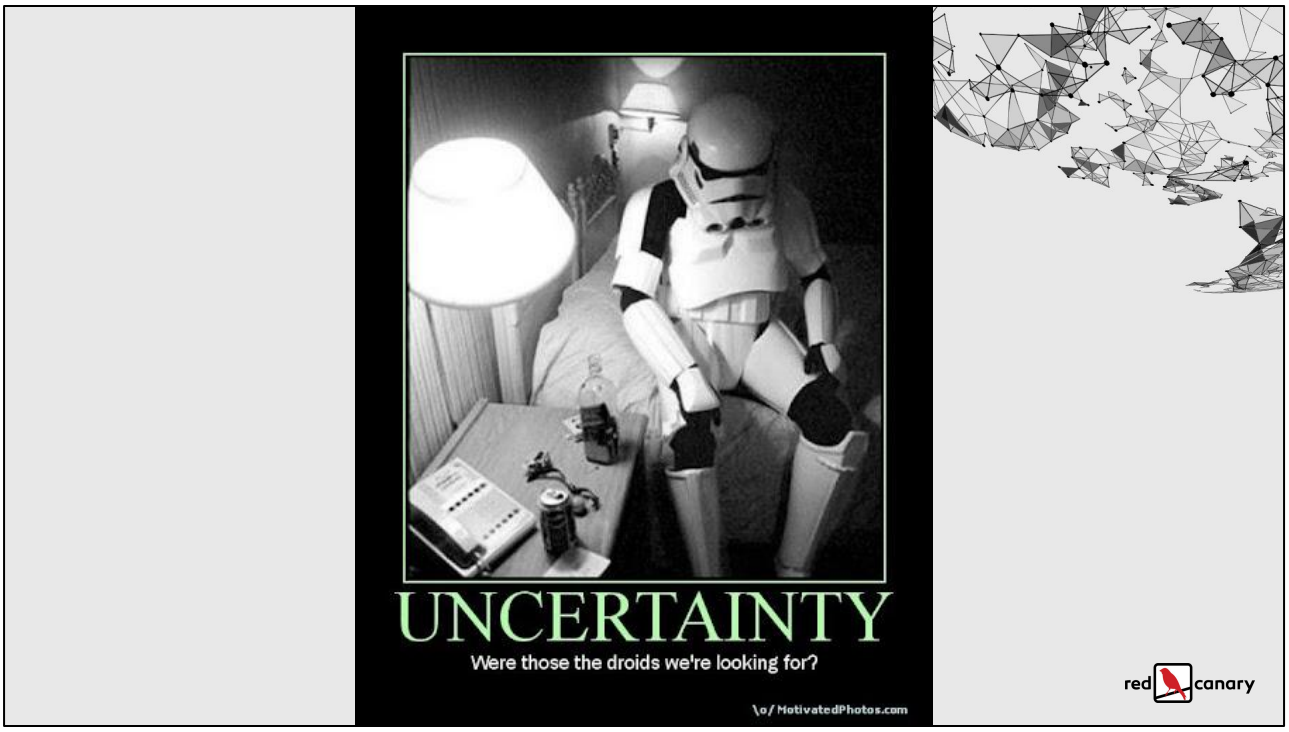
It hasn't spawned any child processes.
Its module loads are all signed, but we don't know if they are all commonly loaded modules for svchost.exe.
Its one registry modification is uninteresting.
It has no network connections.
It hasn't changed anything on the file system.

Big picture, it doesn't appear to be doing anything malicious. So let's move on.

Let's pop back up the stack a bit. Based on the high-level behaviors of svchost.exe, does it appear to be doing anything obviously malicious?

It hasn't spawned any child processes.
Its module loads are all signed, but we don't know if they are all commonly loaded modules for svchost.exe.
Its one registry modification is uninteresting.
It has no network connections.
It hasn't changed anything on the file system.

Big picture, it doesn't appear to be doing anything malicious. So let's move on.



We simply can’t tell from what we know so far whether or not this is benign or malicious, but as we frequently say within the Detection Engineering team at Red Canary when reviewing events of this sort -- “These are probably not the droids we’re looking for.”

Are you being primed?

Stop the ride, I want to get off.

Possibly.

red canary

As we've seen in our quick tour, makers of EDR platforms have a difficult job. They are trying to take a (virtual) reality that is subject to frequent changes and distill it into a map of information that's useful to DFIR practitioners.

Overview

~~Introductions~~

- ~~Who are you?~~
- ~~Who am I?~~

Why are Detection and Security Incident Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

Quick update on where we are.

Overview

~~**Introductions**~~

- ~~Who are yo~~
- ~~Who am I?~~

Missing the forest for the trees.

Why are Detect… …lent Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

We've looked at some EDR platforms and examples of detection work. We've seen that it can be easy to get bogged down in the minutiae and miss the forest for the trees, so to speak.

Overview

~~**Introductions**~~

- ~~Who are you?~~
- ~~Who am I?~~

Why are Detect[illegible]lent Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

Some of this could be mitigated if the EDR platforms gave us more insight into base rates -- how often does conhost.exe spawn cmd.exe and how often does it load given dlls, etc.

Overview

~~**Introductions**~~

- ~~Who are you?~~
- ~~Who am I?~~

False positives.

Why are Detect… …lent Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

We saw an example of what appeared to be something terrible happening on the endpoint that turned out to be a false positive. Now, to be fair, if I had a user trying and failing to dump lsass process memory, I would still want to know about it. So I'd say this is not a great example of a false positive.

Overview

~~**Introductions**~~

- ~~Who are yo~~
- ~~Who am I?~~

False negatives.

Why are Detect lent Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

The false negative, missing the dumping of lsass process memory by another process is more egregious than the false positive.

Overview

~~**Introductions**~~

- ~~Who are you?~~
- ~~Who am I?~~

Make it easier to read.

Why are Detect[illegible]lent Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

We saw an example of a platform that makes no effort to format the data so it is easier for the analyst to read, this slows down the analyst and adds to the cognitive load.

Overview

~~Introductions~~

- ~~Who are you?~~
- ~~Who am I?~~

Why are Detect[illegible]lent Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

UTC time wasted.

Little things like having to convert from one time zone to another, or to a time zone format that isn't even a thing -- UTC is 24 hour time and should have no concept of AM or PM -- this increases cognitive load on the analyst and wastes time.

Overview

~~**Introductions**~~

- ~~Who are you?~~
- ~~Who am I?~~

Crossing the streams on PID reuse.

Why are Detect[illegible]lent Response Hard Problems(™)?

- Examples
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

We saw an example of a platform providing confusing information, telling us that svchost.exe loaded the PowerShell dll, when it fact it appears to be related to quick PID reuse. The process start information appeared to be correct, but the process termination data was confused.

Overview

~~Introductions~~

- ~~Who are you?~~
- ~~Who am I?~~

Why are Detection and Security Incident Response Hard Problems(™)?

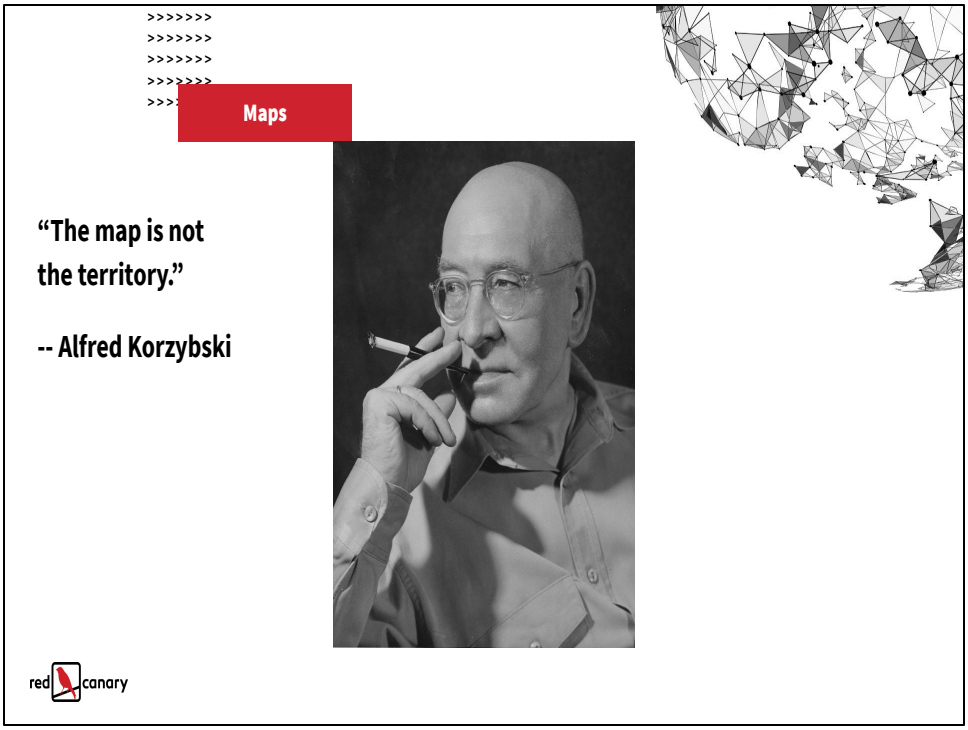
- ~~Examples~~
- Trouble with maps
- Systems of cognition

How can we make it easier?

red canary

So we've seen some examples of things that make the SOC analyst's job difficult. Let's talk about maps.

One problem is that maps are hard.



I listen to a few different podcasts including one called Farnam Street, which typically focuses on decision making and mental models. I caught the host of the show, Shane Parrish, talking about Alfred Korzybski who was the founder of "general semantics," which is beyond the scope of this talk. Korzybski famously said, "The map is not the territory," which is obvious, but it has profound implications.

Korzybski believed that human perception of reality is limited by our nervous system and our language. He wasn't wrong, we don't see, hear or smell as well as many other animals on the planet, that's well known.

For detection engineers and incident responders the telemetry we get from endpoint detection and response tools, from Sysmon, from logging systems, SIEMs, Splunk, etc. is a map. That data is not the territory.

As we've seen during the course of this conversation, those of us doing detection engineering and incident response want the territory. We want full pcaps, full memory dumps, full disk images, complete and accurate logging, etc. If we could reset the entire environment to a point in time and play it forward and backward, make observations and take notes, we might be happy -- we would have, effectively, the territory.

Problems with ~~maps~~ EDR tools

Maps may be incorrect

- Did conhost.exe really spawn cmd.exe?
- Did svchost.exe really modload System.Management.Automation?

Maps are lossy

- They can't tell you everything. **What did those crossproc threads do?**
- Some of what they don't tell you is **critically important**.

Maps require interpretation

- Readers of maps **may not understand** what they are looking at.
- Maps need maps, need maps, need maps...

red canary

Korzybski enumerated three primary problems with maps.

First, they could be wrong. We've seen EDR products mis-represent cross-process directionality. Some software doesn't play well with EDR telemetry -- Run docker on some EDR platforms and try to make sense of the results. You may conclude that the map-maker was drunk.

Second, by definition, maps are lossy. They are reductionist. They can't represent the territory with full fidelity or even high fidelity as the storage demands would make them unusable, impractical, etc. Some of the information that is lost will be critically important to understanding what happened in the territory.

Third, maps require interpretation. Ever been to a strange city and needed to use the map on your phone to walk a few blocks? Did you ever have to walk a block to figure out how the orientation on the phone corresponds to reality? Often maps have legends, or maps for understanding what's on the map. As maps become higher fidelity, more maps of the maps are needed.

Overview

~~Introductions~~

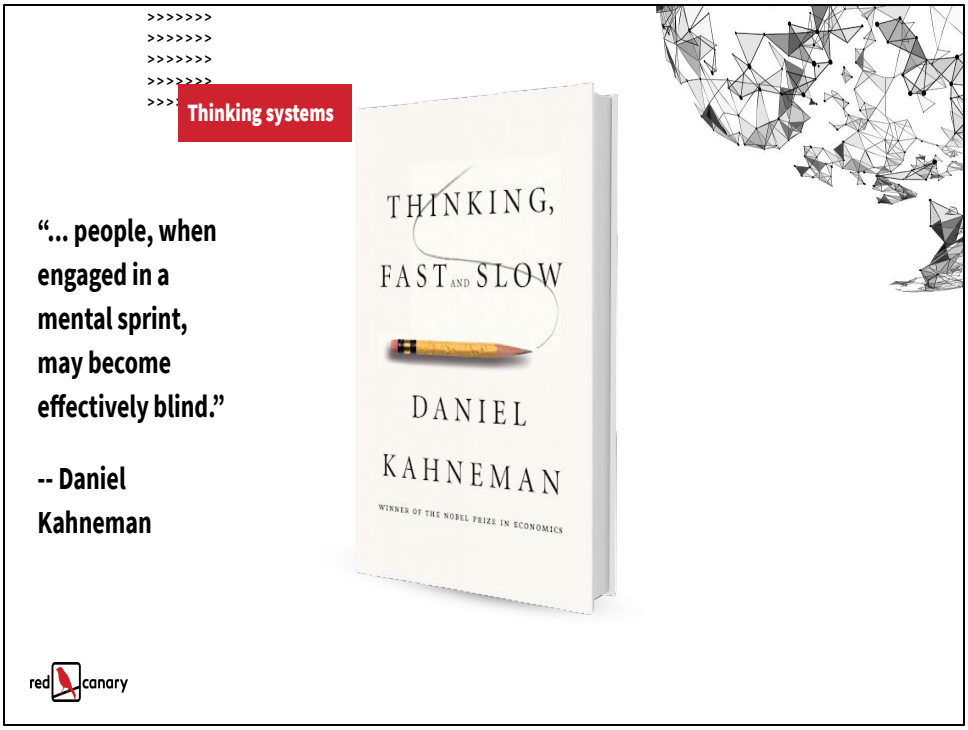
- ~~Who are you?~~
- ~~Who am I?~~

Why are Detection and Security Incident Response Hard Problems(™)?

- ~~Examples~~
- ~~Trouble with maps~~
- Systems of cognition

How can we make it easier?

red canary



A few years ago I picked up Thinking Fast and Slow by Daniel Kahneman, again it was a book I'd heard mentioned on a few different podcasts. I've since read this book a couple times through and I've highlighted more things in this book than any other book I've ever read. It covers a wide range of human behaviors, many of which are relevant to information security practitioners.

Let's play a different game

What is this person feeling?

red canary

Correct answers only: This kid is scared, filled with dread, something along those lines.

Let's play a different game

What is this person feeling?

red canary

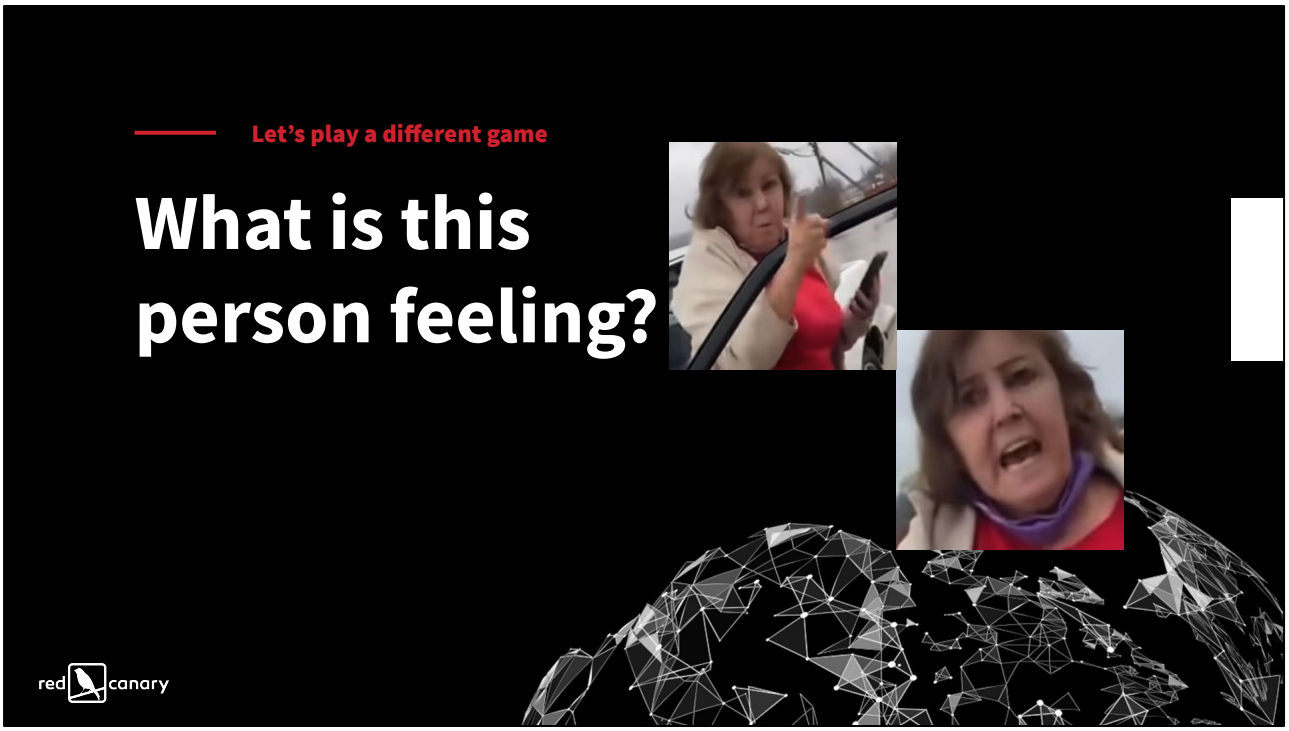
This person appears to be angry. What makes us think that may be the case?

Let's play a different game

What is this person feeling?

red canary

This person appears sad, maybe pleading their case? Distraught?



This person appears to be angry.

Let's play a different game

What is this person feeling?

red canary

This person appears to be surprised, shocked, but not in an unpleasant way.

Let's play a different game

What is this person feeling?

red canary

The person on the right appears to be upset. The person the left?

Let's play a different game

What is this person feeling?

red canary

The person on the left appears to be laughing. The person on the right is actually a grumpy cat.

Let's play a different game

What is this person feeling?

red canary

The person on the left looks surprised. The person on the right may be SwiftOnSecurity.

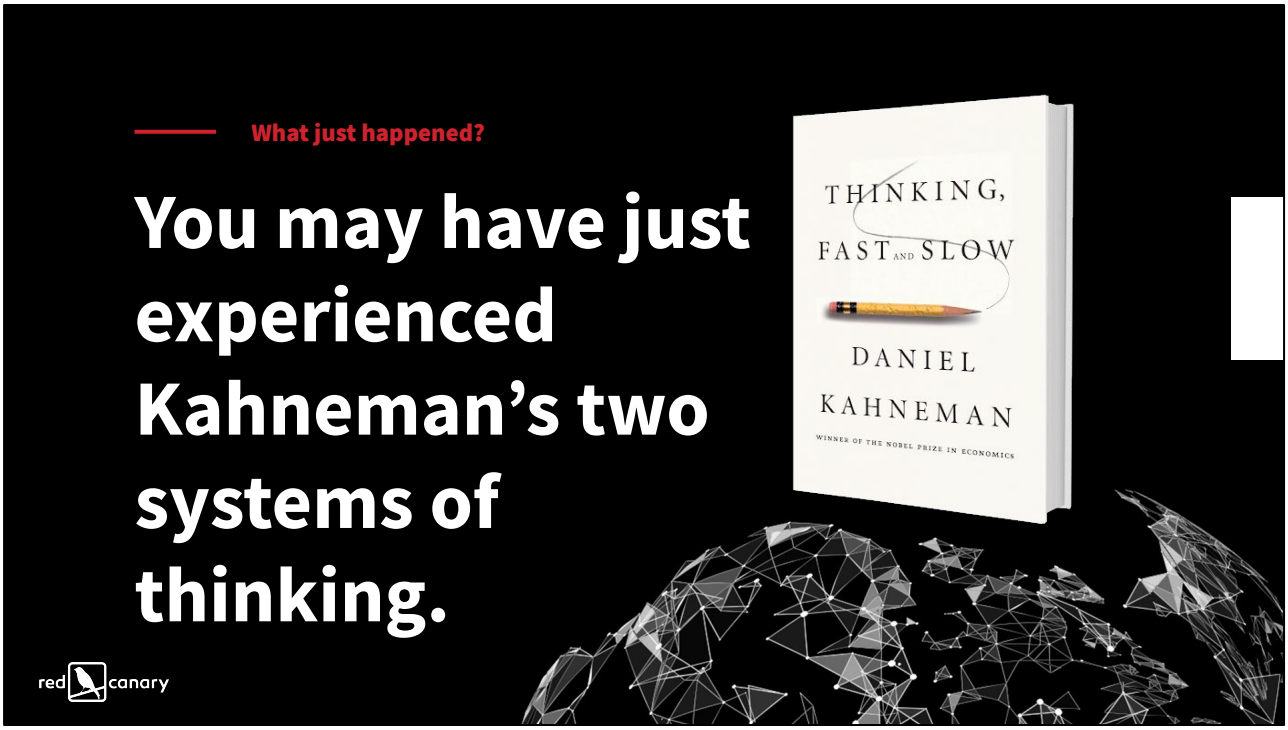
Let's play a different game

What is white's next move?

5, 4, 3, 2, 1

red canary

F4 to H6



In the pictures of people (and a cat), you were probably able to instantly identify with very high fidelity the emotional state.

In the Chess puzzle, unless you've played many hours of Chess or your a suivant, you were probably not sure what the next best move for white would be, or at the very least, you were unlikely to pick out the move instantly.

This concept of two different system of thinking is foundational to Kahneman's book, but it is full of many more great insights.

	System One	System Two
Descriptions	Automatic, effortless, emotional, intuitive, perceptive, reactive	Expensive (calorically and from a risk perspective), lazy, rational, slow
Examples	Recognizing faces & expressions, dodging thrown / falling objects, chess puzzles (if you're a great chess player)	347 * 67.4, chess puzzles (if you're not a great chess player)
Implications	95% of our decisions System two tasks can become system one with enough training (chess puzzles, driving a stick-shift) Jumps to conclusions without understanding the size of the jump(s)	System two generally accepts system one's conclusions, which leads to biases (e.g. familiarity is a product of system one, system two relies on that to make judgements about truth and falsity)

Thinking Fast and Slow presents two systems of thinking -- fast and slow.

System one, the fast system, is generally responsible for 95% of the decisions we make, many decisions we're not even aware we're making. Reacting to someone's facial expression, their tone of voice, the way they move toward us, etc. System one thinking is effortless, inexpensive (from an evolutionary perspective) and has kept us alive for millennia as it quickly evaluates motion, noises, etc.

System two is slow and expensive. Concentrating on a problem consumes more calories than system one and it distracts us from what is happening around us, putting us at risk from snakes in the grass or trees, etc. As a result, we use system two sparingly. It is lazy and will generally accept whatever system one postulates as true -- this is where many biases come from.

Interestingly, system two tasks can become system one tasks with enough practice and repetition. Grand master level chess players can look at a chess position on a chess board and instantly recognize what the next best move will be because they have spent thousands of hours studying and playing the game and have developed a rich pattern recognition system. If you know how to drive a car with manual transmission, you've likely converted a system two task to system one through practice.

This book is a goldmine of interesting and impactful ideas and worth putting on your shortlist of books to read, at least the first third to one half. Example:

“... before an issue is discussed, all members of the committee should be asked to write a very brief summary of their position. This procedure makes good use of the value of the diversity of knowledge and opinion in the group. The standard practice of open discussion gives too much weight to the opinions of those who speak early and assertively, causing others to line up behind them.”

Other K implications

- **Activated ideas**
- **Base rates** and **regression to the mean**

red canary

Thinking Fast and Slow had a big impact on my thinking. It's the most highlighted book I've ever read. It contains many other ideas worth exploring and we'll briefly take a look at a couple that I think are applicable to detection and incident response work.

Other ideas that are worth a mention here, but won't be explored further:

Incident responders should be familiar with **post-mortems** (blameless post-mortems, please) wherein teams get together after an incident to review what happened, how the investigation went, what the findings were, how security posture, detections and investigative capabilities can all be improved going forward, etc., but you may be less familiar with **pre-mortems**. **Pre-mortems** are discussions that are had before a new policy, procedure, technology or similar is implemented, deployed, etc. in which the parties consider that the project is months or years down the road and has been a failure -- pre-mortems ask the questions: What went wrong? What mistakes were made? What should have been done differently?

Pre-meeting one pagers ask that meeting participants come to the meeting having prepared a document of no more than one page with their thoughts on the topic of discussion. These documents should be shared with and reviewed by everyone in the meeting in advance. The idea is to prevent halo-effect, first-mover, frequency, recency and other biases from dominating the decision making process.

Overview

~~**Introductions**~~

- ~~Who are you?~~
- ~~Who am I?~~

Why are Detection and Security Incident Response Hard Problems(™)?

- ~~Examples~~
- ~~Trouble with maps~~
- ~~Systems of cognition~~

How can we make it easier?

red canary

So what?

What do we do with this knowledge?

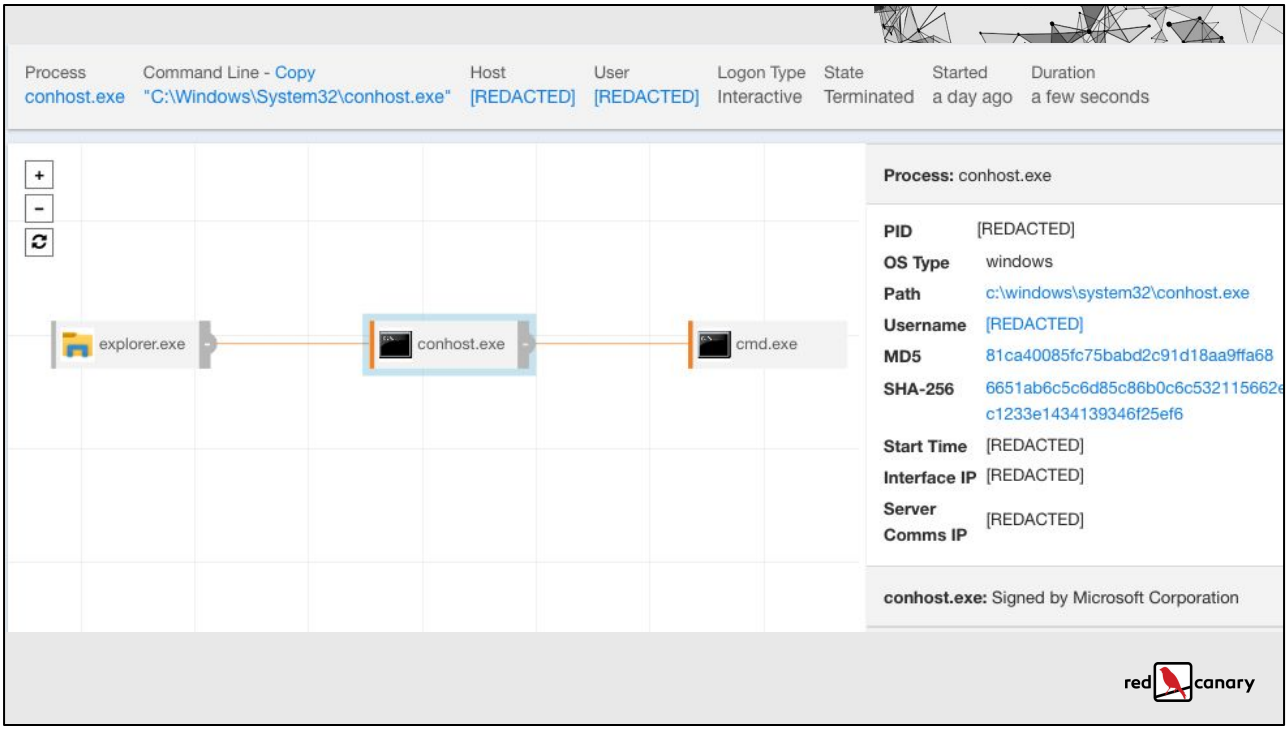
red canary

Great, there may be two systems of thinking. What does instantly recognizing someone's facial expression have to do with detection engineering?

How do we make recognizing that this is good or bad...

red canary

Great, there may be two systems of thinking. What does instantly recognizing someone's facial expression have to do with detection engineering?

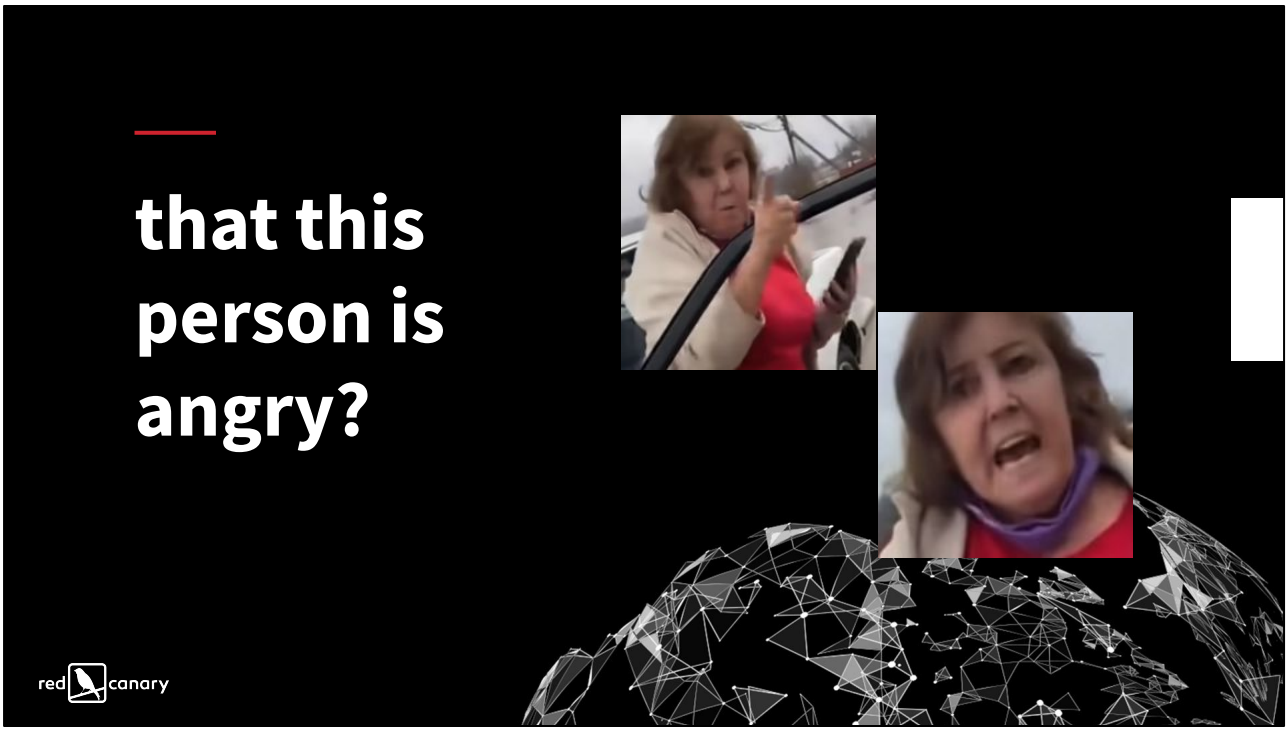


How do make recognizing that this is something bad as easy as

as easy as recognizing that this kid is terrified or...

red canary

as easy as recognizing that this kid is terrified or



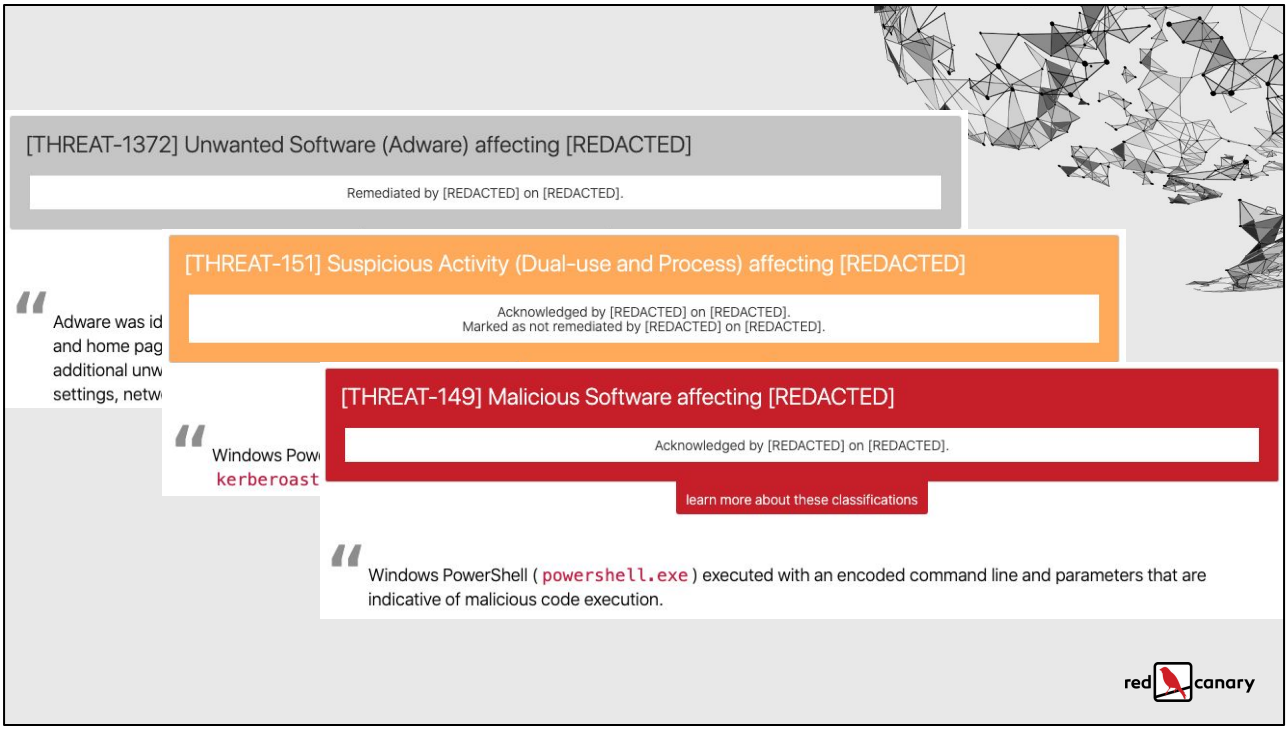
that this person is angry?

Start simple.

Here are some things we do today.

red canary

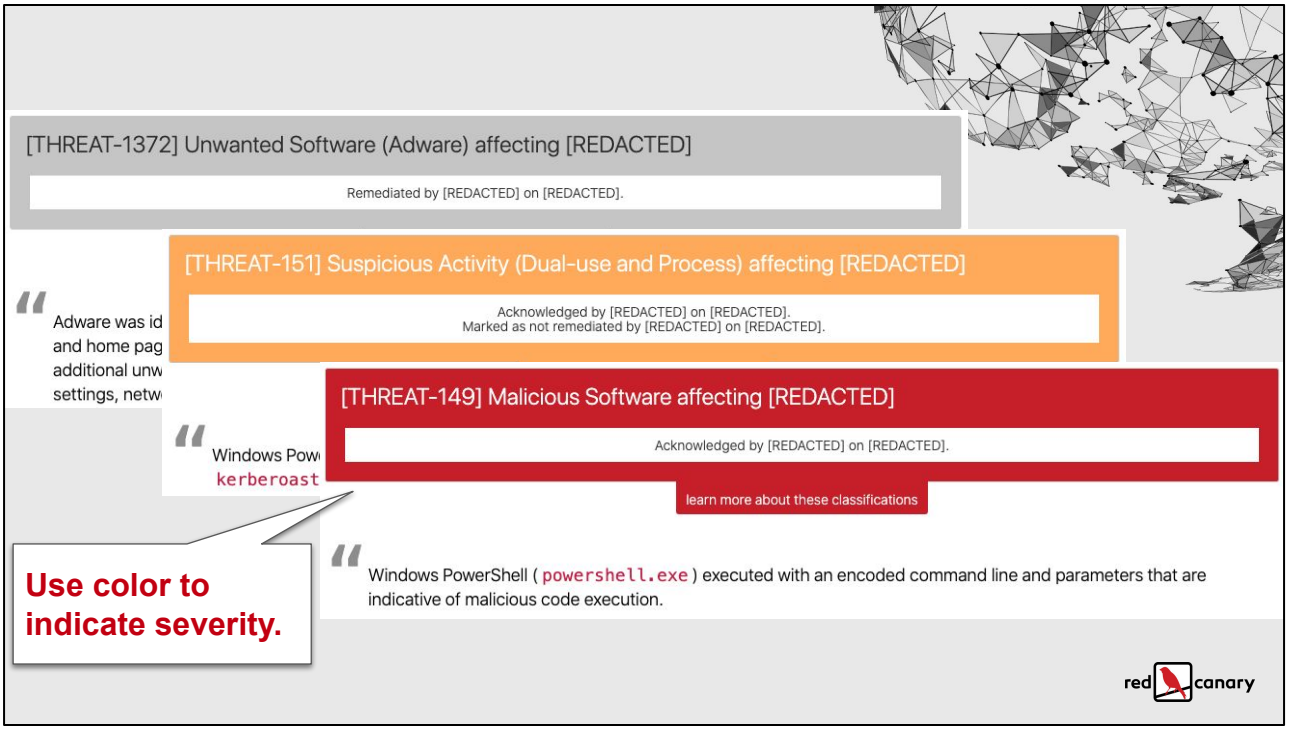
I don’t have all the answers, but I can provide some examples of things we’re doing and would appreciate any input any of you may have.



When we publish detections, the color of the banners corresponds to the severity of the detection. Unwanted software (Adware) is published with a grey banner and customers have the option to disable alerts of PUPs.

Activity that we deem to be of medium consequence is published with a yellowish / orange banner and activity that we believe is of high consequence is published with a red banner.

Each detection has a high level threat description, typically a sentence or two that captures the essence of the activity.



When we publish detections, the color of the banners corresponds to the severity of the detection. Unwanted software (Adware) is published with a grey banner and customers have the option to disable alerts of PUPs.

Activity that we deem to be of medium consequence is published with a yellowish / orange banner and activity that we believe is of high consequence is published with a red banner.

Each detection has a high level threat description, typically a sentence or two that captures the essence of the activity.

Activated ideas

“What you see is all there is” -- maps have gaps

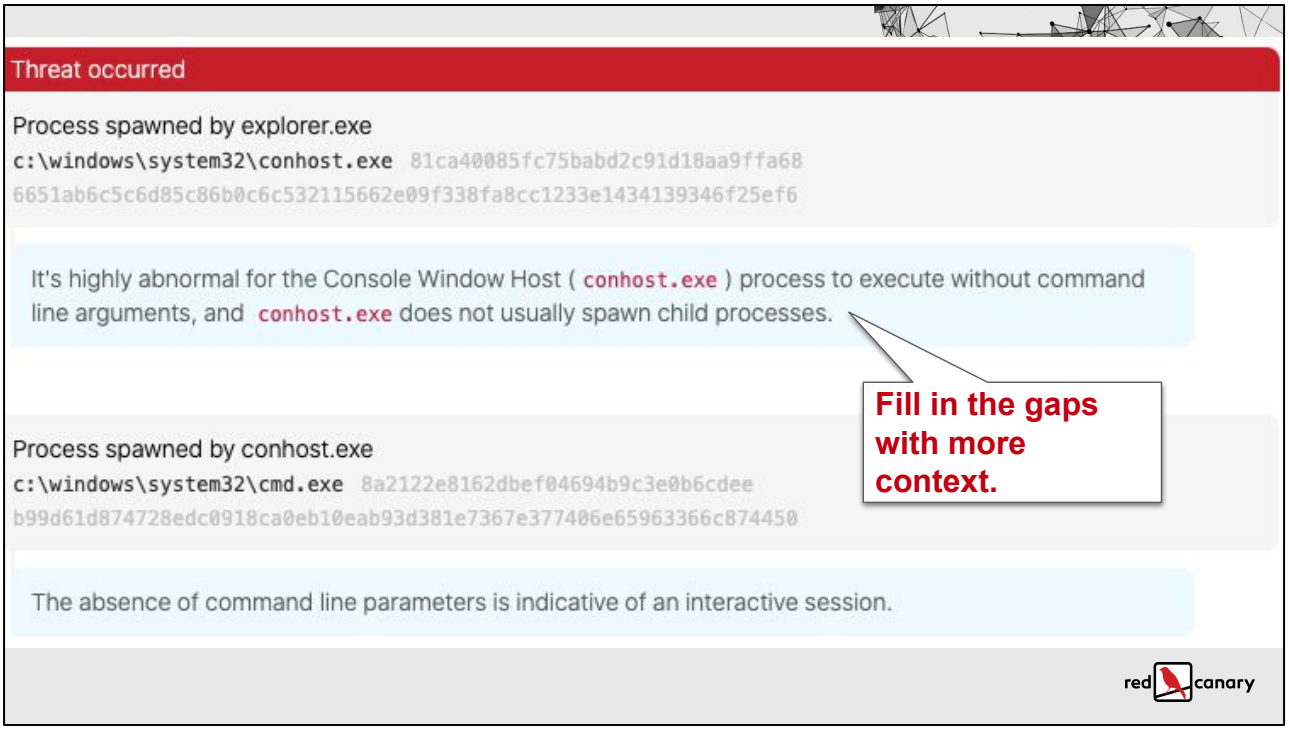
Activated ideas comes from this quote: “An essential design feature of the associative machine is that it represents only activated ideas. Information that is not retrieved (even unconsciously) from memory might as well not exist. System 1 excels at constructing the best possible story that incorporates ideas currently activated, but it does not (**cannot**) allow for information that it does not have.” **Emphasis** mine.

Kahneman refers to this as **“What you see is all there is.”** When you’re looking at a problem, you can’t bring to bear knowledge that you don’t have. Recall that EDR platforms are maps and maps are lossy, they can’t give you all of the information and some of the information they leave out may be critical to understanding the situation at hand.

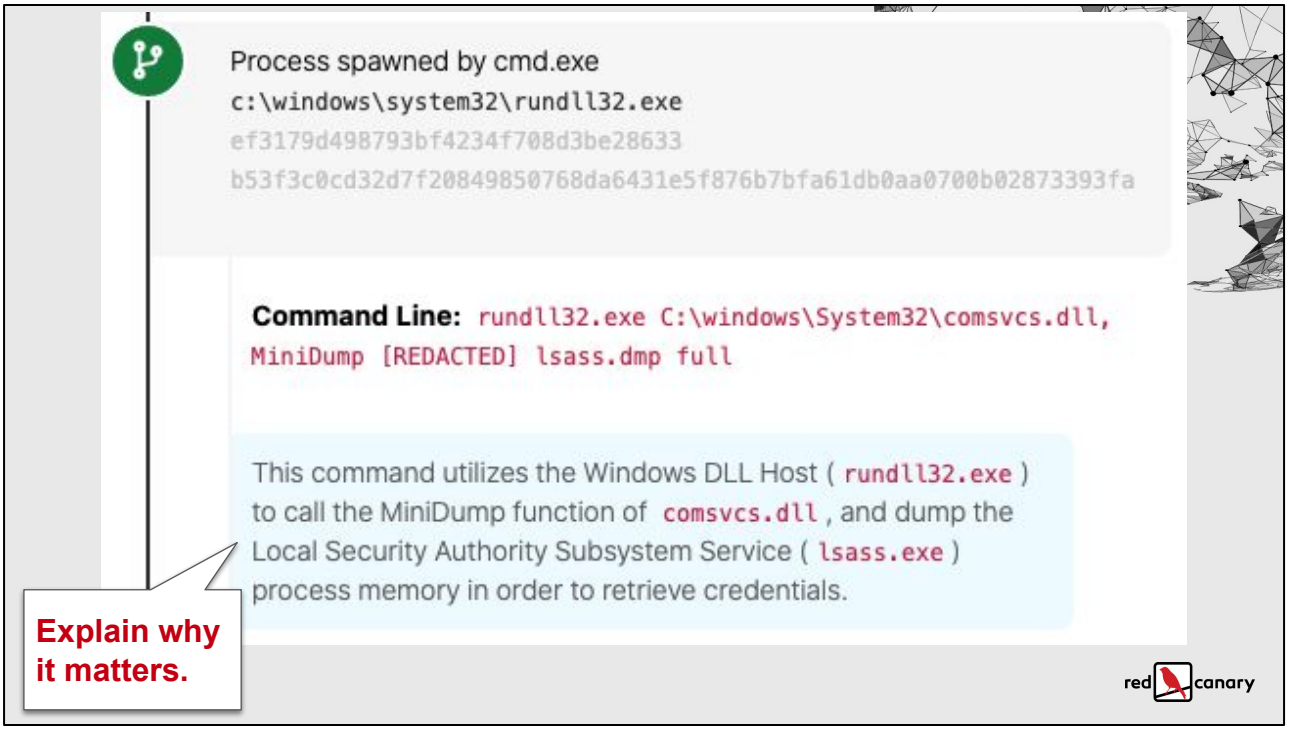
In the clip above, we see that explorer.exe spawned conhost.exe and that conhost.exe spawned cmd.exe. If you aren’t intimately familiar with conhost.exe and that it is the console host application in Windows and runs as a child process of Microsoft’s console based applications, you wouldn’t have the knowledge to even recognize that something strange has happened. Your explanation for this may be that software is buggy and that either this resulted from a Windows bug or a bug in the EDR tooling, end of story.

On the other hand, if you have knowledge or process injection, you know that attackers can inject arbitrary code into processes of their choosing (assuming rights issues aren’t blocking them). If an attacker can inject into explorer.exe, they could cause it to spawn conhost.exe, they could in turn inject into conhost.exe and cause it

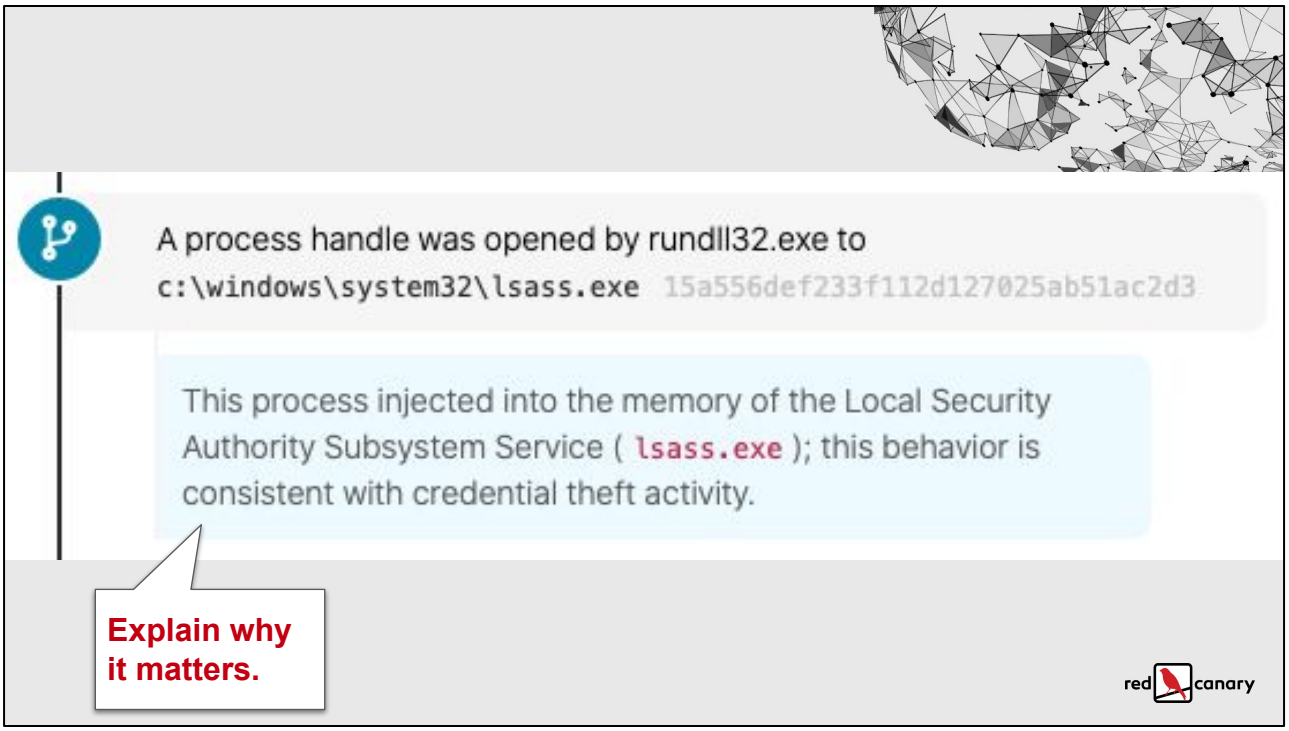
to spawn cmd.exe. If you know about process injection, you know this is possible. It still seems unlikely, as it introduces noise that may trigger detections as it did in this case.



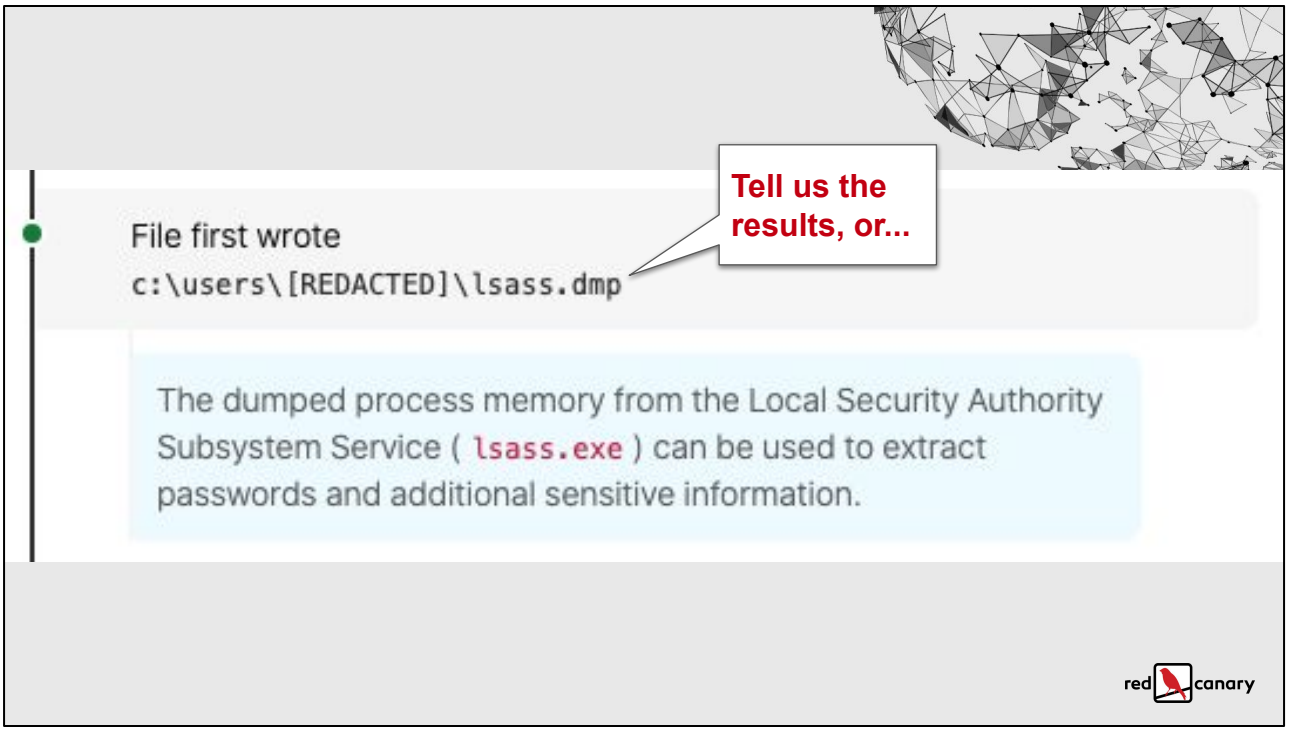
Our analysts add context via annotations to the timeline to help customers understand the implications of the activity.



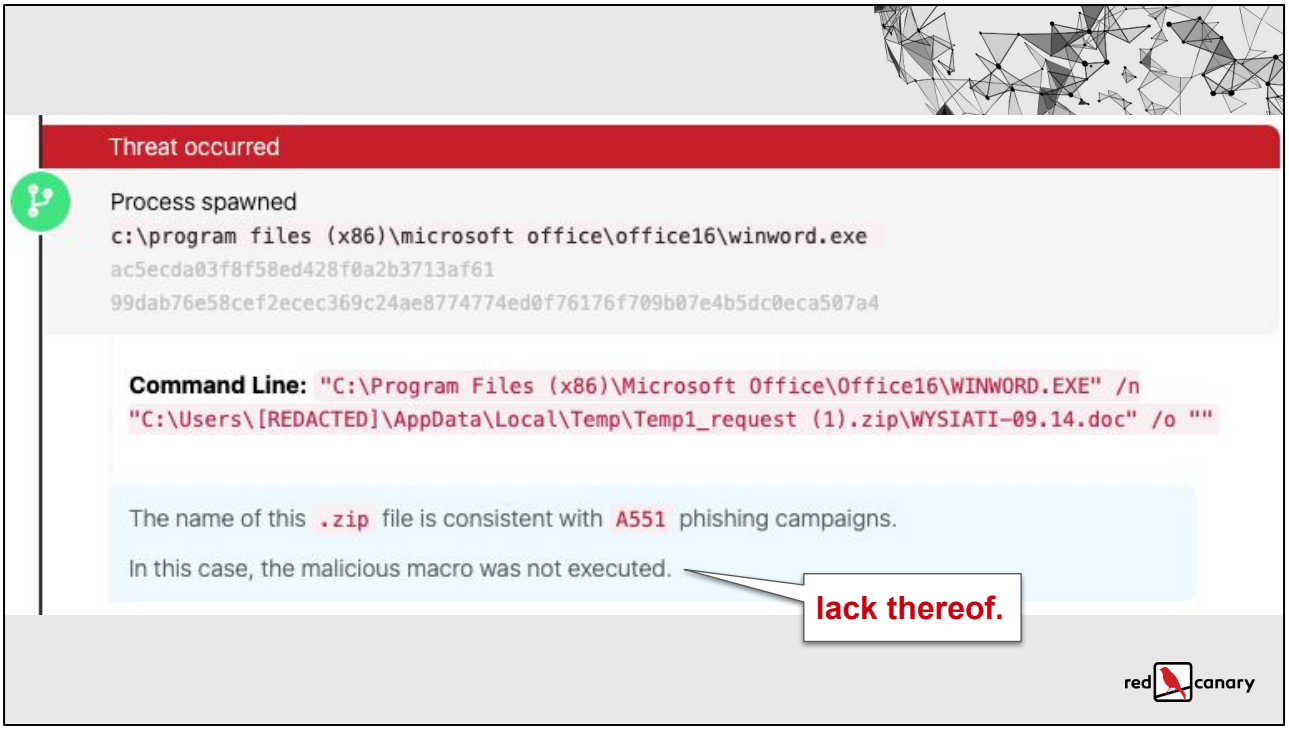
The annotations provide additional context. Here we see an example of rundll32.exe being used with the comsvcs.dll's minidump functionality to create a memory dump of lsass. The annotation explains exactly what is happening and how it is useful to attackers.



From the same timeline as the previous detection, here we see that rundll32 has successfully opened a handle to the lsass process and again the annotation explains what's happening and why.



Lastly we show the reader that the dump file was created and again the annotation explains what's happening and why it matters.



Lastly we show the reader that the dump file was created and again the annotation explains what's happening and why it matters.

Base rates / regression to mean

Why intel matters...

- Direct Qbot phishing lures in early 2021 have consisted of ZIP attachments containing an macro-laden XLS dropper
    - Some campaigns followed the pattern of 1-2 words (seprated by dash or underscore), followed by 6-12 digits, followed by the date (MMDDYYYY), with the same name for both ZIP and XLS file, for example:
      `refusal-778578283-12072020.zip\refusal-778578283-12072020.xls`

red canary

Understanding base rates and regression to the mean gives defenders the ability to anticipate attacker objectives and goals. Red Canary's intel team doesn't do attribution to specific attacker groups, but they do carefully monitor attacker techniques, tactics and procedures and associate those TTPs to clusters of activity. As you can see from our Qbot profile, we know what Qbot phishing lures typically look like.

Base rates / regression to mean

Why intel matters...

- As of June 2021, Qbot uses `regsvr32.exe` to register a randomly named DLL located in a subfolder of the `AppData\Roaming\Microsoft` directory, for example, `regsvr32.exe -s "C:\Users\[redacted]\AppData\Roaming\Microsoft\[random_a-z_chars]\[15_random_lowercase_a-z_chars].dll`.

red canary

Based on our observations, we know Qbot often uses the technique described above.

Base rates / regression to mean

Why intel matters...

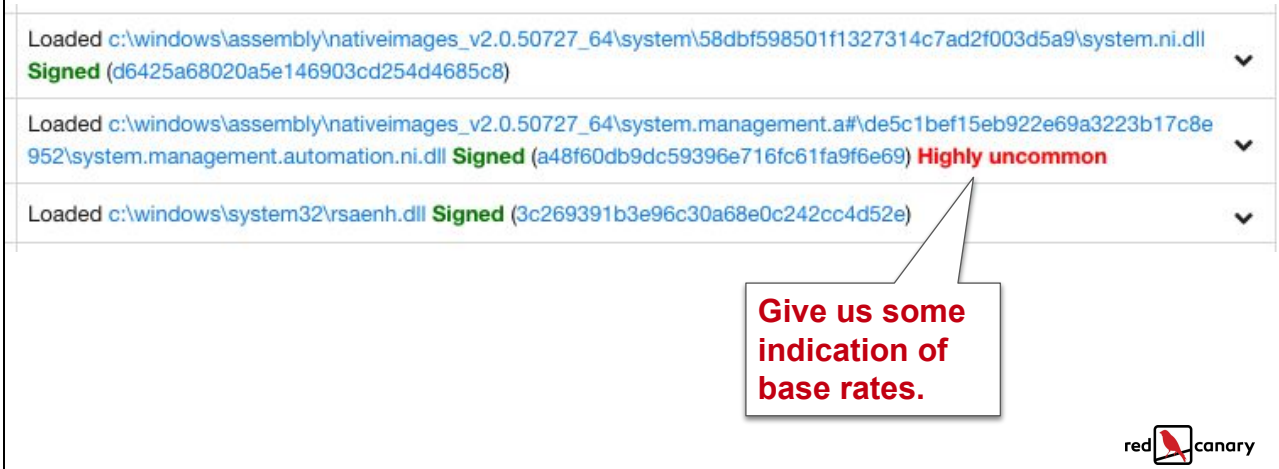
Qbot was historically seen in conjunction with other malware such as Emotet or TrickBot. These cases often begin with Emotet as the initial infection vector followed by delivery of Qbot, TrickBot, or ransomware variants as follow-on payloads. If Qbot is present in an environment, these other malware families may be present as well. In late 2020, environments impacted by Egregor ransomware have been observed to also be infected with widespread, laterally moving Qbot infections.

red canary

When we see these kinds of activities in customer environments, we know because of base rates or what typically happens, that ransomware is likely -- thus we can escalate detections as appropriate.

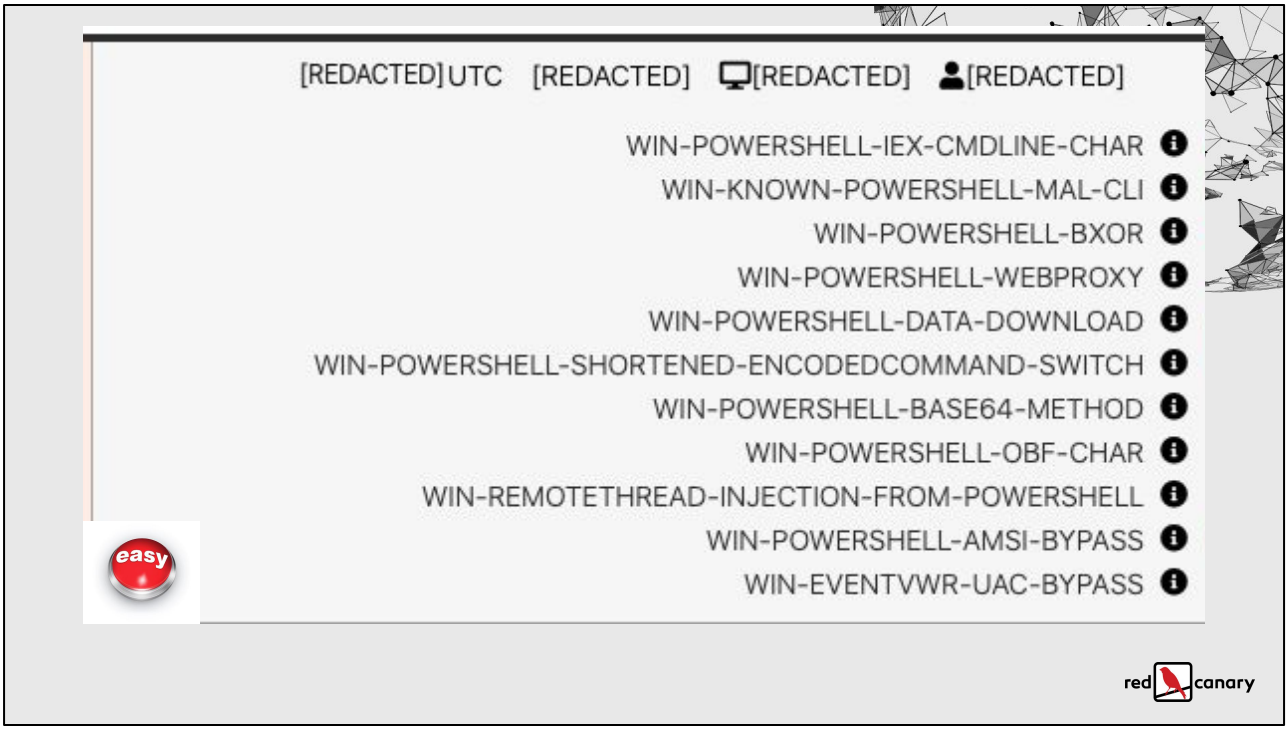
Similarly, if you see suspicious activity on a workstation in your environment understanding base rates can help you prioritize and pursue investigative leads in the proper order. If the activity is present on one workstation (so far) and if phishing is the most common attack vector into enterprise networks, it probably makes sense to review emails for anything suspicious, browser history and egress web traffic for the endpoint in question. If the activity is present on many systems in the environment, lateral movement may be at play and you'll want to identify the credentials in use, relevant authentication logs, attempt to identify patient zero and what the vector of attack was into that system (e.g. phishing, exposed RDP endpoint with a bad or re-used credential?).

Base rates / regression to mean

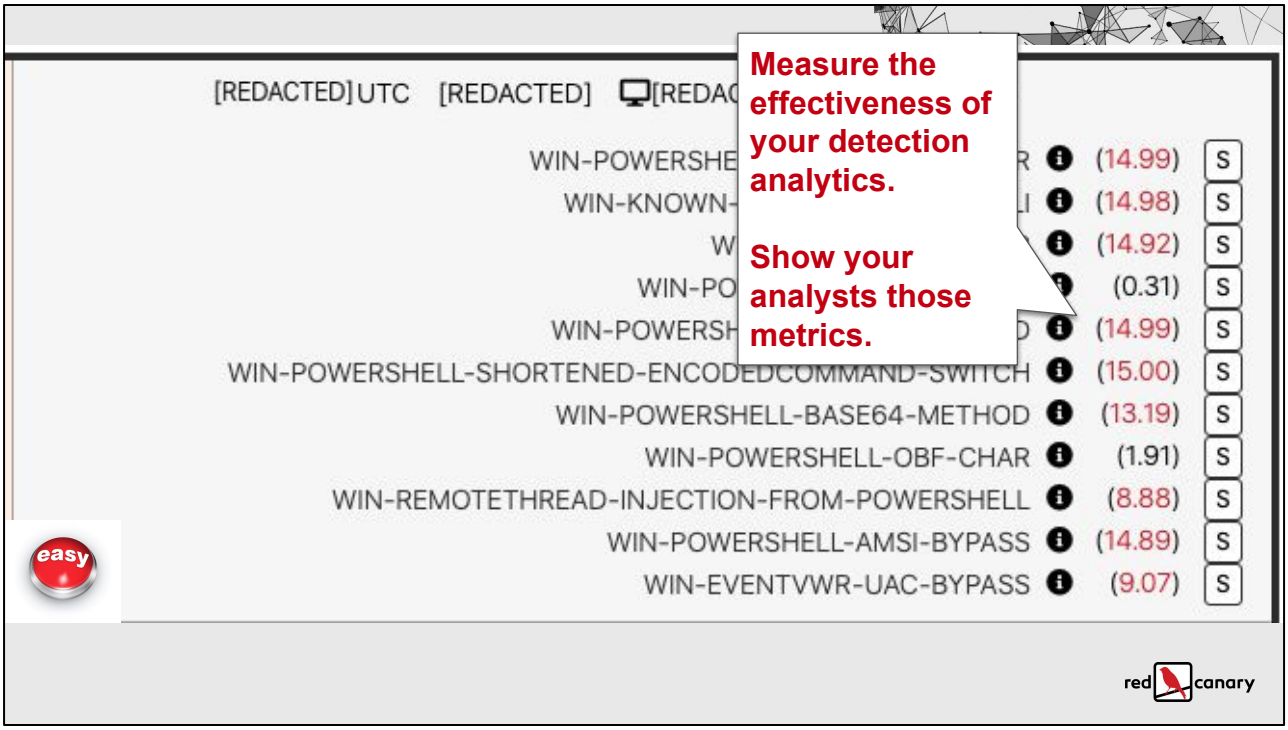


All platforms should provide some information about base rates of the data they present. It's great that they give us all the modloads for a process and tell us that they are signed or unsigned, but they should do more and tell us how common or uncommon it is for a given process to load a given dll or spawn a given child process or make a network connection, etc.

Why should detection engineers have to go to sites like https://echotrail.io to look up common parent/child process relationships, when our EDR platforms have this data and could enrich their presentation layers with it?



The enrichments shown so far are meant to help the recipients of published detections, but we also want to make life easier for detection engineers. The slide above was shown earlier in the presentation, but this is a cropped version of what the detection engineer actually sees. The next slide shows the list of detections that triggered on this event without the cropping.



This version is what the detection engineers see. Note the presence of the numeric values inside parentheses. This value represents the “expected impact” of the given analytic and is based on the previous conversion rate for that analytic multiplied by the analytic’s classification where classification is malicious or suspicious. A score of 15 means the analytic has a 100% conversion rate for malicious detections.

Quick recap

- Maps v territories
- Systems of cognition
- Making things easier

red canary

Let’s quickly recap the key points.

Maps v territories

- **EDR platforms are maps**
  - Defenders want the territory (all the data)
  - Maps **are lossy**
  - Maps **require interpretation** (maps of maps of maps)
  - Maps **may be incorrect**
  - Lossiness leads to "**what you see is all there is**"

red canary

We began by looking at some endpoint behavior that was potentially, but not obviously malicious. I postulated that part of what makes it difficult to adjudicate these events one way or another is that our EDR platforms are maps when what we want is the territory, that is to say, we want all of the data (e.g. full packet captures, full memory dumps, full disk images and the kitchen sink).

Instead of getting the territory, we get maps. Maps are lossy, they have to be. Lossiness means that some information is missing and this information may be critical to our investigations. In addition, maps require interpretation (we need maps of the maps of the maps of the maps, it's maps all the way down). Maps can be wrong. We've seen examples.

The incorrectness or lossiness of our maps contributes to Kahneman's "what you see is all there is" issue -- we're prone to believe what the maps tell us and if we don't have a good understanding of how operating systems and attack techniques work, it may be impossible for us to fill in the gaps in the story. We know from Kahneman's work on systems of cognitions that neither system one nor system two is good at considering information it doesn't have.

Systems of cognition

- **System one and system two**
  - System one is automatic, **jumps to conclusions**
  - System two is expensive, **believes system one**
  - Neither system is aware of what it isn't aware of -- "**what you see is all there is**"
  - System two tasks **+ experience ~=** system one tasks

red canary

In addition to the issues with maps our systems of cognition didn't evolve to do the difficult work that we're routinely tasked with as defenders. We briefly considered Kahneman's system one and system two concepts. We saw system one in action with instantly recognizing the expressions on people's faces and we learned about some of the issues that arise as side effects of the interplay between aspects of systems one and two.

We know system one jumps to conclusions with no awareness of how big those jumps are. We know system two is prone to believe system two. These conditions lead to a variety of interesting biases -- halo effect, frequency, primacy and recency biases, etc.

As mentioned previously, we know maps have gaps and we learned from Kahneman about the principle of "what you see is all there is" -- in other words, neither system one nor system two are capable of taking into account what they do not know.

On a positive note, we learned that difficult system two tasks like playing chess can become system one tasks with enough repetition and practice. Experience, practice and study can also reduce the "what you see is all there is" issue as we learn more about what is possible.

Making things easier

- **Minimize reliance on System two**
  - **Color code** and **Annotate** detections
  - **Measure** detection analytics
  - **Show** the analytic scores
  - **Study** OS internals
  - **Practice** incidents and attacker TTPs
  - Understand and present **base rates** and **regression to the mean**

red canary

Lastly, we looked at some things we can do to make things easier. Primarily we want to minimize the reliance on system two.

We can do this by:

- color coding detections based on criticality
- Annotating detections to provide missing context
- Measuring detection analytics over time
- Displaying those metrics to detection engineers
- Studying OS internals to understand how things work
- Practicing incidents and attacker TTPs to understand how attackers may take advantage of OS internals
- Understanding base rates and regression to the mean
  - Knowing how most attacks begin can inform priorities for investigations
  - Knowing that a specific set of attack techniques that have been a pre-cursor to specific follow-up activity (ransomware) can help defenders prioritize response actions

What else?

- We’re really just scratching the surface.
- What are you doing?
- What else should be done?

red canary

Thank you!

Q & A

PDF of slides available at...

red canary