RESOURCES • BLOG

THREAT DETECTION

Babysitting child processes

Our third Diary of a Detection Engineer explores why baseline knowledge of common executables—such as whether they normally

GET A DEMO ›

MATT GRAEBER • SARAH LEWIS

Originally published June 16, 2021. Last modified April 30, 2024.

In early March 2021, an event fired based on `scrcons.exe` spawning a child process. The child process was `dllhost.exe` with no command-line arguments. Sarah Lewis, the detection engineer working the queue at the time, needed to quickly make sense of the event. She knew that lacking sufficient context, her series of questions in the resulting triage that would follow would be invaluable when she ultimately decided to ship a detection to the customer. In order to arrive at her conclusion though, Sarah needed to be armed with a healthy dose of suspicion and an ability to translate that suspicion into a confident situational awareness of `scrcons.exe`, `dllhost.exe`, and the overall customer environment.

Read on to find out how Sarah handled this unusual child process in our **Diary of a Detection Engineer** spin-off: The SOC Analysts Club!

What is `scrcons.exe`?

No detection engineer can be intimately familiar with every conceivable system executable. Even with knowledge of a subset and how attackers may abuse them, SOC analysts working the queue know very well that executables quite often exhibit behavior that runs counter to our understanding. Baseline knowledge of the

Some background information on `scrcons.exe`:

- It is located in the `%windir%\System32\wbem` directory. "wbem" stands for "Web-Based Enterprise Management" and this directory houses core components related to **Windows Management Instrumentation** (WMI), which is Microsoft's implementation of the WBEM standard
- It has the following file description: `WMI Standard Event Consumer - scripting`. An event consumer is a WMI component that performs an action in response to a trigger. In the case of `scrcons.exe`, it executes **Windows Script Host (WSH) code** (i.e., JScript and VBScript) in response to some event occurring. More specifically, it is the code that implements the **ActiveScriptEventConsumer** WMI event class
- Because WMI is implemented as DCOM, `scrcons.exe` is expected to launch as a child process of the `DcomLaunch` service (command line: `%windir%\system32\svchost.exe -k DcomLaunch -p`), as was the case in this instance (*see image below*)
- It is expected to have the following command line: `%windir%\System32\wbem\scrcons.exe -Embedding`
- The process is expected to execute as **NT AUTHORITY\SYSTEM** and any child processes of `scrcons.exe` are expected to execute with system privileges, which was also the case with `dllhost.exe`

So when `scrcons.exe` launches, it starts in response to the creation of an ActiveScriptEventConsumer instance and will execute registered JScript or VBScript code as a result. To view active, registered ActiveScriptEventConsumer instances with their corresponding script code, run the following PowerShell command:

```
Get-CimInstance -Namespace root/subscription -ClassName ActiveScriptEventCons
```

Our observations have shown that if a child process is spawned from `scrcons.exe`, it implies that the script code in a ActiveScriptEventConsumer instance was responsible for spawning the process, which we observe as uncommon behavior. It was on the basis of how uncommon it is for `scrcons.exe` to spawn child processes that we developed generic detector logic to have an analyst review all related child process creation—as Sarah did in this case. This detector has served us well. It yields a relatively high percentage of events that convert into detections that are shipped to customers. In other words, this detector has a relatively high signal and low volume.

What is `dllhost.exe`?

`dllhost.exe` is a **surrogate executable** designed to launch COM server components in a dedicated process so that if the component crashes, it wouldn't take other COM server components down with it. For detection engineers interested in learning more, **this article** is a good reference.

Some background information on `dllhost.exe`:

- Under normal circumstances, `dllhost.exe` will have the following command line: `dllhost.exe /Processid:{}`
- The AppID CLSID value corresponds to a registered COM server AppID in `HKCR\AppID` in the registry. "`/Processid:{}`" is automatically populated when the `DllSurrogate` value in `HKCR\AppID\{}` is set to a blank string, which is most often the case
- `dllhost.exe` will most often run as a child process of the `DcomLaunch` service (command line: `%windir%\system32\svchost.exe -k DcomLaunch -p`). We have observed exceptions but they are relatively infrequent

Suspicious observations

Along with the background necessary to better understand `scrcons.exe` and `dllhost.exe`, Sarah made the following observations that raised her suspicion level:

- `dllhost.exe` is not a common child process of `scrcons.exe`
- It is uncommon for `dllhost.exe` to launch without command-line arguments
- The `dllhost.exe` process loaded both `samlib.dll` and `vaultcli.dll`, modules that when loaded together often point to credential dumping-related behavior
- The `dllhost.exe` process also obtained a read handle to `lsass.exe`, another **behavioral indicator** related to credential dumping

The above image illustrates the context in which `scrcons.exe` executed `dllhost.exe`

The circumstantial evidence seemed to be pointing in the direction of a credential theft attempt, but there were a few pieces still out of place. First, `dllhost.exe` did not have any unusual file modifications—a potential indicator that credentials were dumped to

relationship. Did this prevalence mean she was looking at some normal, albeit strange, admin-related activity in this customer environment? Or did it mean that multiple endpoints were compromised?

The limited artifacts formed a cloudy image of the activity. Still, a decision had to be made, and the `dllhost.exe` module loads and subsequent interactions with `lsass.exe` were currently outweighing any potential normalcy. On another endpoint in the environment Sarah found the `scrcons.exe` process, which spawned `dllhost.exe`, also spawned an instance of `powershell.exe` with reconnaissance-related command-line arguments. The additional find further increased her confidence that something was off in this environment and the activity warranted further investigation.

> More often than not, there is no smoking gun indicating a malicious event with near 100 percent certainty.

The final call

While the decision-making process that led to classifying this event as suspicious was relatively straightforward, it was only made possible in a timely manner with the following conditions:

- Having a baseline understanding of the sources of behavior associated with the relevant executables, `scrcons.exe` and `dllhost.exe` in this case
- Knowing how to quickly obtain additional situational awareness related to the relevant processes and the timeline of that particular host and the customer's broader environment

More often than not, there is no smoking gun indicating a malicious event with near 100 percent certainty, and this case was no exception. Sarah used her experience, knowledge, and intuition to collect a preponderance of evidence that led this event to lean more towards the malicious side of the spectrum than the benign side. Shortly after shipping the detection, we were informed that it was confirmed red team activity.

that topic can be saved for another day. In this post, we wanted to highlight the rapid decision-making process involved in shipping a detection to a customer.

RELATED ARTICLES

THREAT DETECTION

Artificial authentication: Understanding and observing Azure OpenAI abuse

THREAT DETECTION

Apple picking: Bobbing for Atomic Stealer & other macOS malware

THREAT DETECTION

Keep track of AWS user activity with

THREAT DETECTION

Trending cyberthreats and techniques from the first half of 2024

Subscribe to our blog

You'll receive a weekly email with our new blog posts.

First Name

Last Name

Email Address

SUBSCRIBE ›

GET A DEMO ›