Analysis of a Convoluted Attack Chain Involving Ngrok

By Abraham Camba, Khristoffer Jocson, Gilbert Sison, Jay Yaneza (words)

Published: 2020-09-14 · Archived: 2026-04-05 20:36:18 UTC

One of the primary benefits of an Endpoint Detection and Responseproducts (EDR) security solution is that it gives blue teams (security personnel responsible for maintaining and analyzing an organization's network defenses) the visibility required to detect an attack in its early stages and visualize an incident as it is happening. While these kinds of security technology innovations benefit the cybersecurity industry as a whole, it is often matched by a corresponding evolution in the tools and techniques that malicious actors use as a response.

The Trend Micro ™ Managed XDRproducts team recently handled an incident involving one of Trend Micro's customers. The incident revealed how a malicious actor incorporated certain techniques into an attack, making it more difficult for blue teams and security researchers alike to analyze the chain of events in a clean and easily understandable manner.

Initial Investigation

In July 2020, we noticed the following suspicious event in our customer's environment via the Trend Micro Apex One ™ Endpoint Security Solution:

Process: c:\windows\system32\reg.exe CommandLine: REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v <*value*> /t REG_SZ /d "\"c:\Windows\system32\<*random name*>\"" /f

A few points to note: first, the value name of the registry being created was patterned after a certain security vendor. Second, there was a mistake (or maybe intentional) in the spelling of the registry name (as seen in the <value> portion, which we did not include for privacy reasons). Finally, there is the randomly named executable in the system directory. When considering all of these together, the alert presented us with obvious red flags.

The executable file turned out to be a keylogger that sends the mouse and key clicks it sniffs to a Gmail account. We found hardcoded information in the binary, providing evidence that it was created specifically for the targeted organization. Furthermore, we also learned from the binary that the attackers had existing knowledge about the organization.

We searched records and events using the keylogger's file name and hash and found the following:

File Events

Based on the created analysis chains, the keylogger is dropped by the ntoskrnl.exe process. This tells us that the file was dropped either via network share or through the use of an exploit affecting the kernel.

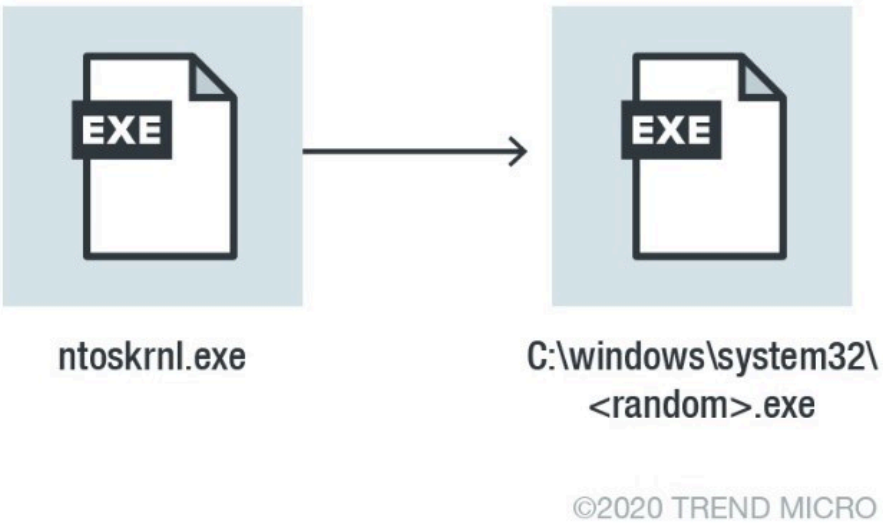

Figure 1. The keylogger is dropped by Ntoskrnl.exe

Events With Command-Line Parameters

While it was not surprising to find reg.exe process events containing the identified random file name (given that it triggered the investigation), the chain of processes that preceded reg.exe was more difficult to predict.

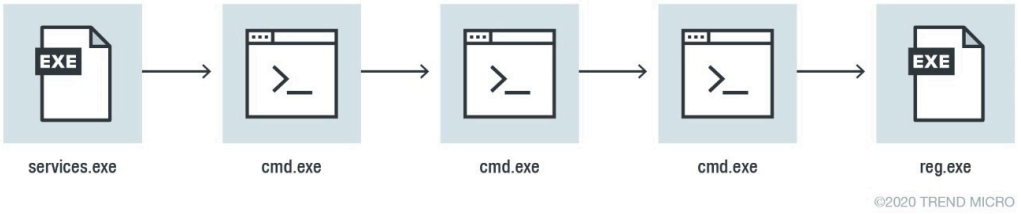

Figure 2. The chain of events leading up to the reg.exe process

Instead of the keylogger calling reg.exe to generate its persistence mechanism, another process — services.exe — seems to be the root of the chain. This means the attacker managed to create a service that would launch a series of cmd.exe processes to build the persistence mechanism using reg.exe.

Registry Data

During our investigation, we found service registry entries containing the command line parameter passed to reg.exe:

Reg Key: hklm\system\currentcontrolset\services\<random>
Reg Value: imagepath
Reg Data: %COMSPEC% /C echo REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v <*value*> /t REG_SZ /d "\"c:\Windows\system32\<*random*>\"" /f ^> %SYSTEMDRIVE%\WINDOWS\Temp\<*random*>.txt > \WINDOWS\Temp\<*random*>.bat & %COMSPEC% /C start %COMSPEC% /C \WINDOWS\Temp\<random>.bat

This registry entry explains the multiple cmd.exe processes in the chain leading to the reg.exe shown above. The command line of the first cmd.exe will contain the string after %COMSPEC%, while the second will contain the string after the second %COMSPEC%, and so on, in sequential order.

Digging Deeper

The act of creating a service to launch a command was uitlized for implementing the shell portion of a backdoor shell. Using the string %COMSPEC%, we searched and examined other service registry entries, eventually discovering multiple tools and commands that were executed using this method.

We observed commands that were derived from service registry entries like “query user”, “net user <user account> /domain”, “ping <ip address>”, etc. on multiple machines. If a new backdoor shell was indeed implemented, it meant that we still needed to find the component that communicates with the outside world, as well as the part that creates the service entries.

We got our first break when we found the command “<random> tcp --authtoken <token> -config <config file> <ip>:445”. Based on the parameters alone, this particular tool looked like it had something to do with network communication, utilizing a common port (SMB/TCP 445) that is widely used within a corporate environment.

After collecting the file, we determined that the tool is a copy of ngrok, a software program that allows an internal machine to be visible to the outside world by routing the traffic through the ngrok website. The existence of the tool on two machines meant that those machines were externally visible. We have previously written how ngrok can be usednews- cybercrime-and-digital-threats for malicious purposes.

How the service entries are created is still a mystery at this point. We searched instances of psexesvc.exe (the server part of PsExec) and remotesvc.exe, but the result of indicator of compromise (IOC) sweeping was inconclusive. However, we did discover the string “admin/smb/psexec_command” in one of the service entries, which resembles the command execution of Metasploit. What we know is that Metasploit’s PsExec module does not drop a binary on the target machine. Further research shows that some versions of Mimikatz and Impacket do possess such a feature.

Simulating the Attack

To try to figure out the attack’s mechanics, we performed a simple simulation by installing ngrok on one of the machines (Machine A) that was neither visible nor accessible externally.

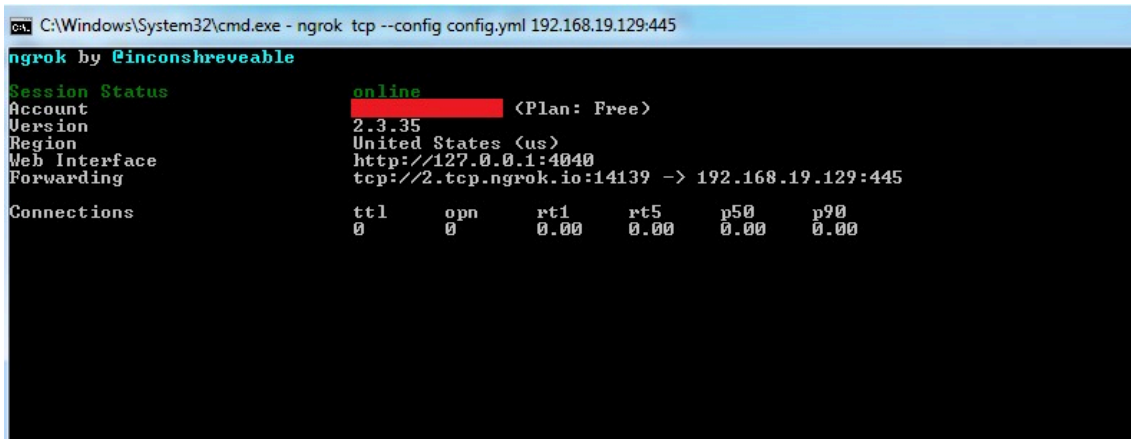

Figure 3. Installing ngrok on Machine A

Ngrok can expose to the internet any open IP port within the internal network that is accessible to Machine A (including itself). In our example, ngrok exposed another machine (Machine B) 192.168.19.129:445 through the ngrok server. We were then able to access 192.168.19.129:445 via 2.tcp.ngrok.io:14139.

By using the service Smbexec module of the Impacket tool kit and the credentials of Machine B, we were able to send simple ping commands to Machine B from an external machine.

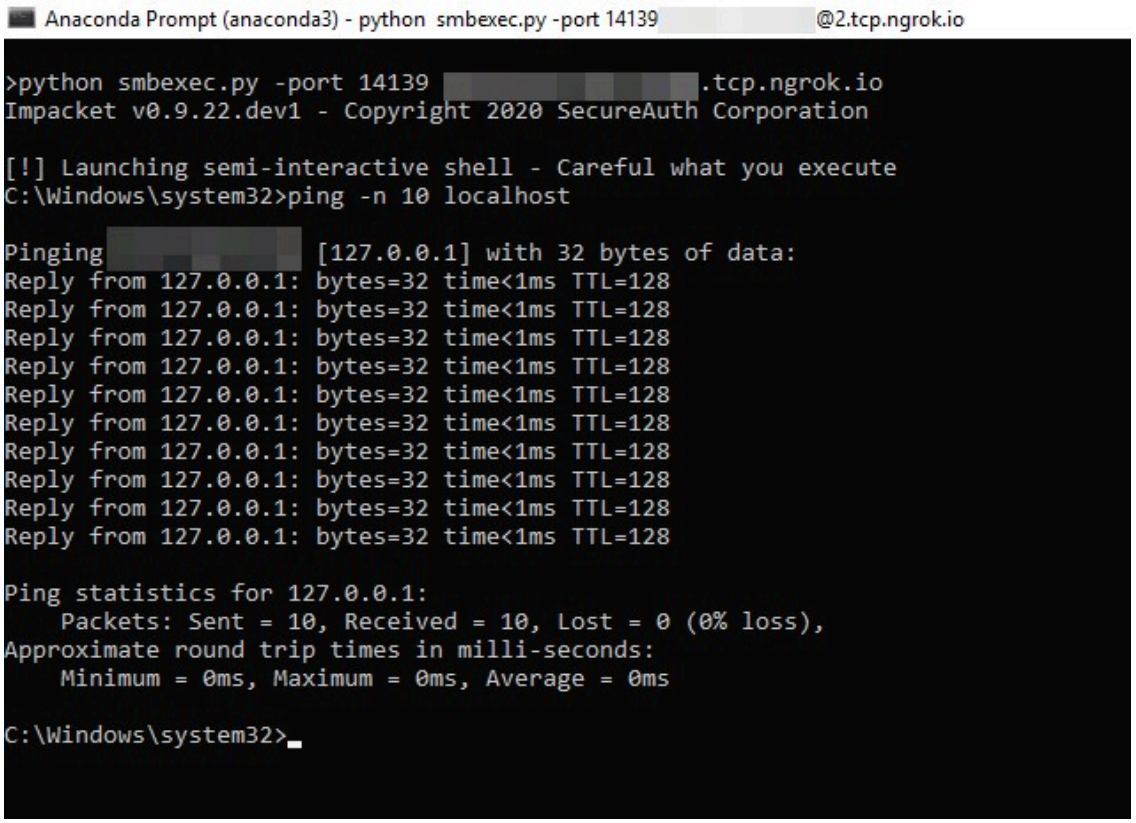

Figure 4. Using an external machine to send ping commands to Machine B

The resulting behavior was similar to what we observed in the customer's environment, where randomly named service entries were created and then deleted. The commands were executed without needing to drop a binary on the target machine.

Furthermore, since the commands were executed as a service, it runs with elevated privileges. Given that network traffic was tunneled via the ngrok service, the command and control server was effectively hidden. As long as the attacker knows the ngrok-assigned public address, it can connect to the compromised endpoint from anywhere, at any time.

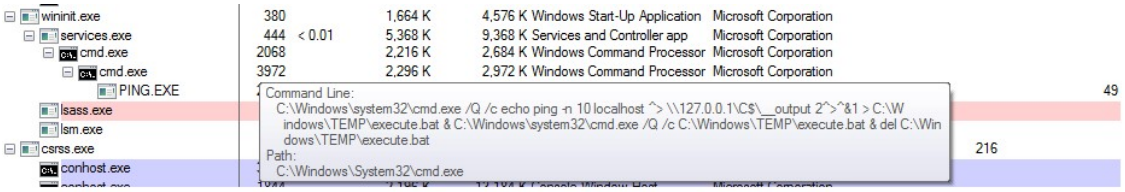

Figure 5. Connecting to a compromised machine using the ngrok-assigned public address

While we did not expect the simulation to recreate the actions of the attacker completely, it provided valuable context on how the attack possibly happened.

In the case of the simulation, it required ngrok installed on the internal machine, the ngrok server domain and port, and an administrator account. We believe that the attacker possessed all three, given that they were the ones who installed ngrok on the machine, and they seemed to have been present long enough to know certain details about the environment. They were also able to compromise a high-privilege account. Based on this, the simulation we conducted fits the attack characteristics.

Fighting Back With EDR

The diagram in Figure 6 shows a typical backdoor shell root cause analysis chain.

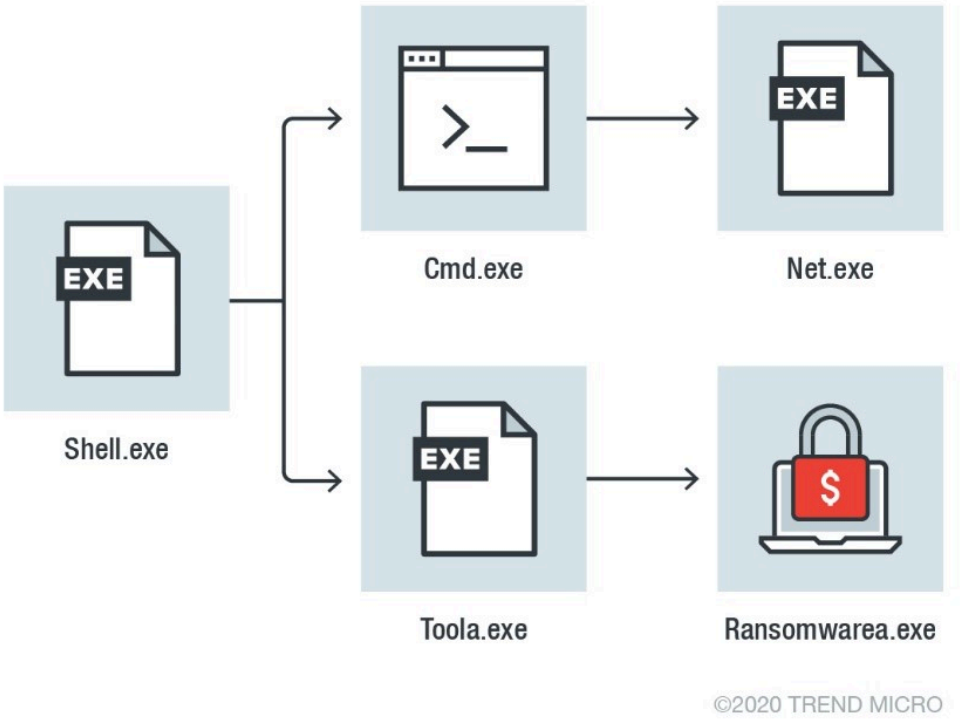

Figure 6. Root cause analysis chain of a typical backdoor shell

Shell.exe launches cmd.exe, which then launches the tool to execute the given command. It could also be shown launching tools it has installed — such as Toola.exe — in the diagram. This kind of diagram is straightforward, making it easy to identify suspicious objects and determine the basic flow of how things work in an attack.

In this incident, the root cause analysis begins with services.exe and ends with the executed tool or command. There was no evidence that a multi-stage tool that drops other tools were ever used, and based on the access the attackers had in this model, it is highly probable they had no use for such a tool. The machines were so accessible that the attackers could run any tool they needed without having to think of clever mechanisms to install the binary (such as moving it laterally from one machine to another) — in other words, tools can be run virtually on demand. Take, for instance, the installation of the keylogger; the attackers dropped the keylogger file via server message block (SMB) and issued a separate command to create its persistence mechanism. They did not use a keylogger that creates its own persistence mechanism, likely because it is not difficult for them to create the registry entry for persistence remotely.

Meaningful root cause analysis chains are difficult to come by because everything starts with services.exe (or another windows process in cases of file dropping, for instance) running one command/tool at a time. The resulting chain can look more like a “tree” with services.exe at the center, where each branch represents a command executed through services.exe.

Our analysis shows that the technique used in the attack hinders the ability of security researchers to piece together the sequence of events via a short diagram. However, certain features of EDR are designed to handle incidents like this.

Mitigation Through Suspicious Events

Suspicious events are effective triggers of an EDR solution, and the capability to mitigate through the same solution is ideal for blue teams. In this incident, Trend Micro Managed XDR utilized the Apex One capabilities to both investigate and mitigate the threat in the same software suite.

Investigation Through Logged Events

Conventional incident response methodology would often require running a tool to acquire evidence from a suspect host. In this investigation, everything was done through the examination of EDR logged events. No memory or disk image acquisition was done for the investigation, which means that the data collected by EDR was enough to determine how similar attacks work. The chronological order of the commands were taken from the time stamps of the events. Even without the self-explanatory diagram EDR creates, it is still possible to determine how the attack took place.

New Alerts

EDR allows for the painless creation of alerts to trigger an investigation. In this case, new alerts can be created whenever services.exe launches cmd.exe and when %comspec% is written to an autostart registry entry, which can help future threat hunting capabilities for blue teams.

Trend Micro Solutions

The Trend Micro XDRproducts solution protects connected emails, endpoints, servers, cloud workloads, and networks by using powerful AI and security analytics to correlate data and provide an optimized set of alerts via a single console. With this technology, organizations can quickly identify threats and remediate their impact in a timely manner.

Trend Micro Managed XDRproducts offers expert threat monitoring, correlation, and analysis from experienced cybersecurity industry veterans, providing 24/7 service that allows organizations to have one single source of detection, analysis, and response. This service is enhanced by solutions that combine AI and Trend Micro's wealth of global threat intelligence.

Tags
