UAT-9244 targets South American telecommunication providers with three new malware implants

By Asheer Malhotra

Published: 2026-03-05 · Archived: 2026-04-05 12:53:36 UTC

- Cisco Talos is disclosing UAT-9244, who we assess with high confidence is a China-nexus advanced persistent threat (APT) actor closely associated with Famous Sparrow.
- Since 2024, UAT-9244 has targeted critical telecommunications infrastructure, including Windows and Linux-based endpoints and edge devices in South America, proliferating access via three malware implants.
- The first backdoor, "TernDoor," is a new variation of the previously disclosed, Windows-based, CrowDoor malware.
- Talos also discovered that UAT-9244 uses "PeerTime," an ELF-based backdoor that uses the BitTorrent protocol to conduct malicious operations on an infected system.
- UAT-9244's third implant is a brute force scanner, which Talos tracks as "BruteEntry." BruteEntry is typically installed on network edge devices, essentially converting them into mass-scanning proxy nodes, also known as Operational Relay Boxes (ORBs) that attempt to brute force into SSH, Postgres, and Tomcat servers.

Introducing TernDoor: A variant of CrowDoor

UAT-9244 used dynamic-link library (DLL) side-loading to activate multiple stages of their infection chain. The actor executed "wsprint[.]exe", a benign executable that loaded the malicious DLL-based loader "BugSplatRc64[.]dll". The DLL reads a data file named "WSPrint[.]dll" from disk, decrypts its contents, and executes them in memory to activate TernDoor, the final payload.

TernDoor is a variant of CrowDoor, a backdoor deployed in recent intrusions linked to China-nexus APTs such as FamousSparrow and Earth Estries. CrowDoor is a variant of SparrowDoor, another backdoor attributed to FamousSparrow. CrowDoor has also been observed in previous Tropic Trooper intrusions, indicating a close operational relationship with FamousSparrow. Based on the overlap in tooling; tactics, techniques, and procedures (TTPs); and victimology, we assess with high confidence that UAT-9244 closely overlaps with FamousSparrow and Tropic Trooper.

Although UAT-9244 and Salt Typhoon both target telecommunications service providers, Talos has not been able to verify or establish a solid connection between the two clusters.

The DLL-based loader

The DLL-based loader, "BugSplatRc64.dll", will load the "WSPrint.dll" file from the current directory, which will be decoded using the key "qwiozpVngruhg123".

```
xor     ecx, ecx        ; lpAddress
mov     edx, 100000h    ; size
mov     r8d, 3000h      ; flAllocationType
lea     r9d, [rcx+(PAGE_EXECUTE_READWRITE)] ; flProtect
call    rax ; VirtualAlloc
mov     rbp, rax
mov     rax, cs:ReadFile
test    rax, rax
jnz     short loc_7FFA6BE312D9
mov     ecx, 44FB28C3h
call    _GetProcAddress_
mov     cs:ReadFile, rax

9:                      ; CODE XREF: decode_and_call_next_stage+13
lea     r9, [rsp+778h+NumberOfBytesRead] ; lpNumberOfBytesRead
mov     qword ptr [rsp+778h+dwCreationDisposition], rsi ; lpOverla
mov     r8d, 100000h    ; nNumberOfBytesToRead
mov     rdx, rbp        ; lpBuffer
mov     rcx, rbx        ; hFile
call    rax ; ReadFile
mov     rax, cs:CloseHandle_0
test    rax, rax
jnz     short loc_7FFA6BE3130E
mov     ecx, 0C2B8DE85h
call    _GetProcAddress_
mov     cs:CloseHandle_0, rax

E:                      ; CODE XREF: decode_and_call_next_stage+16
mov     rcx, rbx        ; hObject
call    rax ; CloseHandle_0
xor     edx, edx        ; Val
lea     rcx, [rsp+778h+var_327] ; void *
mov     r8d, 0FFh       ; Size
call    memset
xor     edx, edx        ; Val
lea     rcx, [rsp+778h+var_724] ; void *
mov     r8d, 3FCh       ; Size
mov     edi, esi
mov     ebx, esi
call    memset
movups  xmm0, xmmword ptr cs:aQwiozpvngruhg1 ; "qwiozpVngruhg123"
movzx   eax, byte ptr cs:aQwiozpvngruhg1+10h ; ""
lea     rcx, [rsp+778h+var_118+11h] ; void *
```

Figure 1. DLL-based loader reading the encoded payload.

The decoded shellcode is position-independent and decodes and decompresses the final payload. The final payload is the TernDoor implant.

TernDoor

The final shellcode consists of the TernDoor backdoor. TernDoor is a variant of CrowDoor, actively developed and used by UAT-9244 since at least November 2024. TernDoor deviates from CrowDoor in the following aspects:

- TernDoor consists of command codes that are different from previously disclosed variants of CrowDoor.
- The TernDoor shellcode also consists of an embedded Windows driver (SYS file). The driver is encrypted using AES in the shellcode. The driver is used to suspend, resume, and terminate processes.

Persistence

The TernDoor infection chain is persisted on the system using either a scheduled task or the Registry Run key.

The scheduled task is named “WSPrint” and created using the command:

```
schtasks /create /tn WSPrint /tr "C:\ProgramData\WSPrint\WSPrint.exe" /ru "SYSTEM" /sc onstart /F
```

Furthermore, TernDoor modifies the following task-related registry keys to hide the task:

- Deletes HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\WSPrint | SD
- Modifies HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\WSPrint | Index = from 1 to 0

A Registry Run key may also be set to run the executable on user login:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run | Default = C:\ProgramData\WSPrint\WSPrint.exe
```

Command line switch

Unlike CrowDoor, TernDoor only supports one command line switch: “-u”, passed to WSPrint.exe. This is the switch for uninstalling the malware from the system and it deletes all malware files from the operating directory, as well as terminates malicious processes.

Decoding the configuration

Like previous variants of CrowDoor, TernDoor also checks to ensure it has been injected into “msiexec[.]exe”. The implant decodes its configuration that can specify the following information:

- Command and control (C2) IP address
- Number of tries to connect to the C2
- C2 port number
- User-Agent to use while connecting to C2 (if applicable)

```
00  31 35 34 2E 32 30 35 2E 31 35 34 2E 38 32 00 00  154.205.154.82..
10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
80  10 00 BB 01 00 00 00 00 00 00 00 00 00 00 00 00  ..».............
```

Figure 2. TernDoor configuration blob.

TernDoor functionality

TernDoor's capabilities resemble those of previously disclosed CrowDoor samples:

- Communicates with the C2 IP address
- Creates processes and runs arbitrary commands via remote shell and independently
- Reads and writes files
- Collects system information such as computer and user name, IP address information, and OS bitness
- Uninstalls itself from the infected system
- Deploys the accompanying driver to hide malicious components and perform process management

The accompanying Windows driver, WSPrint.sys, is dropped to disk and then activated using a windows service:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WSPrint]
"ImagePath"="\\??\\C:\\ProgramData\\WSPrint\\WSPrint.sys"
"Type"=dword:00000001
```

Figure 3. Malicious driver service on the infected endpoint.

The driver creates a device named "\\Device\\VMTool" and symbolically links it to "\\DosDevices\\VMTool". It can terminate, suspend, or resume processes specified by TernDoor — likely a means of evasion.

TernDoor infrastructure

All the C2 IP addresses discovered by Talos were associated with the following SSL certificate on port 443:

```
SSL_fingerprint_sha256= 0c7e36683a100a96f695a952cf07052af9a47f5898e1078311fd58c5fdbdecc8
SSL_fingerprint_SHA1= 2b170a6d90fceba72aba3c7bc5c40b9725f43788
Data:
Version: V3
Serial Number: 1
Thumbprint: 2b170a6d90fceba72aba3c7bc5c40b9725f43788

Signature Algorithm:
Issuer: C=US ST=Some-State O=Internet Widgits Pty Ltd CN=8.8.8.8
Validity
Not Before: 2022-09-04 12:54:51
```

```
Not After: 2023-09-04 12:54:51
Subject: C=US ST=Some-State O=Internet Widgits Pty Ltd CN=8.8.8.8
```

Pivoting off this certificate, Talos found an additional 18 IPs likely being used by UAT-9244. This list is provided in the indicators of compromise (IOCs) section.

One of the DLL-based loaders was also hosted on the IP “212.11.64[.]105”. On this server, we discovered a set of shell scripts and an accompanying malware family we track as “PeerTime.”

PeerTime: UAT-9244's peer-to-peer (P2P) backdoor

PeerTime is an ELF based backdoor that is compiled for a variety of architectures such as ARM, AARCH, PPC, MIPS etc., indicating that UAT-9244 can use it to infect a variety of embedded systems.

PeerTime is deployed through a shellscript that downloads the PeerTime loader ELF binary and an instrumentor binary.

The instrumentor ELF binary will check for the presence of docker on the compromised host using the commands docker and docker –q.

If docker is found, then the PeerTime loader is executed using:

```
docker <path_of_PeerTime_loader_ELF>
```

The instrumentor consists of debug strings in Simplified Chinese, indicating that it is a custom binary created and deployed by Chinese-speaking threat actors:

```
获取当前程序路径错误:     //Error retrieving current program path:
删除当前程序错误:                // Error deleting current program:
```

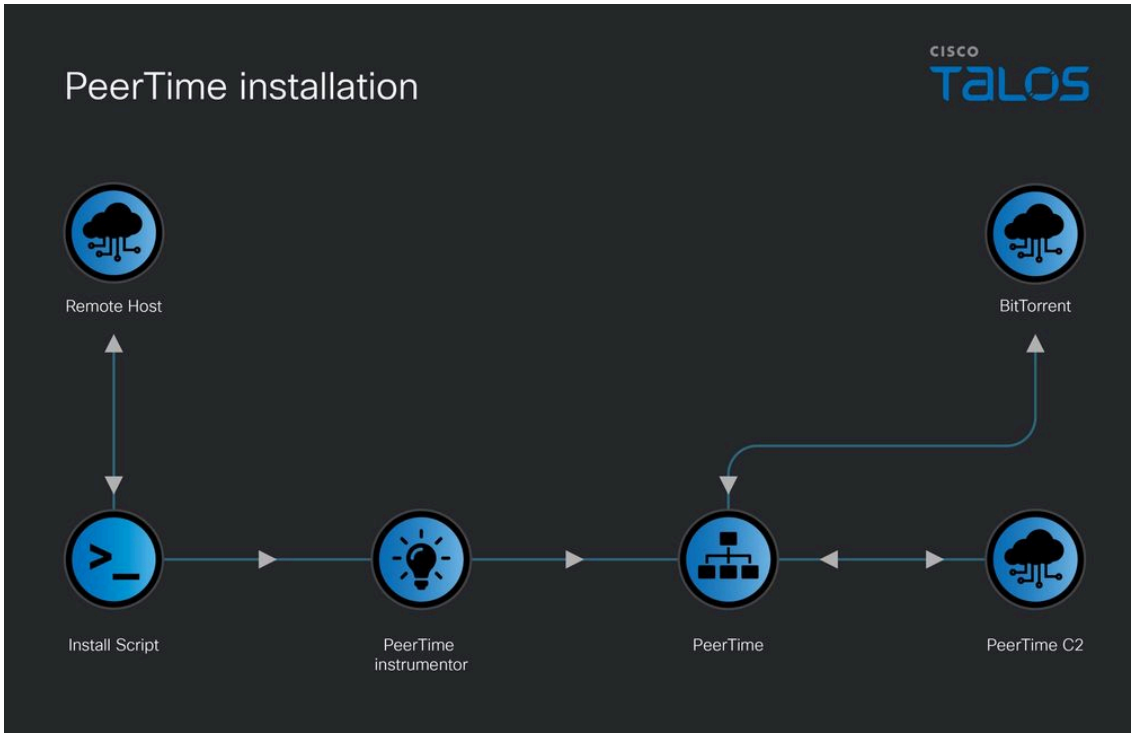

Figure 4. PeerTime installation/infection chain.

PeerTime consists of a loader that will decrypt and decompress the final PeerTime ELF payload and run it in memory. The PeerTime loader has the ability to rename its process to a benign process to evade detection.

PeerTime uses the BitTorrent protocol to obtain C2 information, download files from its peers, and execute them on the infected host. The payloads are written to disk and copied to the specified locations using BusyBox. As of now, PeerTime consists of two versions: one written in C/C++ and a newer version written in Rust.

```
if ( write_file(filename, a1, a2) < 0 || (unsigned int)sub_4973B3(v17, v15) && (unsigned int)sub_4973B3(filename, v15) )
  return 0;
sys_chmod(filename, v16 | 0x40);
v7 = v15;
v8 = sub_4973B3(filename, v15);
if ( !(v8 | v16 & 0x40) )
{
  v11 = sub_49AAF0((__int64)v17);
  v12 = sub_491430(v11 + 21);
  sprintf_0(
    v12,
    (unsigned int)"%s %s %s",
    (unsigned int)"cp",
    (unsigned int)"/bin/busybox",
    (unsigned int)filename,
    v13);
  sys_rmdir_0(filename);
  execute_file_via_bin_sh(v12);
  sys_chmod(filename, v16 | 0x40);
  v7 = a1;
  write_file(filename, a1, a2);
  sub_491140(v12);
}
p_unlink(v17, (__int64)v7);
if ( (unsigned int)p_rename(filename, v17) )
{
  v9 = sub_423AD0();
  return (unsigned int)p_sys_execve(filename, v9) != -1;
}
else
{
  v10 = sub_423AD0();
  return (unsigned int)p_sys_execve(v17, v10) != -1;
}
```

Figure 5. PeerTime uses busybox to copy payloads.

PeerTime is also known as “angrypeer” and can be tracked in VirusTotal using the “malware_config:angrypeer” query. Malware configurations in VirusTotal are identified using Mandiant’s/GTIG’s Backscatter tool.

Setting up ORBs via BruteEntry

Infrastructure used by UAT-9244 also hosts another set of shell scripts and payloads designed to establish compromised Linux based systems including edge devices as operational relay boxes (ORBs) that scan and brute force Tomcat, Postgres, and SSH servers.

The shell script will download two components:

- An instrumentor and daemon process that activates the actual brute forcer
- The actual brute forcer (named BruteEntry) that obtains target IPs from the C2 server and scans the IPs

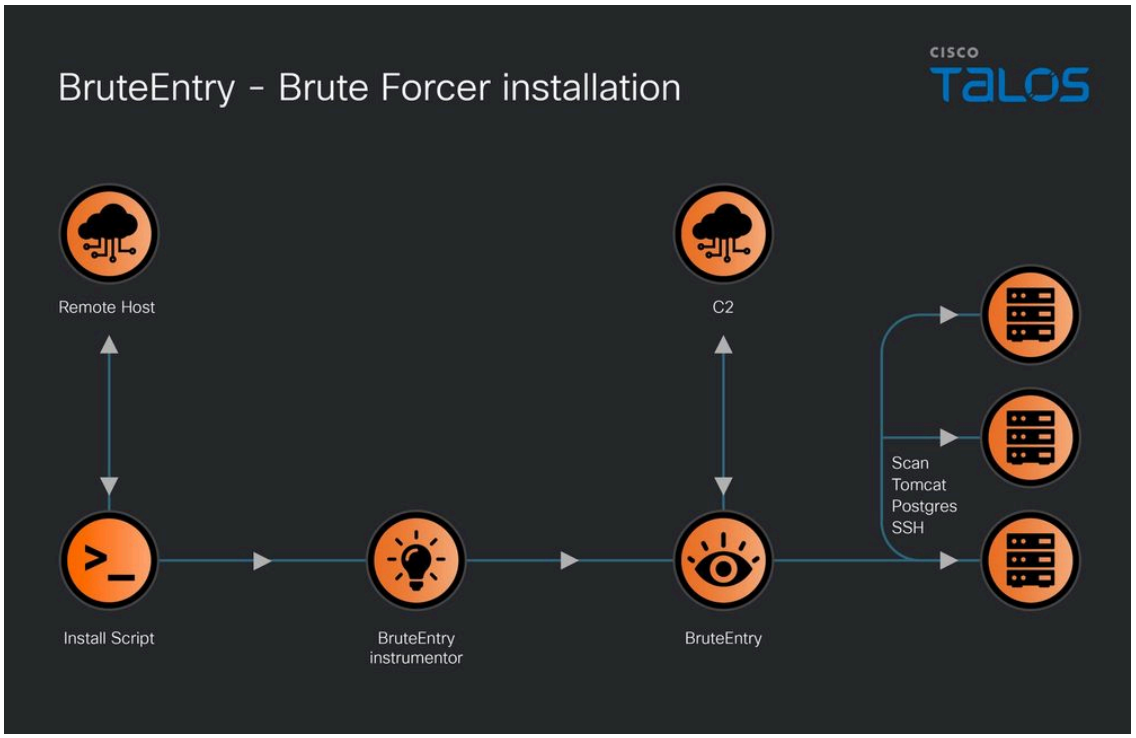

Figure 6. BruteEntry infection chain.

The instrumentor binary

The instrumentor binary is an ELF file written in GoLang. It checks if the BruteEntry is already running on the system using “pgrep”:

```
pgrep <path_to_BruteEntry>
```

And then starts the brute forcer agent:

```
./<path_to_BruteEntry>
```

BruteEntry

BruteEntry is also written in GoLang and begins by registering with the C2 server by providing it with the infected system's IP address and computer name:

```
{“ip”:“value”, “hostname”:“value”}
```

The C2 responds with a JSON that assigns an agent_id to the infected host:

```
{“agent_id”:“value”, “server”:“value”}
```

where “server” = version string of BruteEntry such as “brute-force-server-v1.0”

BruteEntry will then ask the C2 for tasks to perform by sending a GET request to the C2 at the URI, where limit=1000 is the maximum number of vulnerable IPs to scan:

```
/tasks/<agent_id>?limit=1000
```

The C2 responds with a JSON that consists of “tasks” containing the list of IPs to brute force:

```
{"tasks":[
{"id":,"target":":","type":""},
{"id":,"target":":","type":""},
. . . . .
] }
```

The “type” field in the json defines the type of scan to conduct — either “tomcat”,“postgres”, or “ssh”.

The agent will then use a set of embedded credentials to attempt to brute force into either a Tomcat server application at the URL “https[://]<IP>:<Port>/manager/html”, or will brute force into a Postgres instance, either defined in the JSON (<IP><Port>) from the C2 or using the port 5432 if no port is specified.

```
cmp     rsi, 6
jnz     short loc_82B236
cmp     dword ptr [rdi], 'cmot'
jnz     short loc_82B262
cmp     word ptr [rdi+4], 'ta'
jnz     short loc_82B262
mov     rdi, r8         ; cred
                        ; string
mov     service, r9     ; cred
mov     r8, r10         ; cred
mov     r9, r11         ; cred
call    main__ptr_Agent_bruteTomcat
add     rsp, 38h
pop     rbp
retn
-----------------------------------------------------------------

                        ; CODE XREF: main__ptr_Agent_bruteForce+
cmp     rsi, 8
jnz     short loc_82B262
mov     rdx, 'sergtsop'

cmp     [rdi], rdx
jnz     short loc_82B262
mov     rdi, r8         ; cred
                        ; string
mov     service, r9     ; cred
mov     r8, r10         ; cred
mov     r9, r11         ; cred
call    main__ptr_Agent_brutePostgres
```

Figure 7. BruteEntry selecting the type of service to brute force into.

Any successful logins are then POSTED back to the C2:

```
{"batch":[
{"task_id":<task_id>,"success":<true/false>,"note":" <notes on the task>"},
{"task_id":<task_id>,"success":<true/false>,"note":" <notes on the task>"},
......
]}
```

In this instance, “success” indicates if the brute force was successful (true or false), and “notes” provides specific information on whether the brute force was successful. If the login failed, the note reads “All credentials tried.” If it succeeded, the note reads “Cracked by agent <agent_id> | Version <agent_version>”.

Coverage

The following ClamAV signatures detect and block this threat:

- Win.Loader.PeerTime
- Win.Malware.TernDoor
- Unix.Malware.BruteEntry
- Txt.Malware.PeerTime
- Unix.Malware.PeerTime

The following SNORT® rules (SIDs) detect and block this threat: **65551**

IOCs

TernDoor Loader DLL

```
711d9427ee43bc2186b9124f31cba2db5f54ec9a0d56dc2948e1a4377bada289
3c098a687947938e36ab34b9f09a11ebd82d50089cbfe6e237d810faa729f8ff
f36913607356a32ea106103387105c635fa923f8ed98ad0194b66ec79e379a02
```

Encoded TernDoor payload

```
A5e413456ce9fc60bb44d442b72546e9e4118a61894fbe4b5c56e4dfad6055e3
075b20a21ea6a0d2201a12a049f332ecc61348fc0ad3cfee038c6ad6aa44e744
1f5635a512a923e98a90cdc1b2fb988a2da78706e07e419dae9e1a54dd4d682b
```

Windows driver

```
2d2ca7d21310b14f5f5641bbf4a9ff4c3e566b1fbbd370034c6844cedc8f0538
```

UAT-9244 C2 IPs used by TernDoor

```
154[.]205[.]154[.]82:443
207[.]148[.]121[.]95:443
207[.]148[.]120[.]52:443
212[.]11[.]64[.]105
```

Suspected UAT-9244 IPs

```
149[.]28[.]25[.]33
154[.]205[.]154[.]194
154[.]205[.]154[.]65
154[.]205[.]154[.]70
154[.]223[.]21[.]130
154[.]223[.]21[.]194
158[.]247[.]238[.]240
```

```
216[.]238[.]112[.]222
216[.]238[.]123[.]242
216[.]238[.]94[.]37
38[.]54[.]125[.]134
38[.]60[.]199[.]34
45[.]32[.]106[.]94
45[.]77[.]34[.]194
45[.]77[.]41[.]141
47[.]76[.]100[.]159
64[.]190[.]113[.]170
64[.]95[.]10[.]253
```

PeerTime installation script

```
ebcb2691b7c92cdf2b2ff5e2d753abeea8cb325c16596cd839e6bd147f80e38a
00735a8a50d2856c11150ef1e29c05acebce7ad3edad00e37c7f043aacb46330
74fbc8360d4c95d64d7acaa4d18943dce2d41f91d080b0b5e435d8bce52861a5
babc81fc9c998e9dc4ab545f0e112e34d2641e1333bc81aaa131abd061a5b604
e34c9159e6e78c59518a14c5b96bddfee094b684f99d4f69b13371284a014e87
2c3f2261b00ea45e25eb4e9de2b7ff8e41f311c0b3d986461f834022c08b3b99
3fcced9332301ff70b20c98c9434c858400013d659afa6bb5149cffb0206357d
a313f76fca50fff1bcd6f2c6cbc1268985f8c0a3a05fe7f43c4fc0ac3aff84dc
03eac9eb7f4b4bc494ef0496ee23cabbf38f883896838ed813741d8f64ac9fde
17652d7bb5fe0454023db4fc7f608df0dbe6af237be31258e16ba52f0e895e26
74d1a678bdc4bb9f33321e94e3bd1bc1740472ed734231fc46af720072ecb77e
```

PeerTime instrumentor binary

```
c9fc2af30f769d856b88b3051f19fdb663b3e0a0916279df9bbcba93c6a110c9
```

PeerTime malware

```
34d64b3cd9430e85edefcb883973a086dd5de9917e05fabec89b1f4ab9627e91
1cedf01dd4b7e50181d0e781825c66957b862941395d77c8bd7705114f319c80
bfc35f12d00fa4b40c5fbce9e37d704e12a52262709bcbdf09f97890bc40cad5
f3e899789b56429f483e5096e1f473335024f1f763e2d428132338e30352b89e
6ec070457d1f6f239cb02c5e1576a3660cca98f3a07eec6e4e107f698d7fe555
15d937803f90c2b9e277ff94d3e98ff30015ecc7f4623a158e3c98861e5cb278
7b70cd956f082b1029d02b4cb7608893f2de7fa9c500d7d7febdd0f745ac3cb6
d78b3c6df8f3756a7e310cf7435fdba201dd03ec9f97420a0db683489a01a7c9
3fcadde4b414a18b2fed56c1ec59d97977123615fbbf411a1c78425445a6e71c
3d9fbfc2c056eac857ba54e5ed134aa45a4b8322ee9f9353ba32e5b2ca71b0e3
c9a42423ef08bd7f183915780d39530eba5e4e25968c51965ff8bb3026965a28
38eeaa4eaad72feb3f8e6993565fcc548d8e7bb93642590f00fa24aacc0e2862
```

```
56bead2933e91366e4a0d5761daf5b238a7f2c22e597664ef67b3ecae20ab326
6a2d23cc8746a83e9a3b974788fce0e414706b8e75ff390426dd7e10b19967b3
9a7225c17e4bad3ffe7f080530d36f4f8aca5c116b913caa91ab9b0cee85638e
870e791af14caaf395c56028176a9c3f4c1ff0318ef3112d57ecd3d4a1be2ef9
```

PeerTime remote locations

```
185[.]196[.]10[.]247
xtibh[.]com
xcit76[.]com
```

PeerTime C2s

```
bloopencil[.]net
185[.]196[.]10[.]38
```

BruteEntry installation script

```
1fcdd5a417db31e5e07d32cecfa69e53f0dce95b7130ad9c03b92249f001801d
```

BruteEntry instrumentor binary

```
66ce42258062e902bd7f9e90ad5453a901cfc424f0ea497c4d14f063f3acd329
d5eb979cb8a72706bfa591fa57d4ebf7d13cecdc9377b0192375e2f570f796df
```

BruteEntry agent

```
66adeedfb739774fcc09aa7426c8fad29f8047ab4caee8040d07c0e84d011611
66bdce93de3b02cf9cdadad18ca1504ac83e379a752d51f60deae6dcbafe4e31
```

BruteEntry infrastructure

```
212[.]11[.]64[.]105
185[.]196[.]10[.]247
```

Additional malicious scripts

```
023467e236a95d5f0e62e26445d430d749c59312f66cf136e6e2c2d526c46ba1
f8066833e47814793d8c58743622b051070dac09cb010c323970c81b59260f84
06b23d84fd7afd525dfd7860ebd561dcdd72ccbeb51981d5d9a75acf068d0a2a
```

Source: https://blog.talosintelligence.com/uat-9244/