grep is fun

run grep for vulnerable ()’s

whoami

Dhiraj Mishra
Metasploit contributor
Pentester at Cognosec DMCC

mishradhiraj_

Before we get started.

- I don’t know much code review in C,C++
- This might be basic for someone, but a good kick start for someone else

My blind roadmap

when i initially started

- Find list of vulnerable C,C++ ()’s from CWE pages
  - i.e. CWE 120,20 etc.
- Use grep recursive search to find that vulnerable ()’s in open source projects
- Submit that bug to the repo maintainer

Don’t like grep ?

Flawfinder - A static analysis tool for C,C++

Reference: https://www.dwheeler.com/flawfinder/

Running grep on what ?

- Open source projects on github.com or gitlab.com
- OSS projects

Specific to C,C++ codes, because we will be targeting vulnerable functions which affect C,C++ binaries.

Reference: https://opensource.com/article/16/12/yearbook-top-10-open-source-projects

Best way

- https://github.com/trending/c
- https://github.com/trending/c++

Quick look on vulnerable ()’s in C,C++ ?

- realpath()
- strncat()
- access()
- SetSecurityDescriptorDacl()

But, there are many more...

realpath() - Resolves a path name

According to POSIX standard this function is vulnerable by design it self.

- The output buffer needs to be at least of size PATH_MAX
- Set NULL for resloved_path

Reference: http://man7.org/linux/man-pages/man3/realpath.3.html

Actual implementation of realpath()

```
static int
        path_check(ARCHD *arcn, int level)
   {
                char buf[MAXPATHLEN];
                char *p;
                if ((p = strrchr(arcn->name, '/')) == NULL)
                        return 0;
                         *p = '\0';
                if (realpath(arcn->name, buf) == NULL) {
```

Usage of realpath() may cause overflow

If resloved_path is not set to NULL or output buffer is not at least size of PATH_MAX then a buffer overflow will occur.

Case study for realpath()

This issue was observed in Facebook open source project name **osquery**, so we simply use grep recursive search over here to find realpath() in entire repository.

```
input0@zero:~/Desktop/osquery-3.2.6$ grep -iRn 'realpath' .
./external/CMakeLists.txt:15:    get_filename_component(full_child "${PWD}/${child}" REALPATH)
./tools/codegen/gentable.py:23:SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))
./tools/codegen/gentable.py:477:    SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))
./tools/codegen/gentargets.py:12:SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))
./tools/codegen/gentargets.py:13:REPO_ROOT_DIR = os.path.realpath(os.path.join(SCRIPT_DIR, "../.."))
./tools/codegen/genapi.py:28:SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))
./tools/codegen/genapi.py:307:    SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))
./tools/tests/test_osqueryd.py:196:                rpath = os.path.realpath(pth)
./tools/tests/test_release.py:78:    SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))
./tools/tests/test_base.py:88:SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))
./tools/tests/test_http_server.py:32:SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))
./tools/tests/test_example_queries.py:46:    SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))
./tools/analysis/fuzz.py:29:sys.path.append(os.path.dirname(os.path.realpath(__file__)) + "/../tests/")
./tools/analysis/fuzz.py:30:sys.path.append(os.path.dirname(os.path.realpath(__file__)) + "/../codegen/")
./tools/analysis/profile.py:29:sys.path.append(os.path.dirname(os.path.realpath(__file__)) + "/../tests/")
./tools/provision/ubuntu.sh:27:  package realpath
./tools/provision/formula/python.rb:257:      if os.path.realpath(sys.executable).startswith('#{rack}'):
./osquery/tables/system/freebsd/mounts.cpp:37:    r["device_alias"] = std::string(realpath(mnt[i].f_mntfromname, real_path)
./osquery/tables/system/linux/processes.cpp:60:  char* link_path = realpath(attr_path.c_str(), full_path);
./osquery/tables/system/linux/mounts.cpp:36:        realpath(ent->mnt_fsname, real_path) ? real_path : ent->mnt_fsname);
./osquery/tables/system/darwin/mounts.cpp:37:    r["device_alias"] = std::string(realpath(mnt[i].f_mntfromname, real_path)
input0@zero:~/Desktop/osquery-3.2.6$
```

Case study for realpath()

I found multiple file in **osquery** using realpath(), moving further I manually put efforts to read those codes.

So the file: /system/darwin/mounts.cpp

```
r["path"] = TEXT(mnt[i].f_mntonname);
r["device"] = TEXT(mnt[i].f_mntfromname);
r["device_alias"] = std::string(realpath(mnt[i].f_mntfromname, real_path)
                    ? real_path
                    : mnt[i].f_mntfromname);
```

I simply compared this code from the code which is on slide number 9, and two things were confirmed that output buffer is not at least size of PATH_MAX and nowhere NULL is passed for resloved_path

Case study for realpath()

Moving forward i simply reported this Facebook, and the bug was accepted, and Facebook team responded back saying yes this may cause an overflow and a patch was also pushed.
Reference: https://github.com/facebook/osquery/issues/4408

Hi Team,

`https://github.com/facebook/osquery/blob/master/osquery/tables/system/darwin/mounts.cpp#L37`

i.e

```
r["device_alias"] = std::string(realpath(mnt[i].f_mntfromname, real_path)
```

This function does not protect against buffer overflows, and some implementations can overflow internally references (CWE-120/CWE-785!).

Ensure that the destination buffer is at least of size `MAXPATHLEN`, and to protect against implementation problems, the input argument should be checked to ensure it is no larger than `MAXPATHLEN.`

Request team to please have a look and advise.

Thank you
Team w00t

muffins added bug macOS labels on May 21

muffins commented on May 21 — Contributor

@teamwoot thanks for reporting! We'll get a fix up for this asap!

Case study for realpath()

But, I still don't know how this can be exploited in **osquery**, but it general realpath could cause a shell execution or a privilege escalation.

Example:
https://www.rapid7.com/db/modules/exploit/linux/local/glibc_realpath_priv_esc

Case study for realpath() - 2 (CVE-2018-14939)

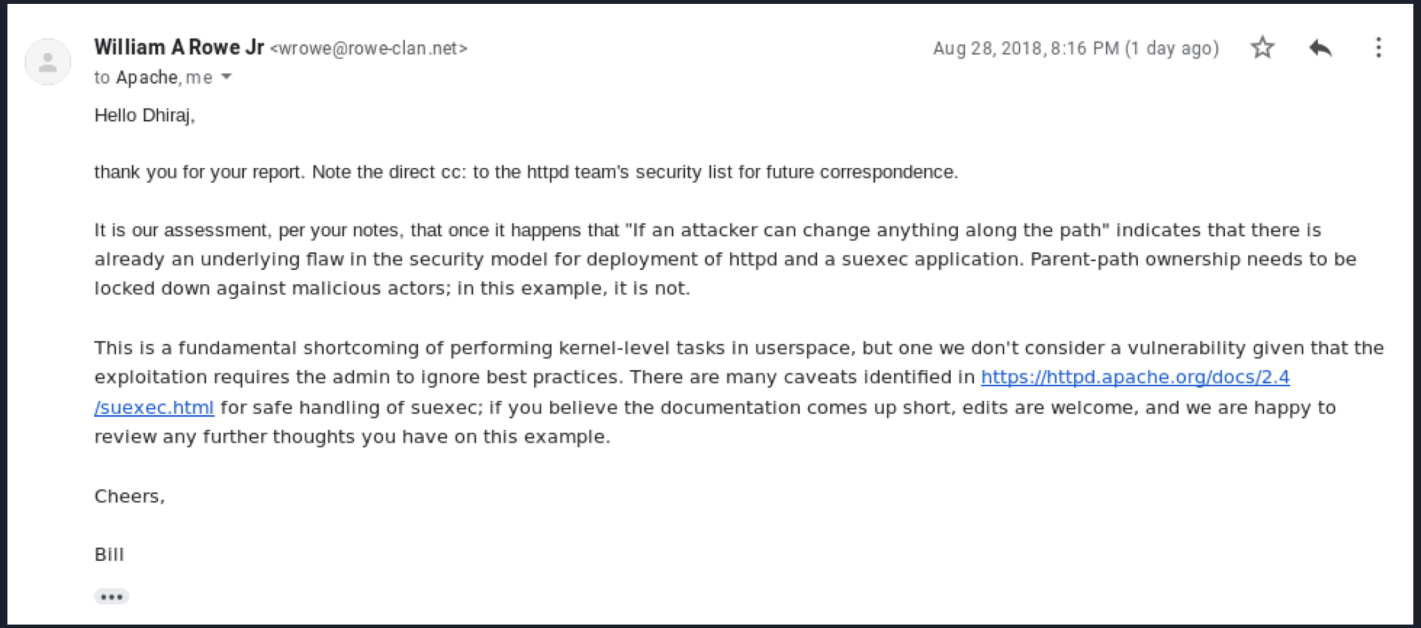
I replicated the same thing in LibreOffice source code, but again there was no fully working PoC that causes buffer overflow in LibreOffice.

Reference:
https://bugzilla.redhat.com/show_bug.cgi?id=1614165
https://access.redhat.com/security/cve/cve-2018-14939

In general I was just capable of finding bugs by doing code review and understanding how the code is working if realpath is in used for multiple open source projects.

Case Study for realpath() - 3 Apache

Found same thing in **httpd - apache** but most of the time developers are aware of such issues that it may cause an overflow, but they document it.



William A Rowe Jr <wrowe@rowe-clan.net> Aug 28, 2018, 8:16 PM (1 day ago)

to Apache, me

Hello Dhiraj,

thank you for your report. Note the direct cc: to the httpd team's security list for future correspondence.

It is our assessment, per your notes, that once it happens that "If an attacker can change anything along the path" indicates that there is already an underlying flaw in the security model for deployment of httpd and a suexec application. Parent-path ownership needs to be locked down against malicious actors; in this example, it is not.

This is a fundamental shortcoming of performing kernel-level tasks in userspace, but one we don't consider a vulnerability given that the exploitation requires the admin to ignore best practices. There are many caveats identified in https://httpd.apache.org/docs/2.4/suexec.html for safe handling of suexec; if you believe the documentation comes up short, edits are welcome, and we are happy to review any further thoughts you have on this example.

Cheers,

Bill

strncat() - appends char from strings

Improper use of the strncpy() and strncat() functions can result in buffer overflow vulnerabilities.

strncat() - appends char from strings

The last argument to strncat() should not be the total buffer length.

Vulnerable code:

```
strncpy(record, user, MAX_STRING_LEN - 1);
                    strncat(record, cpw, MAX_STRING_LEN - 1);
```

Actual Code:

```
strncat(dest, source, dest_size-strlen(dest)-1);
```

Case study for strncat() - TOR browser

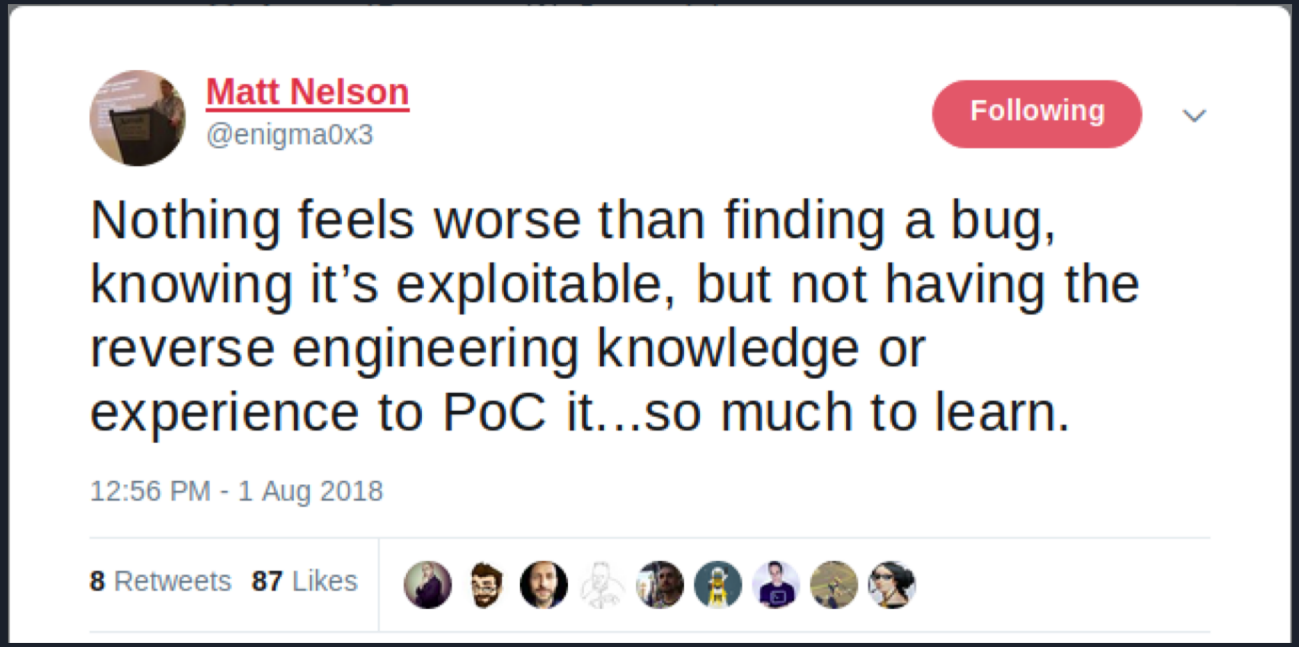
https://github.com/torproject/tor/commit/d0525c38d607504aee4ab8451d4651c2668997c0

9 src/lib/err/backtrace.c — View

```
@@ -35,6 +35,7 @@
 #include <errno.h>
 #include <stdlib.h>
 #include <string.h>
+#include <stdio.h>

 #ifdef HAVE_CYGWIN_SIGNAL_H
 #include <cygwin/signal.h>
@@ -264,16 +265,12 @@ dump_stack_symbols_to_error_fds(void)
 int
 configure_backtrace_handler(const char *tor_version)
 {
-  char version[128];
-  strncpy(version, "Tor", sizeof(version)-1);

   if (tor_version) {
-    strncat(version, " ", sizeof(version)-1);
-    strncat(version, tor_version, sizeof(version)-1);
+    snprintf(version, sizeof(version), "Tor %s", tor_version);
   }

-  version[sizeof(version) - 1] = 0;
-
   return install_bt_handler(version);
 }
```

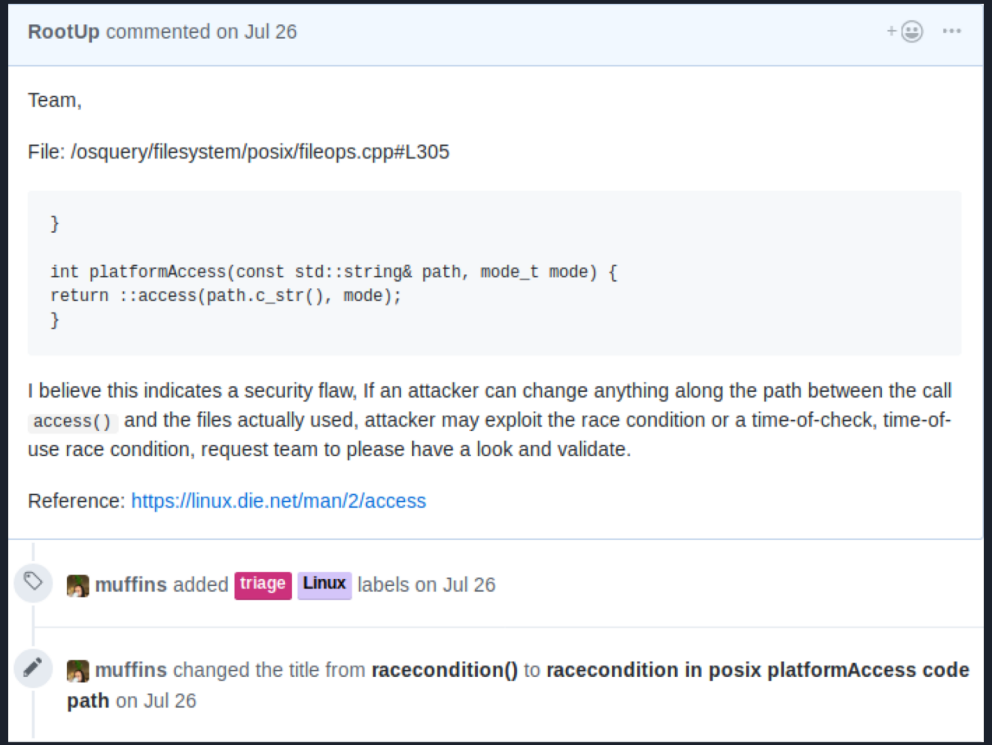
```
@@ -35,6 +35,7 @@
 #include <errno.h>
 #include <stdlib.h>
 #include <string.h>
+#include <stdio.h>

 #ifdef HAVE_CYGWIN_SIGNAL_H
 #include <cygwin/signal.h>
@@ -265,16 +266,12 @@ dump_stack_symbols_to_error_fds(void)
 int
 configure_backtrace_handler(const char *tor_version)
 {
-  char version[128];
-  strncpy(version, "Tor", sizeof(version)-1);

   if (tor_version) {
-    strncat(version, " ", sizeof(version)-1);
-    strncat(version, tor_version, sizeof(version)-1);
+    snprintf(version, sizeof(version), "Tor %s", tor_version);
   }

-  version[sizeof(version) - 1] = 0;
-
   return install_bt_handler(version);
 }
```

I was feeling something like this...



But, I started trying harder...

This bug is about access() function. In general access() can lead racecondition() which may cause privilege escalation.



RootUp commented on Jul 26

Team,

File: /osquery/filesystem/posix/fileops.cpp#L305

```
}

int platformAccess(const std::string& path, mode_t mode) {
return ::access(path.c_str(), mode);
}
```

I believe this indicates a security flaw, If an attacker can change anything along the path between the call `access()` and the files actually used, attacker may exploit the race condition or a time-of-check, time-of-use race condition, request team to please have a look and validate.

Reference: https://linux.die.net/man/2/access

muffins added triage Linux labels on Jul 26

muffins changed the title from **racecondition()** to **racecondition in posix platformAccess code path** on Jul 26

But, I started trying harder...

muffins commented on Jul 26 — Contributor

Hey there, thanks for reporting! I re-titled the issue, hope you don't mind :), to be a bit more descriptive. I *think* you might be right that there's a TOCTOU bug here, have you been able to trigger it?

RootUp commented on Jul 26

Thank you for taking look into this,
This was identified while code review, however I think it 'might' be possible via hardlink case if so then.

exploit.c

```
#include <unistd.h>
int main()
{
  if (geteuid()!=0) exit(1);
  setuid(geteuid());
  char *args[] = { "/bin/sh", 0 };
  return execve(args[0], args, 0);
}
```

Trigger an race condition by creating a hardlink to the vulnerable program.

```
while :; do ln -f ./vulnpoc; ln -f ./exploit; done
```

SetSecurityDescriptorDacl()

Never create NULL ACLs, an attacker can set it to Everyone (Deny All Access), which would even forbid administrator access which may lead to privilege escalation.

Case study of SetSecurityDescriptorDacl() - MariaDB

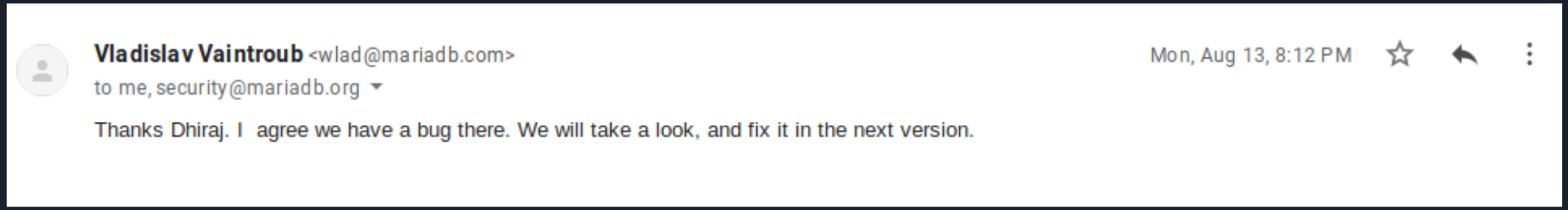
File: /server/blob/10.3/sql/mysqld.cc#L2761

```
}
if (!SetSecurityDescriptorDacl(&sdPipeDescriptor, TRUE, NULL, FALSE))
{
```

Never create NULL ACLs, an attacker can set it to Everyone (Deny All Access), which would even forbid administrator access which may lead to privilege escalation.

Source code: https://github.com/MariaDB/server

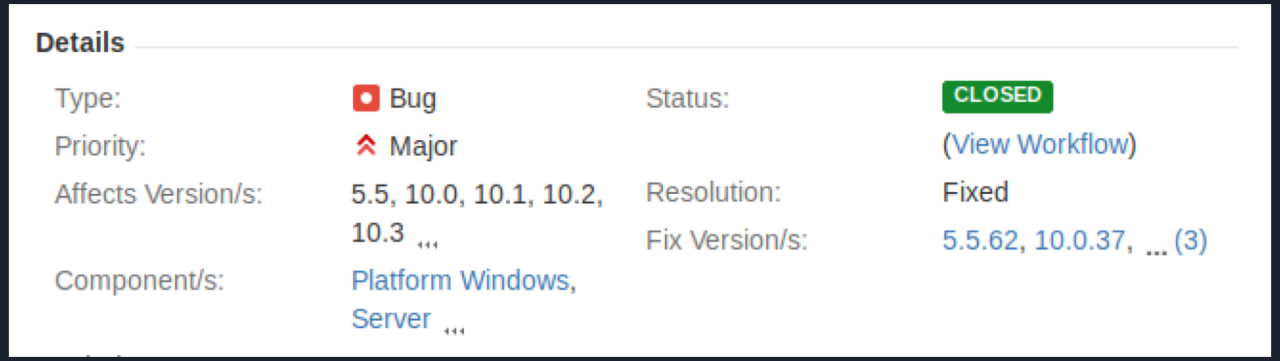
Case study of SetSecurityDescriptorDacl() - MariaDB



Vladislav Vaintroub <wlad@mariadb.com> Mon, Aug 13, 8:12 PM

to me, security@mariadb.org

Thanks Dhiraj. I agree we have a bug there. We will take a look, and fix it in the next version.



Details

Type:	Bug	Status:	CLOSED
Priority:	Major		(View Workflow)
Affects Version/s:	5.5, 10.0, 10.1, 10.2, 10.3 ...	Resolution:	Fixed
		Fix Version/s:	5.5.62, 10.0.37, ... (3)
Component/s:	Platform Windows, Server ...		

Bug thread : https://jira.mariadb.org/browse/MDEV-16963

Case study of SetSecurityDescriptorDacl() - MariaDB (Try exploiting it)

https://www.blackhat.com/presentations/bh-dc-07/Cerrudo/Paper/bh-dc-07-Cerrudo-WP.pdf - page -7- to -13-

This has been really hard:

- Open IDA
- Open oracle.exe binary
- Locate SetSecurityDescriptorDacl() and press Ctrl+X (Jump to cross reference)
- Look at SetSecurityDescriptorDacl() parameters and security descriptor usage.
- Total time spent: 5 minutes (being lazy)

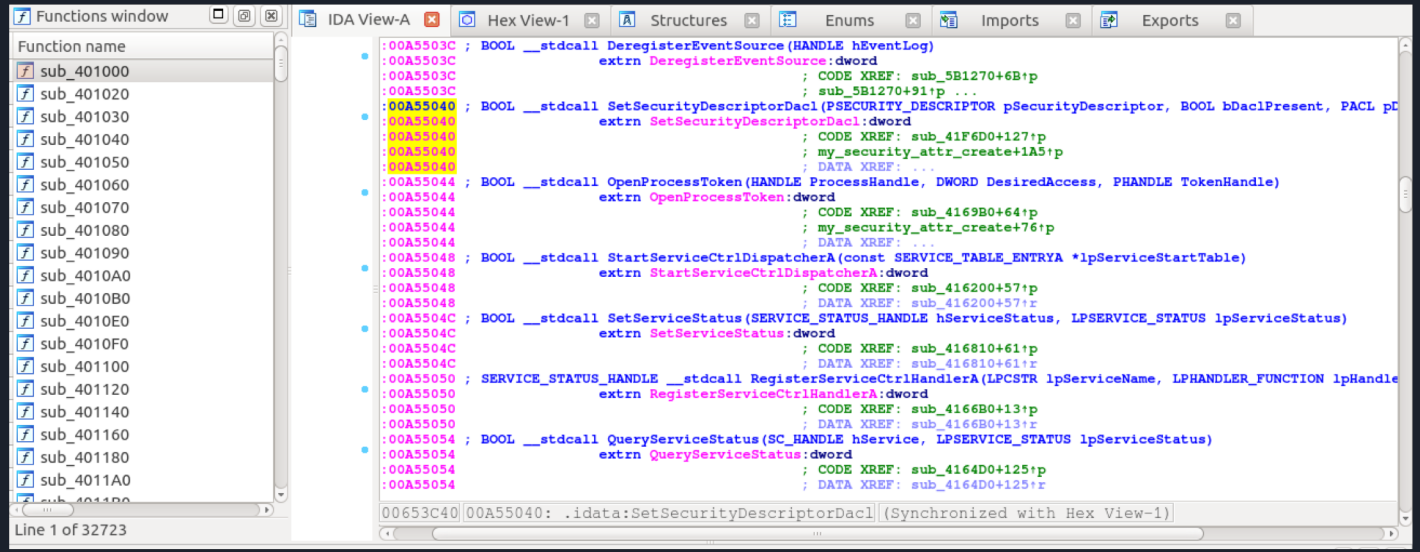
Case study of SetSecurityDescriptorDacl() - MariaDB (Try exploiting it)

File: /server/blob/10.3/sql/mysqld.cc

Download https://downloads.mariadb.org/interstitial/mariadb-10.3.8/win32-packages/mariadb-10.3.8-win32.zip/from/http%3A//mirror.truenetwork.ru/mariadb/

Search for mysqld.exe, I am was using 32 bit system and IDA for this.
Open mysqld.exe in IDA and repeat the steps which is done by Cesar Cerrudo in above slide.

Case study of SetSecurityDescriptorDacl() - MariaDB (Try exploiting it)




```
1 SetSecurityDescriptorDacl(PSECURITY_DESCRIPTOR pSecurityDescriptor, BOOL bDaclPresent, PACL pDacl, BOOL bDaclDefaulted)
 extrn SetSecurityDescriptorDacl:dword
                        ; CODE XREF: sub_41F6D0+127↑p
                        ; my_security_attr_create+1A5↑p
                        ; DATA XREF: ...
```

Case study of SetSecurityDescriptorDacl() - MariaDB (Try exploiting it)

Exploit PoC by Cesar: https://pastebin.com/Nh6nF3Yt

However, I still need to understand the exploit PoC and create something like it, so this could lead to successful local privilege escalation bug.

Case study of SetSecurityDescriptorDacl() - MariaDB

Moving further MITRE told this issue exists in Oracle ….

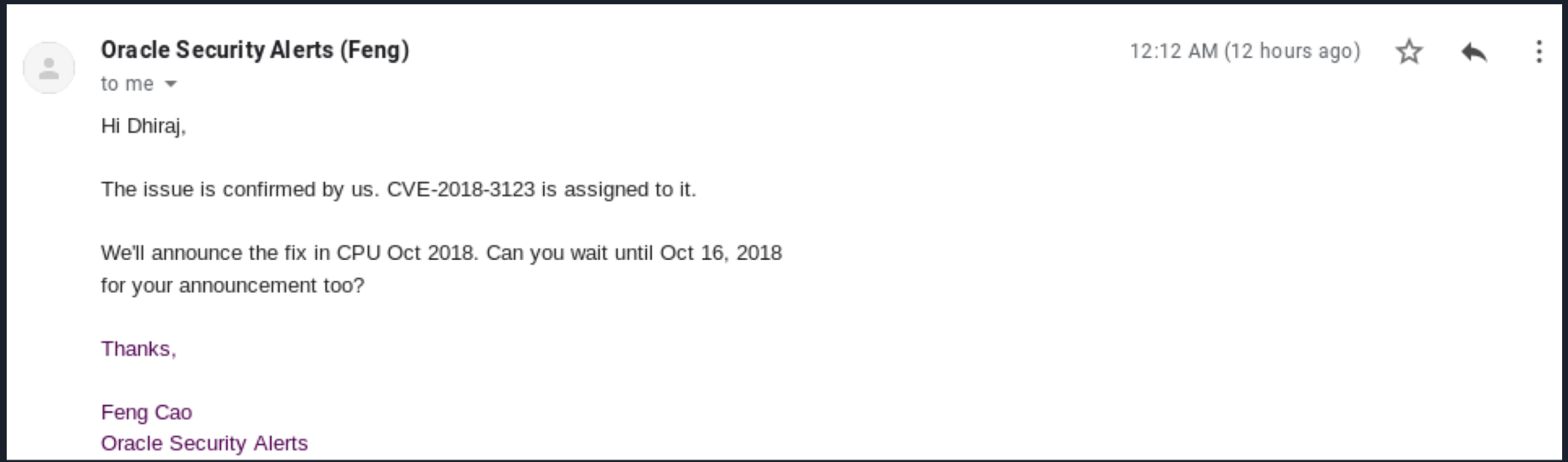
“”

The issues occurs in sql/mysqld.cc in MySQL version 5.1.30 from the https://downloads.mysql.com/archives/community/ web site. Version 5.1.30 is from 2008, and MariaDB was first released in 2009.

“”

PS: The working of cesar’s exploit might also work in this case.

Case study of SetSecurityDescriptorDacl() - MariaDB



Oracle Security Alerts (Feng) 12:12 AM (12 hours ago)

to me

Hi Dhiraj,

The issue is confirmed by us. CVE-2018-3123 is assigned to it.

We'll announce the fix in CPU Oct 2018. Can you wait until Oct 16, 2018 for your announcement too?

Thanks,

Feng Cao
Oracle Security Alerts

Hmmmmm.....

So that was my entire journey, of finding bugs using grep and contributing to open source projects.

But now…

CVE-2019-6498 Detail

Current Description

GattLib 0.2 has a stack-based buffer over-read in gattlib_connect in dbus/gattlib.c because strncpy is misused.

Source: MITRE
Description Last Modified: 01/21/2019
View Analysis Description

Impact

CVSS v3.0 Severity and Metrics:
Base Score: 8.8 HIGH
Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H (V3 legend)
Impact Score: 5.9
Exploitability Score: 2.8

Attack Vector (AV): Adjacent
Attack Complexity (AC): Low
Privileges Required (PR): None
User Interaction (UI): None
Scope (S): Unchanged
Confidentiality (C): High
Integrity (I): High
Availability (A): High

CVSS v2.0 Severity and Metrics:
Base Score: 5.8 MEDIUM
Vector: (AV:A/AC:L/Au:N/C:P/I:P/A:P) (V2 legend)
Impact Subscore: 6.4
Exploitability Subscore: 6.5

Access Vector (AV): Local_Network
Access Complexity (AC): Low
Authentication (AU): None
Confidentiality (C): Partial
Integrity (I): Partial
Availability (A): Partial
Additional Information:
Allows unauthorized disclosure of information
Allows unauthorized modification

stack-based bufferoverflow #81

Open RootUp opened this issue 4 days ago · 0 comments

RootUp commented 4 days ago • edited

Hi Team,

Summary

While fuzzing **gattlib** using **clang 6.0** with ASAN a stack-based buffer-overflow was observed in gattlib.c and discover.c

Vulnerable code from gattlib.c

```
        // Transform string from 'DA:94:40:95:E0:87' to 'dev_DA_94_40_95_E0_87'
        strncpy(device_address_str, dst, sizeof(device_address_str));
        for (i = 0; i < strlen(device_address_str); i++) {
                if (device_address_str[i] == ':') {
                        device_address_str[i] = '_';
                }
}
```

Vulnerable code from discover.c

```
        if (argc != 2) {
                printf("%s <device_address>\n", argv[0]);
                return 1;
        }

        connection = gattlib_connect(NULL, argv[1], BDADDR_LE_PUBLIC, BT_SEC_LOW, 0, 0);
        if (connection == NULL) {
                fprintf(stderr, "Fail to connect to the bluetooth device.\n");
                return 1;
}
```

Also, I have figured a simple way to reproduce this rather than using AFL poc in this case.

Future ToDo’s

Understand multiple vulnerable C,C++ functions more correctly and look around on ways to exploit them.

Happy to collab with any one to work on this and contribute to open source projects.

QnA - Feedback ?

@mishradhiraj_