Metasploit Framework
Telephony

Black Hat Briefings USA 2009
July, 2009

I)ruid, C^{2}ISSP
<druid@caughq.org>
http://druid.caughq.org/

Abstract

An important attack vector missing in many penetration testing and attack tools available today is the tried-and-true telephony dialup. With the recent surge in popularity of VoIP connectivity, accessing such attack vectors has become both cheap and easy. Using the new Metasploit telephony components, users are now able to both scan for and dial up directly to telephony-accessible exploitation targets.

Contents

Chapter 1

Metasploit Framework Telephony

1.1 Telephony Library

The Metasploit Framework[1] Telephony Library is the core of telephony support within the Metasploit Framework. It is intended to provide support for targeting and interfacing with remote devices accessible via telephony such as dialup modems connected to remote systems or PBX and voice-mail systems. This capability extends Metasploit's potential target pool to systems only accessible via dialup connection, vulnerabilities in software such as /bin/login, getty software, Pluggable Authentication Modules (PAM), and Bulletin Board System (BBS) software that are otherwise not accessible via a standard network connection or over the Internet.

The source code for the Metasploit Framework Telephony Library can be found in your Metasploit Framework directory under the *lib/telephony* directory. Currently there is only one source file provided, *modem.rb*, which provides the *Modem* object. The *Modem* object exports a number of methods and accessors used to configure and interface with a local Serial Port Modem:

Methods:

- Modem.put_command(command, timeout)

  Flushes the Modem.

  Sends command to the Modem.

  Reads result of the command using Modem.get_response(timeout).

  Returns result.

- Modem.get_response(timeout)

  Reads characters from the Modem until the next newline (0x0a) character.

  Returns the characters read or "TIMEOUT" if no characters were read in the allotted time.

- Modem.commandstate

  Puts the Modem into command state.

- Modem.datastate

  Puts the Modem into data state.

- Modem.hangup

  Sends "ATH0" to the Modem if the Modem is in command state.

Sets DTR to off if the underlying Serial Port responds to the command.

Puts the modem into command state if it is in data state and the underlying Serial Port does not respond to DTR.

Sends "ATH0" to the Modem.

- Modem.flush

  Flushes all data from the read buffer of the Serial Port.

Accessors:

- serialport

  String defining the Serial Port that the Modem is attached to, such as "/dev/ttyS0" or "COM1".

- sp

  The SerialPort Object created from the *serialport* variable.

- sock

  A Socket abstraction of the *sp* SerialPort object.

- baud

  The Baud rate to set the Modem to, such as 9600 or 19200.

- data_bits

  The number of data bits per byte, usually 7 or 8.

- parity

  The parity to use, either "NONE", "EVEN", or "ODD".

- stop_bits

  The number of stop bits to use, usually 1.

- flowcontrol

  The type of flow control to use, either "NONE", "HARD", or "SOFT".

- display

  A Boolean indicating whether Modem commands and responses should be displayed to the Metasploit console. This output will be prefixed with the "[m]" identifier.

- commandstate

  A Boolean indicating whether the Modem is in command or data state. When set to "true", the modem is in command state. When set to "false", the modem is in data state.

1.2 Dialup Exploitation Module

Dialup exploitation is accomplished in Metasploit by providing the *Exploit::Remote::Dialup* module. This module provides a number of useful dialup-related methods which in turn employ the underlying Modem methods. The code for these dialup-related methods can be found in your Metasploit Framework directory under the *lib/msf/core/exploit* directory in a file called *dialup.rb*. This code registers a bunch of new options related to the Modem and deregisters the "RHOST" option as the exploit target is connected to via the modem rather than over the network:

- BAUDRATE

  This is an Integer option which specifies the Baud Rate for the modem, defaulting to 19200.

- DATABITS

  This is an Enum option which specifies the Data Bits to be used for the modem. Available values are 4, 5, 6, 7, and 8, defaulting to 8.

- DIALPREFIX

  This is a String option which specifies the Modem command prefix to use when dialing, defaulted to "ATDT *67, *70,".

- DIALSUFFIX

  This is a String option which specifies the Modem command suffix to use when dialing, defaulted to unset.

- DIALTIMEOUT

  This is a Integer option which specifies the timeout in seconds for expecting a connection when dialing, defaulted to 60.

- DISPLAYMODEM

  This is a Boolean option which indicates whether or not to display Modem commands and responses on the Metasploit console.

- FLOWCONTROL

  This is a Enum option which specifies what type of flow control to use with the Modem. Options are "None", "Hardware", "Software", and "Both", defaulted to "None".

- INITSTRING

  This is a String option which specifies the Modem initialization string to use, defaulted to "AT X6 S11=80".

- NUMBER

  This is the number that the Modem will dial, defaulted to unset.

- PARITY

  This is an Enum option which specifies the parity to use with the Modem. Options are "None", "Even", "Odd", "Mark", and "Space", defaulted to "None".

- SERIALPORT

  This is a String option which specifies the Serial Port that the Modem is attached to, defaulted to "/dev/ttyS0".

- STOPBITS

  This is an Enum option which specifies the stop bits to use with the Modem. Options are 1 and 2, defaulted to 1.

The *Exploit::Remote::Dialup* module provides the following methods:

- connect_dialup

  Creates a Modem object attached to the Serial Port specified by the SERIALPORT option.

  Configures the Modem with the supplied parameter options.

  Initializes the Modem.

  Dials the number provided in the NUMBER option.

  Checks the Modem response to determine if a connection has been established or not.

- disconnect_dialup

  Calls Modem.flush to flush the Modem buffer.

  Calls Modem.hangup to hang up the line.

  Calls Modem.close to destroy the Modem object.

- dialup_getc

  Reads and returns a single character from the Modem read buffer.

- dialup_gets

  Reads and returns a newline-terminated string of characters up to and including the terminating newline character (0x0a) from the Modem read buffer.

- dialup_expect(regexp, timeout)

  Reads and stores in a buffer characters from the Modem read buffer until either the provided regular expression matches the buffer or the timeout value (in seconds) is reached.

  Returns a hash with :match set to true or false depending on whether the regexp matched or not and with :buffer set to the stored character buffer.

- dialup_putc(c)

  Sends a single character to the Modem write buffer.

- dialup_puts(string)

  Sends a string of characters to the Modem write buffer.

- handler

  Calls the handler specified by the selected payload.

1.3 UNIX TTY Interact Payload

When using Metasploit, systems are traditionally targeted by address across an IP network. When successfully exploited, a shell session is established via a payload such as a reverse shell which connects back to Metasploit, by attaching a shell to a listening port on the target system, which is then connected to via a new socket connection across the network, or in some cases a shell is attained over the existing network socket that was used to attack the system. When exploiting a system over a dialup connection, a traditional network socket is not being employed. Instead, a local file descriptor attached to a serial device is being used, so a new Payload type that is able to handle this type of interface in the same manner that sessions attached to network sockets are was required.

To fill this need, a new payload for TTYs called "UNIX TTY Interact" was created which allows the user to interact directly with a target system's TTY over an established connection attached to a file descriptor. When an exploit's platform is set to "unix" and it's architecture is set to *ARCH_TTY*, the *tty/unix/interact* payload becomes available. When the payload's handler is called from an exploit, it connects a Metasploit session to the active file descriptor for the Modem, allowing the user to interact directly with the target system. This session can then be managed with the Metasploit session manager.

Chapter 2

Applied Metasploit Telephony

The following sections will outline ways in which the Metasploit Framework Telephony Library and the Dialup Exploitation Module have been applied to various tasks such as interacting with a dialup system, exploitation of a dialup system, and scanning number ranges for dialup systems, or wardialing[1].

2.1 Interactive Dialup

Within the Metasploit Framework a test "exploit" has been provided for use in testing Metasploit's basic telephony capability, verifying that your Modem configuration options are correct, and that your target system receives calls and will establish a connection. It's source code can be found in the *modules/exploits/test/dialup.rb* file. It is expected to be used with the *UNIX TTY Interact* payload and as such is set to platform "unix" and architecture *ARCH_TTY*. In all, it is about the most basic "exploit" you could possibly write using the Dialup Exploitation Module, as seen below:

```
def exploit
   connect_dialup
   handler
   disconnect_dialup
end
```

In practice, it would be used in the following manner:

```
> use exploit/test/dialup
> setg NUMBER 512.867.5309
> setg BAUDRATE 19200
> setg SERIALPORT /dev/ttyS0
```

... set other relevant options ...

```
> set PAYLOAD tty/unix/interact
> exploit
```

[1] Wardialing is a technique of using a modem to automatically scan a list of telephone numbers, usually dialing every number in a local area code to search for computers, BBS systems and fax machines.

Launching this exploit results in output such as:

```
msf exploit(dialup) > exploit

[*] Initializing Modem
[*] Dialing: ###.###.#### (60 sec. timeout)
[*] Carrier: CONNECT 14400/ARQ/V32/LAPM/V42BIS
[*] Trying to use connection...
[*] Interactive TTY session 1 opened (Local Pipe -> Remote Pipe)

Login: druid
Password:

Last login: Mon Jun 27 07:20:30 on term/a
Sun Microsystems Inc.   SunOS 5.6      Generic August 1997
$
```

2.2 Scripted Interactive Dialup

As an extension to the Interactive Dialup scenario outlined in Section 2.1, the *dialup_puts* and *dialup_expect* methods can be used to script an initial portion of the session prior to handing the connection off to the handler. The authentication portion of the example in Section 2.1 could be scripted as shown below, using authentication credentials from the exploit's options datastore:

```
def exploit
   connect_dialup
   dialup_expect(/ogin: /i, 4)
   dialup_puts(datastore['USERNAME'])
   dialup_expect(/assword: /i, 4)
   dialup_puts(datastore['PASSWORD'])
   dialup_expect(/[$#] /, 4)
   handler
   disconnect_dialup
end
```

In this manner you can script execution of local exploits on systems that you have unprivileged access to. Simply script an interactive dialup to connect, log in, write out an exploit to file, run the exploit, and (hopefully) elevate privileges.

2.3 Exploit for System V Derived /bin/login

A more realistic example can be found in the exploit provided for CVE-2001-0797[2, 3] which is included with the Metasploit Framework. This vulnerability exists in /bin/login implementations which were derived from System V's /bin/login. When a large number of environment variables are passed to it via the "login:" prompt, an overflow condition is triggered. Due to the input vector, exploitation of this vulnerability can be accomplished entirely through direct interaction with /bin/login over any method of invoking it, network, dialup, or otherwise, and the resultant shell access is provided over the same connection as is used for exploitation. The code for this exploit can be found in *modules/exploits/dialup/multi/login/manyargs.rb*, and is used as follows:

```
> use exploit/dialup/multi/login/manyargs
> setg NUMBER 512.867.5309
> setg BAUDRATE 19200
> setg SERIALPORT /dev/ttyS0
```

... set other relevant options ...

```
> set PAYLOAD tty/unix/interact
> exploit
```

Successful exploitation of a vulnerable system using this exploit yields the following results:

```
msf exploit(manyargs) > exploit

[*] Targeting: Solaris 2.6 - 8 (SPARC)
[*] Dialing Target
[*] Initializing Modem
[*] Dialing: ###.###.#### (60 sec. timeout)
[*] Carrier: CONNECT 19200/ARQ/V32/LAPM/V42BIS
[*] Waiting for login prompt
[*] Sending evil buffer...
[*] Waiting for password prompt
[*] Password prompt received, waiting for shell
[*] Success!!!
[*] Trying to use connection...
[*] Interactive TTY session 1 opened (Local Pipe -> Remote Pipe)


#
```

2.4 The Metasploit Wardialer

What use are dialup exploitation methods and weaponized exploits if you don't have suitable targets to attack? Using the *Metasploit Wardialer*, such systems can be found whether they be soft targets in Eastern Europe or that dialup access to the file server that the absent-minded IT Administrator left answering calls on a spare phone line belonging to the subject of your current penetration testing engagement.

2.4.1 Target Detection

The Metasploit Wardialer will detect and log all standard, and many nonstandard, modem word responses such as "CONNECT", "+FCO", "BUSY", "NO DIALTONE", and so forth. As do most wardialers (WarVOX notably excluded), the Metasploit Wardialer relies upon the underlying Modem's detection capability to identify systems. By setting your modem to data, fax, or even voice mode, various types of systems can be scanned for and identified, however this entirely depends on the types of systems that your Modem can detect, and usually what mode you have your modem set to. Obviously a more versatile modem used for scanning will yield better results.

2.4.2 Results Logging

Output is stored in the user's Metasploit working directory under the *logs/wardial* sub-directories in two types of files, the global *found.log* file and the individual scan's *.dat* files.

Each scan range database for all numbers within that range are stored in a *.dat* file of the range's name which contains a gzipped, Marshaled Ruby object. Using this database file, the Metasploit Wardialer is able to stop and resume scanning of any given range of numbers individually. 3rd party applications may also use these files to attain scan data as well as glean real-time updates from the Metasploit Wardialer as the files on disc are written after each completed dial.

Interesting detections such as data carriers, faxes, and tones are logged to the text log file *found.log*, as well as to the Metasploit database abstraction layer, if the user has one configured. Notes are logged to the database using the dialed number as the host identifier with a note type of "wardial_result" and the connection string with any banner information as the data.

In an upcoming version, the Metasploit Wardialer will also be capable of logging interesting detections to a centralized database called *TIDbITS*, or *The Internet Database of Information Telephony Systems*[4].

2.4.3 Configuration

The Metasploit Wardialer provides many of the familiar options and settings that you would expect (slightly edited due to formatting constraints):

```
Module options:

   Name        Current Setting  Description
   ----        ---------------  -----------
   DIALMASK    202.358.XXXX     Dial Mask (e.g. 1.800.95X.99XX, (202) 358-XXXX, 358.####, etc.)
   DIALPREFIX  ATDT             Dial Prefix
   INITSTRING  AT X6 S11=80     Initialization String
   SERIALPORT  /dev/ttyS0       Serial Port (e.g. 0 (COM1), 1 (COM2), /dev/ttyS0, etc.)
   THREADS     1                The number of concurrent threads

Module advanced options:

   Name           : BaudRate
   Current Setting: 19200
   Description    : Baud Rate

   Name           : ConnTimeout
   Current Setting: 45
   Description    : Timeout per data connection in seconds

   Name           : DataBits
   Current Setting: 8
   Description    : Data Bits (4 is Windows Only) (accepted: 4, 5, 6, 7, 8)

   Name           : DialDelay
   Current Setting: 1
   Description    : Time to wait between dials in seconds (rec. min. 1)

   Name           : DialSuffix
```

```
Current Setting:
Description    : Dial Suffix

Name           : DialTimeout
Current Setting: 40
Description    : Timeout per dialed number in seconds

Name           : DisplayModem
Current Setting: false
Description    : Displays modem commands and responses on the console

Name           : FlowControl
Current Setting: None
Description    : Flow Control (accepted: None, Hardware, Software, Both)

Name           : InitInterval
Current Setting: 30
Description    : Number of dials before reinitializing modem

Name           : LogMethod
Current Setting: File
Description    : Log Method (accepted: File)

Name           : NudgeString
Current Setting: \x1b\x1b\r\n\r\n
Description    : Nudge String

Name           : Parity
Current Setting: None
Description    : Parity (Mark & Space are Windows Only) (accepted: None, Even,
   Odd, Mark, Space)

Name           : RedialBusy
Current Setting: false
Description    : Redials numbers found to be busy

Name           : StopBits
Current Setting: 1
Description    : Stop Bits (accepted: 1, 2)
```

2.4.4 Use

Given the configuration options outlined in Section 2.4.3, the Metasploit Wardialer is fairly simple to use:

```
> use auxiliary/scanner/telephony/wardial
> set DIALMASK 512.867.XXXX
> set DIALPREFIX ATDT *67
```

... set other relevant options ...

```
> run
```

```
[*] No previous scan data found (/home/druid/.msf3/logs/wardial/512.276.XXXX.dat)
[*] Detected 4 masked digits in DIALMASK (512.276.XXXX)
[*] Generating storage for 10000 numbers to dial
[*] Initializing Modem
[*] 10000 of 10000 numbers unidentified, 0 carriers found, 0 faxes found, 0 busy
[*] Dialing: 512.276.#### (45 sec. timeout, previously undialed)
[*] Timeout
[*] 9999 of 10000 numbers unidentified, 0 carriers found, 0 faxes found, 0 busy
[*] Dialing: 512.276.#### (45 sec. timeout, previously undialed)
[*] Fax: +FCO
[*] Initializing Modem
[*] 9998 of 10000 numbers unidentified, 0 carriers found, 1 faxes found, 0 busy
[*] Dialing: 512.276.#### (45 sec. timeout, previously undialed)
...
```

In the above example, the Modem was set to Fax mode and was therefore scanning for fax machines. In the output above, a Fax machine was detected on the second dial. The modem's output is displayed following the "Fax:" identifier and the identified Fax machine is then represented in the following status lines preceding each subsequent dial.

Chapter 3

Deficiencies and Future Goals

As with any young project, there are a number of deficiencies in the current implementation with many opportunities for future development paths and goals. The following key areas have been identified and may garner future development effort.

3.1 Direct VoIP Support

While Voice-over-IP (VoIP) technologies can be used with the Metasploit Framework, it is currently relegated to being used externally to a physical Serial Port and attached Modem. The physical Modem may be attached to a VoIP Analog Telephone Adapter (ATA) as you would any other analog telephony device, which in turn uses VoIP protocols to interface with your VoIP service provider. This limitation primarily restricts the number of simultaneous calls that could be made down to a single call at a time. While this limitation is fairly insignificant for dialup exploitation, it severely limits the capability of the Metasploit Wardialer, which could greatly increase efficiency with the ability to dial more than a single number at a time.

This limitation currently exists due to lack of adequate VoIP Digital Signal Processing (DSP) software. There are a few key projects that exist and show promise, however none of them are currently usable "out of the box" or easily integrated. IAXModem[5] is one of the more promising VoIP-enabled software modems, however the underlying DSP that it uses currently only supports Fax signal processing and not Data. Other DSPs exist but are not easily coupled with VoIP software.

Notably, this limitation in available DSP software is the primary reason that the WarVOX Project[6] approached target detection by performing audio signal analysis. While this proved to be a benefit to the WarVOX project as it enabled the tool to detect many more disparate systems than just data or fax carriers, as well as group and associate like-sounding systems, it also obviously introduced limitations in the area of data system identification of not actually connecting to a target to determine such parameters as a carrier negotiation speed or reading a system's login banner to help identify the type of system.

3.2 More Exploits

Due to the age of this project, much more work has been allocated to development of the core software and the wardialer than to pre-packaged exploits. Some additional vulnerabilities have been initially identified which may be applicable to exploitation over a dialup connection:

1, CVE-2002-1391[7, 8]

mgetty ¡ 1.1.20 CallerID Excessive Name Length cnd-program() Argument Buffer Overflow

This vulnerability should be supportable once direct VoIP support is added and the CallerID value able to be controlled.

2, CVE-2003-0574[9, 10]

SGI IRIX Scheme Login Privilege Escalation

3, CVE-2003-0686[11, 12]

PAM SMB module (pam_smb) ¡= 1.1.6 /bin/login Buffer Overflow

3.3 Non-Carrier Signal Processing

It is intended that the Metasploit Wardialer will be used as the primary distributed wardialer for the *TIDbITS* database. As such, there will eventually be a need for analysis of non-data carrier targets such as PBXs and Voice-mail systems which will require some other form of signal analysis. The WarVOX Project has made significant advances in this area specifically tied to analysis of information systems. As a sister project to the Metasploit Framework, it is hopeful that some of the WarVOX code may be integrated back into the Metasploit Wardialer.

Bibliography

[1] The Metasploit Project. The metasploit framework. http://www.metasploit.com/, May 2009.

[2] System v derived /bin/login many arguments buffer overflow. CVE 2001-0797, Mitre, June 2002.

[3] Mark Dowd. Multiple vendor system v derived 'login' buffer overflow vulnerability. BID 3681, ISS X-Force, December 2001.

[4] I)ruid. Tidbits. http://tidbits.caughq.org/, July 2009.

[5] faxguy. Iaxmodem. http://sourceforge.net/projects/iaxmodem/, June 2009.

[6] HD Moore. Warvox. http://warvox.org/, May 2009.

[7] mgetty $<$ 1.1.29 callerid excessive name length cnd-program() argument buffer overflow. CVE 2002-1391, Mitre, September 2004.

[8] Mgetty caller id excessive name length buffer overrun vulnerability. BID 7303, mgetty, November 2002.

[9] Sgi irix scheme login privilege escalation. CVE 2002-1391, Mitre, July 2003.

[10] Sgi irix scheme login privilege escalation vulnerability. BID 8212, SGI, July 2003.

[11] Pam smb module $<=$ 1.1.6 /bin/login buffer overflow. CVE 2002-1391, Mitre, August 2003.

[12] Craig Miskell. Pam_smb remote buffer overflow vulnerability. BID 8491, August 2003.