



# Contents

1 Foreword



### B.3.21 packet\_add\_exception

# Chapter 1

## Foreword

**Abstract:** Meterpreter, short for *The Meta-Interpreter* is an advanced payload

## Chapter 2

### Introducti5







## Chapter 3

# Technical Reference





Name	Value
TLV_META_TYPE_NONE	0 << 0
TLV_META_TYPE_STRING	1 << 16
TLV_META_TYPE_UINT	1 << 17
TLV_META_TYPE_RAW	1 << 18
TLV_META_TYPE_BOOL	1 << 19
TLV_META_TYPE_GROUP	1 << 20
TLV_META_TYPE_COMPLEX	1 << 21

Based on the above meta-types the following predefined TLVs have been generated which are used to provide core functionality to the meterpreter client and server.

**TLV\_TYPE\_ANY**

Meta-Type	Identifier
TLV_META_TYPE_NONE	0

The ANY

TLV\_TYPE

## TLV\_TYPE\_LENGTH

Meta-Type	Identifier
TLV_META_TYPE_UINT	25



This TLV holds the target path to upload a library to when it's being saved to disk on the remote client's machine.

#### TLV\_TYPE\_CIPHER\_NAME

Meta-Type	Identifier
TLV_META_TYPE_STRING	500

Holds the name of the cipher that is be used to encrypt the data stream between



with the method set to `core_crypto_negotiate`





```
{
    packet_add_tlv_uint(response, TLV_TYPE_RESULT, 0);

    packet_transmit(remote, response, 0);
}

return ERROR_SUCCESS;
}
```

Using this simple event drive422-sriev422-driisle422-extensispong422-e(e)-xp}

```
        identifier => "echo",  
        description => "Sends an echo request to the server.",  
        handler    => \&echo,  
    },  
);
```

The above code block will cause the following output to be displayed when a



```
        method => "echo");

$client->writeConsoleOutput(text =>
    "Sending echo request to server...\n");

$client->transmitPacket(
    packet            => \$request,
    completionHandler => \&echoComplete);

return 1;
}
```

When the user types echo into the command line, an

## Chapter 4

# Using Meterpreter

Meterpreter has been fully integrated into the Metasploit Framework in version 2.3 and can be accessed through a number of different payloads. At the time of this writing meterpreter has only been implemented on Windows but its principals and design are fully portable to a variety of other operating sys-



start the Metasploit client interface. Though Metasploit provides a number of interfaces (including `msfweb`), `msfconsole` will be used for illustration purposes.

```
$ ./msfconsole
```

$\frac{\text{ms}_{>}}{\text{a(t)-080(the)-09(pur)1omep thefipset t0ige(t)1oedotpicanntor(of)-09(dt)1(e)-1mon-|}$

```
msf Tester(win32_reverse_meterpreter) > exploit
[*] Starting Reverse Handler.
[*] Sending 270 bytes to remote host.
[*] Got connection from 127.0.0.1:5556 <-> 127.0.0.1:2029
[*] Sending Stage (2835 bytes)
[*] Sleeping before sending dll.
[*] Uploading dll to memory (69643), Please wait...
[*] Upload completed
meterpreter>
[ -= connected to == ]
[ -= meterpreter server == ]
[ -= v. 00000500 == ]
meterpreter>
```

```
loadlib: Loading library from 'ext950591.dll' on the remote machine.  
meterpreter>  
loadlib: success.  
meterpreter>
```





# Appendix A

## Command Reference

### A.1 Built-in Commands

#### A.1.1 use

Usage: use -m module1,module2,module3 [ -p path ] [ -d ]

#### Arguments

## Arguments

-f	Specifies the path from which the library should be loaded. If the -l parameter is specified, the path is rel-
----	--

## Arguments

--	--



This command closes a channel and frees its resources. After a

This command provides the client with the ability to enable an arbitrary cipher which will as a result encrypt the



src	One or more files on the remote server that are to
-----	--

-a	Indicates that the port forward is to be added. This instruction is mutually exclusive with -r and -v.
----	--



pid	The unique process identifier or one or more processes that should be terminated.
-----	---

This command is similar to the `kill` command that is found on most UNIX derivatives. Its syntax is

Reverts the server's thread to the identify that was associated with



## Appendix B

# Common API

The common API is an interface that is shared between the interpreter client and server. It provides things like channel management, packet management, and other various interfaces that are common to both the client and the server. The following subsections define the C interface for these subsystems. The interface also matches nearly one to one with the interface supplied in perl by `metasploitTdd39` framework. This interface can be found

```
Channel *channel_find_by_id(DWORD id);
```

### **Arguments**









### B.1.8 cha8nel\_close

Prototype

### B.1.9 channel\_interact

#### Prototype

```
DWORD channel_interact(Channel *channel, Remote *remote,  
    Tlv *addend, DWORD addendLength, BOOL enable,  
    Channel CompletionRoutine *completionRoutine);
```

#### Arguments

channel	The channel instance that is to be closed.
remote	The remote connection management object that is used for the transmission of packets.
addend	An array of TLV addends to be included in the core_channel_close request. This parameter is optional and should be NULL if there are no addends.
addendLength	The number of bytes in the addend array.





## Summary

This function deregisters a command handler that was previously registered with the `command_register` function.

```
Packet *packet_create_response(Packet *packet);
```

### **Arguments**

packet	The packet that is to be duplicated.
--------	--------------------------------------

**Returns**

On success a pointer to a valid Packet instance is returned that is a duplicate of the packet passed in. Otherwise, NULL is returned.

**Summary**

This function create204.03(Su-1(20t)1(e)-327of)-327(the)-327(pa)1(c)27(k)(O)1(34a)eci0t)-326pa

\_\_\_\_\_

```
DWORD packet_add_tlv_uint(Facket *packet, TlvType type,  
    UINT val);
```

### Arguments

packet	The packet instance that is to be operated on.
type	The unique TLV type identifier to add. This type should have a meta-type of TLV_META_TYPE_UINT.
val	

## Summary



## Arguments

packet	The packet instance that is to be operated on.
type	The unique TLV type identifier to add. This type should have a meta-type of TLV_META_TYPE_RAW.
buf	The raw data that should be used as the value for the TLV.
length	

### B.3.13 packet\_is\_tlv\_null\_terminated

#### Prototype

```
DWORD packet_is_tlv_null_terminated(Packet *packet,  
                                     Tlv *tlv);
```

#### Arguments



## Returns

On success, zero is returned. Otherwise, a non-zero value is returned



## Returns

On success, zero is returned. Otherwise, a non-zero value is returned that indicates the error that occurred.

## Summary

This function populates the buffer supplied in `tlv` with information about the TLV type specified by `type` and the index supplied in `index`.

If the `type` parameter is set to `TLV_TYPE`.

## Arguments

packet type	The packet instance that is to be operated on.
----------------	--



### B.3.23 packet\_transmit

#### Prototype

```
DWORD packet_transmit(Remote *remote, Packet *packet,
```



## Returns

## Summary

This function initializes the local half of the encrypted channel and populates the packet supplied in `initializer` with the parameters that will be necessary for the remote half to complete its portion of the negotiation.

### B.4.2 `remote_get_cipher`

#### Prototype

```
CryptoContext *remote_get_cipher(Remote *remote);
```

#### Arguments

--	--

### B.5.1 scheduler\_



# Bibliography

- [1] skape and Jarkko Turkulainen. *Remote Library Injection*.