# About the Metasploit Meterpreter

---

## What is Meterpreter?

Meterpreter is an advanced, dynamically extensible payload that uses *in-memory* DLL injection stagers and is extended over the network at runtime. It communicates over the stager socket and provides a comprehensive client-side Ruby API. It features command history, tab completion, channels, and more.

Metepreter was originally written by skape for Metasploit 2.x, common extensions were merged for 3.x and is currently undergoing an overhaul for Metasploit 3.3. The server portion is implemented in plain C and is now compiled with MSVC, making it somewhat portable. The client can be written in any language but Metasploit has a full-featured Ruby client API.

*Contents*



## How Meterpreter Works

- The target executes the initial stager. This is usually one of *bind, reverse, findtag, passivex*, etc.
- The stager loads the DLL prefixed with Reflective. The Reflective stub handles the loading/injection of the DLL.
- The Metepreter core initializes, establishes a TLS/1.0 link over the socket and sends a GET. Metasploit receives this GET and configures the client.
- Lastly, Meterpreter loads extensions. It will always load stdapi and will load priv if the module gives administrative rights. All of these extensions are loaded over TLS/1.0 using a TLV protocol.

## Meterpreter Design Goals

### *Stealthy*

- Meterpreter resides entirely in memory and writes nothing to disk.
- No new processes are created as Meterpreter injects itself into the compromised process and can migrate to other running processes easily.
- By default, Meterpreter uses encrypted communications.
- All of these provide limited forensic evidence and impact on the victim machine.

### *Powerful*

- Meterpreter utilizes a channelized communication system.
- The TLV protocol has few limitations.

### *Extensible*

- Features can be augmented at runtime and are loaded over the network.
- New features can be added to Meterpreter without having to rebuild it.

## Adding Runtime Features

New features are added to Meterpreter by loading extensions.

- The client uploads the DLL over the socket.
- The server running on the victim loads the DLL in-memory and initializes it.
- The new extension registers itself with the server.
- The client on the attackers machine loads the local extension API and can now call the extensions functions.

This entire process is seamless and takes approximately 1 second to complete.

In the next Metasploit Unleashed tutorial we'll discuss some of the various Meterpreter Commands available to us in this new environment.