POSIX Meterpreter

By Sebastian Fernandez

About me

=> Sebastian Fernandez

=> Estudiante de Ingeniería Electronica

=> Participante en varios proyectos Open Source

=> Actualmente desarrollando el Meterpreter(POSIX) para Metasploit.

Presentacion

=> Introducción a Meterpreter

=> Por que la necesidad de un Meterpreter para POSIX y sus ventajas.

=> Problemas y soluciones a la implementacion en los sistemas *nix (POSIX)

=> Demostracion

=> Questions

Introduccion

Por que Meterpreter para *nix ?

=> Metasploit es uno de los frameworks mas usados para desarrollo de exploits y pen-testing ; Meterpreter es la payload por defecto para win32.

=> Encriptacion del canal de transmicion de datos.
Atacker -> SSL Channel -> Target

=> Facil automatizacion a traves de la API de Meterpreter integrada al framework.

=> Soporte para extensiones (plugins).

Anteriores implementaciones

=> Meterpreter win32:

=> Direccion conocida de GetProcAddress y LoadLibrary

=> Usa un PE loader customizado para cargar el core de Meterpreter y sus extensiones de memoria.

=> Dll estandar, no se usan “trucos” de compilacion.

Problemas en *NIX

=> Direccion de dlopen() no conocida(libc/libdl) en cambio de GetProcAddress(win32).

=> Resolucion de simbolos hechas por el runtime loader

=> Incompatibilidad entre los metodos para cargar las librerias entre las distintas implementaciones de *nix, depende de cada rtld/libc.

Soluciones existentes

=> Meterpretux:

=> No es libre y no se conoce mucho de su arquitectura y extensibilidad.

=> Shellcodes con resolucion de simbolos:

=> No funcionan en programas linkeados estaticamente.

=> Problemas para agregar encriptacion y extensiones debido a que deberia resolver simbolos con librerias locales.

Solucion

=> Combinar tecnicas conocidas, uso de tecnicas de compilacion con gcc/ld.

=> Crear una shellcode estatica de memoria fija para facilitar inyeccion.

=> Incorporar un RTLD en nuestra shellcode que cargue las librerias en memoria.

=> Independizar el codigo de las librerias locales.

- **=>** (micro) Libc, libcrypto y libssl comprimidas enviadas dentro de la shellcode .

=> Producir la explotacion en 2 stages:

- **=>** La primer stage carga la segunda y se encarga de mapearla en la memoria prefijada.
- **=>** La segunda shellcode se descomprime a si misma y ejecuta el RTLD, resuelve simbolos entre las librerias y corre la rutina principal.

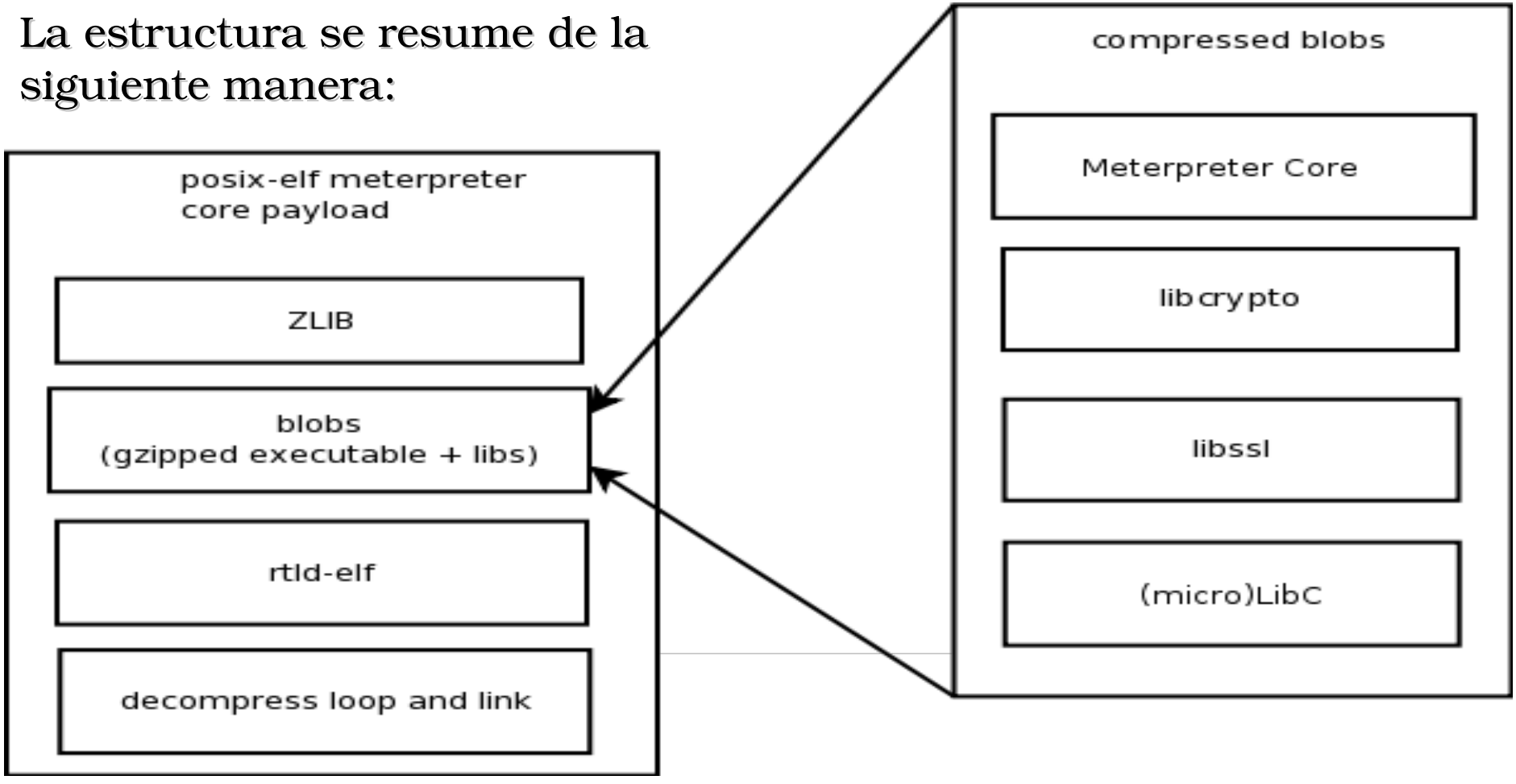
=> Compilar todo como PIC, el RTLD se encarga de alinear las tablas de offsets.

=> Para portar el Meterpreter a otras plataformas solo es necesario extender la libc.

Estructura

Estructura

La estructura se resume de la siguiente manera:



Loader y RTLD

=> Partes del rtld-elf portado de FreeBSD.

=> No puede contener llamadas a libc ni otras librerias que no se encuentren estaticas en la shellcode.

=> No usa el disco rigido, uso intensivo de mmap para cargar las librerias de la memoria.

Meterpreter Core

=> Codigo reusado del meterpreter para win32

=> Se adapta todo el API a POSIX usando la libc customizada.

=> Se mantienen las interfases anteriores, consecuentemente manteniendo el api de Meterpreter (casi) intacto.

=> Compilado en posicion independiente (PIC) para ser cargado por el RTLD.

Librerias

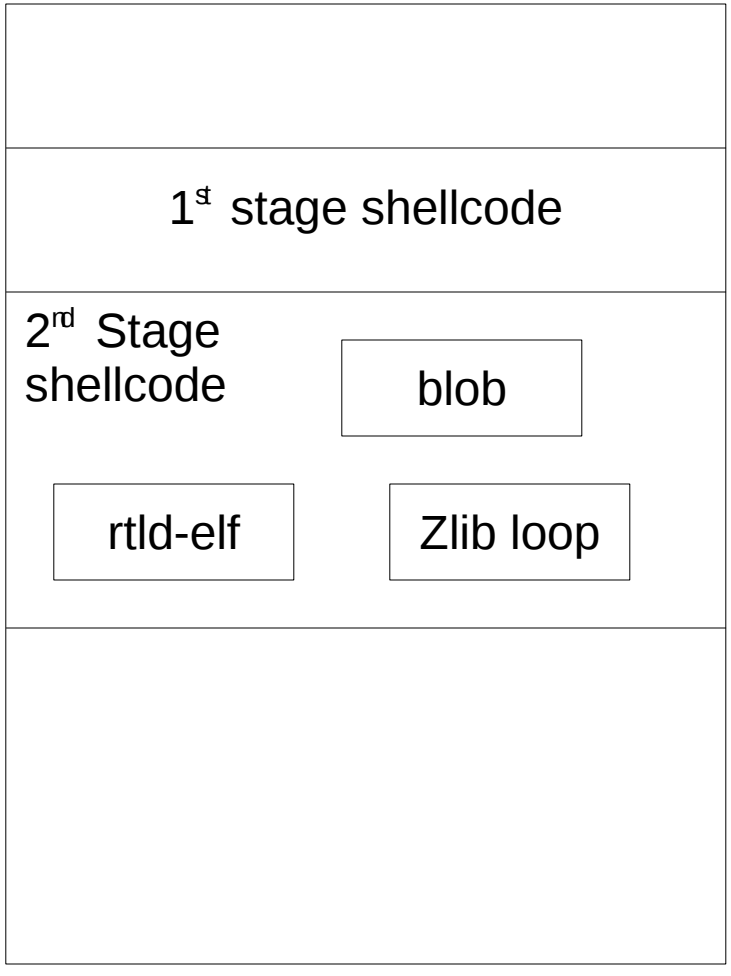
=> Tamanos:

- zlib	~ 90kb
- ulibc (gzipped)	~ 17kb
- libcrypto (gzipped)	~ 620kb
- libssl (gzipped)	~ 130kb
- rtld	~ 90kb
- core (gzipped)	~ 800kb
- total	~ 2.5mb

Funcionamiento

Funcionamiento [1ª Stage]

En la primera etapa la shellcode mapea la seccion de memoria , carga el elf remotamente y ejecuta la rutina principal



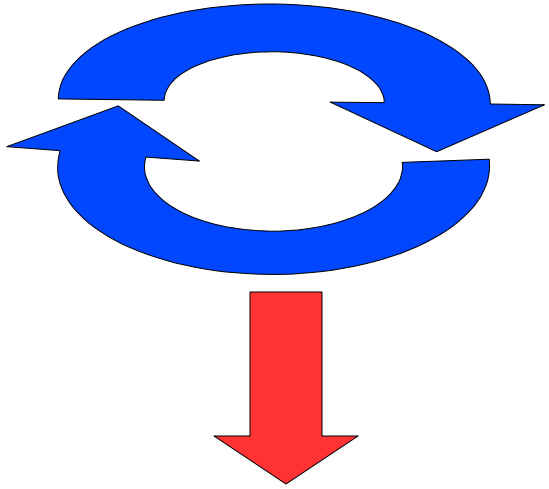
Memoria prefijada, definida en el linkeo de la 2da stage shellcode

Funcionamiento [2ª stage]

Un loop se encarga de descomprimir las librerias (zlib) y el RTLD las carga resolviendo simbolos entre el core y las librerias.
Salta a la rutina principal dentro del Meterpreter Core.

Funcionamiento [loop]

El core realiza la negociacion SSL reusando el socket de la 1ra shellcode.



Se ejecuta el loop que recibe comandos y los procesa, carga librerias, extensiones y transmite datos; hasta que el cliente corte la conexion.

Estado Actual de Meterpreter

=> Meterpreter core funcionando en Linux32, FreeBSD32/64

=> Loader funcionando (beta) en Linux32, FreeBSD32/64. Sobreescribe el proceso corriendo.

=> Extension Stdapi implementada parcialmente(fs).

TODO

=> Terminar de implementar la extension stdapi, entre otras cosas agregar un sistema de migracion de proceso.

=> Probarlo bajo la mayor cantidad de entornos.

=> Unir el codigo con el de Win64 y prepararlo para el release 3.3 de Metasploit (?).

=> Portar libc a linux64 y MacOSX.

DEMO

Gracias!

PREGUNTAS?

<Sebastian Fernandez>

Referencias

ELF Specification
http://www.skyfree.org/linux/references/ELF_Format.pdf

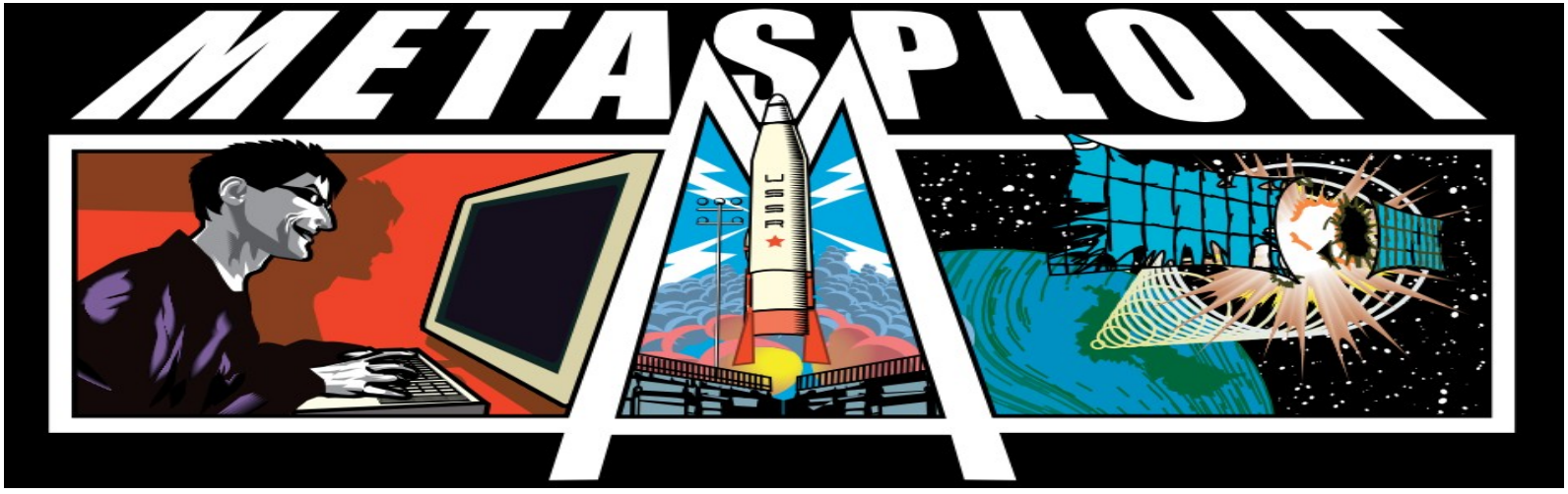
Metasploit Framework
http://www.metasploit.com/framework/

[3] Impurity Project
http://archives.neohapsis.com/archives/vuln-dev/2003-q4/0006.htm

FreeBSD Project
http://www.freebsd.org

Palabras finales

Muchos creditos y agradecimientos a JR quien comenzo el proyecto y me incorporo al equipo y a Roberto Cavalcanti por colaborar en esta presentacion.



Sebastian N. Fernandez - sebastianfernandez6@gmail.com