Università degli Studi di Catania
Corso di Laurea Specialistica in Ingegneria Informatica
Sicurezza nei Sistemi Informativi
Docente prof. O. Tomarchio

Relazione su

METASPLOIT FRAMEWORK

Studenti
Coco Ignazio Andrea
Lazzara Alessandro

Definizione di Exploit

- Exploit è il termine usato in informatica per identificare un metodo che, sfruttando un bug o una vulnerabilità, porta all'acquisizione di privilegi o al denial of service (DoS)
- Lo scopo di molti exploit è quelli di prendere i privilegi di root su un sistema.
- È' comunque possibile usare exploit che dapprima acquisiscono un accesso con i minimi privilegi e che poi li alzano fino ad arrivare a root.
- Normalmente un exploit può sfruttare solo una specifica falla e quando è pubblicato questa falla è riparata e l'exploit diventa obsoleto
- Per questo motivo alcuni blackhat hacker non divulgano gli exploit trovati ma li tengono riservati per loro o per la loro comunità. Questi exploit sono chiamati *zero day* exploit.

Organizzazione della memoria di un processo

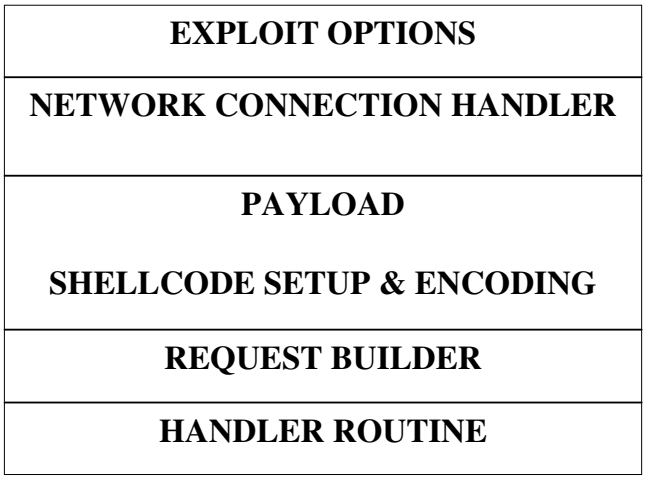
- ***memoria istruzioni***: un segmento in sola lettura che contiene il codice del programma
- ***memoria dati***: accesso in lettura e scrittura; contiene dati statici, globali, inizializzati e non, e le variabili usate dal programma
- ***stack***: struttura dati basata su ordinamento LIFO. Usato per memorizzare il contenuto di un processo a run-time
- ***stack pointer(SP):*** registro che punta alla cima dello stack.
- ***instruction pointer(IP):*** registro che punta all'indirizzo della successiva istruzione che deve essere eseguita. Usato per tenere traccia del corretto flusso di istruzioni di un normale programma
- ***heap*****:** parte restante di memoria che viene assegnata ad un processo. Vi sono memorizzate le variabili che hanno bisogno di essere allocate a run-time.

Overflow

Evento che si verifica quando un segmento di memoria viene saturato ed il contenuto in eccesso viene scritto in aree di memoria diverse da quelle attese.

- **Buffer overflows:** Lo stack di un programma memorizza i dati nel seguente ordine: prima i dati passati alla funzione, il return address, il precedente puntatore allo stack e di seguito le variabili locali. Se le variabili (come gli array) sono passati senza controllo dei limiti, scrivendo in essi un largo quantitativo di dati si va a corrompere lo stack del processo. In questo modo, l'indirizzo di ritorno viene sovrascritto conseguentemente ad un *segmentation fault*. Permette la modifica del buffer in modo che l'indirizzo di ritorno punti ad una specifica locazione per poter eseguire un codice arbitrario.
- **Heap overflows:** è organizzata come una doppia linked list. Effettuando un overflow su questa struttura dati possiamo modificare la linked list in modo da puntare ad una locazione di memoria arbitraria. L'overflow dell'heap è difficilmente sfruttabile in generale, ma in ambiente Windows tale vulnerabilità è sfruttabile facilmente

Struttura di un Exploit



Struttura di un Exploit

- **Exploit Options:** parametri da passare all'exploit (ip, porta, etc.)
- **Network connection Handler**: contiene varie routine per gestire le connessioni di rete
- **Shellcode:** payload che va eseguito dopo l'exploit. Generalmente il flusso di esecuzione del programma è ridirezionato in modo che il payload iniettato venga eseguito tramite la modifica del return address. Lo shellcode è un blocco di istruzioni (generalmente in assembly) che viene codificato come una stringa binaria ed esegue determinate operazioni. Un buon pezzo di shellcode deve essere un compromesso tra dimensione e complessità.
- **Request builder:** permette di attivare l'exploit
- **Handler Routine:** un gestore per lo shellcode che effettua operazioni come il collegamento di una shell remota o la connessione di una console ad un socket

Metasploit Framework

"The Metasploit Framework is a complete environment for writing, testing, and using exploit code. This environment provides a solid platform for penetration testing, shellcode development, and vulnerability research."

- Il progetto include una serie di tool: il framework, lo shellcode ed il database dei moduli
- Il framework rappresenta la principale interfaccia verso l'insieme di librerie che permette lo sviluppo degli exploits e dei payloads. Permette di effettuare un penetration test in base a specifici requisiti.
- Estremamente flessibile
- Scritto in Perl: lo rende facilmente portabile su molti sistemi operativi. I componenti sono stati scritti in C, assembler e Python.
- Lo Shellcode contiene un archivio di payload già compilati come moduli del sistema e pronti per essere usati all'interno dell'ambiente di lavoro.
- Il database degli OpCode fornisce un metodo per la ricerca degli OpCode che verranno usati nei moduli. Evita la ricerca su Internet.

Caratteristiche del MSF

- Scritto in Perl: si traduce in pulizia ed efficienza del codice e rapido sviluppo dei plug-in
- Supporta tool per l'estensione delle funzionalità: debug, encoding, logging timeout, SSL e random NOPS
- Un set di API per gli exploit intuitivo e modulare
- Ambiente multipiattaforma ottimizzato con un set di payload comuni
- Ottimizzato per generare exploit corti ed efficienti
- Include exploit supplementari per testare le tecniche di exploit più comuni e usare alcuni di essi come base di partenza per la creazione di nuovi
- E' open source!

Ambiente Operativo

Collega le variabili ed i moduli; le varie interfacce utente lo usano per configurare le impostazioni, i payload lo usano per impostare gli opcode, gli exploit lo usano per definire i parametri e viene internamente usato per passare le opzioni tra i vari moduli.

- **Global Environment:** permette di definire le impostazioni globali per l'intero ambiente. Ad ogni avvio del sistema le impostazioni globali precedenti, salvate su disco, vengono caricate
- **Temporary Environment:** le impostazioni di questo ambiente si applicano solamente al modulo correntemente caricato; il cambio ad un altro exploit comporterà un cambio dell'ambiente temporaneo. Quello vecchio viene salvato mentre viene caricato quello nuovo

Principali Variabili d'ambiente

Permettono di configurare l'ambiente di lavoro secondo le proprie esigenze

- Configurazione della rete: RecvTimeout, RecvTimeoutLoop, Proxies, ForceSSL, UdpSourceIp, ConnectTimeout
- Configurazione dell'ambiente: DebugLevel, Logging
- Configurazione degli Exploit: Encoder, EncoderDontFallThrough, Nop, NopDontFallThrough

Exploit Integrati (esempi)

- **Microsoft ASN.1 Library Bitstring Heap Overflow:** la libreria che gestisce ASN.1 contiene una vulnerabilità. Questo bug permette una sequenza ostile che sovrascrive da remoto la sezione di heap memory tramite il servizio che esegue il parsing della BITSTRING ASN.1. Servizi affetti: NetBIOS, SMB, IPSEC, Kerberos, SSL e IIS. Con un particolare pacchetto dati è possibile eseguire del codice con i privilegi del componente "bacato".
- **Microsoft LSASS MSO4-011 Overflow:** Un overflow remoto è presente in Windows (tristemente noto per essere stato sfruttato da numerosi worm). L'LSA (Local Security Authority) fallisce nel controllo dell'input ricevuto sulla pipe LSARPC sulla porta TCP 139 e 445 causando un buffer overflow.
- **Internet Explorer WebViewFolderIcon setSlice() Code Execution:** IE contiene un bug che causa DoS. Quando viene chiamato il metodo "setSlice" su un certo oggetto ActiveX viene attivato il bug, passando come paramentro il valore 0x7fffffff. Ciò causa un'eccezione che permette di eseguire codice arbitrario e/o una perdita di funzionalità del browser.

Payload

Codice che verrà eseguito sul sistema bersaglio, una volta che l'*exploit* avrà avuto successo

Può essere scritto:

- direttamente in linguaggio macchina byte per byte direttamente in un file
- in C, compilato e trasferito in un file mediante l'uso del debugger
- in assembly, assemblato e trasferito in un file mediante l'uso del debugger.
- in Perl

Payload Integrati (esempi)

- **Win32_adduser:** permette di inserire in sistemi operativi Microsoft Windows multiuser degli utenti con diritti di amministratore permettendo di specificare userid e password.
- **Linux IA32 Reverse Shell:** permette di ottenere una shell remota su un sistema Linux
- **Windows Bind Shell:** permette di ottenere una shell remota su un sistema Windows
- **Windows Reverse VNC Server Inject:** inietta una DLL che apre un server VNC in ascolto sulla macchina remota. Una volta avviato il server, viene attivata sulla macchina che ha eseguito l'attacco un client VNC che viene collegato all'host remoto

NOP Generator

- **Significato**: "No Operation". Effettivamente questa rappresenta una particolare operazione che non esegue alcuna azione.
- **Scopo**: Sincronizzazione temporale.
- **Utilizzo "malevolo"**: far in modo che sia eseguito un codice arbitrario. Mediante essi si può giungere, dopo un buffer overflow, alla locazione del codice che vogliamo sia eseguito

Encoder

- Permettono di mescolare le NOP ed il payload.
- La maggior parte degli encoder si basa su un algoritmo che modifica parte del payload. Questo spesso offre anche un decoder in modo che quando il target riceve il payload esso possa 'capire' il contenuto dopo avere applicato il decoder.
- L'encoder permette di mascherare un certo payload mischiando i dati e facendo sembrare così il payload come normale traffico dati in modo da sfuggire ai moderni sistemi di intrusion detection
- La maggior parte degli amministratori di sicurezza cerca di bloccare le azioni degli encoder studiandone il funzionamento e cercando di prevedere il risultato dopo l'applicazione su un payload.

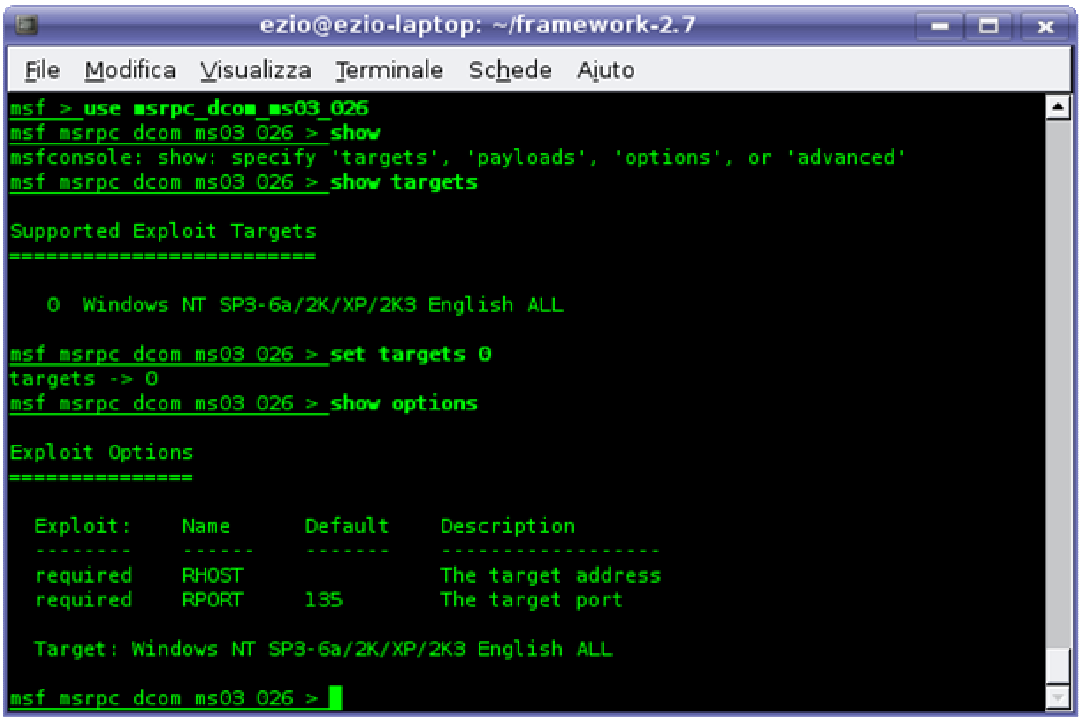
Encoder Integrati (esempi)

- **Countdown:** usa un byte decrescente come chiave, ha come svantaggio il fatto di non funzionare bene con i payload di grosse dimensioni
- **PexAlphaNum:** affinchè questo payload sia alfanumerico al 100%, devono essere impostati tutti i caratteri non validi in una lista detta 'BadChar'. I vari IDS/IPS basati sulla signature cercano la stringa ASCII usata da questo encoder, quindi il suo utilizzo risulta facilmente rilevabile
- **PexFnstenvMov:** standard XOR encoder, usa l'OpCode Fnstenv per ottenere il valore dell'EIP e decodificare il payload

Interfaccia Console

- Rapida, efficiente e flessibile
- Più difficile da usare rispetto all'interfaccia Web
- Intuitiva, grazie alla guida facilmente accessibile ('?' o 'help') ed alla tab completions

Interfaccia Console



Interfaccia Console

- **show + nome elemento**:mostra una lista di elementi del database del framework
  - *exploits:*
  - *payloads:*
  - *encoders:*
  - *targets:* tipologia di obiettivi cui può essere applicato l'exploit
  - *options:* opzioni disponibili per l'exploit/payload selezionato
  - *advanced:* opzioni avanzate

Interfaccia Console

- **use:** permette di scegliere un certo exploit
- **setg** / **set:** permettono di impostare le variabili a livello globale / locale.
  - la variabile viene settata tramite il comando

    set/setg variabile valore
  - è possibile impostare qualunque tipo di parametro visualizzato con 'show option'
  - permette di impostare il payload ed il target
  - usata sempre per settare l'RHOST (host remoto)

Interfaccia Console

- **unsetg** / **unset**: usata per porre ai valori di default variabili globali / locali
- **info**: permette la visualizzazione di informazioni sui vari moduli dell'MSF (siano essi exploit, payload o encoder)

Interfaccia MSFcli

- Particolare interfaccia per consentire l'uso dell'MSF in uno script batch
- Permette l'esecuzione di un exploit con una sola riga di comando
- Utilissima per l'automatizzazione dei test

Interfaccia MSFcli

- La riga di comado ha la seguenti sintassi:

  ./msfcli match_string options(VAR=VAL) action_code

  - **./msfcli:** eseguibile
  - **match_string:** nome dell'exploit
  - **options(VAR=VAL):** varie opzioni nel formato variabile=valore (nome payload, host, porta, etc)
  - **action_code:** un codice (lettera alfabetica S O A P T C E), con cui si specifica l'operazione da effettuare

Interfaccia Web

- Semplice ed intuitiva
- Web Server: http://localhost:55555 di default
  - Limitato
  - Insicuro

Interfaccia Web

EXPLOITS | PAYLOADS | SESSIONS

os :: winxp | Filter Modules

- 3Com 3CDaemon FTP Server Overflow
- AOL Instant Messenger goaway Overflow
- Alt-N WebAdmin USER Buffer Overflow
- Apache Win32 Chunked Encoding
- BakBone NetVault Remote Heap Overflow
- Blue Coat Systems WinProxy Host Header Buffer Overflow
- CA BrightStor Agent for Microsoft SQL Overflow
- CA BrightStor Discovery Service Overflow

Interfaccia Web

EXPLOITS | PAYLOADS | SESSIONS

Microsoft LSASS MS04-011 Overflow

Name:	lsass_ms04_011 v (remote)
Authors:	H D Moore <hdm [at] metasploit.com> Brian Caswell <bmc [at] shmoo.com>
Disclosure:	Apr 13 2004
Arch:	x86
OS:	win32, win2000, winxp

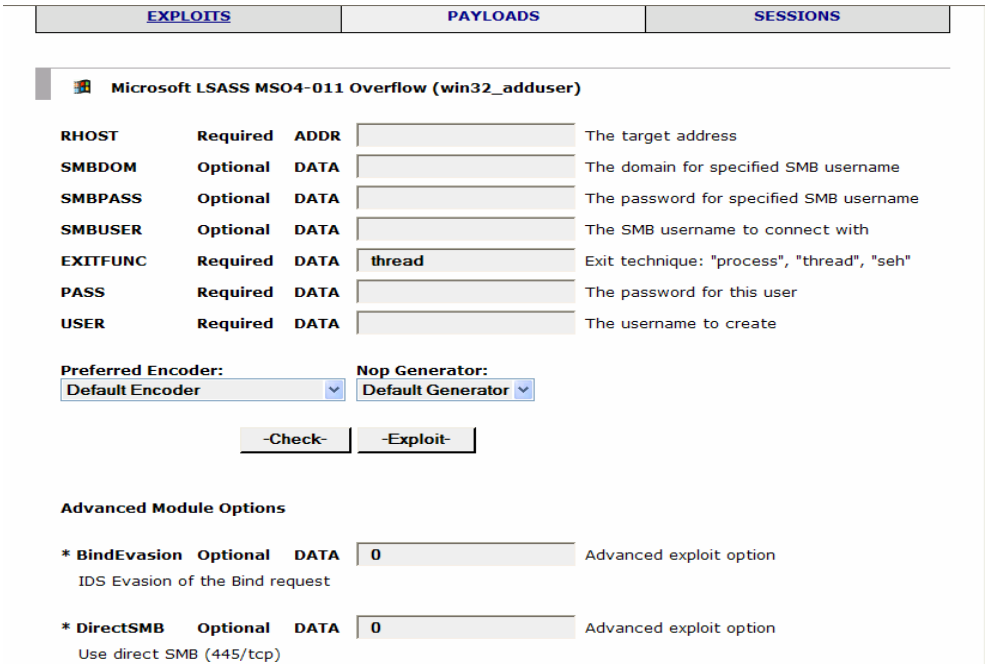
This module exploits a stack overflow in the LSASS service, this vulnerability was originally found by eEye. When re-exploiting a Windows XP system, you will need need to run this module twice. DCERPC request fragmentation can be performed by setting 'FragSize' parameter.

- http://www.osvdb.org/5248
- http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx
- http://www.milw0rm.com/metasploit/36

Select Target:

- 0 - Automatic (default)
- 1 - Windows 2000
- 2 - Windows XP

Interfaccia Web



Meterpreter

- Payload con caratteristiche speciali
- Consente di creare sull'host remoto un *meta interpreter*
- Invisibile prima e dopo l'attacco; caricato direttamente nello spazio di memoria del processo remoto senza causarne il crash
- Prescinde dal toolkit di sistema
- È estendibile
- Include dei moduli di base per le principali funzioni
  - gestione filesystem
  - gestione rete
  - gestione processi di sistema

Conclusioni

- Ottimo tool di penetration testing
- Consente lo sviluppo ed il test di exploit e payloads
- Buon punto di partenza per “neofiti” ma non sufficiente
- Exploit e Payloads integrati “obsoleti”
- Proteggersi
  - Aggiornamenti di sistema
  - Buon Antivirus
  - Buon Firewall
  - MOLTA ATTENZIONE!!!