Penetration Testing mit MetaSploit

Stefan Schumacher
www.sicherheitsforschung-magdeburg.de
Stefan.Schumacher@Sicherheitsforschung-Magdeburg.de

Open Source Workshop Erfurt
DB Systel



Inhaltsverzeichnis

1. Intro
2. Penetration Testing
3. Pen-Test Walkthrough
4. Einbruchsversuche
5. Meterpreter
6. Automatisierung
7. Social-Engineering-Toolkit
8. Beispielanwendung



Inhaltsverzeichnis

1. Intro
2. Penetration Testing
3. Pen-Test Walkthrough
4. Einbruchsversuche
5. Meterpreter
6. Automatisierung
7. Social-Engineering-Toolkit
8. Beispielanwendung



Über Mich

- Direktor des Magdeburger Instituts für Sicherheitsforschung
  Forschungsprogramme zur Unternehmenssicherheit
- Berater für Unternehmenssicherheit
  Organisationssicherheit, Social Engineering, Security Awareness
- Geek, Nerd, Hacker seit knapp 20 Jahren
- Herausgeber des Magdeburger Journals zur Sicherheitsforschung
- `www.Sicherheitsforschung-Magdeburg.de`
- CLT 2005: IDS; 2007: Systrace; 2010: KI und IDS



Inhaltsverzeichnis

1 Intro

2 Penetration Testing

3 Pen-Test Walkthrough

4 Einbruchsversuche

5 Meterpreter

6 Automatisierung

7 Social-Engineering-Toolkit

8 Beispielanwendung



Warum das alles?

- Penetration Tests untersuchen Systeme/Netzwerke auf Schwachstellen
- notwendig um die »Unsicherheiten« eines Systems zu finden
- Werkzeugkasten für Angreifer oder SysAdmins
- Sicherheit lässt sich nicht beweisen

  Schumacher, S. »Sicherheit messen. Eine Operationalisierung als latentes soziales Konstrukt«. In: *Die sicherheitspolitische Streitkultur in der Bundesrepublik Deutschland. Beiträge zum 1. akademischen Nachwuchsförderpreis Goldene Eule des Bundesverbandes Sicherheitspolitik an Hochschulen (BSH)*. Hrsg. von S. Adorf, J. Schaffeld und D. Schössler. Magdeburg: Meine Verlag, S. 1–38.
- Pen-Test beinhaltet Koordinierung von Gegenmaßnahmen
- eingesetzt im Vietnam-Krieg mit sog. Tiger-Teams die FOBs testeten
- komplexes Thema, kurze Einführung



Vulnerability, Exploit, Scanner

- Vulnerability: Sicherheitslücke, Sicherheitsproblem
- Exploit: Programm/Skript, das eine Vulnerability ausnutzt
  US-CERT Vulnerability Notes, Microsoft, BugTraq, Full Disclosure
- Scanner: Satan, Saint, Nessus, Google
- ebenso: IDS-Signaturen :-)
- Scanner arbeiten nur ihr Lexikon ab, keine Validierung $\rightsquigarrow$ False Positives
- False Positive: Meldung einer Vulnerability die keine ist
- False Negative: Meldung keiner Vulnerability wo eine ist
- Payload: Nutzlast eines Exploits, dass, was eigentlich auf dem Zielrechner ausgeführt wird :-)



Pen-Test

Phasen eines Pen-Test

1. **Vorbereitung:** Ziele definieren, Vorgehensweise, Kontaktpersonen, Black/White List, Compliance
2. **Informationen sammeln und auswerten:** Details über Ziele, Google, Scanner, MetaSploit, Angriffsvektoren, Karte des Netzwerks ...



Pen-Test

Phasen eines Pen-Test

1. **Vorbereitung:** Ziele definieren, Vorgehensweise, Kontaktpersonen, Black/White List, Compliance
2. **Informationen sammeln und auswerten:** Details über Ziele, Google, Scanner, MetaSploit, Angriffsvektoren, Karte des Netzwerks ...
3. **Risikoanalyse:** Angriffspotential aus Schwachstellen errechnen,



Pen-Test

Phasen eines Pen-Test

1. **Vorbereitung:** Ziele definieren, Vorgehensweise, Kontaktpersonen, Black/White List, Compliance
2. **Informationen sammeln und auswerten:** Details über Ziele, Google, Scanner, MetaSploit, Angriffsvektoren, Karte des Netzwerks ...
3. **Risikoanalyse:** Angriffspotential aus Schwachstellen errechnen,
4. **Einbruchsversuche:** Vulnerabilities exploiten, Beobachtung der Systeme sicherstellen



Pen-Test

Phasen eines Pen-Test

1. **Vorbereitung:** Ziele definieren, Vorgehensweise, Kontaktpersonen, Black/White List, Compliance
2. **Informationen sammeln und auswerten:** Details über Ziele, Google, Scanner, MetaSploit, Angriffsvektoren, Karte des Netzwerks ...
3. **Risikoanalyse:** Angriffspotential aus Schwachstellen errechnen,
4. **Einbruchsversuche:** Vulnerabilities exploiten, Beobachtung der Systeme sicherstellen
5. **Abschlussbericht:** Aufbereitung und Auswertung des Pen-Tests, der Logs etc.



Pen-Test

Phasen eines Pen-Test

1. **Vorbereitung:** Ziele definieren, Vorgehensweise, Kontaktpersonen, Black/White List, Compliance
2. **Informationen sammeln und auswerten:** Details über Ziele, Google, Scanner, MetaSploit, Angriffsvektoren, Karte des Netzwerks ...
3. **Risikoanalyse:** Angriffspotential aus Schwachstellen errechnen,
4. **Einbruchsversuche:** Vulnerabilities exploiten, Beobachtung der Systeme sicherstellen
5. **Abschlussbericht:** Aufbereitung und Auswertung des Pen-Tests, der Logs etc.



Pen-Test

- Pen-Test erfordert einen wohlorganisierten Werkzeugkasten
- Angriffsvektoren müssen überprüft werden: Passwörter, Trojaner, Rootkits installieren, Buffer Overflow, Format String, ...
- Portscanner, Servicescanner, Vulnerabilityscanner, Passwortscanner, Exploits, Payload, Passwort-Cracker
- hunderte Ziele, komplette Netzwerke, dutzende Betriebssysteme $\rightsquigarrow$ komplexeres Projektmanagement
- Metasploit ist ein Werkzeugkasten, der dafür sorgt dass alle Werkzeuge schön griffbereit liegen :-)
- Manchmal reicht auch ein einzelnes Werkzeug aus!



Pen-Test

- Pen-Test erfordert einen wohlorganisierten Werkzeugkasten
- Angriffsvektoren müssen überprüft werden: Passwörter, Trojaner, Rootkits installieren, Buffer Overflow, Format String, ...
- Portscanner, Servicescanner, Vulnerabilityscanner, Passwortscanner, Exploits, Payload, Passwort-Cracker
- hunderte Ziele, komplette Netzwerke, dutzende Betriebssysteme $\rightsquigarrow$ komplexeres Projektmanagement
- Metasploit ist ein Werkzeugkasten, der dafür sorgt dass alle Werkzeuge schön griffbereit liegen :-)
- Manchmal reicht auch ein einzelnes Werkzeug aus!

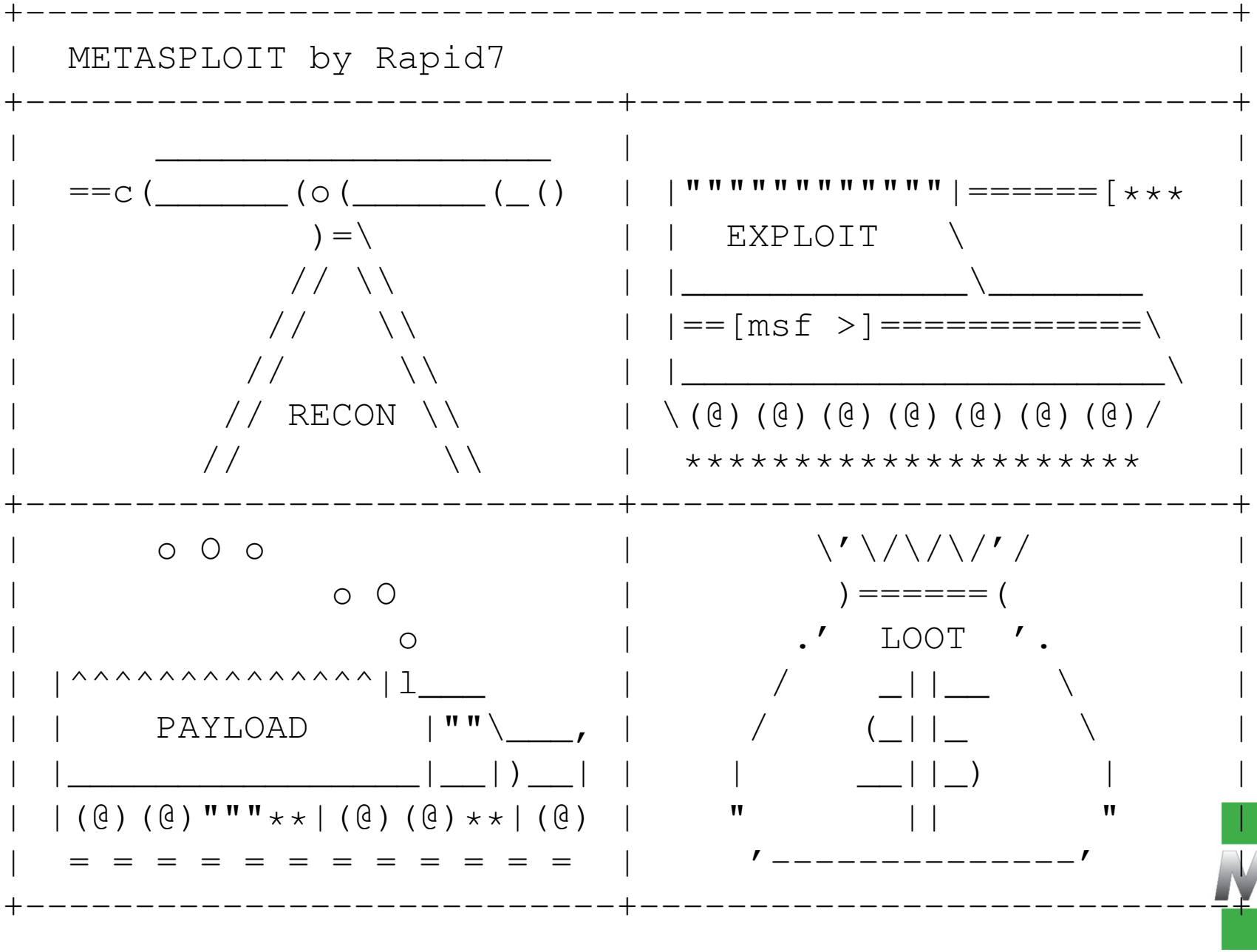


Metasploit

Das Framework

- Open Source, Modular, z. Zt. 1,2 Mio LoC
- ursprünglich in Perl, später in Ruby neu geschrieben
- das komplexeste und größte Ruby-Projekt am Markt :-)
- Rex: Ruby Extension Lib; Herzstück
- Metasploit Framework Base und Core
- Tools und Plugins
- GUIs: Armitage, msfgui, msfcli, msfconsole
- Modules: Exploit, Payload, Encoder, NOP, Auxiliary machen Metasploit erst nutzbar :-)
- Anbindung an PostgreSQL, MySQL, SQLite3




METASPLOIT by Rapid7
==c(_____(o(_____(_()
)=\
// \\
// \\
// RECON \\
// \\
|"""""""""""|======[***
| EXPLOIT \
|__________
|==[msf >]==========\
|_____\
\(@)(@)(@)(@)(@)(@)(@)/
****
o O o
o O
o
|^^^^^^^^^^^^^|l___
| PAYLOAD |""___,
|_____|__|)__|
|(@)(@)"""**|(@)(@)**|(@)
= = = = = = = = = = =
\'\/\/\/'/
)======(
.' LOOT '.
/ _||__ \
/ (_||_ \
| __||_) |
" || "
'--------------'

Inhaltsverzeichnis

1 Intro

2 Penetration Testing

3 Pen-Test Walkthrough

4 Einbruchsversuche

5 Meterpreter

6 Automatisierung

7 Social-Engineering-Toolkit

8 Beispielanwendung



Vorbereitung

- Ziele definieren, Vorgehensweise, Kontaktpersonen, Black/White List, Compliance
- organisatorische Maßnahmen :-)



Informationen sammeln und auswerten

- Details über Ziele sammeln, Netzwerk erkunden, Lagekarte erstellen
- Auxiliary Modules
- `search type:auxiliary`



Inhaltsverzeichnis

1 Intro

2 Penetration Testing

3 Pen-Test Walkthrough

4 Einbruchsversuche

5 Meterpreter

6 Automatisierung

7 Social-Engineering-Toolkit

8 Beispielanwendung



Einbruchsversuche

- Vulnerabilities exploiten, Beobachtung der Systeme sicherstellen
- gesammelte Informationen auswerten
- `search type:exploit`
- stark vom Zielsystem abhängig
- Exploit und Payload
- Exploit-Ranking: excellent, great, good, normal, average, low, manual



Inhaltsverzeichnis

1 Intro

2 Penetration Testing

3 Pen-Test Walkthrough

4 Einbruchsversuche

5 Meterpreter

6 Automatisierung

7 Social-Engineering-Toolkit

8 Beispielanwendung



Meterpreter

- Meta-Interpreter, Payload auf Opfer-System
- läuft im RAM statt auf Festplatte
- bindet sich an existierende Prozesse statt einen neuen zu erstellen
- unterstützt Staging (dynamisches Nachladen)
- SSL/HTTPS als Kommunikationsakanal vermindert Enttarnungsrisiken
- Anti-Virus-Evading-Funktion
- Systemunabhängigkeit
- Privilege Escalation
- Pivoting (Port-Forwarding)



Meterpreter Beispiele

- `ipconfig` - Netzwerkkonfiguration
- `ps` - Prozesse
- `sysinfo` - Systeminformation
- `hashdump` - dumpt die Passwort-Hashes unter Windows
- `uictl disable keyboard` - :-)
- `winenum` - sammelt Informationen über Zielhost



Inhaltsverzeichnis

1 Intro

2 Penetration Testing

3 Pen-Test Walkthrough

4 Einbruchsversuche

5 Meterpreter

6 Automatisierung

7 Social-Engineering-Toolkit

8 Beispielanwendung



makerc

- makerc speichert die bisher ausgeführten Befehle in einer .rc-Datei
- erstellt quasi ein Makro
- kann mit `<ruby> </ruby>` erweitert werden :-)



sshscan.rc

```
use auxiliary/scanner/ssh/ssh_login
set USER root
set PASSWORD root
set THREADS 256
<ruby>
print_status("Setze Ziel-Adresse auf 127.0.0.1")
</ruby>
set RHOSTS 127.0.0.1
run
```



db_autopwn

- automatisierte Exploitation kompletter Netze
  aka: Globale Thermonukleare Kriegsführung :-)
- verarbeitet Input von nmap, nessus, satan, saint ...
- `nmap -v -sSV -A 192.168.1.0/24 -oX nmap.xml`
- `msf> db_import nmap.xml`
- `hosts`
- `services`
- `vuln`
- Spass mit MeterPreter :-)



Inhaltsverzeichnis

1 Intro

2 Penetration Testing

3 Pen-Test Walkthrough

4 Einbruchsversuche

5 Meterpreter

6 Automatisierung

7 Social-Engineering-Toolkit

8 Beispielanwendung



Social-Engineering-Toolkit

- Vereinfachung technischer SE-Attacken
- Spear-Phishing
- Credential Harvesting
- Java-Applet-Injection
- Tabnapping
- Mass-Mailer
- SMS-Spoofing
- Metasploit Payloads etc.



Inhaltsverzeichnis

1 Intro

2 Penetration Testing

3 Pen-Test Walkthrough

4 Einbruchsversuche

5 Meterpreter

6 Automatisierung

7 Social-Engineering-Toolkit

8 Beispielanwendung



Module finden

```
msf > search type:auxiliary ssh
Matching Modules
================
Name                                                Rank    Description
----                                                ----    -----------
auxiliary/fuzzers/ssh/ssh_kexinit_corrupt           normal  SSH Key Exchange Init
auxiliary/fuzzers/ssh/ssh_version_15                normal  SSH 1.5 Version Fuzzer
auxiliary/fuzzers/ssh/ssh_version_2                 normal  SSH 2.0 Version Fuzzer
auxiliary/fuzzers/ssh/ssh_version_corrupt           normal  SSH Version Corruption
auxiliary/scanner/ssh/ssh_identify_pubkeys          normal  SSH Public Key Accepta
auxiliary/scanner/ssh/ssh_login                     normal  SSH Login Check Scanne
auxiliary/scanner/ssh/ssh_login_pubkey              normal  SSH Public Key Login S
auxiliary/scanner/ssh/ssh_version                   normal  SSH Version Scanner
```



ssh_version

```
msf > use auxiliary/scanner/ssh/ssh_version
msf  auxiliary(ssh_version) > set rhosts 127.0.0.1
rhosts => 127.0.0.1
msf  auxiliary(ssh_version) > run

[*] 127.0.0.1:22, SSH server version: SSH-2.0-OpenSSH_5.6
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```



ssh_login

```
msf > use auxiliary/scanner/ssh/ssh_login
msf  auxiliary(ssh_login) > set USER root
USER => root
msf  auxiliary(ssh_login) > set PASSWORD root
PASSWORD => root
msf  auxiliary(ssh_login) > set THREADS 256
THREADS => 256
msf  auxiliary(ssh_login) > set RHOSTS 127.0.0.1
RHOSTS => 127.0.0.1
msf  auxiliary(ssh_login) > run

[*] 127.0.0.1:22 SSH - Starting bruteforce
[*] 127.0.0.1:22 SSH - Trying: username: 'root' with password: ''
[-] 127.0.0.1:22 SSH - Failed: 'root':''
[*] 127.0.0.1:22 SSH - Trying: username: 'root' with password: 'root'
[-] 127.0.0.1:22 SSH - Failed: 'root':'root'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```



Literatur

- Magdeburger Institut für Sicherheitsforschung
- www.sicherheitsforschung-magdeburg.de
- stefan.schumacher@sicherheitsforschung-magdeburg.de
- Magdeburger Journal zur Sicherheitsforschung
- »Informationstechnologie und Sicherheitspolitik« (Mitte Oktober)
- youtube.de/Sicherheitsforschung

