Exploits & Rootkits

Betriebssystemdienste &
-administration
Seminar

Betriebssysteme und Middleware

Sebastian.Roschke@hpi.uni-potsdam.de

Steffen.Ryll@hpi.uni-potsdam.de

Exploits & Rootkits

- 1. Begriffserklärung
- 2. Exploits & Demo
- 3. Rootkits & Demo
- 4. Schutz vor Rootkits
- 5. Nach dem Angriff
- 6. Zusammenfassung & Quellen

Begriffsklärung

- Exploit: Ausnutzung einer Sicherheitslücke, um ungewollten Code auszuführen
  - Code läuft mit Rechten des angegriffenen Prozesses
  - evtl. → root-Rechte erlangbar
- Rootkit: Programm zum Verstecken des Angreifers (Prozesse, Verbindungen, Dateien,…)
  - Angreifer Logins vereinfachen (Backdoor)
  - Daten (Passwörter etc.) mitschneiden

Angriffsabsichten

- Erlangung von Benutzerrechten auf einer fremden Machine (remote exploit)
  - Angreifen von Nutzerprozessen (Officeanwendungen, Browser, …)
  - Angriff auf Netzwerk-Systemdienste (Web-Server, Mail-Server, RPC-Dienst, …)
- Ausweitung von Nutzerrechten zu root-Rechten (local root exploit)
  - Kernel oder Prozess mit root-Rechten angreifen

Angriffsvektoren

1. Buffer Overflow / Heap Overflow
2. Format String-Attacke
3. Verarbeitung unmaskierter Sonderzeichen
4. Race Conditions

- i.d.R. bei allen: Einschleusung von Shellcode

Buffer Overflow

- Überschreiben der Rücksprungadresse
- durch geschickte Präparierung Ausführung von eingeschleustem Code auf dem Stack
- alternativ: Präparierung von Argumenten auf dem Stack und dann Sprung zu einer libc-Funktion (*return-to-libc*)
  ➔ funktioniert auch ohne ausführbaren Stack

Format-String-Attacke(1/2)

- ausnutzbar: printf()-Formatanweisung %n schreibt in eine (Zeiger-)Variable
  - Beispiel – verwundbarer Code (GNU mailutils imap4d):

```
asprintf (&tempbuf, "%s %s%s %s\r\n",
        command->tag, sc2string (rc), command->name, format);
va_start (ap, format);
vasprintf (&buf, tempbuf, ap);
```

- beliebiger Speicher des Prozesses überschreibbar, kein ausführbarer Stack nötig
- mit anderen Formatanweisung lässt sich auch der Stack inspizieren

Code aus: http://www.idefense.com/application/poi/display?id=246&type=vulnerabilities&flashstatus=true

Format-String-Attacke(2/2)

```
#include <stdio.h>
int main(int argc, char **argv) {
    int i = 1;
    char buffer[64];
    char tmp[] = "\x01\x02\x03";
    snprintf(buffer, sizeof buffer, argv[1]);
    buffer[sizeof (buffer) - 1] = 0;
    printf("buffer : [%s] (%d)\n", buffer, strlen(buffer));
    printf ("i = %d \n", i);
}
```

- "./stack \x64\xf6\xff\xbf%.496x%n"

Code von: www.linuxfocus.org / siehe Quellen

unmaskierte Sonderzeichen

- Programm übergibt Benutzereingabe, Dateiname, etc. ungeprüft an eine Shell
  - Beispiel – verwundbarer Code (ClamAV):

```
sprintf(ditto, "/usr/bin/ditto --rsrc %s %s", src, dest);
if(system(ditto)) {     [....]
```

  - benennen Datei

```
;echo "test";
```

  - ausgeführt wird also:

```
system("/usr/bin/ditto -rsrc ;echo "test"; /tmp/;echo "test“  ");
```

Code aus: http://www.sentinelchicken.com/advisories/clamav/

Race Conditions

- zwischen einer Operation und vorhergehender Überprüfung vergeht Zeit
  - Szenario: Überprüfung erfolgreich
  - Angreifer hat Glück, kann Gegebenheit so ändern, dass die Überprüfung fehlschlagen würde
  - → Operation wird unter unzulässigen Bedingungen ausgeführt
- Beispiel: uselib-exploit für Linux 2.4/ 2.6

Erkennung von Einbrüchen

- Hostbasiertes IDS einsetzen
  - z.B. tripwire
- nach verdächtigem Verhalten Ausschau halten (Dateien, Protokolle)
  - /var/log/messages bzw. Event Log überprüfen
  - Benutzerliste beobachten
  - ~/.ssh/authorized_keys
  - Datenaufkommen beobachten

Fragen über Fragen

- Was tun, wenn ein kompromittiertes System im Netz erkannt wurde?
- Was tun Angreifer, wenn sie ein System kompromittiert haben?

Absichten der Angreifer

- viele denkbare Ziele für einen Angreifer
- typische Ziele:
  - Datendiebstahl, Sabotage
  - dauerhafte Zugriffsmöglichkeit auf System
  - Verschleierung des Angriffs
  - Informationen zum Angreifen anderer Rechner
    - Passwörter : /etc/passwd und /etc/shadow
    - Informationen über Netzstruktur und angreifbare Rechner

Wege zum Ziel (1/2)

- Möglichkeiten der Angriffsverschleierung:
  - automatische Tools und Skripte, die Logfiles säubern
- Möglichkeiten der Informationssammlung:
  - Netzwerksniffer bieten gewissen Aufschluss über Netzstruktur und aktive Verbindungen
  - Keylogger zeigen eingegebene Passwörter

Wege zum Ziel (2/2)

- Möglichkeiten „root“ zu bleiben:
  - User mit uid = 0 einrichten
  - eigenen Service starten / Shell an Port binden
  - Remote-Login-Services verändern (z.B. sshd)
  - Authentication-Key für ssh eintragen
  - Rootkit installieren

Rootkit-Typen

- sind in der Regel nicht ohne weiteres sichtbar und teilweise schwer zu entfernen
- Memory-Based oder Persistente Rootkits
- Dateibasierte oder Kernelbasierte Rootkits
- auf vielen Systemen einsetzbar: Linux, Windows, Solaris

Rootkits – Geschichte (1/2)

- erstes Rootkit – Dateibasiertes Rootkit
  - Austauschen von Systemprogrammen welche eine Möglichkeit zur Entdeckung des Angreifers boten
  - bspw. ps, netstat, passwd, sshd, top, w, users, finger, wtmp, etc.
- auch Modifizierung der Libraries möglich
- erkennbar durch Prüfsummencheck der Programmdateien bzw. Libraries

Rootkits – Geschichte (2/2)

- neuere sind oft Kernelbasierte Rootkits:
  - LKM-Rootkits – Loadable Kernel Module
    - als Kernelmodul in den Speicher geladen
    - Ersetzen von wichtigen Betriebssystemfunktionen
  - z.B. über /dev/kmem
    - Kernelspeicherabbild
    - Pointer auf die Syscalltable (Tabelle mit Verweisen auf Betriebssystemfunktionen) wird auf Kopie der Syscalltable verbogen

Rootkits

- Demo:
  - SuckIT 1.2
  - bietet Passwortsniffer
  - leichte Installation
  - bietet unbemerkten root-Login
  - Verstecken von Dateien und Prozessen möglich

Schutz vor Angreifern

- Programme wie chkrootkit (www.chkrootkit.org) oder RootkitRevealer (Sysinternals) bieten Erkennungsmöglichkeit
- “host-based“ Intrusion Detection Systeme wie tripwire oder bsign bieten Schutz vor Rootkits
- spezielle Kernelpatches wie RSBAC oder SELinux bieten Schutz
- nur wirklich benötigte und vertrauenswürdige Programme ausführen; nur mit den nötigen Recht ausführen
- Patches zeitnah einspielen → Sicherheitsnews verfolgen (CERT, Advisories)

Nach dem Angriff

- Sichern von Informationen über Angriff
  - **Wer?** Wann? **Wie?** Warum? **Was?**
  - aufwendige Dokumentation vor Gericht benötigt
- ALLE vom Angreifer eingebauten Hintertüren entfernen
  - Neuinstallation entfernt alle Hintertüren auf einem Rechner!!!
- bekannte Sicherheitslücken schließen

Zusammenfassung

- kein System ist 100%ig sicher
  - jegliche Programme bieten potentielle Angriffsziele
  - lokaler Zugang eines Nutzers auf einem Rechner ist eine potentielle Gefahr
- Sicherheitsaspekte müssen schon bei der Planung mit bedacht werden
- zum Erkennen eines Angriffs ist eine gewisse regelmäßige Kontrolle notwendig
- nur Neuinstallation entfernt alle Rootkits sicher

Quellen

- www.google.de (Suche nach Exploits)
- www.de.wikipedia.org (Exploits & Rootkits, Angriffstechniken)
- www.heise.de/security/
- www.packetstormsecurity.org
- www.phrack.org
- www.dolle.net
- exzellente Einführung zu Exploit-Techniken: http://www.linuxfocus.org/Deutsch/March2003/article282.shtml#282lfindex11