Hiding the honey in the pot

(Versteckspiele auf Linuxsystemen)

unseen@magic-opcodes.de

Inhalt

- Wer will Was verstecken?
- “old school” Ansätze im Userland
- neuer Ansatz Kernelspace
- Vor-/Nachteile der beiden “Varianten”
- Wo kann man mit LKMs ansetzen?
- Konkrete Beispiele
- Resume und Blick in die Zukunft

Wer hat etwas zu verbergen?

- “Hacker” nach erfolgreichem Angriff

  Spuren des Angriffs

  “systemfremde” Dateien (sniffer, snifferlogs, etc.)

  Anwesenheit im System

- Administratoren

  siehe oben. ;-)

  Bsp.: Honeypots kaschieren, Sensible Systeme, etc.

Was will man verbergen?

- Dateien/Verzeichnisse

  Binaries von Sniffern, Backdoors, etc.

  Logfiles zum Beispiel von Keyloggern

- Prozesse

  aktive shells, sniffer (ach!?), monitoring tools, etc.

- Netzwerkaktivität

  Listening sockets

  Aktive Verbindungen

“old school” Ansätze im Userland

- Geschickt gewählte Namen

  Beispiel: /dev/output, [kl0gd]

- Vorteile

  Keine besonderen Vorausetzungen. Schnell machbar.

- Nachteile

  Wird schnell entdeckt.

“old school” Ansätze im Userland

- Binaries ersetzen/patchen

  Beispiele: ps, ls, sshd, bash, login, etc.

- Vorteil

  Ebenfalls relativ leicht machbar

- Nachteile

  Koennte das System “instabil” machen

  Nicht schwer zu finden

Auswandern ins Kernelland

- Die Idee

  Alle Informationen kommen irgendwie vom Kernel

  Also können sie auch von dort geändert werden

- Der Vorteil

  Nahezu totale Kontrolle über den Userspace

- Die Nachteile

  Relativ schwer zu implementieren

  Hohes Risiko von Systeminstabilität

Ansätze im Kernel (2.4)

- Systemcalls und andere Funktionen “hijacken”

  Ersetzen von vorhandenen Funktionen durch Eigene

  ist an vielen Stellen und auf vielen Ebenen möglich.

- Eingebaute Funktionen nutzen

  netfilter um Netzwerkverkehr zu “sehen”

  devices um Befehle zu senden oder logs zu lesen

  procfs mein Favorit gegenüber devices

Lesetipps

- Linux Kernel Code (lxr)
- Phrack Magazine
- Addison-Wesley

  Linux Kernelprogrammierung

  Linux Netzwerkarchitektur
- siehe Linkliste

Die Zukunft

- Kernel 2.6

  Keine globalen syscalls mehr

  Folgt: Ansatz auf anderen Ebenen

  Beispiel: VFS oder INT-Handler

  Erste Module bereits verfügbar

- Trusted Computing

  Vielleicht eine Lösung?

Resume

- Gegenmassnahmen

  Systeme überwachen. (tripwire, tiger, etc.)

  LKM support nicht verwenden

  Eigene LKMs einsetzen um Kernel zu sichern

- Folgerungen

  Mehr Aktivität im Kernel

  Linux kann böse sein

  Paranioa lohnt sich

Besonderer Dank

Linus Torvalds und Alan Cox

Stealth (und Team Teso)

Pragmatic (und The Hackers Choice)

Phrack Staff

Chaos Computer Club

... und tschüss!

DANKE SCHÖN!