Einbruchserkennung

Einbruchserkennung in Computersystemen

Dr. Carsten Gnörlich

Rechnerbetriebsgruppe

Übersicht

1. Historie und Motivation
2. Kenne Deinen Gegner...
3. Klassen von Einbruchserkennungs-Systemen
4. Diskussion konkreter Systeme
5. Ausblick

Historie

und

Motivation

Historie

2. November 1988: "Internet Worm"

- VAX und Sun-3 (BSD 4.2/4.3)
- Pufferüberlauf in finger
- Hintertür in sendmail

→ Dienstverweigerung durch exzessive Reproduktion

→ aber keine explizite Schadfunktion

Auswirkung: Paradigmenwechsel

vorher: Unix als offenes System

- Teilen von Ressourcen
  - Offene mail/news – Server
  - RMS's paßwortloses Account

nachher: Abgeschlossene Systeme

- Administration interner Ressourcen
- Abschottung nach außen

Randbemerkungen

Internet brach für mehrere Stunden zusammen

➞ praktisch ein Nicht-Ereignis in den Nachrichten!

➞ wie wäre das heute ;-)

- keine Beteiligung von Microsoft-Produkten ;-)
- HTML, Java(script), ActiveX, ... gab es noch nicht

Kenne Deinen Gegner...

Das A und O der Einbruchserkennung

- Kenne Deinen Gegner
- aber erwarte das Unerwartete!

(nicht in vorgefertigten Kategorien denken!)

Systemschwächen

Fehlverhalten der Benutzer

- schwache Paßwörter, Zettel auf dem Bildschirm, falsche Berechtigungen, social engineering

Konfigurationsfehler

- falsche Sicherheitseinstellungen
- offener Zugriff auf Konfigurations-Details (Portscans, Webserver-Version, ...)

Systemschwächen

Implementierungsfehler

- Pufferüberläufe
- Abstürze durch falsche Parameter

Designfehler in Protokollen, Diensten, Anwendungen

- Authentisierungsschwächen (➞Spoofing, Hijacking,...)
- systematisches Ausprobieren von Paßworten

Motivation von Angreifern

"Skript-Kid"

- Motivation: Imponiergehabe
- Angriffsziele: diffus / zufällig
- Aktionen: typischerweise zerstörerisch

Technisch versierte Cracker

- Motivation: Anerkennung, Rache
- Angriffsziele: konkrete Institutionen / Personen
- Aktionen: verdeckt, zielgerichtet

Motivation von Angreifern

Organisierte Gruppen

- Motivation: wirtschaftliche Ziele
  (Industriespionage, Spam-Bot-Netze)
- Angriffsziele: konkrete Institutionen
- Aktionen: technische Angriffe
  "social Engineering"
  Einschleusen eigener Leute

Einordnung

und

Klassifikation

"Sicherheit" : Begriffsbestimmung

allgemein: Schutz vor Risiken und Gefahren

technisch: Störungs- und gefahrenfreie Funktion

EDV: Schutz von Informationen und Diensten

Einordnung

- physische Sicherheit (z.B. Zugangskontrolle)
- operationale Sicherheit (wie ist etwas zu tun)
- personelle Sicherheit (wer darf was tun)
- Systemsicherheit
- Netzwerksicherheit

} → Einbruchserkennung

Einbruchserkennung

Sicherheit = Schutz von Informationen und Diensten

- kein unautorisierter Zugriff / Veränderung
- Verfügbarkeit des IT-Dienstes sicherstellen

↪ "root werden", Identitätsdiebstahl Spezialfälle

Einbruchserkennung:

- Handlungen *erkennen*,
  die System im obigen Sinne kompromittieren
- es geht nicht um das *Verhindern* solcher Aktionen!

Deutsche/Englische Begrifflichkeiten

Einbruchserkennungs-Systeme

= Intrusion Detection Systems, kurz IDS

Komponenten eines IDS

1. Datensammlung

quantitativ: 150 SYN-Pakete auf Port n, 12:30 Uhr
qualitativ: Schutzverletzung V durch Programm X, Benutzer U, 14:15 Uhr

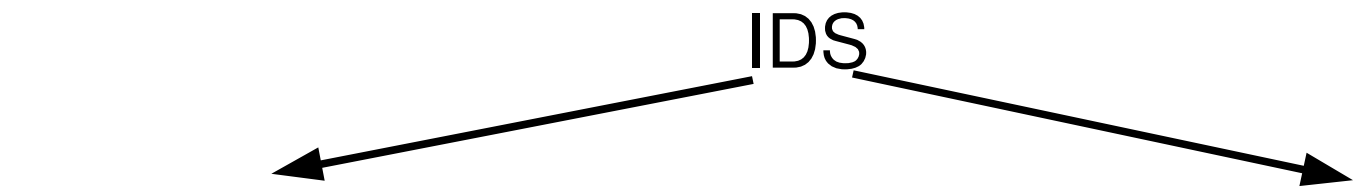
2. Datenanalyse auf mögliche Angriffe

→gleich mehr zu möglichen Techniken

3. Ergebnisdarstellung / Benachrichtigung

Entscheidungshilfe für den Sicherheitsbeauftragten

Datensammlung



Netzwerk-basiert (NIDS)

- Netzwerk-Verkehr

Host-basiert (HIDS)

- Kernel-Daten
- (Dateisyst.-)Rechte
- Log-Dateien
- Registry

Netzwerk-basierte IDS

NIDS: Verkehr auf Netzwerk auswerten

Vorteile:

+ 1 Sensor für das komplette Netz
+ Ausfall von Hosts (DoS) stört nicht

Nachteile:

– Bandbreitenprobleme
– benötigt Mirror-Port or Tap in geswitchten Netzen

Host-basierte IDS

HIDS: Daten auf jedem Host sammeln

Vorteile

+ umfassende Überwachung (Dateien, Prozesse, ...)
+ spezifische Aussagen über Angriff

Nachteile

– nicht hilfreich während DoS-Angriffen
+/- HIDS ist für Benutzer und Angreifer sichtbar

Datenanalyse: Angriffe erkennen

Signaturanalyse

- Arbeitsweise wie Virenscanner
- sucht nach typischen Angriffsmerkmalen (Codesequenzen, typ. Datenpakete, etc.)
- erkennt nur bekannte Angriffe

Datenanalyse: Angriffe erkennen

Anomalieerkennung / Heuristiken

➞ Abweichungen von "Normalzustand"

- statistische Auswertungen
  (Anzahl Seitenfehler, Schutzverletzungen,
  Verbindungsfehler auf bestimmten Ports, ...)
- dynamische Auswertungen
  via Lernkomponente / Expertensysteme
  (Nutzerprofile, Jobprofile nach Kalender, ...)

Schwäche aller Datenanalyse-Verfahren

- Fehlalarme / "falsche Positive"
- Individuelle Anpassung erforderlich
- kontinuierliche Anpassung erforderlich

➞ IDS brauchen Eingriff durch Menschen:

- zeitnah
- permanent

Ergebnisdarstellung / Benachrichtigung

Ergebnisdarstellung

- sehr hohes Datenaufkommen

➞ Filter + Visualisierung für relevante Ereignisse

Benachrichtigung

- Pager / Handy
- Nachricht über internes Netz (Problem: DoS)
- Alarm an entferntes System über sichere Verbind.
- Lokaler Alarm / eigene Konsole

Übersicht:

NIDS

Freeware - NIDS

Snort (www.snort.org)

- regelbasiert:
  - Signaturanalyse
  - Protokollanalyse
  - Anomalieerkennung

- Einsatzmöglichkeiten
  - Packet sniffing (debugging / cracking)
  - Einbruchserkennung
  - Einbruchsverhinderung (Pakete verwerfen)

Scanner als Bausteine für NIDS

Typische Portscanner (von beliebiger Maschine):

nmap (www.insecure.org/nmap)

Nessus (www.nessus.org)

Hostseitige Information über Ports:

netstat (8)

➞ Anwendung für NIDS:
Datenbank gewünschter offener Ports pflegen

Übersicht:

HIDS

Kleine Erinnerung

HIDS verwenden Informationen aus

- Kernel-Daten
- Dateisystem
- Log-Dateien
- Registry

Spezialfall: System Integrity Verifier (SIV)

- bilden Prüfsummen von Dateien

➡ zyklisches Prüfen statt permanenter Kontrolle

Freie System-Integritäts-Wächter

Tripwire (www.tripwire.com, SourceForge)

- "Vater" der SIW
- Nach Aufspaltung Closed/OpenSource inaktiv

Aide (SourceForge)

- Freier Nachfolger von Tripwire, gleiche Syntax

Samhain (www.la-samhna.de/samhain/)

- zentrale Überwachung mehrerer Hosts
- Überlegungen zum Schutz der Clients und Daten

Genereller Ansatz

Initialisierung (einmal)

Für alle zu überwachenden Dateien $(datei):

Nimm Prüfsumme($datei) in Datenbank auf

Überwachung (täglich)

Für alle zu überwachenden Dateien $(datei):

Vergleiche Prüfsumme($datei) mit Datenbank
➞ Melde Nicht-Übereinstimmungen

Auswählen von Dateien

- Kniffliger als es aussieht
- rekursives Abgrasen aller Dateien ab "/" geht nicht!

Beispiele:

- Arbeitsverzeichnisse: /home, /tmp, /var/tmp
- Prüfsummenlose Dateien: /dev/*, /dev/zero
- Pseudo-Verzeichnisse: /proc, /lost+found
- Log-Dateien, Backups: /var/log/*
- Spool-Bereiche: /var/mail, /var/run

Auswahl durch reguläre Ausdrücke

Tripwire, Aide: Verzeichnis- und Dateilisten

Beispiel:

```
/etc
!/etc/mtab$
/var
!/var/tmp$
!/var/log/syslog$
!/var/log/\.[0-9]$
```

Stolperfallen (1)

Reguläre Ausdrücke (RA) passen auf Teilstrings

Beispiel:

!/etc/mtab paßt auf

- */etc/mtab*
- */etc/mtab*-verzeichnis-mit-root-kit

➡ RA gegen Zeichenketten-Ende verankern ($):

!/etc/mtab$

Stolperfallen (2)

Der Punkt hat spezielle Bedeutung:

!/var/log/syslog.1$ paßt auf

/var/log/syslog.1
/var/log/syslogx1

➞Entschärfen von Sonderzeichen nicht vergessen:

!/var/log/syslog\.1$

Datei-Eigenschaften überwachen

➞ Attributgruppen erzeugen und daneben schreiben

NoChecksum = p+i+n+u+g+s+b+c+m
Ignore-Time = p+i+n+u+g+s+b +md5+sha1

/dev
/dev/null NoChecksum
/dev/zero NoChecksum
/var
/var/log$ Ignore-Time

Verfügbare Datei-Eigenschaften

Inhalt (Prüfsummen)

- md5, crc32, sha1, ...
- möglichst mehrere

Datei-Attribute

p	Permissions		
i	inode	n	number of links
u	user	g	group
s	size	b	block count
a	access time	c	change time (Status)
m	modified time (Inhalt)	S	growing size

Schwachstellen

Keine lückenlose Überwachung aller Dateien

- Rootkit in /tmp, /proc, /dev
- Kernel-(Modul)-Rootkit (gefälschte Prüfsummen, ...)
- Zeitfenster zwischen IDS-Läufen

Samhain:

- IDS läuft als Dämon; nicht transparent austauschbar
- verschlüsselte Datenbank, Schlüssel im Binary
- erkennt kompromittierte Linux- und FreeBSD-Kernel

Schwachstellen

Kompromittierung des IDS selbst

- Binary des IDS
- Datenbank mit Prüfsummen etc.

allgemein:

- Binary, Datenbank read-only bereitstellen
- GPG-Signaturen verwenden

Samhain:

- Schlüsselpaar Binary/Datenbank
- dedizierten Datenbank-Server, verschlüsselte Kommunkation mit den Clients

IDS in der RBG

Aktuell: Aide auf dem Linux-Server sputnik

- Intrusion detection
- Überwachung von Software-(Neu-)Installationen

Geplant:

- Samhain ausprobieren
- Rudimentäres NIDS via nmap

Vielen Dank fürs Zuhören :-)