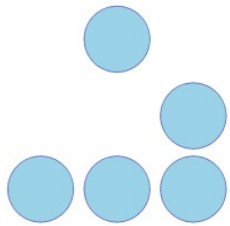


Rootkits

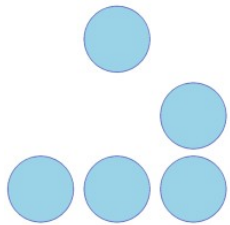
SLAC, 08.12.06

Johannes Plötner
<jploetner@ploetner-it.de>



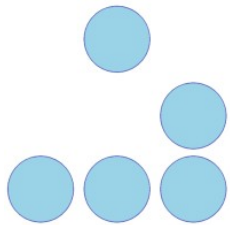
zur Person

- Johannes Plötner
 - jploetner@ploetner-it.de
- IT-Fachautor
- IT-Berater
 - Unix/Linux
 - IT-Sicherheit
 - IT-Projektmanagement
- Unternehmer
 - Plötner IT (<http://www.ploetner-it.de>)
 - IT-Consulting, Solutions, Coaching



Ablauf

- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- Rootkit-Technologien
 - Application-Rootkits (historisch)
 - Kernel-, Userland- und Speicher-Rootkits (aktuell)
 - Virtual-Machine-Rootkits (Zukunft?)
 - Tunneling
- Szenario „Compromise Detection“
- Diskussion



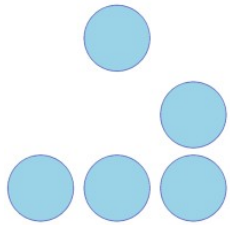
Grundlagen und Definitionen

Rootkits

- **Rootkits** (*klassische Definition*)
 - „Malware“
 - nach erstem Einbruch installiert
 - künftiger Zugang für Angreifer

- **Rootkit-Technologien** (*neue Definition*)
 - verbergen
 - Prozesse,
 - Dateien,
 - Traffic,

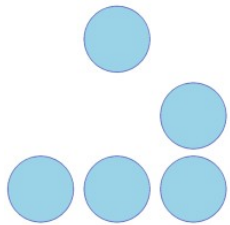
vor den Augen des Admins



Grundlagen und Definitionen

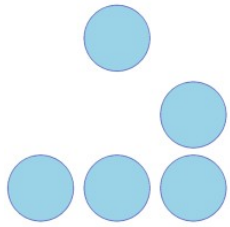
Botnetze

- Netzwerk gehackter Rechner
 - von Würmern/Trojanern kontrolliert
 - durch **Rootkit-Technologien** versteckt
 - langer „Nutzen“



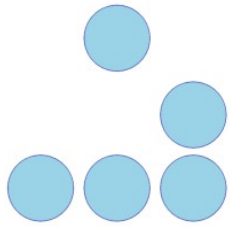
Ablauf

- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- Rootkit-Technologien
 - Application-Rootkits (historisch)
 - Kernel-, Userland- und Speicher-Rootkits (aktuell)
 - Virtual-Machine-Rootkits (Zukunft?)
 - Tunneling
- Szenario „Compromise Detection“
- Diskussion



Szenario „Angriff“ - Hacker **Motivation**

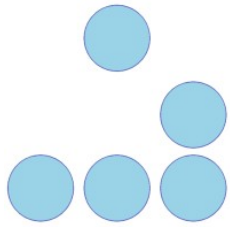
- Motivation eines Hackers
 - Früher / klassisch:
 - Ego
 - Geltungsdrang
 - Interesse an Technik
 - Heute:
 - Geld / Buisness



Szenario „Angriff“ - Hacker Geschäftsmodelle

- früher:
 - DDOS-Attacken durch Botnetze
 - Erpressungsversuche
- heute:
 - Spam-Versand
 - Adware-Verteilung
 - Ausspähen von Benutzerdaten (Phishing)

http://www.computerbase.de/news/internet/hacker_sicherheit/2006/april/studie_rootkits

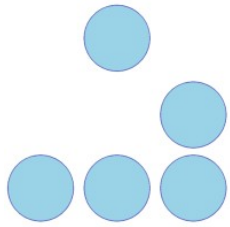


Szenario „Angriff“ - Hacker Wirtschaftlichkeit

- Preise
 - Botnet (11000 Rechner): ab 80 Euro/h
 - Einzelner Rechner: 0.10 \$/Woche

Quelle: McAfee

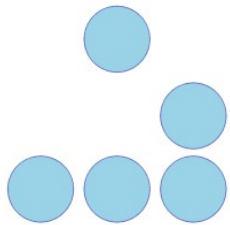
- Wirtschaftlichkeit von Spam
 - Wieviele Mails können über einem Rechner in einer Stunde verschickt werden?
 - Kosten pro Mail
 - notwendige Returnquote für Wirtschaftlichkeit



Szenario „Angriff“ - Hacker

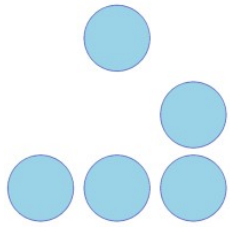
„ungezielter Angriff“

- Entdeckung einer Sicherheitslücke
 - „Buffer-Overflow in xyz“
- Schreiben eines Exploits
 - Einbau in Wurm/Virus
 - **Rootkit-Technologien**
 - Freisetzung
 - Fortpflanzung
 - generiert „zufällige“ IPs und prüft auf Verwundbarkeit
- Auf dem Host:
 - Nachladen von Schadcode
 - „nach Hause telefonieren“



Ablauf

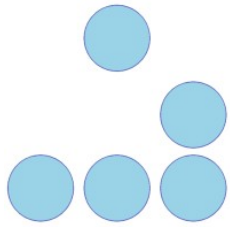
- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- Rootkit-Technologien
 - Application-Rootkits (historisch)
 - Kernel-, Userland- und Speicher-Rootkits (aktuell)
 - Virtual-Machine-Rootkits (Zukunft?)
 - Tunneling
- Szenario „Compromise Detection“
- Diskussion



Szenario „Angriff“ - Wirtschaftsspionage

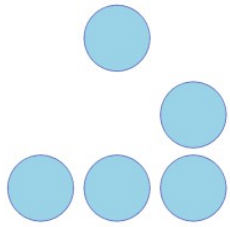
Echelon

- globales Abhörssystem **Echelon**
 - für private und wirtschaftliche Kommunikation
„Abhörsystem ECHELON (2001/2098 (INI)“, EU, 2001
 - Wer?
 - USA
 - Vereinigtes Königreich
 - Kanada
 - Australien
 - Neuseeland
 - Was?
 - Globales Abhören von Satelliten-Kommunikation
 - Teilweises Abhören drahtgebundener Netze
 - Wirtschaftsspionage



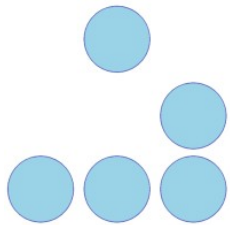
Szenario „Angriff“ - Wirtschaftsspionage **Echelon (2)**

- Warum?
 - Entscheidungsprozesse im Unternehmen
 - Unternehmens- & Marktstrategien
 - Zusammenschlüsse, Absprachen
 - Preisgestaltung & Konditionen (Ausschreibungen)
 - Hochtechnologie
- Beispiel:
 - Enercon '94
 - Windkraftanlagen plötzlich bereits in den USA patentiert
 - möglich durch per Echelon abgefangene Sicherheitscodes



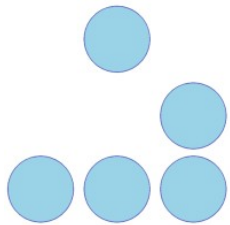
Szenario „Angriff“ - Wirtschaftsspionage „gezielter Angriff“

- Footprinting
 - Sammeln von Infos (Google, DNS, trace, ...)
- Portscan / Vulnerability Scan
 - OS-Version, Dienste-Versionen, Schwachstellen
- Exploitation
 - Ausnutzen einer Sicherheitslücke
- Verstecken und Festsetzen
 - Einsatz von **Rootkits**, Backdoors, ...
 - Spuren verwischen
- System benutzen



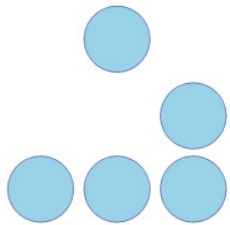
Ablauf

- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- **Rootkit-Technologien**
 - **Application-Rootkits** (historisch)
 - Kernel-, Userland- und Speicher-Rootkits (aktuell)
 - Virtual-Machine-Rootkits (Zukunft?)
 - Tunneling
- Szenario „Compromise Detection“
- Diskussion



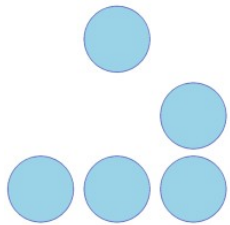
Rootkit-Technologien - **Application-Rootkits**

- Idee:
 - Ersetzen von Admin-Werkzeugen durch modifizierte Varianten
 - Unix: eigenes „ps“, „ls“, ...
- Schutz:
 - hostbasierte Intrusion Detection
 - bspw.: aide/tripwire
 - speziell gesicherte Tools
 - Nachladen von CD
 - ...



Ablauf

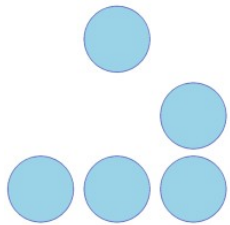
- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- **Rootkit-Technologien**
 - Application-Rootkits (historisch)
 - **Kernel-, Userland- und Speicher-Rootkits** (aktuell)
 - Virtual-Machine-Rootkits (Zukunft?)
 - Tunneling
- Szenario „Compromise Detection“
- Diskussion



Rootkit-Technologien – Kernel Rootkits

Syscalls

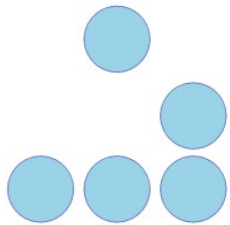
- Usermode vs. Kernelmode
 - **Ring 3:** Eingeschränkter CPU-Befehlssatz
 - Benutzerprogramme
 - **Ring 0:** Zugriff auf Peripherie usw. möglich
 - Kernel
- Systemaufrufe („Syscalls“)
 - Zugriff auf Funktionalität des Kernels
 - Beispiele:
 - read_dir
 - read
 - write



Rootkit-Technologien – Kernel Rootkits

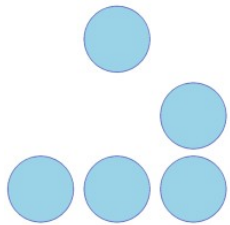
Syscalls (2)

- Syscall-Table
 - Speicherbereich im Kernelspace
 - Zuordnung Syscallnummer – Codebereich
- Ablauf Syscall:
 - Angabe der Syscallnummer in Register
 - Auslösung eines Interrupts
 - Auslesen der Nummer durch Kernel
 - Ausführung des zugehörigen Codes



Rootkit-Technologien – **Kernel-Rootkits**

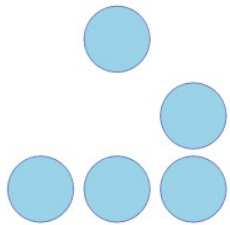
- Idee:
 - Syscall-Table des Kernels manipulieren
 - verweist auf eigenen Code
 - manipulieren von Daten, nicht Tools
- Umsetzung:
 - Code in Kernelspace einbinden
 - nachladbare Treiber (Windows)
 - LKMs (Linux/Unix)



Rootkit-Technologien – Kernel Rootkits

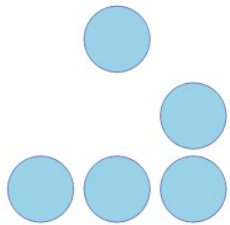
Erkennung / Schutz

- Prinzipiell:
 - Überwachung / Überprüfung der Syscall-Table auf Veränderungen
- Aber:
 - Veränderung des Syscalls-Codes durch den Angreifer (anstatt der Tabelle)
 - vfs-layer
 - ...
- **best practice**
 - Nachladen von Kernelmodulen verbieten (Linux)
 - Windows x64: *MS Patchguard*



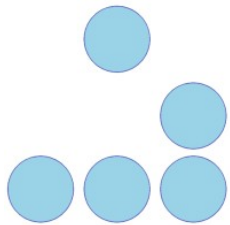
Ablauf

- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- **Rootkit-Technologien**
 - Application-Rootkits (historisch)
 - Kernel-, **Userland-** und **Speicher-Rootkits** (aktuell)
 - Virtual-Machine-Rootkits (Zukunft?)
 - Tunneling
- Szenario „Compromise Detection“
- Diskussion



Rootkit-Technologien - **Userland-Rootkits**

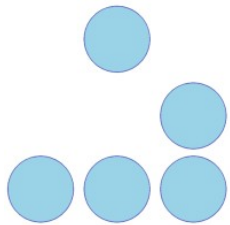
- Idee:
 - Bibliotheksaufrufe umleiten
 - bekannte Programme verhalten sich „anders“
 - keine verdächtigen Programmen gestartet
 - keine Prüfsummen verändert
- Umsetzung:
 - Linux:
 - LD_PRELOAD, Bibliotheken ersetzen, ELF
 - Windows:
 - div. API-Funktionen



Rootkit-Technologien - Userland-Rootkits

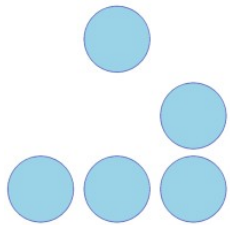
Erkennung / Schutz

- Rechte-Ausweitung durch OS verhindert
 - Beispiel:
 - Unter Linux kein LD_PRELOAD bei *setuid*-Programmen
 - daher: kein „Sicherheitsloch“.
- **Aber:** Restgefahr bleibt
 - bekannte Programme verhalten sich anders
 - weitere Maßnahmen notwendig:
 - Windows: Virenschutz
 - Linux: Programme *statisch* kompilieren (?)



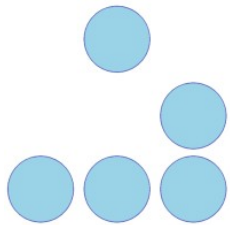
Ablauf

- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- **Rootkit-Technologien**
 - Application-Rootkits (historisch)
 - Kernel-, Userland- und **Speicher-Rootkits** (aktuell)
 - Virtual-Machine-Rootkits (Zukunft?)
 - Tunneling
- Szenario „Compromise Detection“
- Diskussion



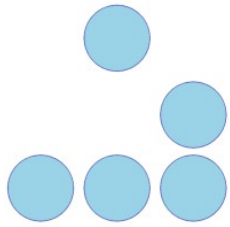
Rootkit-Technologien – Speicher-Rootkits

- Idee:
 - nur im Speicher vorhanden, keine Dateien im System abgelegt
 - in der Regel Kernel-Rootkits
 - Überstehen Reboot nicht
- Umsetzung:
 - Kernel-Modul / Treiber laden,
 - im Kernelspeicher kopieren,
 - entladen,
 - Spuren löschen
 - /dev/kmem-Patching (Linux)



Ablauf

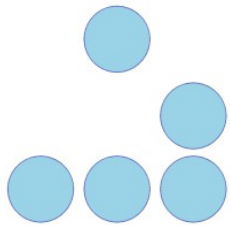
- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- **Rootkit-Technologien**
 - Application-Rootkits (historisch)
 - Kernel-, Userland- und Speicher-Rootkits (aktuell)
 - **Virtual-Machine-Rootkits** (Zukunft?)
 - Tunneling
- Szenario „Compromise Detection“
- Diskussion



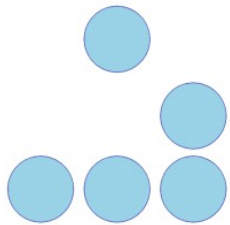
Rootkit-Technologien - **Virtual-Machine-Rootkits**

- Idee:
 - Host-OS wird in **virtuelle Maschine** verfrachtet
 - Rootkit „zwischen“ OS und Hardware
 - nicht(?) zu entdecken
- Umsetzung:
 - bisher nur als proof-of-concept
 - Ausnutzung spezieller CPU-Befehle
 - Kommunikation z.B. durch LKMs

<http://www.eecs.umich.edu/virtual/>

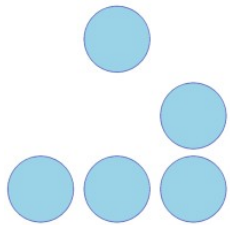


- *Typ 0 Malware*
 - keine Veränderungen am System
 - nicht interessant
- *Typ I Malware*
 - Dinge werden verändert, die nicht verändert werden sollen
 - relativ leicht aufzuspüren
- *Typ II Malware*
 - nur Dinge verändert, die verändert werden sollen
 - nicht aufzuspüren (?)
 - proof-of-concept existiert



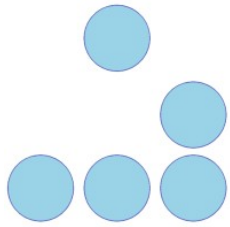
Ablauf

- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- **Rootkit-Technologien**
 - Application-Rootkits (historisch)
 - Kernel-, Userland- und Speicher-Rootkits (aktuell)
 - Virtual-Machine-Rootkits (Zukunft?)
 - **Tunneling**
- Szenario „Compromise Detection“
- Diskussion



Tunneling **Grundgedanken**

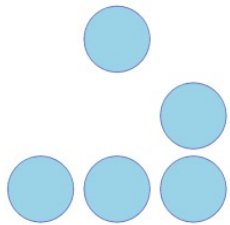
- Idee:
 - Verstecken von Traffic
 - Überwinden von Firewalls & Intrusion Detection
- Umsetzung:
 - kreatives Benutzen von TCP/IP
 - Payload bei ICMP
 - Protokoll-Tunneling („looks-like-HTTP“)
 - Connect-Back
 - ...



Tunneling

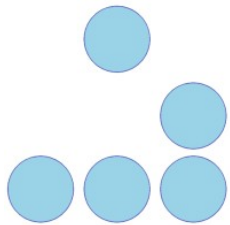
VSST – Tunneling Demotool

- **vstt** – very strange tunneling tool
 - Verbindungen über ICMP / POP3 / ... tunneln
 - <http://www.doomed-reality.org/?sub=vstt>
- Schutz:
 - intelligentes Firewall-Setup :-)
 - Intrusion Detection (bspw. snort)
 - automatische Logfile-Auswertung!
 - bspw. logcheck



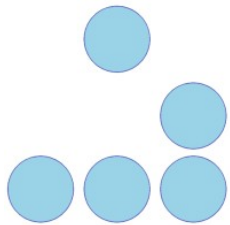
Ablauf

- „Rootkits“
 - Definition und Grundlagen
- Szenario „Angriff“
 - Hacker, Wirtschaftsspionage
- Rootkit-Technologien
 - Application-Rootkits (historisch)
 - Kernel-, Userland- und Speicher-Rootkits (aktuell)
 - Virtual-Machine-Rootkits (Zukunft?)
 - Tunneling
- Szenario „Compromise Detection“
- Diskussion



Compromise Detection

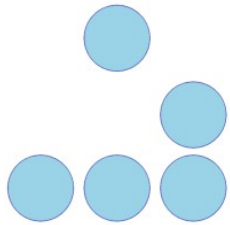
- System inspizieren
 - Startskripte
 - Trojaner-Kandidaten (ps, ls, login, ...)
 - Verdächtige Rechte & SetUID-Binarys
 - Shellhistory (.bash_history)
 - /dev & Co.
 - lsmod
 - ifconfig (promiscuous Mode?)
 - Logfiles
 - Portscan außen/innen
 - (Firewallregeln)
 - ...



Compromise Detection

- Weitere Möglichkeiten:
 - Cross-View-Ansatz (*high- vs. low-level Sichten*)
 - Seiteneffekte
 - verdächtige Dateien, Prozesse, ...
 - Signaturbasierte Erkennung
 - nach bekannten Rootkitsignaturen suchen
 - Integrität substantieller OS-Elemente
 - Syscall-Table etc.

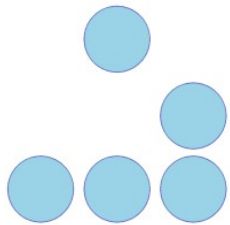
Unterstützung durch diverse Tools!



Compromise Detection

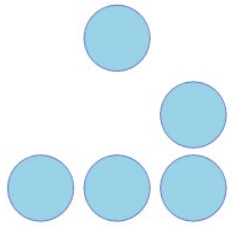
Anti-Rootkit-Tools

- für Windows:
 - *BlackLight*
 - Cross-View
 - erkennt auch aktive Rootkits
 - <http://www.f-secure.com/blacklight/>
 - *Sophos Anti-Rootkit*
 - <http://www.sophos.de/products/free-tools/sophos-anti-rootkit.html>
 - *System Virginty Verifier*
 - überprüft Identität von .DLLs im Speicher und auf HD
 - <http://invisiblethings.org/tools.html>



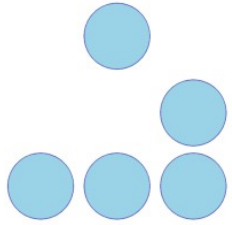
Anti-Rootkit-Tools (2)

- für Linux:
 - *chkrootkit*
 - findet viele bekannte Rootkits
 - <http://www.chkrootkit.org/>
 - *rkhunter*
 - sucht nach suspekten Rechten, versteckten Dateien usw.
 - <http://rkhunter.sourceforge.net>
 - *rkscan*
 - LKM-Rootkits
 - <http://www.hsc.fr/ressources/outils/rkscan/>



- Rootkits auch in Zukunft Bedrohung
 - öffentlich zugänglich & einfach anwendbar
 - immer ausgefallenerere Techniken
- Leider auch „legale“ Einsatzgebiete
 - Sony Rookit
 - ...

false positives?



<http://www.ploetner-it.de>

Vielen Dank für Ihre Aufmerksamkeit.