

Sicherheit in komplexen DV-Systemen

Integrität und Rootkits

Christoph Hermann
Matrikelnummer 300944

Inhaltsverzeichnis:

Kurze Zusammenfassung	3
Einleitung	4
Rootkits	4
- Kurze Erwähnung: was sind Rootkits	4
- Welche Arten von Rootkits gibt es	4
- Wie werden sie benutzt und wozu	4
- Gefahr der Rootkits	4
- Normale Rootkits	5
o Funktionsweise / Aus was besteht ein Rootkit	5
o Vorstellung der Vorgehensweise eines Angreifers	5
o Vorstellung von t0rnkit, einem relativ altbekannten Rootkit	6
- Loadable Kernel Module Rootkits	6
o Funktionsweise/Besonderheit	6
o Vorstellung: Adore	7
- Full Stealth Rootkits (TCP-Layer Ebene)	7
o Funktionsweise/Besonderheit	7
o Vorstellung Kernel Intrusion System (KIS) von Optyx	8
Gegenmaßnahmen	8
- Wie finde ich Rootkits, bin ich überhaupt kompromittiert ?	8
- Tools zur Vorbeugung und Entdeckung	9
o checkrootkit	9
o checkps	9
o KSTAT – Entdeckung von LKMs	9
o Samhain	10
Intrusion Detection	10
- NIDS (Network Intrusion Detection Systems)	10
- NNIDS (Network Node Intrusion Detection Systems)	11
- HIDS (Host-Based Intrusion Detection Systems)	11
- Anomaly-Based Intrusion Detection Systems	11
- dIDS (Distributed Intrusion Detection Systems)	11
- Ein Beispiel eines IDS: Tripwire	12
HoneyNet Forensic Challenge	13
- Einführung	13
- Die Herausforderung	13
- Ergebnisse der Challenge	14
o Vorgehensweise des Angreifers	14
o Spuren, die zur Entdeckung führten	15
Literaturverzeichnis	16

Kurze Zusammenfassung

Im Rahmen der Veranstaltung „Sicherheit in komplexen DV-Systemen“ befasse ich mich in meinem Seminarteil mit dem Thema „Integrität und Rootkits“. Diese Rootkits sind in einer Vielzahl im Internet zu finden und bieten selbst einem unerfahrenen User die Möglichkeit seine Einbruchsspuren auf einem fremden Rechner professionell zu verschleiern. Sie geben ihm auch die Möglichkeit mittels sogenannten Backdoors später jederzeit wieder Zugriff zu dem System zu erlangen.

Es gibt eine Vielzahl verschiedener Rootkits, deren Funktionsweise, Verschleierungstechniken und Gefahren im weiteren Verlauf dieses Textes erläutert werden.

Wir werden dabei auf drei verschiedene Arten von Rootkits eingehen: „Normale Rootkits“, „Loadable Kernel Module Rootkits“ und sogenannte „Full Stealth Rootkits“.

Diese werden dann auch anhand von Beispielen vorgestellt, wobei speziell auf die Rootkits „t0rn“ und „Adore“ Bezug genommen werden wird.

Anhand von „normalen Rootkits“ wird auch die Vorgehensweise eines Angreifers (meist von sogenannten Script-Kiddies) vorgestellt.

Natürlich ist man der Gefahr, die von Rootkits ausgeht bieten nicht komplett hilflos ausgeliefert; es gibt verschiedene Maßnahmen zur Bekämpfung und Prävention eines Angriffes. Der Leser wird lernen die Angriffsart von und mit Rootkits zu verstehen, um dann im Falle eines Falles selbst Gegenmaßnahmen ergreifen zu können.

Das Problem ein Rootkit zu entdecken wenn das eigene System kompromittiert wurde, wird ausführlich diskutiert werden, auch Tools wie z.B. „chkrootkit“ oder „KSTAT“, die die Entdeckung von Rootkits vereinfachen, werden vorgestellt.

Zur Vorbeugung eines Angriffes durch Rootkits und für schnelle Erkennung eines solchen werden wir uns dann mit Intrusion Detection Systemen befassen, dabei auf die Vor- und Nachteile von Host-based und Network Based Intrusion Detection eingehen.

Es wird erklärt werden, was NIDS, NNIDS, HIDS und dIDS bedeutet und welches dieser Systeme wann und warum an welcher Stelle eingesetzt werden sollte.

Dann wird noch ein Beispiel eines Intrusion Detection System vorgestellt: Tripwire, das sowohl ein HIDS als auch NIDS sein kann.

Zum Schluss wird noch die HoneyNet Forensic Challenge vorgestellt. Bei dieser geht es darum, Informationen über die Techniken eines Angreifers und seine Vorgehensweise zu bekommen, und aus diesen mittels eines Wettbewerbes zu lernen. Den Teilnehmern des Wettberwerbes wird als Anreiz eine Prämie geboten, und sie sollen die Spuren des Angreifers aufdecken und genauestens dokumentieren.

Einleitung

Im Rahmen der Veranstaltung „Sicherheit in komplexen DV-Systemen“ befasse ich mich in meinem Seminar mit dem Thema „Integrität und Rootkits“. Im Folgenden werde ich auf die sogenannten Rootkits eingehen, von denen eine Vielzahl im Internet zu finden sind. Mit ihnen ist es selbst einem unbedarften Computernutzer möglich, sich Zugriff auf ein fremdes, mangelhaft geschütztes System zu verschaffen und seine Einbruchsspuren professionell zu verschleiern. Es ist es dem Angreifer möglich, die Spuren seiner Tat zu verwischen, völlig unentdeckt zu bleiben und das kompromittierte System für weitere Angriffe oder sonstige eigene Zwecke zu nutzen. Ich werde die verschiedenen Arten von Rootkits, deren Funktion und Gefahr sowie Gegen- und Präventivmassnahmen vorstellen.

Dazu erläutere ich einige Programme und Verfahren von Host-based und Network-based Intrusion Detection, die es einem Systemadministrator ermöglichen sich gegen solche Angriffe zu wappnen und ihnen vorzubeugen. - oder im Falle einer Kompromittierung eines Systems den Angriff zu entdecken, nachzuvollziehen und die Spuren bzw. Folgen zu bereinigen.

Rootkits:

Rootkits sind eine Sammlung von Tools die ein Eindringling auf ein System nach vorherigem (gewaltsamen) Eindringen mitbringt um seine Aktivitäten zu verschleiern und um den Zugriff, den er sich (mittels eines Exploits o.ä.) erarbeitet hat, zu einem späteren Zeitpunkt möglicherweise noch einmal nutzen zu können. Im Normalfall enthält ein Rootkit eine Sammlung von Sniffern, Scripten die Logs säubern und trojanisierte Binaries die jene des angegriffenen Systems ersetzen, um einerseits unentdeckt zu bleiben und um weiteren Zugriff auf den kompromittierten Host zu einem späteren Zeitpunkt zu erlangen. Beispiele von Binaries die auf dem Fremdsystem ersetzt werden, sind z.B. ps, ls, netstat, ifconfig, killall oder daemons, die auf dem System laufen.

Obwohl der Angreifer sich zuerst einmal Zugriff auf das entsprechende System verschaffen muss (mittels eines vorher angesprochenen Exploits z.B.), so sind diese Rootkits doch aufgrund der Einfachheit Ihrer Handhabung und ihrem gewaltigen Zerstörungspotential ein großes Problem für Systemadministratoren.

Wenn der Angreifer es erst einmal geschafft hat sich Zugriff auf ein fremdes System zu verschaffen, so ist es ihm dann durch ein Rootkit ein leichtes, später wieder diesen Zugriff auf das System zu erlangen.

Das primäre Ziel von Rootkits ist es aber eigentlich, die Anwesenheit und die Aktionen des Angreifers auf dem kompromittierten System zu verschleiern und dessen Spuren zu verwischen. (Dabei sind die Rootkits mehr oder weniger erfolgreich: Es gibt verschiedene Arten von Rootkits, die sich selbst (also Ihre eigenen Dateien) besser oder schlechter tarnen, um nicht von einem findigen Systemadministrator entlarvt zu werden.)

Prinzipiell gibt es drei verschiedenen Arten von Rootkits, die dann natürlich in verschiedenen Varianten auftreten und immer weiter entwickelt werden, um sich und den Angreifer besser zu tarnen. Die drei Arten von Rootkits die im folgenden hier vorgestellt werden, sind sogenannte „normale Rootkits“, die relativ schnell und einfach auszumachen sind, dann „Loadable Kernel Module Rootkits“ (LKMs) und noch eine Besonderheit LKM auf TCP/IP-Layer Ebene.

„Normale Rootkits“

Normale Rootkits bestehen – wie schon oben genannt – aus einer Sammlung von Scripts, Tools, und Binaries die zwei Zwecke erfüllen. Zum einen verschleiern sie die Anwesenheit des Angreifers, seiner Taten und des Rootkits, und zum anderen ermöglichen sie späteren Zugriff auf das bereits kompromittierte System.

Zum Verschleiern der Anwesenheit sind meist Scripte oder Tools vorhanden, die die Log-Files von den Einträgen des Angreifers säubern.

Solche Tools sind z.B. z2 und wted, die Log-File-Einträge von den wtmp, utmp und lastlog Files entfernen, um die Präsenz und den Einbruch zu verbergen.

Dann gibt es noch Binaries, die die des angegriffenen Systems ersetzen, damit das Rootkit nicht auffällt.

Binaries die meistens ersetzt werden sind z.B. folgende:

ls, du, find :	damit die Dateien des Rootkits nicht entdeckt werden
ps, top, pidof:	die Prozesse die der Angreifer startet werden nicht angezeigt
netstat:	zeigt keinerlei vom Angreifer verursachten Netzwerktraffic an und zeigt laufende Daemonen wie sshds, Eggdrops (IRC Bot) o.ä. nicht an
killall:	tötet die Prozesse des Angreifers nicht
ifconfig:	zeigt nicht an, dass das PROMISC-Flag gesetzt ist, falls der Angreifer einen Sniffer verwendet.
crontab:	zeigt die crontabs des Angreifers nicht an
tcpd:	wird Verbindungen nicht mitloggen
syslogd:	wird entsprechende Log-Einträge nicht anzeigen

Oft ist dem Rootkit dann noch ein Utility namens „fix“ beigefügt, mit dem es möglich ist, die mitgebrachten Binaries den schon vorhandenen so anzupassen, dass die Ersetzung durch etwaige abweichende Größe oder Datumsangaben nicht auffällt, bzw. zunichte gemacht wird.

Auch ein Sniffer ist bei den meisten Rootkits dabei, um z.B. von dem kompromittierten Rechner aus neue Ziele für weitere Angriffe zu suchen, oder um Netzwerktraffic mittels des PROMISC-modus der Netzwerkkarte unbefugt mitzulesen.

Der spätere Zugriff auf das System wird dann durch das Rootkit ermöglicht, wobei es zwei Möglichkeiten gibt. Entweder es werden lokale Programme, oder Netzwerk Daemonen mit Trojanern infiziert. Lokale Programme die oftmals infiziert werden sind chfn, chsh, login, oder passwd. Gibt man dann das entsprechende rootkit-Passwort ein, so erlangt man Zugriff zu einer root-shell. Im anderen Fall werden Backdoors in Daemons installiert (meist sshd oder telnetd o.ä.) oder ein eigener Remote-Daemon installiert, der dann meist auf einem anderen Port (z.B. 37337 - eleet) läuft als es Standard wäre.

Bei der Installation vieler Rootkits sind meist nur ein paar simple Einstellungen zu tätigen, den Rest erledigen „Scripte“, die der Programmierer des Rootkits gleich mitgeliefert hat, was unter anderem auch zu der Beliebtheit der Rootkits führt.

Generelle Vorgehensweise eines Angreifers

Im Folgenden wird nun die generelle Vorgehensweise eines Angreifers vorgestellt, um einen Einblick in das Verhalten der meisten Angreifer (meist sogenannte Script-Kiddies) zu ermöglichen:

Als erstes wird dem Angriff in der Regel ein Scan-Vorgang vorausgehen, der es dem Angreifer ermöglicht ein verwundbares System zu ermitteln. Nachdem der Angreifer ein verwundbares System ausgemacht hat, wird er eine Schwachstelle im System ausnutzen um sich Zugriff auf das System zu verschaffen und einen root-Zugang zu erlangen. Das ist sehr oft mit vielen Log-File Einträgen und Spuren verbunden, die der Angreifer verbergen muss. Oftmals ist es jedoch so, dass der Angreifer sehr schlampig arbeitet, und anstatt nur seine Spuren zu entfernen, die kompletten Log-Files löscht, was einem Systemadministrator zu denken geben sollte, wenn er dies feststellt.

Hat der Angreifer dieses vollbracht, wird er sich (wenn er sich sicher fühlt; meist schaut er als erstes nach wer auf dem System ist (who)) das Rootkit herunterladen und installieren, meist mit wget oder per Anonymous ftp. Oft wird das Rootkit in ein verstecktes Verzeichnis „<Leerzeichen>“ oder in /dev/ptys /dev/ptyr o.ä. installiert.

Nachdem die Spuren des Angriffs beseitigt wurden, werden nun ein oder mehrere Trojaner, hinterlassen um weiteren Zugriff auf das System zu ermöglichen. Im Beispiel von Rootkit IV ist z.B. inetd trojanisiert und lauscht auf einem besonderen Port (z.B. 5002) auf Verbindungen. Gibt man bei einer Verbindung zu diesem Port das richtige Passwort an, so bekommt man eine root-shell. Oft wird für diesen Zweck das Programm bindshell verwendet, das eine root-shell auf einem bestimmten Port bereitstellt (37337 – eleet in „cracker“-Sprache), wobei kein Passwort verlangt wird. Hat der Angreifer einen Trojaner hinterlassen, installiert er manchmal noch Programme wie einen Eggdrop (IRC-Bot), einen IRC-Bouncer (mit welchem er das kompromittierte System benutzen kann um „anonymen“ IRC-Zugriff zu erlangen) oder einen Sniffer, um z.B. Passwörter mitzuzugreifen, die er dann für weitere Tätigkeiten verwenden kann.

Diese installierten Programme werden mit Hilfe der Konfigurationsdateien (oftmals zu finden in /dev/ptyr /dev/ptys etc..) für die mitgelieferten, ersetzten Binaries versteckt, so dass z.B. „ls -la“ die installierten Dateien nicht anzeigt.

T0rnkit – ein altbekanntes Rootkit

Ein mehr oder minder harmloses (da zu bekanntes und nicht sehr kompliziertes) Rootkit ist T0rnkit. Dieses Rootkit kann jeder unerfahrene Angreifer verwenden, da keine komplizierten Einstellungen zu tätigen sind. Das Rootkit installiert sich von selbst und versteckt die eigenen Dateien, Prozesse und Aktionen. Während der Installation versucht t0rnkit zuerst den Syslog Daemon auszuschalten. Dann ersetzt es einige Programme (du, ls etc...), speziell den ssh-daemon mit einer trojanisierten Version. Danach startet t0rnkit einen Sniffer als Hintergrundprozess, aktiviert telnetd, rsh und finger daemons in der "/etc/inetd.conf" und startet den Inetd sowie den Syslog-Daemon neu.

T0rnkit versucht zwar sich relativ gut zu verstecken, ist aber anhand einfacher Merkmale zu erkennen. Zum einen haben alle Dateien des t0rnkits die Größe von 31.336 Bytes, das Rootkit kann meist in /usr/src/.puta gefunden werden und Port 47017 ist nach außen offen um den Angreifer root-Zugriff zu ermöglichen.

Loadable Kernel Module Rootkits

Kommen wir nun zu einer fortgeschritteneren Art von Rootkits, den sogenannten Loadable Kernel Module Rootkits.

Die Besonderheit an LKMs ist, dass Sie auf eine ganz andere Art und Weise funktionieren als die bereits vorgestellten Rootkits. LKMs laden sich (wie der Name schon sagt) als Modul in den Kernel und modifizieren

dort System-Aufrufe wie z.B. `open()` um Ihre Anwesenheit zu verschleiern. Der Befehl „ps“ z.B. benutzt den `open()`-Systemaufruf und liest Informationen von den Dateien in `/proc`, wo auch Informationen über laufende Prozesse gehalten werden.

Jede Anwendung wird vom Kernel kontrolliert und überwacht, bzw. ausgeführt. Die Anwendung ruft einen System-call auf, der Kernel bearbeitet diese Anfrage und liefert das Ergebnis an die Anwendung zurück. Dementsprechend kann ein Rootkit also, indem es die Rückgabewerte des Kernel modifiziert sich viel besser verstecken, da es eine viel bessere Kontrolle über alle Informationen hat, die im System angefordert und bearbeitet werden. Der Effekt ist also derselbe wie bei den „normalen Rootkits“; Prozesse, Dateien und Netzwerkverbindungen des Angreifers werden aufgrund der Manipulation der Informationen durch das Kernel-Modul nicht angezeigt.

Das Rootkit modifiziert z.B. den `sys_getdents` Kernel Systemaufruf, welcher Verzeichnisse ausliest um Ihren Inhalt anzuzeigen. Dies geschieht natürlich um gewisse Dateien (wie z.B. die zusätzlich installierten Programme oder Konfigurationsdateien) zu verstecken. Aufgrund der Tatsache, dass jedes Programm, das ein Dateilisting anfordert, diesen Systemaufruf verwendet, wird keines dieser Programme in der Lage sein jemals Dateien des Rootkits anzuzeigen.

Um eben diese System-Calls zu modifizieren, bzw. um diese mit eigenem Code zu ersetzen modifiziert das LKM die Tabelle in der die Adressen für die Systemaufrufe stehen und ersetzt die vorhandenen Werte so, dass sie nun an die Stelle des Codes des LKMs zeigen. Das wiederum gibt dem Systemadministrator die Möglichkeit, die modifizierten Systemaufrufe zu entdecken. Beim Kompilieren eines Kernel wird im selben Verzeichnis wie der Kernel eine Karte dieser Systemaufrufe mit dem Namen „System.map“ (manchmal ist noch eine Kernel-Version angefügt) angelegt. Eine Möglichkeit die falschen Systemaufrufe zu entlarven ist also, die Adressen der Systemaufrufe, mit denen die in dieser Datei stehen, zu vergleichen, um so möglicherweise ein LKM zu entdecken.

Vorstellung eines LKMs: Adore

Adore ist ein LKM, das folgende Systemaufrufe modifiziert:

```
sys_getdents:          versteckt Dateilistings
sys_write, sys_close, sys_kill, sys_fork, sys_clone, sys_symlink
sys_execve:           leitet Programmaufrufe um (dadurch muss man die Binaries nicht mehr wie
                      bei „normalen Rootkits“ ersetzen)
```

Adore kann Dateien, Prozesse, Services verstecken und Programme (unter anderem mit Root-Rechten) ausführen (z.B. `/bin/sh`). Adore ist mittels eines Programms namens „ava“ kontrollierbar, tarnt sich selbst (ist also nicht mittels „`lsmod`“ oder „`cat /proc/modules`“ erkennbar) und kann nicht entfernt werden.

Full Stealth Rootkits auf TCP-Layer Ebene

Es gibt noch weitere Arten von LKMs, die im folgenden als „Full Stealth Rootkits“ bezeichnet werden. Wie wir bereits festgestellt haben sind LKMs viel mächtiger und flexibler als „normale Rootkits“. Eine Variante, die jedoch auf dem TCP-Layer agiert ist noch ausgereifter als die bisher vorgestellten. Das Ziel dieses LKMs ist es, so unauffällig wie möglich zu sein, seine Präsenz so gut wie möglich zu verbergen und dem Angreifer so viele

Möglichkeiten zu geben, Zugriff zu bekommen wie nur möglich. Dabei wird wie bisher die Technik der LKMs verwendet - mit der Besonderheit, dass der TCP/IP Stack modifiziert wird.

Im Kernel ist es so, dass jedes Protokoll seine Handler-Routine im `*inet_protocol_base` Pointer und `*inet_protos[MAX_INET_PROTOS]` Hash registriert. Wenn das System die Initialisierung durchläuft, so werden alle Protokolle in `inet_protocol_base` registriert und werden dem `inet_protos` Hash hinzugefügt. Wenn der Kernel dann ein Paket empfängt, wird dieser Hash auf den entsprechenden Handler überprüft. Dies ist genau die Stelle, an der das Full Stealth Rootkit angreift: Es ersetzt die originalen Protokoll-Handler mit eigenen, so dass das LKM das Paket empfängt und analysieren kann. Hat das Paket die entsprechende Form, so wird der Befehl den es enthält ausgeführt; wenn nicht, so wird die normale Routine ausgeführt. Das ganze funktioniert für TCP und UDP (für den Fall dass ein Teil durch einen Packet-Filter oder einen Firewall blockiert würde). Dadurch, dass auch jeweils nur ein Paket an das kompromittierte System durch den Angreifer gesendet werden muss, kann die IP gespoof werden, was es erschwert den Angreifer ausfindig zu machen.

Das Kernel Intrusion System (KIS) von Optyx

Eine Variante dieses Rootkits funktioniert sogar, wenn der Kernel statisch kompiliert wird, d.h. es keine Möglichkeit gibt Kernel Module während des Betriebs zu laden.

Das KIS von Optyx ist momentan wohl das ausgereifteste Rootkit, das die Systemsicherheit gefährdet. Anstatt die Kernel-Systemaufrufe durch eigene zu ersetzen, schreibt das KIS direkt in den RAM des Systems und ändert somit die Ausgabe der Programme. KIS benutzt dafür sogenanntes „Kernel-memory-patching“. Daher ist keine Unterstützung für LKMs notwendig, nur Schreibzugriff zu Teilen des RAMs (bei Linux mit `/dev/kmem`).

Gegenmaßnahmen

Wie finde ich Rootkits

Um Rootkits zu entdecken, muss ein ziemlicher Aufwand betrieben werden, da ja erst einmal überhaupt festgestellt werden muss, dass das System kompromittiert und ein Rootkit installiert wurde.

Erste Anzeichen für eine Kompromittierung des Systems sind ein anomales Verhalten des Servers bzw. diverser Befehle. Es kann z.B. sein, dass ein Parameter bei einem Programm wie „ls“ oder „ps“ nicht mehr funktioniert, was daran liegt, dass der Angreifer diese System-Binaries mit eigenen ersetzt hat, die aber vielleicht eine andere Version haben und deshalb die verwendeten Parameter nicht mehr existieren.

Ein weiterer Anhaltspunkt ist ein gewaltiger Anstieg des Netzwerktraffics (durch die Verteilung von Warez o.ä. auf einem kompromittierten Webserver z.B.) oder ein Unterschied in der Ausgabe von Netstat mit einem Portscan des Systems.

Andere Möglichkeiten zur Entdeckung sind verschiedene Tools (wie z.B. chkrootkit), von denen einige hier nun vorgestellt werden, oder der Einsatz von Intrusion Detection Systemen.

Tools zur Vorbeugung und Entdeckung

chkrootkit

„chkrootkit“ ist eine Sammlung von 7 kleinen Tools, die es einem ermöglichen die Präsenz von Rootkits aufzudecken. „chkrootkit“ wird verwendet um bekannte Signaturen von Rootkits zu erkennen und funktioniert prinzipiell wie ein Virenschanner. Außer dass es nach bekannten Rootkit-Dateien zu suchen, durchsucht es auch wichtige Systemdateien und ausführbare Dateien nach „böartigem Inhalt“. „ifpromisc“, ein weiteres Tool, das im „chkrootkit“ enthalten ist, stellt fest ob das Netzwerkinterface im PROMISC-Modus ist, da man der Ausgabe von Netstat nicht trauen kann. „chklastlog“, „chkwtmp“, und „check_wtmpx“ entdecken, dass aus den Log-Files Einträge gelöscht wurden. „chkproc“ entdeckt LKMs und getarnte Prozesse. Dann ist noch „strings“ enthalten, eine Version des Unix Strings-Tools.

Mit „chkrootkit“ ist es ein Leichtes die meisten Rootkits zu entdecken um sie dann hinterher zu entfernen oder – je nach Anforderungen – vielleicht sogar den Server komplett neu zu installieren. Wird ein Rootkit entdeckt, muss auf jeden Fall der Server erst mal vom Netz getrennt werden und das Rootkit und alle damit verbundenen Dateien gelöscht, bzw. durch die Originale ersetzt werden. Je nach Sicherheitsrichtlinien muss sogar der Server komplett neuinstalliert werden, da man nicht weiß, ob man alle Veränderungen die durch das Rootkit gemacht wurden gefunden hat.

checkps

„checkps“ ist ein kleines Tool, das erkennt ob die Ausgabe von „ps“ korrekt ist, oder ob „ps“ durch ein Rootkit ersetzt wurde um die Ausgabe desselben zu manipulieren.

checkps generiert seine eigenen Prozessliste, indem es die Informationen aus /proc ausliest oder alle möglichen Prozessnummern ausprobiert mit Hilfe des Kill() Systemaufrufs. Es vergleicht dann diese Informationen mit denen die „ps“ liefert um Unterschiede festzustellen.

„checkps“ bietet dann die Möglichkeit Ungereimtheiten zu loggen, entweder per syslogd und in ein Log-File oder per E-Mail, da der Syslog-Daemon möglicherweise ebenfalls ersetzt wurde.

KSTAT

KSTAT (Kernel Security Therapy Anti-Trolls) ist ein Tool um LKMs aufzuspüren. KSTAT überprüft dazu den Speicher (/dev/kmem) auf Informationen über den Host (auch über LKMs) und vergleicht auch die Adressen der Systemaufrufe mit denen in der „System.map“. In der neuesten Version wird sich allerdings nicht mehr auf die Informationen in der „System.map“ verlassen, da ein Rootkit diese manipulieren könnte. Man muss jetzt jedes Mal wenn man den Kernel neu kompiliert auch KSTAT neukompilieren, da die ganzen Adressen jetzt statisch miteinkompiliert werden. Des Weiteren lädt sich KSTAT als LKM in den Kernel um genauere Informationen zu bekommen. Mit KSTAT ist es nun ohne weiteres möglich LKM Rootkits wie z.B. Adore zu finden, obwohl sie bei „lsmod“ o.ä. nicht angezeigt werden. KSTAT bietet einem bei dem Parameter „-m“ die Möglichkeit den Speicher nach LKMs zu durchsuchen und liefert einem dann mögliche Speicheradressen für „getarnte“ Kernel-Module zurück. Überprüft man diese Adressen dann genauer, so findet KSTAT an diesen Stellen z.B. Adore, falls dieses Rootkit installiert war und kann das Modul wieder der Modulliste hinzufügen, so dass es mit „lsmod“ sichtbar wird und auch ohne weiteres entfernt werden kann.

Samhain Integritätschecker

Der Samhain Integritätschecker bietet ähnliche Funktionalität wie KSTAT. Mit ihm ist es einfach die Adressen der augenblicklichen Kernel Systemaufrufe mit denen in der „System.map“ zu vergleichen um wiederum LKMs aufzuspüren. Der Unterschied zu KSTAT ist, dass Samhain auch noch Checksummen von Dateien verwenden kann um Veränderungen aufzuspüren und Dateien mit abnormalen SUIDs auf der Festplatte aufspüren kann. Des Weiteren kann Samhain seine Log-Files signieren und in einem Stealth Modus laufen.

Er bietet dann noch die Möglichkeit im Client/Server-Modus zentrale Überwachung und die Speicherung Logs auf einem Log-Server (z.B. MySQL oder postgresQL).

IDS – Intrusion Detection Systeme

Zur Vorbeugung von Angriffen auf den oder die eigenen Rechner, und zur sofortigen Feststellung einer Manipulation von Dateien kann man sogenannte Intrusion Detection Systeme verwenden, deren Funktionalität derer des vorgestellten Integritätscheckers von Samhain schon sehr nahe kommt.

Dabei muss zwischen Host Based Intrusion Detection Systemen und Network Based Intrusion Detection Systemen unterschieden werden. Im Folgenden werden nun die Funktionalität solcher Systeme erklärt und ein paar dieser IDS vorgestellt.

IDS dienen im allgemeinen dazu den Systemadministrator frühzeitig zu warnen, und ihn im Falle eines Falles darauf aufmerksam zu machen, dass eines seiner Systeme kompromittiert wurde. IDS sind dafür entwickelt die Angriffsfälle zu erkennen oder abzufangen, die durch die Firewall (falls vorhanden) durchgekommen sind.

IDS sind entweder darauf ausgelegt einen gerade laufenden Angriff abzufangen oder um festzustellen, dass das System kompromittiert wurde. Im letzteren Fall ist es zwar zu spät um den kompletten Schaden abzuwenden, aber man weiß wenigstens um die Kompromittierung und kann entsprechende Gegenmaßnahmen einleiten.

Wird ein IDS verwendet, so sollte dies aus den verschiedenen Varianten von IDS zusammengesetzt sein. Ein Systemadministrator der viel auf Sicherheit hält, sollte Network Intrusion Detection systems (NIDS), Network Node Intrusion Detection systems (NNIDS), Host Based Intrusion Detection systems (HIDS), Anomaly Based Intrusion Detection Systems, und die analytische Stärke von Distributed Intrusion Detection System (dIDS) kombinieren.

NIDS (Network Intrusion Detection Systems)

NIDS können als „Breitbanderkennungs-System“ genutzt werden um Angriffe gegen ein Teilnetzwerk innerhalb eines größeren Netzwerkes festzustellen. Diese Systeme bieten den größtmöglichen Überblick an Überwachung und können als generelles IDS, das nicht-kritische Systeme überwacht, eingesetzt werden. Ein guter Punkt, an dem NIDS eingesetzt werden können, ist zwischen dem Router eines Subnetzwerkes und den Rechnern eben dieses Subnetzes. Somit kann ein Angriff gegen jeden Rechner in diesem Subnetz aus einem anderen Netzabschnitt sofort der Administration gemeldet werden. NIDS können auch auf Switches und HUBS eingesetzt werden, oder an jedem anderen Punkt, an dem Subnetze zusammengeführt werden.

NNIDS (Network Node Intrusion Detection Systems)

NNIDS sind am besten auf kritischen Systemen (wie z.B. Datenbank oder Backup-Servern) einzusetzen. Diese Art von IDS stellt nur Angriffe auf das System fest, auf dem es installiert ist und kümmert sich nicht um andere Hosts im Netzwerk. Das vermindert zwar in dieser Sicht die Fähigkeiten von NNIDS, führt aber dazu, dass spezielle IDS-Fähigkeiten (wie genaue Änderungsfeststellungen) auf dem speziellen Host eingesetzt werden können.

HIDS (Host-Based Intrusion Detection Systems)

HIDS Systeme sind weniger damit beauftragt Angriffe aus dem Netzwerk bzw. über Netzwerkprotokolle festzustellen, vielmehr sind sie dazu gedacht kontinuierlich Systemdateien zu überwachen. Systemdateien die sich verändert haben – es aber nicht sollten, es sei den ein Update o.ä. wurde vom Administrator durchgeführt – sind ein sofortiger Hinweis darauf, dass das System kompromittiert wurde. Das erste was ein Angreifer meist macht nachdem er Root - Zugriff erlangt hat (um z.B. später die Möglichkeit zu haben wieder Zugriff zu dem System zu erlangen), ist Dateien zu verändern. Hier ist auch der Punkt an dem das IDS ansetzt. Tools wie z.B. Tripwire überwachen verschiedene Aspekte des Dateisystems und vergleichen diese mit gespeicherten Dateien in einer Datenbank. Sie können so konfiguriert werden, den Administrator zu alarmieren falls Veränderungen stattgefunden haben. Klar ist natürlich, dass solche Programme nur auf „sauberen“ Systemen installiert werden sollten, da sonst die Dateien in der Abgleichdatenbank keinen wirklichen Wert haben.

HIDS werden normalerweise auf kritischen Workstations installiert, oder auf Servern die ein extra Maß an Sicherheit brauchen.

Anomaly-Based Intrusion Detection Systems

Anomaly-Based Intrusion Detection Systems sind eine relativ neue Entwicklung. Die Idee die dahinter steckt, ist dass man eine Art von „normaler Aktivität“ festlegt, z.B. welcher Netzwerktraffic in welchen Größenordnungen unter normalen Bedingungen von wo nach wo geht. Jedwede Abweichung von diesen normalen Werten kann dann als Anomalie gewertet und dem Systemadministrator berichtet werden. Anomaly-Based Intrusion Detection Systems werden immer wichtiger im Zusammenhang mit internen Angriffen. Das liegt unter anderem daran, dass diese Systeme das Problem lösen, wer Zugriff zu bestimmten Ressourcen hat und wer nicht. Diese Systeme filtern nur den Traffic des Users der von den normalen Werten abweicht, was eine viel kürzere Analysezeit zur Folge hat. Anomaly-Based IDS werden meist genau wie NIDS an Stellen eingesetzt, an denen verschiedenen Teilnetzwerke zusammenlaufen.

dIDS (Distributed Intrusion Detection Systems)

Für viele Systemadministratoren ist es zu aufwendig die generierten log Files und Reports komplett durchzusehen, da die Arbeitszeit dafür einfach nicht ausreicht.

Diesem Problem kann Abhilfe geschaffen werden durch ein dIDS System. DIDS sammeln und aggregieren Logs von verschiedenen IDS und Firewall-Systemen. Das erlaubt den Systemadministratoren Angriffsinformationen und Log Files in einer aggregierten Form an einem zentralen Ort anzusehen. Das verkleinert die

Bearbeitungszeit und ermöglicht es den Systemadministratoren einen generellen Überblick über die Angriffe und Angriffsstrategien im gesamten Netzwerk zu bekommen.

Das dIDS hilft dem Systemadministrator Angriffe vorzeitig zu verhindern und zu erkennen, indem es die Zeit verkürzt, alle Log-Files auf verschiedenen Rechnern durchzusehen. Des Weiteren bietet es dem Administrator die Möglichkeit aus Angriffen zu lernen und somit zukünftige Angriffe bereits früher zu erkennen und abzuwehren, oder Schwachstellen im gesamten System zu erkennen und sie abzudichten.

IDS Sicherheit wird sehr oft überschätzt. Der Nutzen von IDS ist nicht von der Hand zu weisen, jedoch ist ein IDS kein Allheilmittel gegen Angriffe. Auf jeden Fall sollten die Logs von IDS oder Firewall Systemen auf einem zentralen Log-Server gespeichert werden, damit die Logs nicht von den Angreifern modifiziert werden können. Aus diesem Grund sollten auch auf dem Log-Server sowenige Dienste wie möglich laufen um eine Kompromittierung des Log-Systems so schwer wie möglich zu machen.

Tripwire

Tripwire ist ein IDS das Attribute von wichtigen Systemdateien überwacht, die sich nicht ändern sollten. Unter anderem kann Tripwire den Zeitpunkt des letzten Zugriffs, die Anzahl der Blöcke, die Dateibesitzer GID (Group ID), den Zeitpunkt der letzten Veränderung, die Dateiattribute (rwx), Dateigröße, Dateityp, UserID, einen CRC-32, MD5, oder SHA Hash Wert überprüfen.

Die Überwachung der Dateien durch Tripwire kann in einer Konfigurationsdatei vollständig an die Belange des Systemadministrators angepasst werden.

Was Tripwire für Linux als Opensource Lösung nicht bietet ist NIDS, dies gibt es nur in der kommerziellen Version, die dann auch für andere Plattformen verfügbar ist. Die kommerzielle Version von Tripwire bietet einiges mehr als die Opensource Version die „lediglich“ als HIDS ausgelegt ist.

Unter Unix überwacht die kommerzielle Version von Tripwire 14 Systemattribute, unter anderem Dateiänderungen (neue Dateien, Änderungen, Gelöschte Dateien), UserID, GroupID, Größe, Datum der letzten Änderung, Größenveränderungen oder das Datum des letzten Zugriffs. Tripwire bietet auch die Möglichkeit die Dateien mittels einer CRC-32, MD5, SHA, HAVAL Hash-Signatur auf Veränderungen zu überprüfen. Jedwede Veränderung wird sofort dem Tripwire Manager gemeldet und in das System Log-File eingetragen.

Die mächtige Konfigurationssprache ermöglicht es sogar bei Regelbrechung einen bestimmten Befehl auszuführen, was z.B. dazu genutzt werden kann den Administrator zu benachrichtigen oder direkt die veränderten Dateien aus einem Backup wiederherzustellen.

Auch die Benachrichtigungsmöglichkeiten durch Tripwire nach einer Veränderung sind sehr vielfältig, man kann per E-Mail oder durch den Tripwire-Manager informiert werden, und Tripwire kann XML Dokumente mit detaillierten Reports über die Veränderungen erstellen.

Der Tripwire-Manager kommuniziert mit den einzelnen Hosts, die von Tripwire überwacht werden, per SSL, so dass sämtliche Befehle und Daten die ausgetauscht werden, über eine sichere, verschlüsselte Verbindung laufen. Tripwire besteht prinzipiell aus sechs Teilen. Zum einen ist da das „Information Infrastructure Monitoring“, d.h. „Tripwire für Server“ macht einen digitalen Schnappschuss von der aktuellen Systemkonfiguration im „sauberen“ Systemzustand, und vergleicht dann im laufenden Betrieb den Systemzustand mit diesem Schnappschuss und identifiziert jede Abweichung.

Dann gibt es noch die „Intrusion Detection“: Tripwire verhindert keinen unerlaubten Zugriff, sondern macht ihn bemerkbar und warnt den Systemadministrator - egal ob der Angriff nun außerhalb oder innerhalb des eigenen Netzwerkes seinen Ursprung hat.

Das „System-Konfigurations-Management“ bietet die Möglichkeit, aufgrund eines Vergleichs mit einer Basisdatenbank zu überwachen, ob sich Systemdateien verändert haben.

Mittels „System Lockdown“ bietet Tripwire die Möglichkeit zu zertifizieren, dass keinerlei unbefugte Software installiert wurde.

Das „Damage Assessment and Recovery“ von Tripwire bietet die Möglichkeit nach einem Angriff oder interner Misskonfiguration den Schaden einzuschätzen, zu klassifizieren und anhand dessen einen Bericht abzuliefern welche Dateien repariert oder ersetzt werden müssen.

Die „Data Forensics“-Fähigkeiten von Tripwire helfen dabei den Angreifer genauer ausfindig zu machen und seine Spuren klar aufzuzeichnen.

Die HoneyNet Forensic Challenge

Einführung

Täglich haben Systemadministratoren überall auf der Welt mit kompromittierten Systemen zu tun. Auf diesen Systemen laufen - meist ohne Wissen des Administrators - eine Reihe von unbekanntem Programmen, die dem Eindringling ungewünschte Dienste zur Verfügung stellen.

In den meisten Fällen ist die Vorgehensweise nach diesen unbefugten Zugriffen, die betroffenen und kompromittierten Rechner so schnell wie möglich auf irgendeine Art und Weise in den normalen Betrieb zu überführen.

Das führt dazu, dass die Gegenmaßnahmen ungenügend oder ineffektiv sind, da die genaue Angriffsursache und deren Folgen unbekannt ist. Ein wiederholtes Auftreten dieses Missbrauchs ist deshalb sehr wahrscheinlich.

Die HoneyNet Forensic Challenge ist der Versuch, Ordnung in dieses Chaos (mit dem Systemadministratoren bei Angriffen zu kämpfen haben) zu bringen, indem man die Angreifer ihre Veränderungen machen lässt: Man schaut Ihnen dabei nur sehr genau auf die Finger und protokolliert alle ihre Aktivitäten. Aus den gewonnenen Informationen kann man lernen, sich besser gegen solche Angriffe zu schützen um diese in der Zukunft besser abwehren zu können.

Das primäre Ziel des HoneyNet Projects ist es, diese Daten zu sammeln um sie bei der Abwehr solcher Angriffe verwenden zu können, durch diese Erfahrung aus den Angriffen zu lernen und die Prävention gegen solche zu verbessern.

Die Herausforderung

Die Forensic Challenge ist der Versuch es allen Systemadministratoren möglich zu machen, die selben Daten eines Angriffs einzusehen (ein Image des selben kompromittierten Systems) um herauszufinden, wer am meisten Informationen über den abgelaufenen Angriff aus diesen wenigen Daten erkennen kann. Diese Informationen sollen dann allen Leuten zur Verfügung gestellt werden, um sich gegenseitig darüber auszutauschen.

Das ganze ist eine nichtwissenschaftliche Studie über die Tools und Techniken, die zur Beseitigung der Folgen eines Angriffs, deren Dokumentation und rechtliche Verfolgung eingesetzt werden.

Die Herausforderung soll den Teilnehmern Spaß machen, ein globales Problem lösen und jedem die Möglichkeit geben daraus zu lernen. Als Anreiz wird den besten 20 Teilnehmern ein Exemplar des Buches „Hacking Exposed“ von Foundstone geboten.

Mit den wenigen gegebenen Informationen (ein paar Logfiles, wenigen Informationen über das System und Diskimages von den Partitionen nach der Kompromittierung des Systems) soll nun von den Teilnehmern in Erfahrung gebracht werden, wer was, wann, wie und wo auf dem kompromittierten System verursacht hat.

Es steht den Teilnehmern frei alle Tools und Techniken ihrer Wahl einzusetzen, solange die Ergebnisse von der Jury im Nachhinein verstanden und überprüft werden können. Jeder Schritt und alles was zur Ermittlung der Tatsachen führt muss dokumentiert und einsichtig niedergelegt werden. Sämtliche verwendeten Tools, Programme die zur Ermittlung beigetragen haben, müssen frei verfügbar sein, selbsterstellte Skripte und Source-Codes müssen der Jury zur Verfügung gestellt werden.

Ergebnisse der Challenge

Lediglich 13 Einsendungen aus der ganzen Welt gingen in die Endauswertung ein. Auch eine „offizielle“ Analyse der Daten wurde von „Dave Dittrich“ (einem Mitarbeiter aus dem HoneyNet Project) durchgeführt. Im folgenden werden die Ergebnisse der Analyse des erstplatzierten Teilnehmers dargestellt.

Die Vorgehensweise des Angreifers

Der Rechner (apollo.honeyp.edu) wurde am 7. November mittels eines altbekannten Exploits (ein Format String Problem in rpc.statd) angegriffen.

Der Angreifer fügte als nächstes der „inetd.conf“ einen neuen Service als Backdoor hinzu (4545 stream tcp nowait root /bin/sh sh -i), der es ihm ermöglichte später wieder auf das System zurückzukehren. Der Angreifer hat zusätzlich zwei Accounts angelegt („adm1“ – ein normaler User Account UID 5000 GID 5000 und „own“ – ein passwortloser Root-account).

Schon zu diesem frühen Zeitpunkt hat der Angreifer die Log Files in /var/log/wtmp bereinigt und den „klogd“ und „syslogd“ ausgeschaltet. Außerdem hat er alle Log-scripte aus den Startskripten (/etc/rc.d/init.d) entfernt.

Der Angreifer loggte sich dann mittels „telnet“ oder „ssh“ als Benutzer adm1 auf den kompromittierten Host ein und su'te sich dann zu own, was ihm Root - Zugriff verschaffte.

Dann benutzte der Angreifer ftp um sich sein Rootkit zu beschaffen und installierte es in /usr/man/.Ci.

Als nächstes führt der Angreifer eines seiner Scripte aus um .bash_history Files zu löschen und einen Symlink derselben nach /dev/null zu legen. Das Skript versucht dann /bin/ps, /usr/bin/top, /usr/sbin/syslogd, /bin/lis, /bin/netstat, /sbin/ifconfig, /usr/sbin/tcpd, und /usr/sbin/in.identd zu ersetzen, was teilweise misslingt, da der syslogd auf dem angegriffenen System in /sbin installiert wurde, was zur Folge hatte, dass dieser Daemon nicht ersetzt wurde.

Das Skript erzeugt dann Backups der ersetzten Systemdateien im Verzeichnis /usr/man/.Ci/backup.

Bei der Ersetzung achtet das Skript darauf, dass die Timestamps der ersetzten Binaries mit dem der originalen übereinstimmen. Die ersetzte Version von ifconfig zeigt nicht an, ob das Netzwerkinterface in den PROMISC-

Modus versetzt wurde. „ps“ und „top“ zeigen gewisse Programme, und „ls“ zeigt die Dateien des Angreifers nicht mehr an. „netstat“ und „tcpd“ zeigen die Netzwerkverbindungen des Angreifers nicht mehr.

Als nächstes startete der Angreifer einen Sniffer im Hintergrund und loggte Zugriffe per ftp, telnet, rlogin, und imap nach /usr/man/.Ci/tcp.log.

Ein weiteres Script entfernte dann die Spuren des Angreifers aus verschiedenen Log-Files.

Als nächsten Schritt sicherte der Angreifer das System gegen etwaige andere Angreifer indem er diverse rpms und eine Kopie eines IRC Clients (bitchx) in /bin/bx installierte.

Als letztes installierte der Angreifer eine modifizierte Version des sshd, welcher ein Universalpasswort enthält und sämtliche eingegebenen Passwörter nach /usr/tmp/nap logt und benutzte ein Script um seine erstellten User-Accounts zu löschen. Ein Nebeneffekt davon war, dass der User „shutdown“ auch gelöscht wurde, da der Substring „own“ darin enthalten ist. Die Zeile die der Angreifer der /etc/inetd.conf hinzugefügt hat, wurde dann per Hand entfernt.

Spuren die zur Entdeckung führen

- Der Angreifer löschte den Account „shutdown“ indem der User aus der etc/passwd und /etc/shadow entfernt wurde.
- Das Systemlogging wurde ausgeschaltet
- Es befanden sich untypische Dateien in /dev die mittels „find /dev -type f -print0 | less“ ausgemacht werden konnten.
- Systemdateien wurden mit libc.so.5 verlinkt auf einem System das normalerweise die GNU libc verwendet.
- „ifconfig“ zeigt nicht an, dass das Netzwerkinterface im PROMISC Modus ist, was aber durch „strings /sbin/ifconfig | fgrep PROMISC“ herausgefunden werden kann.
- Software Upgrades oder Downgrades wurden nicht vom Systemadministrator selbst gemacht

Literaturverzeichnis:

Rootkits

<http://www.usenix.org/publications/login/1999-9/features/rootkits.html>
<http://rr.sans.org/linux/toolkit.php?sansrr=b5ee7643a65bafd55a0166a446e55d6d>
<http://www.networknewz.com/2002/0417.html>
<http://linux.oreillynet.com/pub/a/linux/2001/12/14/rootkit.html>
<http://linux.oreillynet.com/pub/a/linux/2002/02/07/rootkits.html>
<http://www.linuxvoodoo.com/security/enemy31.php>
<http://www.cs.wright.edu/~pmateti/InternetSecurity/Lectures/RootKits/>
<http://neworder.box.sk/newsread.php?newsid=4182>

Kernel Rootkits LKM

<http://www.w00w00.org/files/articles/lkmhack.txt>
<http://members.prestige.net/tmiller12/papers/lkm.htm>
<http://it.rising.com.cn/safety/safetyschool/ywyb/020129lkm.htm>

Tools

<http://www.chkrootkit.org/>
<http://checkps.alcom.co.uk/>
<http://s0ftpj.org/en/site.html>

IDS

<http://www.tldp.org/HOWTO/Security-Quickstart-HOWTO/intrusion.html>
<http://www.tripwire.org/>
<http://www.snort.org/>
<http://online.securityfocus.com/infocus/1558>
http://rr.sans.org/intrusion/intrusion_list.php?sansrr=3f3ebaf64b33500c26b2df2f8af25066

Download

<http://www.thenewbiesarea.com/unix.shtml>
<http://packetstormsecurity.nl/UNIX/penetration/rootkits/>