Hostbasierte Intrusion Detection – Ein Einblick

Steffen Wendzel

Schwierigkeitsgrad

Um Angriffe auf Unix-Systeme detektieren zu können, steht eine Vielzahl von Programmen zur Verfügung. Einige davon sollen in diesem Artikel vorgestellt werden.

Mit der Zunahme an Bedrohungen innerhalb von Netzwerken steigt auch der Wunsch nach der Detektion von netzwerkbasierten Angriffen. Um diese Detektion zu ermöglichen, entwickelte man Intrusion Detection Systeme (IDS) für Netzwerke, die man als Network Intrusion Detection Systems (NIDS) bezeichnet. Ist ein Angreifer erst einmal in einem Netzwerk, so wird er – zumindest meistens – ein bestimmtes Ziel haben, dass er angreifen möchte. Die Detektion von Angriffen auf einzelne Systeme können NIDS allerdings nur zum Teil abdecken. Für die Detektion von lokalen Angriffen entwickelte man daher Host Intrusion Detection Systeme (HIDS). Dieser Artikel wird die wichtigsten Möglichkeiten, die Ihnen HIDS bieten sowie einige HIDS-Software vorstellen. Weitere Informationen zum Thema Network Intrusion Detection (Systems) finden Sie allerdings in den am Ende des Artikels gelisteten Büchern. Die Anwendung der HIDS-Systeme ist dabei in der Regel gar nicht sonderlich schwierig. Bedenkt man, wie wichtig HIDS sind, dann könnte man also schonmal sagen, dass man *viel* (Sicherheit) für *wenig* (Lernaufwand) bekommt.

Doch bevor es mit den eigentlichen HIDS-Softwareprojekten los geht, soll noch kurz besprochen werden, welche Typen von HIDS es überhaupt gibt, und worin ihr Unterschied besteht.

Zunächst einmal gibt es zwei erschiedene Verfahrensweisen, wie ein IDS einen Angriff detektiert. Die erste Möglichkeit basiert auf Signaturen. Dabei werden i.d.R. durch die Konfiguration bestimmte Signaturen (etwa bestimmte Strings in Netzwerkstreams) definiert, mit denen das IDS überprüft, ob es sich um einen Angriff handelt (eine Signatur aus einer

In diesem Artikel erfahren Sie...

- Verschiedene Typen von HIDS;
- HIDS-Softwareprojekte

Was Sie vorher wissen/können sollten...

- TCP/IP Grundkenntnisse;
- Erfahrung mit Unix Netzwerken.

Menge von Signaturen kommt in einer Menge von zu überprüfenden Daten vor) oder ob es sich um keinen Angriff handelt (also keine Signatur *matcht*, d.h. gefunden wird).

Die zweite Möglichkeit basiert auf verschiedenen Formen künstlicher Intelligenz (KI) und nennt sich Anomalie-Detektion. Dabei muss das IDS zunächst lernen, welche Daten *>erlaubt< oder >gut< sind*, um später Abweichungen entdecken und bewerten zu können.

Filesystem IDS

Eine ganz typische Art von hostbasierten IDS sind die so genannten Filesystem Intrusion Detection Systems (FSIDS). Diese Überwachen das Dateisystem (bzw. Teile davon) auf Veränderungen.

Besonders bekannt sind dabei vier FSIDS-Vertreter: Tripwire, AIDE, AFICK und mtree. Tripwire war früher eine zwar freie, aber nicht Quelloffene FSIDS-Implementierung, die in diesem Artikel nicht besprochen werden soll. AIDE gilt als OpenSource Replacement für Tripwire und ist für Linux, diverse BSD-Derivate, diverse Unix-Derivate sowie für cygwin verfügbar. Das Tool mtree hingegen stammt aus der BSD-Welt und ist u.A. Bestandteil des OpenBSD-Projektes. Verfügbar war mtree allerdings schon zu Zeiten von 4.3BSD-Reno.

AFICK

AFICK (Another File Integrity Checker), das in Kombination mit ActivePerl auch für Windows zur Verfügung steht, ist hingegen etwas für all Jene, die eine grafische Oberfläche (Tk-basiert) bevorzugen. Ein Webinterface in Form eines Webmin-Modules ist im Übrigen ebenfalls verfügbar.

AIDE

AIDE wird über die Datei aide.conf, die sich meistens in */etc* oder */etc/aide* befindet, konfiguriert. Standardmäßig verwendet AIDE */etc/aide.conf*, was man aber durch den *Parameter –config=[Pfad]* beim Aufruf ändern kann. Der Syntax der Datei ist recht einfach gehalten und teilweise selbsterklärend, was mich natürlich nicht davon abhält, Ihnen die Details trotzdem darzulegen.

Bevor man AIDE zur Überwachung eines Systems verwenden kann, muss AIDE zunächst eine Datenbank des zu überwachenden (Teils des) Dateisystems erstellen. In der Konfigurationsdatei wird dabei angegeben, welche Verzeichnisse und welche Attribute der einzelnen Dateien dafür berücksichtigt werden sollen. Die beiden Zeilen im Folgenden Listing geben an, in welcher Datei AIDE eine neu erstellte Datenbank ausgeben soll, und zum anderen, welche Datei schließlich als Datenbank verwendet werden soll:

```
database=file:/var/lib/aide/ \
aide.db
database_out=file:/var/lib/ \
aide/aide.db.new
```

Weiterhin können in der Konfigurationsdatei von AIDE Definitionen gesetzt werden, die wie simple Variablen verwendet werden können. Diese Definitionen findet man in der Standardkonfiguration zur Regelung der zu überprüfenden Attribute der Dateien. Beispielsweise schaltet die Definition `>OwnerMode<` (s. nächstes Listing) die Überprüfung für Veränderungen der Zugriffsrechte (`p`), des Eigentümers der Datei (`u`) und der Gruppenzugrhörigkeit der Datei (`g`) ein.

```
OwnerMode = p+u+g
```

Wahrscheinlich erahnen Sie nun bereits den Syntax für die zu überprüfenden Attribute: Diese werden durch kurze Strings (bzw. einfache Buchstaben) abgekürzt und mit Plus-Zeichen kombiniert. Möchte man nun beispielsweise das *Verzeichnis /etc* auf die Attribute der OwnerMode-Definition sowie auf deren Inode-Nummern hin überprüfen lassen, so würde dies folgendermaßen aussehen:

```
/etc OwnerMode+i
```

Nachdem man die entsprechenden Verzeichnisse, die AIDE überprüfen soll, in der Konfigurationsdatei spezi-

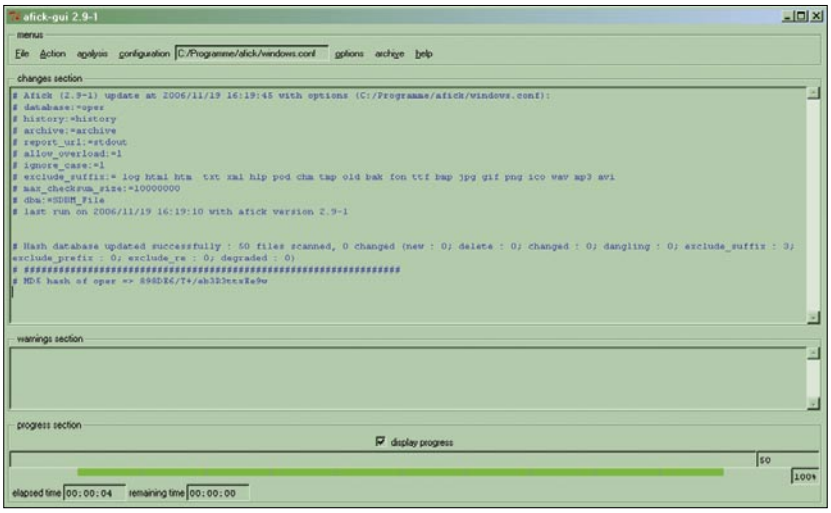

Abbildung 1.AFICK *(Another File Integrity Checker)*

Kurz und bündig: Wozu HIDS?

- um möglichst unmittelbar über Angriffe informiert zu werden,
- um möglichst unmittelbar auf Angriffe reagieren zu können,
- um Beweise für Angriffe zu sammeln,
- für forensische Analysen,
- als Vorbereitung/Grundlage für Gegenmaßnahmen (sog. Intrusion Response).

fiziert hat, kann AIDE die Datenbank erstellen, was mit dem Parameter `--init` erledigt wird:

```
# aide --init \
--config=/etc/aide/aide.conf
AIDE, version 0.12
### AIDE database at /var/lib/
aide/aide.db.new initialized.
```

Nun kann die Datenbank als offizielle Datenbank verwendet werden, indem man den Dateinamen der Datenbank in denjenigen ändert, der in der Konfigurationsdatei unter `>database<` angegeben wurde:

```
# mv /var/lib/aide/aide.db.new \
/var/lib/aide/aide.db
```

Möchte man nun (bspw. durch einen *Cronjob*) das Dateisystem auf Veränderungen überprüfen lassen, so geht dies via –check-Paramter. Möchte man die Datenbank gleichzeitig updaten (was sehr sinnvoll ist, um jeweils nur die aktuellen Veränderungen zu sehen), so kann dies via –update-Parameter erledigt werden.

Listing I. zeigt, wie AIDE eine in der *Datei /etc/inetd.conf* neu eingefügte Zeile erkennt und den Administrator auf zweierlei Veränderungen hinweist: Die Größe der Datei nahm um ein Byte zu und die MD5-Prüfsumme der Datei hat sich dadurch ebenfalls verändert. Würde man noch die MAC-Werte der Datei überprüfen lassen, so würden diese natürlich zusätzliche Informationen liefern. Es empfiehlt sich allerdings nur bedingt alle MAC-Werte (speziell A – die Zugriffszeit) zu verwenden, da sich diese Werte ständig ändern und zu riesigen Logdateien führen können.

rkhunter

Weitere typische HIDS sind solche, die lokal installierte Backdoors und Rootkits detektieren können. Ein besonders populärer Vertreter dieser Art von HIDS ist der `>Rootkit Hunter<`, kurz *rkhunter*.

Den rkhunter gibt es unter *http://www.rootkit.nl/projects/rootkit_hunter.html*, Hat man diesen einmal installiert, so kann man das System mit allerlei Überprüfungen nach Sicherheitsproblemen durchsuchen lassen, am Besten, man greift gleich auf den Parameter – checkall zurück. Dabei wird zunächst das Vorhandensein diverser Rootkits überprüft. Anschließend wird nach trojanischen Pferden, nach unsicheren Zugriffsrechten für wichtige Binärdateien, nach eventuell gefährlichen Kernelmodulen sowie nach Netzwerkschnittstellen, die sich im Promiscuous Modus befinden, gesucht.

Tabelle 1 *zeigt die einzelnen Attribute der AIDE-Konfiguration und deren Bedeutung*

String/Zeichen	Bedeutung
p	permissions (Zugriffsrechte)
i	inode (Überprüft die Inode-Einträge)
n	number of links (Anzahl der Hardlinks)
u	user (Eigentümer der Datei)
g	group (Gruppenzugehörigkeit der Datei)
s	size (Dateigröße)
b	block count (Anzahl der Blöcke)
m	modification time (Modifizierungs Timestamp)
a	access time (Timestamp des letzten Zugriffs)
c	creation time (Timestamp der Erzeugung der Datei)
S	growing size (Überprüft Datei auf Wachstum)
md5	Überprüft MD5-Prüfsumme der Datei
sha1	Überprüft SHA1-Prüfsumme der Datei
rmd160	Überprüft RMD160-Prüfsumme der Datei
tiger	Überprüft Tiger-Prüfsumme der Datei
E	empty group (Überprüft, ob Gruppenzugrhörigkeit der Datei fehlt)

Listing 1. *AIDE: Erkennung einer neu eingefügten Zeile*

```
# echo >> /etc/inetd.conf
# aide --update \
--config=/etc/aide/aide.conf
AIDE found differences between
database and filesystem!!
Start timestamp:
    2006-11-17 20:36:44
Summary:
  Total number of files: 1902
  Added files: 0
  Removed files: 0
  Changed files: 1
---------------------------------
Changed files:
---------------------------------
changed: /etc/inetd.conf
---------------------------------
Detailed information about
changes:
---------------------------------
File: /etc/inetd.conf
  Size     : 1448 , 1449
  MD5      :
VdRqohll8ZTrJl68vLDaK
w== ,
h2Tm2kHwK8GMjV1/rq
DAUA==
```

Listing 2. *Eine Detektion für den Benutzer mit der User-ID 1000*

```
$ gcc -o exploit exploit.c
Apr 26 14:39:21 projekt /bsd:
fupids: new programm: 'gcc',
uid: 1000
Apr 26 14:39:21 projekt /bsd:
fupids: new programm: 'cpp0',
uid: 1000
Apr 26 14:39:21 projekt /bsd:
fupids: new programm: 'cc1',
uid: 1000
Apr 26 14:39:21 projekt /bsd:
fupids: new programm: 'as',
uid: 1000
Apr 26 14:39:21 projekt /bsd:
fupids: new programm:
'collect2', uid: 1000
Apr 26 14:39:21 projekt /bsd:
fupids: new programm: 'ld',
uid: 1000
```

Zudem werden Unstimmigkeiten in den Passwortdateien aufgedeckt, die Startskripte des Systems untersucht, versteckte Dateien und sonderbare Gerätedateien in */dev* gesucht und auch einige Anwendungen überprüft. Beispielsweise kann rkhunter Ihre Apache-Konfiguration und die Versionsnummern der Dienste überprüfen. Wird eine veraltete Version (mit Sicherheitslöchern) gefunden, so bekommen Sie den entsprechenden Hinweis. Auch kann bspw. detektiert werden, dass in der SSH-Konfiguration root-Logins möglich sind.

OSSEC-HIDS

Ein ähnliches HIDS ist das OSSEC-HIDS. Dies basiert auf einer Client-Server-Architektur. Auf einem System kann man entweder einen OSSEC-Server, der Logging-Informationen der Clients über eine verschlüsselte Verbindung entgegen nimmt, laufen lassen, oder aber man installiert OSSEC als Agent. Ein Agent überträgt die Logging-Informationen an den OSSEC-Server, auf dem alle Daten dann zentral gesammelt werden. Die dritte Installationsmöglichkeit nennt sich `>local<` und umgeht das Client-Server-Modell. Die local-Installation empfiehlt sich nur auf Einzelsystemen.

Leider verwendet OSSEC eine sehr eigene Verzeichnisstruktur, die man als Distributor nur ungern in sein System aufnimmt, weshalb es für Hardened Linux noch kein OSSEC-Paket gibt. Ich habe dem leitenden Entwickler vorgeschlagen dies zu ändern, und es bleibt zu hoffen, dass dies auch getan wird, denn derzeit scheint (fast?) keine Distribution OSSEC zu beinhalten. Zudem hat das OSSEC-HIDS leider noch einige Bugs, auf die ich bereits hinwies, und die noch nicht gefixt sind, weshalb ich es hier nicht näher beschreiben werde. Es bleibt zu hoffen, dass wir in Zukunft mehr über dieses interessante Projekt hören werden.

FUPIDS(2)

Bei FUPIDS und fupids2 handelt es sich um zwei meiner Eigenentwicklungen, die ich nur am Rande vorstellen werde, um zu zeigen, dass man HIDS auch auf Benutzer-Überwachung ansetzen kann. Es ist allerdings rechtlich bedenklich, sofern die >*Benutzer*< nicht abstrakt als Accounts bestimmter Dienste gesehen werden. Man überwacht also bspw. die Programme, die der >*Benutzer*< des Apache-Daemons ausführt.

FUPIDS ist ein Patch für den OpenBSD-Kernel, der mit Fuzzy-Logic lernt, welche Programme von welchen Benutzern ausgeführt werden. Nach einer gewissen Lernphase weiß der Kernelcode dann, welche Programme typischer Weise ausgeführt werden, und welche nicht. Führt ein Angreifer, der nie zuvor einen C-Compiler ausgeführt hat, bspw. einen solchen aus, dann kann auf einen übernommenen Account geschossen werden, der möglicherweise von einem Angreifer zum kompilieren von Exploit-Code verwendet wird. Eine Detektion für den Benutzer mit der User-ID 1000 zeigt das Listing 2.

Nun kommen wir zu fupids2. Es funktioniert auf eine ähnliche Weise wie FUPIDS, arbeitet jedoch betriebssystemunabhängig und verwendet zur Berechnung der Wahrscheinlichkeit eines übernommenen Accounts neuronale Netze. Ausserdem werden Tageszeit sowie Ort (Gebäude, Etage, Raum, Sitzplatz mit X-/Y-Koordinate) mit in die Berechnungen eingefügt. Beide Projekte findet man bei *freshmeat.net*. Auf *doomed-reality.org* finden Sie zudem noch einige meiner Texte zum entsprechenden Thema.

IPS und IRS

Jeweils ein kurzes Wort muss an dieser Stelle noch zu den Themen Intrusion Prevention Systems (IPS) sowie Intrusion Response Systems (IRS) fallen. Erstere, die IPS, können Angriffe abwehren. Beispielsweise kann mit systrace unter BSD und diversen Erweiterungen unter Linux definiert werden, welche Syscalls ein bestimmtes Programm durchführen darf. Falls ein Programm auf einmal ein `execve()` Syscall durchführt um eine Shell zu starten, so kann dies dadurch vermieden werden. Man bezeichnet diese Handhabung von nicht explizit erlaubten Handlungen auch als >*default deny*< Policy.

Intrusion Response hingegen kann nach einem detektieren Angriff bestimmte Maßnahmen ergreifen, beispielsweise den Rechner komplett abschotten (Achtung! Das ist gefährlich, da man sich kaum vor Denial of Service Angriffen schützen kann.) oder den Gegner attackieren (was rechtlich natürlich sehr bedenklich ist!). ●

Links und Literatur

- rkhunter: *http://www.rootkit.nl/projects/rootkit_hunter.html*
- OSSEC-HIDS: *http://www.ossec.net/*
- OpenSource Tripwire: *http://sourceforge.net/projects/tripwire/*
- AIDE: *http://www.cs.tut.fi/~rammer/aide.html*
- AFICK: *http://afick.sourceforge.net/*
- fupids2: *http://freshmeat.net/projects/fupids2/*
- FUPIDS: *http://freshmeat.net/projects/fupids/*
- Steffen Wendzel: Implementierung eines Fuzzy Userprofile IDS in den OpenBSD-Kernel: *http://www.ploetner-it.de/~dr/site/*
- Steffen Wendzel, Johannes Plötner: Praxisbuch Netzwerksicherheit, Galileo-Press 2005.
- Stephen Northcutt, Judy Novak: Network Intrusion Detection
- Richard Bejtlich: The Tao of Network Security Monitoring. Beyond Intrusion Detection

Über den Autor

Steffen Wendzel ist 22 und beschäftigt sich seit vielen Jahren mit der Sicherheit von Unix-Systemen und TCP/IP-Netzwerken. Er entwickelte diverse OpenSource Software, ist Autor mehrerer Bücher zu den Themen Linux und Netzwerksicherheit, Security Consultant bei Plötner-IT sowie Student der Informatik an der FH-Kempten (Deutschland). Seine Webseite: *http://cdp.doomed-reality.org*