Ralf Spenneberg

Gut bewacht

Intrusion Detection mit OSSEC

Systeme zur automatischen Einbruchserkennung schlagen Alarm, wenn auf einem Rechner oder im Netz etwas Suspektes passiert. Das freie Intrusion-Detection-System Open Source Security (OSSEC) checkt dazu die Integrität von Dateien, sucht Rootkits, analysiert Log-Dateien und überprüft DNS-Anfragen.

Einbruchserkennung, das klingt einfacher als es ist: Die Grenze zwischen normaler Nutzung und Anzeichen für einen Angriff ist nicht immer scharf zu ziehen. Intrusion-Detection-Systeme versprechen, unautorisierte Zugriffe anhand einer Vielzahl von Indikatoren zu erkennen, zu melden und je nach Konfiguration auch zu blockieren – man spricht dann von Intrusion-Prevention-Systemen (IPS).

Host-basierte Intrusion-Detection-Systeme (HIDS) kann man bereits auf einem einzelnen Server sinnvoll einsetzen. Sie analysieren die Daten auf diesem Rechner, darunter Protokolldateien, Anmeldevorgänge, die Ressourcennutzung sowie die Integrität wichtiger Systemdateien. In diesen Daten suchen HIDS nach sogenannten Indicators of Compromise (IoC) – Indizien, die auf einen Einbruch hinweisen (siehe Kasten auf Seite179).

Netzwerkbasierte IDS (NIDS) beziehen die Rohdaten für die Erkennung von Angriffen aus dem Netzwerkverkehr. Bekannte quelloffene netzwerkbasierte Intrusion-Detection/Prevention-Systeme sind Snort und Suricata. Beide analysieren den Netzwerkverkehr und erkennen Abweichungen von der Norm sowie spezifische Angriffe anhand typischer Signaturen.

Da die Host-basierten IDS eine Vielzahl von Daten korrelieren können, sind sie bei korrekter Konfiguration weniger anfällig für falsche Alarme als NIDS – das Erkennen von Abnormalitäten im Netzwerkverkehr in nahezu Echtzeit ist eine erheblich schwierigere Aufgabe. HIDS sind zudem einfacher zu konfigurieren und benötigen wesentlich weniger Ressourcen, was sie preiswerter in Anschaffung und Betrieb macht.

OSSEC

Ein sehr mächtiges und aktiv gepflegtes Host-basiertes Intrusion-Detection-System ist OSSEC (kurz für Open Source Security). OSSEC läuft unter Linux, Solaris, AIX, HP-UX, BSD, Windows, OS X und VMware ESX. Das Programm kann zudem die Protokolle vieler Netzwerkkomponenten und Appliances analysieren. Bei Erkennung eines kritischen Zustands kann OSSEC sowohl einen Alarm auslösen als auch im Sinne einer Intrusion Prevention Zugriffe blockieren.

Die 2004 von Daniel B. Cid begründete Software gehört seit 2009 zu Trend Micro, das bis vor einem Jahr auch Support dafür angeboten hat. Seit März 2014 hat Trend Micro den kommerziellen Support eingestellt, unterstützt das Projekt jedoch weiter. Unternehmen wie AlienVault, Hersteller des „Open Source Security Information and Event Management System“ OSSIM und des kommerziellen Produkts „AlienVault Unified Security Management“, sowie AtomiCorp bieten aber weiterhin Support für OSSEC.

OSSEC kann als verteiltes HIDS mit einer zentralen Verwaltungsoberfläche eine ganze IT-Infrastruktur überwachen, aber auch lokal auf einem einzelnen Rechner installiert werden, wie wir im Folgenden beschreiben. Eine solche Installation lässt sich mithilfe der hervorragenden Dokumentation des OSSEC-Projekts später leicht in eine größere Infrastruktur einbinden.

Installation

OSSEC erhält man als Quelltext und Binärpaket auf der Homepage des Projekts (www.ossec.net). Die Pakete in den Repositories für RHEL, CentOS, Fedora und Debian/Ubuntu, auf die dort verwiesen wird, sind für eine Installation als reiner Agent oder reiner Server in einem verteilten HIDS optimiert. Für einen einzelnen Server ist die Installation aus dem aktuellen Quelltextpaket ossec-hids einfacher. Nach dem Auspacken erledigt das Skript install.sh die Installation; es benötigt einige Entwicklungspakete, wie sie unter Debian und Ubuntu das Metapaket build-essential liefert. Wenn Sie später Systemdateien permanent in Echtzeit auf Veränderungen überwachen wollen statt lediglich in periodischen Scans, benötigen Sie außerdem die inotify-Tools.

Das Installations-Skript stellt einige Fragen. Nach der Sprache, die Sie mit „de“ beantworten, fragt es zunächst nach dem Installationsmodus. Hier wählen Sie „lokal“ – das installiert OSSEC im Stand-alone-Modus. Alternativ können Sie nur den Agenten oder den Server installieren. Den vorgeschlagenen Installationsort /var/ossec können Sie übernehmen. Um per Mail benachrichtigt zu werden, müssen Sie anschließend eine E-Mail-Adresse angeben und den vorgeschla-

genen SMTP-Server bestätigen. Dieser muss die Mails natürlich auch annehmen.

Anschließend können Sie entscheiden, welche Funktionen von OSSEC – Syscheck, Rootcheck, Active Response und Firewall-Drop – Sie nutzen möchten. Standardmäßig werden alle Optionen aktiviert. OSSEC startet über ein SysV-Initskript:

```
/etc/init.d/ossec start
```

Wenn Sie lieber eines der übersetzten Pakete nutzen wollen, wählen Sie den OSSEC-Server. Hier gibt es üblicherweise nur drei Unterschiede zur lokalen Installation:

- In der Datei /etc/ossec-init.conf lautet der Wert der Variablen TYPE „server“ statt „local“.
- In der Datei /var/ossec/etc/ossec.conf nimmt der Abschnitt <remote> die OSSEC-Clients auf; er kann bei einer lokalen Installation gelöscht werden.
- Das Skript /var/ossec/bin/ossec-control startet den Dienst ossec-remoted, der bei einer lokalen Installation nicht gebraucht wird. Dieses Start-Skript wird aus /etc/init.d/ossec aufgerufen.

Integrität von Dateien

Zu den wichtigsten Informationen, die OSSEC analysiert, gehört die Integrität wichtiger Systemdateien. Wie viele andere Host-basierte Intrusion-Detection-Systeme (Tripwire, Samhain, Aide) kann auch OSSEC den Inhalt von Dateien auf Veränderungen prüfen. Allerdings sollten Sie nicht das ganze System überwachen lassen, da sich auf einem Linux-Server regelmäßig Systemdateien ändern. Sind beispielsweise automatische Updates aktiviert, werden immer wieder Binaries und Bibliotheken in Systemverzeichnissen ausgetauscht – jedes Update führt dann zu einem falschen Alarm.

Besser ist es, nur Dateien zu überwachen, die sich im normalen Betrieb nicht ändern. Hierzu gehören typischerweise Konfigurationsdateien oder die von einem Webserver ausgelieferten statischen Inhalte. Letztlich muss aber jeder Admin selbst Erfahrungen sammeln, welche Dateien auf seinem System ausreichend statisch sind und welche von den regelmäßigen Überprüfungen ausgenommen werden müssen, um nicht übermäßig viele falsche Alarme zu generieren.

Die Konfiguration der zu überwachenden Dateien erfolgt im Abschnitt <syscheck> in /var/ossec/etc/ossec.conf:

```
<syscheck>
  <frequency>79200</frequency>
  <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
  <directories check_all="yes">/bin,/sbin</directories>
  <alert_new_files>yes</alert_new_files>
  <ignore>/etc/mtab</ignore>
  <ignore>/etc/mnttab</ignore>
</syscheck>
```

Standardmäßig erfolgt die Integritätsprüfung alle 22 Stunden (79200 Sekunden). Da dieser Check relativ viele Ressourcen benötigt, sollte man den <frequency>-Wert nicht zu niedrig setzen. Statt einer periodischen Überprüfung alle paar Stunden kann man auch eine Uhrzeit (<scan_time>) und einen Wochentag (<scan_day>) vorgeben, um den Scan in ein definiertes Wartungsfenster zu legen.

Der Parameter <directories> gibt die zu überwachenden Verzeichnisse an; check_all steht für die Prüfung von MD5- und SHA1-Prüfsumme, Dateigröße sowie Eigentümer, Gruppe und Rechte der Datei. Bei einer Veränderung eines dieser Werte benachrichtigt OSSEC den Aministrator per E-Mail.

Überwachung in Echtzeit

Sofern Inotify installiert ist, kann OSSEC Dateien auch in Echtzeit über die Inotify-Schnittstelle überwachen. Dies erfordert dann die zusätzliche Angabe von realtime="yes" im Tag <directories>. Die Echtzeitüberwachung beginnt allerdings erst nach einem Neustart von OSSEC: Die Software durchläuft beim Start das Dateisystem und registriert jedes Unterverzeichnis via Inotify. Bei umfangreichen Verzeichnisbäumen kann das einige Zeit dauern.

Häufig möchte der Admin nur bestimmte Dateitypen in einem Verzeichnis berücksichtigen. Dazu dient der Parameter restrict:

```
<directories check_all="yes" restrict=".php|.js|.html">/srv/www</directories>
```

Mit <ignore> lassen sich Dateien und Verzeichnisse von der Überwachung ausschließen. Hier sind auch reguläre Ausdrücke möglich:

```
<ignore type="sregex">.log$|.bak$</ignore>
```

Die Option <alert_new_files> sorgt dafür, dass OSSEC auch bei neuen Dateien in den überwachten Verzeichnissen Alarm schlägt. Standardmäßig schickt OSSEC hierbei keine Alarm-Mail, da Meldungen über neue Dateien über eine Regel in /var/ossec/rules/ossec_rules.xml lediglich mit dem Alarm-Level 0 versehen werden:

```
<rule id="554" level="0">
  <category>ossec</category>
  <decoded_as>syscheck_new_entry</decoded_as>
  <description>File added to the system.</description>
  <group>syscheck,</group>
</rule>
```

Um das Alarm-Level anzuheben, kopiert man den Block nach /var/ossec/rules/local_rules.xml und ändert dort die erste Zeile ab:

```
<rule id="554" level="7" overwrite="yes">
```

Nun verschickt OSSEC Mails für neue Dateien. Die Prüfung funktioniert jedoch nicht im Echtzeit-Modus; neue Dateien werden nur bei den regelmäßigen Komplett-Scans entdeckt.

Falscher Alarm

Grundsätzlich gibt es bei der Einbruchserkennung und -abwehr vier mögliche Fälle:

- Das Intrusion-Detection/Prevention-System hat einen Angriff korrekt erkannt, gemeldet und verhindert.
- Ein legitimer Zugriff wird als erlaubter Zugriff erkannt und nicht gemeldt oder verhindert.
- Falscher Alarm: Das IDS oder IPS stuft einen legitimen Zugriff als potenziellen Angriff ein, löst einen Alarm aus und verhindert den legitimen Zugriff.
- Ein echter Angriff wird nicht erkannt, es wird kein Alarm ausgelöst und der Angriff nicht verhindert.

Während die ersten beiden Fälle das erwartete und erwünschte Verhalten darstellen, erzeugen die letzten beiden Fälle Probleme im Betrieb. Die Fehler sind jedoch nicht unabhängig voneinander: Verringert man das Risiko, dass ein Angriff unerkannt bleibt, indem man das IDS schärfer einstellt, erhöht das die Gefahr von falschen Alarmen. Da das IDS/IPS Angriffe auf jeden Fall verhindern soll, tendieren Intrusion-Detection-Systeme dazu, eher falsche Alarme zu produzieren.

	Angriff	kein Angriff
Alarm	erkannter Angriff (true positive)	falscher Alarm (false positive)
kein Alarm	verpasster Angriff (false negative)	korrekt als legitim durchgelassen (true negative)

You’ve been hacked

Als Indicator of Compromise (IoC) bezeichnet man Merkmale, die bei einer forensischen Analyse mit hoher Wahrscheinlichkeit auf eine Kompromittierung des Systems hinweisen. Typische IoC sind:

- Virus-Signaturen
- IP-Adressen, DNS-Namen oder URLs bekannter Command&Control-Server
- MD5-Hashes von Malware-Dateien
- Malware-spezifische Windows-Registry-Keys

„IDS ist tot“ …

… lautete die zentrale These einer Gartner-Studie im Jahr 2003. Spätestens 2005, so die Marktforscher, sollten sämtliche Intrusion Detection Systeme durch wirksame Firewalls ersetzt sein, die die Netze vor den Angriffen schützen. Die wesentlichen Kritikpunkte von Gartner waren:

– IDS schützen nicht: Anders als ein Patch-Management, das Schwachstellen in Software behebt, und eine Firewall, die Angriffe verhindert, meldet ein IDS Angriffe nur.

– IDS erkennen nur bekannte Angriffe, die sich aber häufig durch Patches und wirksame Schutzmechanismen abwenden lassen.

– IDS sind zu teuer: Der finanzielle und personelle Aufwand steht in keinem Verhältnis zur Wirkung.

Trotz dieser Vorhersagen sind Intrusion-Detection-Systeme keineswegs ausgestorben. Sie haben sich jedoch weiterentwickelt: Viele aktuelle IDS arbeiten als Intrusion Prevention System (IPS) und wehren erkannte Angriffe gleich ab. Allerdings kann kein Intrusion Prevention System wirklich alle potenziellen Angriffe verhindern, da die Gefahr zu groß wäre, bei einem falschen Alarm eine legitime Anfrage abzulehnen. Einige Linux-Systeme nutzen daher die Mandatory-Access-Control-Lösungen AppArmor oder SELinux, die Angreifer nach einem Einbruch stoppen.

Falls OSSEC neue Dateien nur in bestimmten Verzeichnissen melden soll, schränkt man die Regel durch Einfügen einer Match-Bedingung im rule-Block ein:

```
<match>bin/</match>
```

Rootcheck

Häufig versuchen Angreifer nach einem erfolgreichen Einbruch, sich einen dauerhaften Zugang einzurichten. Die Hintertür wird mithilfe eines Rootkits versteckt, damit der Admin die zusätzlichen Prozesse und Netzwerkverbindungen nicht bemerkt. Die Rootcheck-Funktion in OSSEC sucht nach Anzeichen für derartige Rootkits. Das erfolgt im einfachsten Fall über die Dateinamen bekannter Rootkits – /var/ossec/etc/shared/rootkit_files.txt sammelt solche Dateinamen. Einige Rootkits ersetzen einzelne Befehle durch Trojaner, die sich an bestimmten Signaturen erkennen lassen. Die Datei /var/ossec/etc/shared/rootkit_trojans.txt enthält die Signaturen bekannter Rootkits.

Moderne Rootkits verstecken sich im Kernel. Um sie aufzuspüren, sucht OSSEC nach versteckten Prozessen und Netzwerkports. Dazu prüft es sämtliche Prozess-IDs und Ports. So bindet sich der Rootcheck-Test beispielsweise auf jeden Port. Schlägt das fehl, weil der Port belegt ist, ohne dass ihn der netstat-Befehl als belegt ausweist, handelt es sich möglicherweise um eine durch ein Rootkit versteckte Hintertür.

Diese allgemeinen Tests werden durch systemspezifische Prüfungen ergänzt, die in den RCL-Dateien in /var/ossec/etc/shared/ definiert sind. Sie testen, ob ein System verschiedenen Härtungsrichtlinien entspricht. Grundlage der Tests für Red Hat Enterprise Linux und Debian beispielsweise sind die Benchmarks des Center for Internet Security (CIS).

Forensische Analyse

Häufig suchen Anwender und Admins erst dann nach einer Sicherheitslösung, wenn das Kind bereits in den Brunnen gefallen ist. OSSEC kann auch dann helfen und beispielsweise die System-Logs nachträglich auf fehlgeschlagene Anmeldeversuche untersuchen:

```
cat /var/log/auth.log | /var/ossec/bin/ossec-logtest -a | /var/ossec/bin/ossec-reportd
```

OSSEC analysiert neben Syslog-basierten Protokollen die Logdateien unter anderem von Snort, IIS, MySQL, PostgreSQL, dem Apache-Webserver sowie Windows-Event-Logs. Das IDS bringt eine Vielzahl von Decodern mit, um die Protokollformate fast aller UNIX-Dienste und Netzwerkkomponenten zu analysieren – beispielsweise die Log-Dateien des DNS-Servers.

Malware-Erkennung per DNS

Viele Administratoren übersehen die Möglichkeiten, die die Analyse der DNS-Anfragen auf ihren Servern bieten. Fast alle Malware kommuniziert mit der Außenwelt; sei es, um ausgespähte Daten weiterzugeben, sei es, um sich nach der Installation in einem Bot-Netz zu melden. Viele DNS-Domänen und IP-Adressen, die von Angreifern missbraucht werden, sind öffentlich bekannt; DNS-Anfragen nach diesen Zielen sind ein Indikator für eine Kompromittierung durch Malware.

Verschiedene Anbieter stellen DNS-basierte Blackhole-Listen zur Verfügung. Unter mirror1.malwaredomains.com/files/justdomains findet man eine Liste mit über 20 000 Domänen, die in Zusammenhang mit der Verbreitung von Malware und Spyware stehen. Die Verwendung der Liste ist für nicht-kommerzielle Zwecke gestattet.

OSSEC kann diese Liste allerdings nicht direkt verarbeiten, sondern benötigt eine Key-Value-Liste. Die Umformung erledigt ein einfaches Skript:

```
cat justdomains | while read domain; do echo "$domain: Suspicious DNS Domain"; done > justdomains.txt
```

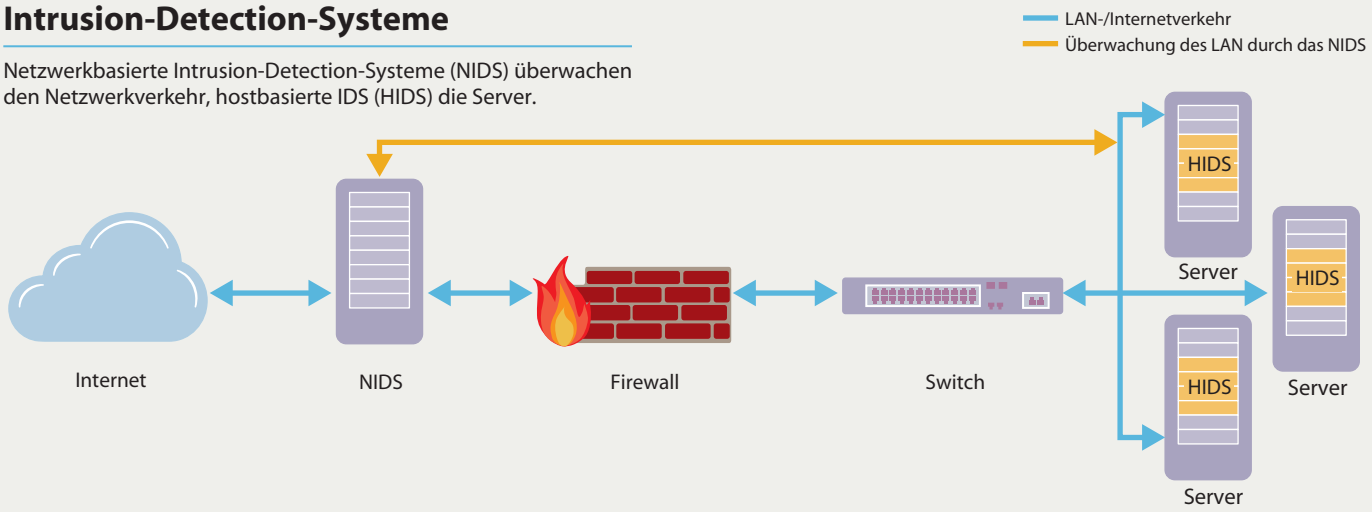
Legen Sie nun in /var/ossec/rules ein Unterverzeichnis lists an, kopieren Sie die Datei justdomains.txt dorthin und ergänzen Sie /var/ossec/etc/ossec.conf um die folgende Zeile:

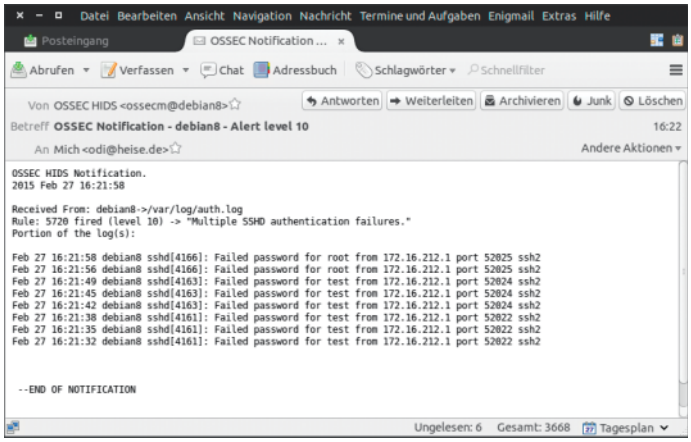
```
<rules> …
<list>rules/lists/justdomains.txt</list> </rules>
```

Ein Aufruf von /var/ossec/bin/ossec-makelists wandelt die Textdatei in eine binäre Datenbank um.

Intrusion-Detection-Systeme

Netzwerkbasierte Intrusion-Detection-Systeme (NIDS) überwachen den Netzwerkverkehr, hostbasierte IDS (HIDS) die Server.





OSSEC informiert den Admin per Mail, wenn verdächtige Aktionen stattfinden.

Damit der verbreitete DNS-Server bind9 die angefragten Domänen protokolliert, ist folgende Konfiguration nötig:

```
logging {
  channel "ossec" {
    syslog local7;
    severity info;
  };
  category lame-servers { null; };
  category "queries" { "ossec"; };
  category "unmatched" { "ossec"; };
};
```

Nach Einfügen der Zeile

```
local7.info   /var/log/named-ossec.log
```

in /etc/rsyslog.conf schreibt Rsyslogd die Meldungen mit der Facility local7 in eine eigene Datei.

DNS-Analyse

Damit OSSEC diese Datei analysiert, muss man die Konfigurationsdatei /var/ossec/etc/ossec.conf ergänzen:

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/named-ossec.log</location>
</localfile>
```

Die Dekodierung der Meldungen unterstützt OSSEC bereits und bringt in /var/ossec/rules/named_rules.xml einige Regeln ab ID 12100 zur DNS-Analyse mit. Der zuständige Decoder extrahiert den angefragten DNS-Namen in die Variable url. Hiermit kann eine Regel erzeugt werden:

```
<rule id="100201" level="10">
  <if_sid>12100</if_sid>
  <list field="url">rules/lists/justdomains.txt</list>
  <group>rootcheck,</group>
  <description>DNS query on a potentially malicious ↵
                                    domain.</description>
</rule>
```

Die Regel prüft zunächst über das <if_sid>-Tag, ob die Regel mit der SID 12100 zutrifft – das bedeutet, dass der DNS-Decoder die Protokollzeile parsen konnte. Anschließend wird getestet, ob der Inhalt von url in der DNS-Blackhole-Liste vorkommt.

Rechtekontrolle

Linux verfügt nur über eingeschränkte Möglichkeiten zur Verwaltung der Dateirechte: Es existieren lediglich die Rechte Lesen, Schreiben und Ausführen, die jeweils für den Eigentümer der Datei, genau eine Benutzergruppe und alle anderen (Others) gesetzt werden können. Es ist wichtig, dass hier das Prinzip der geringstmöglichen Rechte angewendet wird: Dateien sollen nur für diejeni-

Alarm-Regeln

Bei wiederholten erfolglosen Anmeldeversuchen in kurzer Zeit erzeugt das folgende Regelwerk einen „Brute force“-Alarm auf Alarmstufe 10.

```
<group name="syslog,">
  <rule id="100100" level="0" noalert="1">
    <decoded_as>foo</decoded_as>
    <description>FOO messages grouped.</description>
  </rule>
  <rule id="100101" level="4">
    <if_sid>100100</if_sid>
    <match>Connection from</match>
    <description>Foo connection attempt (scan).</description>
    <group>recon,</group>
  </rule>
  <rule id="100102" level="5">
    <if_sid>100100</if_sid>
    <match>Unsuccessful authentication</match>
    <description>Failed attempt to login</description>
    <group>invalid_login,authentication_failed,</group>
  </rule>
  <rule id="100103" level="10" frequency="6" timeframe="120" ignore="60">
    <if_matched_sid>100102</if_matched_sid>
    <description>FOO brute force</description>
    <same_source_ip />
    <group>authentication_failures,</group>
  </rule>
</group>
```

gen Benutzer und Gruppen les- und beschreibbar sein, die diese Rechte unbedingt benötigen.

OSSEC kann den Admin dabei unterstützen und Dateien melden, die world-writable – für jedermann beschreibbar – sind. Zwei einfache Regeln melden diese Dateien:

```
<rule id="100018" level="7">
<if_group>syscheck,</if_group>
<regex>Permissions changed from '\D+' to ➥
                              '\D\D\D\D\D\D\Dw\D'</regex>
<description>World-writable File</description>
</rule>
<rule id="100019" level="0">
<if_sid>100018</if_sid>
<regex>Permissions changed from '\D\D\D\D\D\D\Dw\D' ➥
                                        to '\D+'</regex>
<description>World-writable File</description>
</rule>
```

Beide Regeln filtern über reguläre Ausdrücke Meldungen über geänderte Zugriffsrechte für Dateien aus den Protokollmeldungen des OSSEC-Syscheck-Moduls aus. Die erste Regel meldet alle Dateien, bei denen sich die Rechte geändert haben und bei denen jetzt das Schreibrecht für alle gesetzt ist, mit dem Alarm-Level 7. Die zweite Regel prüft alle Treffer darauf, ob die Datei dieses Recht bereits vorher hatte; wenn ja, versieht sie die Meldung mit dem Level 0, sodass diese Dateien nicht zu einer Meldung führen.

Log-Analyse

Analog kann ein Admin auch für Anwendungen, deren Protokolle OSSEC nicht kennt, eigene Regeln schreiben. Das Modul ossec-logtest analysiert eine Meldung wie

```
Jan 19 13:37:30 station2 sshd[8872]: Failed password for ➥
invalid user test from 192.168.15.5 port 60343 ssh2
```

in drei Phasen. Zunächst ermittelt Logtest das protokollierende Programm und den Namen des Rechners, auf dem die Meldung erzeugt wurde:

```
**Phase 1: Completed pre-decoding.
  full event: 'Jan 19 13:37:30 station2 ...
  hostname: 'station2'
  program_name: 'sshd'
  log: 'Failed password for invalid user test from ➥
                                  192.168.15.5 port 60343 ssh2'
```

Der Inhalt von program_name bestimmt, welcher Decoder die Protokollmeldung in Phase 2 analysiert. Der sshd-Decoder beispielsweise extrahiert die IP-Adresse, von der aus der Login via ssh erfolgt ist:

```
**Phase 2: Completed decoding.
  decoder: 'sshd'
  srcip: '192.168.15.5'
```

Der Decoder bestimmt auch, welcher Regelsatz in der dritten Phase angewendet wird – im Beispiel /var/ossec/rules/sshd_rules.xml. Hier passt Regel 5710 auf die Meldung:

```
<rule id="5710" level="5">
  <if_sid>5700</if_sid>
  <match>illegal user|invalid user</match>
  <description>Attempt to login using a non-existent ➥
                                          user</description>
  <group>invalid_login,authentication_failed,</group>
</rule>
```

Das Ergebnis der Analyse:

```
**Phase 3: Completed filtering (rules).
  Rule id: '5710'
  Level: '5'
  Description: 'Attempt to login using a non-existent user'
```

Ein eigener Decoder

Im Folgenden gehen wir von einem Programm Foo aus, das die folgenden Protokollzeilen erzeugt:

```
Jan 19 13:37:30 station2 foo[512]: Connection from ➥
                                          192.168.15.5
Jan 19 13:37:30 station2 foo[512]: Unsuccessful ➥
                        authentication for ralf from 192.168.15.5
```

Ohne passenden Decoder gibt OSSEC in Phase 2 bloß `No decoder matched` aus. Ein Decoder für Foo ist allerdings schnell an die Datei /var/ossec/etc/local_decoder.xml angehängt:

```
<decoder name="foo">
  <program_name>foo</program_name>
</decoder>
```

Nun greift in Phase 2 der neue Decoder foo. Weitere Decoder analysieren die beiden Protokollzeilen:

```
<decoder name="foo-connection">
  <parent>foo</parent>
  <prematch offset="after_parent">^Connection ➥
                                          </prematch>
  <regex offset="after_prematch">^from (\S+)$</regex>
  <order>srcip</order>
</decoder>
```

Damit wird die erste Meldung bereits richtig dekodiert und die IP-Adresse, von der aus der Zugriff erfolgt ist, als srcip extrahiert. Ein zweiter Decoder wertet die zweite Protokollmeldung aus:

```
<decoder name="foo-authentication">
  <parent>foo</parent>
  <prematch offset="after_parent">
authentication</prematch>
  <regex offset="after_parent">^(\S+) authentication for ➥
                                  (\S+) from (\S+)$</regex>
  <order>status, user, srcip</order>
</decoder>
```

Das Ergebnis:

```
**Phase 2: Completed decoding.
  decoder: 'foo'
  status: 'Unsuccessful'
  dstuser: 'ralf'
  srcip: '192.168.15.5'
```

Nun fehlen nur noch die passenden Alarmregeln. Eigene Regeln tragen eine ID ab 100 000 und werden in /var/ossec/rules/local_rules.xml gepeichert (siehe Listing auf Seite 182).

Echtzeit-Abwehr

OSSEC kann Angriffe auch abwehren. So könnte beispielsweise ein Skript aufgerufen werden, das die Firewall anweist, den betroffenen Rechner zu isolieren, wenn die DNS-Analyse eine Malware erkennt. OSSEC bringt in /var/ossec/active-response/bin fertige Active-Response-Skripte für unterschiedliche Szenarien mit.

Für Linux-Firewalls mit iptables eignet sich das Skript firewall-drop.sh. Dieses Skript muss der Admin zunächst in /var/ossec/etc/ossec.conf aktivieren:

```
<command>
  <name>firewall-drop</command>
  <executable>firewall-drop.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>
```

Dank der Option <timeout_allowed> lässt sich eine gesperrte IP-Adresse nach einer bestimmten Zeit automatisch wieder freigeben.

Für die Sperrung sind die folgenden Zeilen nötig:

```
<active-response>
<command>firewall-drop</command>
  <location>local</location>
  <level>6</level>
  <timeout>600</timeout>
</active-response>
```

Damit werden alle IP-Adressen, die in Alarmen mindestens der Stufe 6 vorkommen, für zehn Minuten blockiert. Mit der zusätzlichen Zeile

```
<rules_group>rootcheck</rules_group>
```

lässt sich die Active Response auf die Malware-Erkennung der Rootcheck-Funktion von OSSEC beschränken. Um Wiederholungstäter stärker zu bestrafen, kann der Admin noch die folgende Option einfügen:

```
<repeated_offenders>30,60,120</repeated_offenders>
```

Während der Timeout in Sekunden definiert wird, handelt es sich hier um Minutenangaben: Bei der zweiten Verletzung wird die IP-Adresse für 30 Minuten, bei der dritten Verletzung für 60 Minuten und ab der vierten Verletzung für zwei Stunden gesperrt.

Tägliche Benachrichtigung

Bei entsprechender Konfiguration in /var/ossec/etc/ossec.conf verschickt OSSEC zusätzlich zu den Alarmen einen täglichen Report per Mail:

```
<ossec_config>
<reports>
<category>authentication_success</category>
<user type="relation">srcip</user>
<title>Daily report: Successful logins</title>
<email_to>me@myemail.com</email_to>
</reports>
</ossec_config>
```

Die Meldungen lassen sich in den täglichen Zusammenfassungen nach Kriterien gruppieren:

- group, category: Filter nach Gruppe oder Kategorie. Die Kategorien sind in OSSEC fest definiert: firewall, ids, syslog, web-log, squid, windows und ossec. Jede Regel ist einer Kategorie zugeordnet. Log-Analyse-Regeln weisen Protokolleinträge beliebigen Gruppen zu, beispielsweise authentication_success. Bei der Filterung sind group und category synonym.
- rule: Filter nach der Regelnummer.
- level: Filter nach dem Schweregrad.
- location: Filter nach dem Dateinamen (zum Beispiel /var/log/auth.log) oder bei verteilten Installationen nach dem Rechnernamen.
- srcip: Filter nach der Quell-IP-Adresse eines Zugriffs.
- user: Filter nach einem Benutzernamen.

Diese Schlüsselwörter lassen sich auch in Kombination mit type="relation" verwenden: Das Beispiel oben liefert pro User eine Liste der IP-Adressen, von denen aus er sich eingeloggt hat.

OSSEC erkennt und meldet bereits in der Default-Installation viele Indicators of Compromise. Eine Anpassung an das eigene System ist jedoch meist unumgänglich – und je mehr Möglichkeiten man nutzen möchte, desto aufwendiger werden die Anpassungen. (odi@ct.de)

c't Downloads, Benchmarks: ct.de/yjd1