



14 Tripwire im Unternehmensnetzwerk

Tripwire stellt ein sehr mächtiges Werkzeug zur Integritätsüberwachung der im Netzwerk befindlichen Rechner dar. In großen Unternehmensumgebungen befinden sich die Rechner meist in unterschiedlichen Netzen und an unterschiedlichen physikalischen Orten. Eine klassische Administration von Hand kann eine konsistente und nachvollziehbare Verwaltung dieser Rechner nicht ermöglichen. Daher sollen in diesem Kapitel Methoden und Werkzeuge vorgestellt werden, die eine zentralisierte Administration und Überwachung von Tripwire ermöglichen. Bei den zusätzlich eingesetzten Werkzeugen handelt es sich ausnahmslos um Open-Source-Werkzeuge, die auf jeder Linux-Distribution eingesetzt werden können und meist auch verfügbar sind.

Um die Tripwire-Installationen zentral verwalten zu können, sollte ein Rechner zur Verfügung stehen, der keine weiteren Dienste anbietet. Er sollte lediglich als IDS-Workstation zur Verarbeitung der Daten dienen. Besteht für den Angreifer die Möglichkeit, auf diesem zentralen Rechner einzubrechen, so kann er, wenn nicht zusätzlich Werkzeuge wie LIDS eingesetzt werden, die Berichte und die Datenbanken nach Gutdünken modifizieren.

14.1 Zentrale Erzeugung der Konfigurationsdateien

Ein ganz wichtiger Punkt in der Verwaltung der Konfigurationsdateien ist deren Einheitlichkeit. Sinnvoll ist die Erzeugung einer einzigen Konfigurationsdatei, die die entsprechenden Regeln für alle zu überprüfenden Rechner enthält. Zusätzlich sollten Werkzeuge existieren, die es ermöglichen, die Konsistenz dieser Dateien über alle verwalteten Rechner sicherzustellen. Desgleichen sollten die Datenbanken auf einem zentralen Rechner gesichert und in regelmäßigen Abständen überprüft werden. Der Aufruf der Integritätsprüfungen mit der Tripwire-Datenbank und die Alarmierung der verantwortlichen Administratoren kann auch zentral gesteuert erfolgen. Die Berichte der Überprüfungen werden zentral gesammelt und ausgewertet (siehe nächster Abschnitt).

Die Konfigurationsdateien *twcfg.txt* und *twpol.txt* werden im Folgenden für die Firma nohup.info beispielhaft entwickelt. Die Firma nohup.info verfügt über die folgenden durch Tripwire zu überwachenden Rechner: Squid-Proxy, E-Mail-Server, Webserven, DNS-Server, Samba-Server und OpenLDAP-Server.

14.1.1 twcfg.txt

Die Datei *twcfg.txt* enthält die Konfiguration der Pfade und der E-Mail-Einstellungen. Diese Datei hat folgendes Format:

```

ROOT                =/usr/sbin
POLFILE             =/etc/tripwire/tw.pol
DBFILE              =/var/lib/tripwire/$(HOSTNAME).twd
REPORTFILE          =/var/lib/tripwire/report/$(HOSTNAME).twr
SITEKEYFILE         =/etc/tripwire/site.key
LOCALKEYFILE        =/etc/tripwire/$(HOSTNAME)-local.key
EDITOR              =/bin/vi
LATEPROMPTING       =false
LOOSEDIRECTORYCHECKING =false
MAILNOVIOLATIONS    =true
EMAILREPORTLEVEL    =3
REPORTLEVEL         =3
MAILMETHOD           =SENDMAIL
SYSLOGREPORTING     =false
MAILPROGRAM         =/usr/sbin/sendmail -oi -t
GLOBALEMAIL         =tripwire@nohup.info

```

Die Berichte sollen später mit entsprechenden Werkzeugen auf den zentralen Server übertragen werden. Dazu ist es sinnvoll, dass der Dateiname des Berichtes nicht den Zeitstempel enthält. Dann ist später ein einfacher Zugriff auf die Datei möglich. Wird der Zeitstempel mit in den Namen aufgenommen, ist die Kenntnis für einen späteren Zugriff erforderlich. Im Weiteren ist zu überlegen, ob eine grundsätzliche E-Mail-Benachrichtigung sinnvoll ist oder ob sie deaktiviert werden sollte. Dann können spezifische E-Mails versendet werden (s.u.).

14.1.2 twpol.txt

Die Datei *twpol.txt* beschreibt die Richtlinien, mit denen die Rechner überwacht werden sollen. Sinnvollerweise wird eine Datei erzeugt, die für sämtliche Rechner anschließend angewendet werden kann. Dies ermöglicht eine einfache zentrale Konfiguration der Richtliniendatei und ihre anschließende Verteilung.

Bei der Erzeugung der Tripwire-Richtliniendatei sollte die spätere Wartbarkeit und Pflege berücksichtigt werden. Dies lässt sich besonders durch eine strukturierte und gut dokumentierte Konfigurationsdatei erreichen. Des Weiteren sollte die Konfigura-

tionsdatei lesbar und überschaubar gehalten werden. Im Folgenden soll beispielhaft die Erzeugung dieser Datei beschrieben werden.

Die Konfigurationsdatei sollte mit der Definition allgemeiner Regeln beginnen. Diese Regeln überprüfen die Integrität der Systemdateien, die allen Rechnern gemeinsam sind.

Listing 14.1 Allgemeine Tripwire-Regeln

```
(rulename=general)
{
  # überwache /etc außer /etc/mtab
  /etc          -> $(ReadOnly);
  !/etc/mtab

  # überwache Binärdateien in /bin und /sbin
  /bin          -> $(ReadOnly);
  /sbin        -> $(ReadOnly);

  # überwache Bibliotheken
  /lib          -> $(ReadOnly);

  # überwache /usr
  /usr         -> $(ReadOnly);

  # überwache Geräte
  /dev         -> $(ReadOnly);

  # überwache Dateien des Bootvorganges /boot
  /boot        -> $(ReadOnly);

  # überwache Protokolldateien
  /var/log     -> $(Growing);

  # überwache Heimatverzeichnis von root
  /root        -> $(ReadOnly);
}
```

Anschließend sollten die Regeln für die einzelnen Rechner hinzugefügt und angepasst werden. Dies erfolgt am einfachsten mit `@ifhost`-Direktiven.

Der Squid-Proxy-Rechner befindet sich in der DMZ. Er dient den internen Benutzern zum Zugriff auf das Internet. Hierzu kontaktieren diese den Squid-Prozess, der dann die Verbindung zu den eigentlichen Webservern im Internet aufbaut. Häufig wird im Zusammenhang mit dem Squid auch eine Authentifizierung der Benutzer verlangt und zusätzliche Werkzeuge wie zum Beispiel *squidGuard* (<http://www.squid-guard.org>) eingesetzt.

Der Squid-Proxy besteht aus einem Binärprogramm und mehreren Authentifizierungs-Modulen. Die Konfiguration befindet sich üblicherweise im Verzeichnis `/etc/squid`. Die Protokolle und der Cache befinden sich in den Verzeichnissen `/var/log/squid` und `/var/spool/squid` (Red Hat) oder `/var/squid/logs` und `/var/squid/cache` (SUSE).

Um nun den Squid-Proxy mit Tripwire zu überwachen, werden die folgenden Regeln benötigt:

Listing 14.2 Squid Tripwire-Regeln

```

@@ifhost squid
(rulename = squid, emailto=squid_admin@nohup.info)
{
    # Squid-Konfiguration
    /etc/squid                -> $(ReadOnly);

    # Squid-Binärdateien
    /usr/sbin/squid          -> $(ReadOnly);

    # Squid-Bibliotheken
    /usr/lib/squid           -> $(ReadOnly);    # Red Hat
    /usr/share/squid        -> $(ReadOnly);    # SUSE

    # Squid-Protokolle
    /var/log/squid          -> $(Growing);     # Red Hat
    /var/squid/logs        -> $(Growing);     # SUSE
}
@@endif

```

Für den E-Mail-Server werden zwei verschiedene Konfigurationen benötigt. Die Firma nohup.info verfügt über zwei E-Mail-Server. Ein E-Mail-Server wird als E-Mail-Relay in der DMZ eingesetzt und ein weiterer übernimmt die Rolle des internen E-Mail-Hubs. Der interne E-Mail-Hub verfügt zusätzlich zum Mail Transport Agent (MTA) auch über einen IMAP-Server. In diesem Fall handelt es sich um den Einsatz von Sendmail als MTA¹ und UW-Imap-Server. Dieser UW-Imap-Server wird über den *xinetd* (Red Hat) oder den *inetd* (SUSE) gestartet.²

Damit Tripwire diese Dienste überwachen kann, werden die folgenden Regeln eingesetzt:

```

@@ifhost mailrelay
(rulename = mailrelay, emailto=relay_admin@nohup.info)
{
    # Sendmail Konfiguration

```

1 Sendmail ist immer noch der Standard Mail Transport Agent unter Linux. Andere Mail-Server wie Postfix oder Qmail sind aber genauso gut möglich. Hier soll Sendmail lediglich als Beispiel dienen.

2 Andere IMAP-Server wie zum Beispiel der Cyrus IMAP-Server laufen als selbstständige Dienste.

```

/etc/sendmail.cf          -> $(ReadOnly);
/etc/aliases             -> $(ReadOnly);
/etc/mail                -> $(ReadOnly);

# Postfix Konfiguration
# /etc/postfix           -> $(ReadOnly);

# Red Hat verwendet ein Alternatives System zur Wahl von Sendmail/Postfix
/etc/alternatives        -> $(ReadOnly);

# Binärdateien (SUSE, Links bei Red Hat)
/usr/bin/mailq           -> $(ReadOnly);
/usr/bin/newaliases      -> $(ReadOnly);
/usr/sbin/sendmail       -> $(ReadOnly);
# Binärdateien (Red Hat)
/usr/bin/mailq.sendmail  -> $(ReadOnly);
/usr/bin/newaliases.sendmail -> $(ReadOnly);
/usr/sbin/sendmail.sendmail -> $(ReadOnly);

# Protokolldateien
/var/log/maillog         -> $(Growing);
}
@@endif

@@ifhost mailhub
(rulename = mailhub, emailto=mail_admin@nohup.info)
{
# Sendmail Konfiguration
/etc/sendmail.cf          -> $(ReadOnly);
/etc/aliases             -> $(ReadOnly);
/etc/mail                -> $(ReadOnly);

# Postfix Konfiguration
# /etc/postfix           -> $(ReadOnly);

# Red Hat verwendet ein Alternatives System zur Wahl von Sendmail/Postfix
/etc/alternatives        -> $(ReadOnly);

# Imap Konfiguration
/etc/inetd.conf          -> $(ReadOnly); # SUSE
/etc/xinetd.conf         -> $(ReadOnly); # Red Hat
/etc/xinetd.d            -> $(ReadOnly); # Red Hat

# Binärdateien (SUSE, Links bei Red Hat)
/usr/bin/mailq           -> $(ReadOnly);
/usr/bin/newaliases      -> $(ReadOnly);
/usr/sbin/sendmail       -> $(ReadOnly);
# Bei Verwendung des inetd

```

```

#/usr/sbin/inetd          -> $(ReadOnly);
#/usr/sbin/imapd         -> $(ReadOnly);

# Binärdateien (Red Hat)
/usr/bin/mailq.sendmail  -> $(ReadOnly);
/usr/bin/newaliases.sendmail -> $(ReadOnly);
/usr/sbin/sendmail.sendmail -> $(ReadOnly);
# Bei Verwendung des xinetd
/usr/sbin/xinetd         -> $(ReadOnly);
/usr/sbin/imapd         -> $(ReadOnly);

# Protokolldateien
/var/log/maillog        -> $(Growing);
}
@@endif

```

Der Webserver bietet Informationen über die Firma *nohup.info* an. Hierzu wird der Apache-Webserver eingesetzt. Dieser Webserver ist auch in der Lage, den Zugriff auf bestimmte Seiten über SSL zu ermöglichen. Die entsprechenden Regeln für einen PHP-gestützten Webserver können folgendermaßen definiert werden:

```

@@ifhost www
(rulename = www, emailto=www_admin@nohup.info)
{
  # Konfigurationsdateien
  /etc/httpd/conf          -> $(ReadOnly);

  # Binärdateien
  /usr/sbin/httpd         -> $(ReadOnly);
  /usr/sbin/ab            -> $(ReadOnly);
  /usr/sbin/apachectl    -> $(ReadOnly);
  /usr/sbin/suexec        -> $(ReadOnly);
  /usr/sbin/logresolve    -> $(ReadOnly);
  /usr/sbin/rotatelogs    -> $(ReadOnly);
  /usr/bin/checkgid       -> $(ReadOnly);
  /usr/bin/htpasswd       -> $(ReadOnly);
  /usr/bin/htdigest       -> $(ReadOnly);
  /usr/bin/dbmmanage      -> $(ReadOnly);

  # Module
  /usr/lib/apache         -> $(ReadOnly);

  # Web Site
  /usr/local/httpd/htdocs -> $(ReadOnly); # SUSE
  /var/www/               -> $(ReadOnly); # Red Hat
  /www/docs               -> $(ReadOnly); # Generic

  # Protokolle

```

```

    /var/log/httpd          -> $(Growing);
}
@@endif

```

Der DNS-Server ist in der DMZ lokalisiert. Er dient als primärer Master DNS-Server der Auflösung der Zone *nohup.info*. Zusätzlich stellt er für alle weiteren Dienste einen cachenden DNS-Server. Das bedeutet, dass alle weiteren Dienste nicht direkt die Nameserver im Internet kontaktieren, sondern dass diese Aufgabe vom cachenden DNS-Server übernommen wird. Als Beispiel für den DNS-Server dient hier der Bind9. Diese Software ist sicherlich der Standard für DNS-Dienste unter Linux. Der Einsatz anderer Programme wie *djbdns* (<http://cr.yip.to/djbdns.html>) oder MaraDNS (<http://www.maradns.org>) ist jedoch genauso möglich.

Listing 14.3 Tripwire-Regeln für Bind9

```

@@ifhost dns
(rulename=dns, emailto=dns_admin@nohup.info)
{
    # Regeln für Bind9

    # Konfigurationsdateien
    /etc/named.conf          -> $(ReadOnly);
    /etc/rndc.conf          -> $(ReadOnly);
    /etc/rndc.key           -> $(ReadOnly);

    # Binärdateien
    /usr/sbin/dns-keygen    -> $(ReadOnly);
    /usr/sbin/dnssec-keygen -> $(ReadOnly);
    /usr/sbin/dnssec-makekeyset -> $(ReadOnly);
    /usr/sbin/dnssec-signkey -> $(ReadOnly);
    /usr/sbin/dnssec-signzone -> $(ReadOnly);
    /usr/sbin/lwresd       -> $(ReadOnly);
    /usr/sbin/named        -> $(ReadOnly);
    /usr/sbin/named-bootconf -> $(ReadOnly);
    /usr/sbin/named-checkconf -> $(ReadOnly);
    /usr/sbin/named-checkzone -> $(ReadOnly);
    /usr/sbin/rndc         -> $(ReadOnly);
    /usr/sbin/rndc-confgen -> $(ReadOnly);

    # Zonendateien
    /var/named              -> $(ReadOnly);
}
@@endif

```

Der Samba-Server bietet ebenfalls unternehmenskritische Dienste an. Die Sicherheit dieser Dienste und die Unversehrtheit der gespeicherten Daten sollen mit Tripwire überprüft werden. Der Samba-Server bietet verschiedene Verzeichnisse für die

Windows- und Linux-Clients an. Unter anderem werden auch verschiedene Read-Only-Verzeichnisse freigegeben. Diese sollen auch von Tripwire überwacht werden.

Listing 14.4 Tripwire-Regeln für Samba 2.2

```

@@ifhost samba
(rulename=samba, emailto=samba_admin@nohup.info)
{
  # Konfigurationsdateien
  /etc/pam.d/samba          -> $(ReadOnly);
  /etc/samba/               -> $(ReadOnly);
  /lib/security/pam_smbpass.so -> $(ReadOnly);

  # Binärdateien
  /usr/bin/make_unicodemap  -> $(ReadOnly);
  /usr/bin/mksmbpasswd.sh   -> $(ReadOnly);
  /usr/bin/smbadduser       -> $(ReadOnly);
  /usr/bin/smbcontrol       -> $(ReadOnly);
  /usr/bin/smbstatus        -> $(ReadOnly);
  /usr/sbin/nmbd            -> $(ReadOnly);
  /usr/sbin/smbd            -> $(ReadOnly);
  /usr/bin/make_printerdef  -> $(ReadOnly);
  /usr/bin/make_smbcodepage -> $(ReadOnly);
  /usr/bin/smbpasswd        -> $(ReadOnly);
  /usr/bin/testparm         -> $(ReadOnly);
  /usr/bin/testprns         -> $(ReadOnly);
  /usr/bin/wbinfo           -> $(ReadOnly);
  /usr/sbin/winbindd       -> $(ReadOnly);

  # Bibliotheken
  /usr/include/libsmclient.h -> $(ReadOnly);
  /usr/lib/libsmclient.a     -> $(ReadOnly);
  /lib/libnss_winbind.so    -> $(ReadOnly);
  /lib/libnss_winbind.so.2  -> $(ReadOnly);
  /lib/libnss_wins.so       -> $(ReadOnly);
  /lib/libnss_wins.so.2     -> $(ReadOnly);
  /lib/security/pam_winbind.so -> $(ReadOnly);

  # Freigaben
  /share/readonly          -> $(ReadOnly);
}
@@endif

```

Der OpenLDAP-Server wird als zentrale Benutzerverwaltung eingesetzt. Die Sicherheit des Rechners ist daher besonders wichtig. Jedoch können nur die Konfigurationsdateien und die Binärdateien des OpenLDAP-Servers überwacht werden. Die LDAP-Datenbanken ändern sich bei jeder Kennwortänderung durch einen Benutzer.

Listing 14.5 Tripwire-Regeln für OpenLDAP

```

@@ifhost ldap
(rulename=ldap, emailto=ldap_admin@nohup.info)
{
    # Konfigurationsdateien
    /etc/openldap                -> $(ReadOnly);

    # Migration Files
    /usr/share/openldap/migration -> $(ReadOnly);

    # Binärdateien
    /usr/sbin/fax500             -> $(ReadOnly);
    /usr/sbin/go500              -> $(ReadOnly);
    /usr/sbin/go500gw            -> $(ReadOnly);
    /usr/sbin/in.xfingerd        -> $(ReadOnly);
    /usr/sbin/mail500            -> $(ReadOnly);
    /usr/sbin/maildap            -> $(ReadOnly);
    /usr/sbin/rcpt500            -> $(ReadOnly);
    /usr/sbin/rp500              -> $(ReadOnly);
    /usr/sbin/slapadd             -> $(ReadOnly);
    /usr/sbin/slapadd-gdbm        -> $(ReadOnly);
    /usr/sbin/slapcat            -> $(ReadOnly);
    /usr/sbin/slapcat-gdbm        -> $(ReadOnly);
    /usr/sbin/slapd              -> $(ReadOnly);
    /usr/sbin/slapindex          -> $(ReadOnly);
    /usr/sbin/slappasswd          -> $(ReadOnly);
    /usr/sbin/slurpd              -> $(ReadOnly);
    /usr/sbin/xrpscomp           -> $(ReadOnly);

    # LDAP Datenbanken (führt zu Problemen bei Kennwortänderungen)
    /var/lib/ldap                 -> $(ReadOnly);
}
@@endif

```

Diese Regeln erlauben nun die Erzeugung der Konfigurationsdateien für alle Rechner basierend auf einem einheitlichen zentralen Regelsatz.

14.2 Zentrale Verwaltung der Schlüssel

Sinnvollerweise werden die Klartextvarianten der Konfigurationsdateien auf dem zentralen Überwachungsrechner vorgehalten. Auf diesem Rechner können auch die binären verschlüsselten Versionen der Konfigurationsdateien erzeugt werden. Dazu muss zunächst der Site-Key für Tripwire generiert werden. Hier genügt ein Site-Key für das gesamte Netzwerk.

```
# twadmin --generate-keys --site-keyfile /etc/tripwire/site.key
# twadmin --create-cfgfile -c /etc/tripwire/tw.cfg /etc/tripwire/twcfg.txt
# twadmin --create-polfile -p /etc/tripwire/tw.pol /etc/tripwire/twpol.txt
```

Schließlich können auch die Schlüssel Local-Key für die einzelnen zu überwachen den Rechner erzeugt werden. Hierbei sollte für jeden Rechner ein individueller Schlüssel mit einer individuellen Passphrase erzeugt werden. Die Passphrasen sollten nicht auf einem Rechner abgespeichert, sondern entsprechend der Unternehmenspolitik verwaltet werden. Dies kann zum Beispiel bedeuten, dass diese Passphrasen niedergeschrieben und in einem Safe aufbewahrt werden.

```
# twadmin --generate-keys --local-keyfile /etc/tripwire/dns-local.key
# twadmin --generate-keys --local-keyfile /etc/tripwire/squid-local.key
# twadmin --generate-keys ...
```

Die so erzeugten Dateien und Passphrasen müssen nun noch auf die zu überwachen den Rechner verteilt werden.

14.3 Distribution der Dateien mit *rsync*

Für die Distribution der Dateien stehen unter UNIX/Linux verschiedene Werkzeuge zur Verfügung. Eines der intelligentesten Werkzeuge ist *rsync*. *rsync* verbindet sich ähnlich dem Befehl *rcp* mit einem anderen Rechner und tauscht Dateien aus. Es ist jedoch in der Lage, nur die veränderten Teile einer Datei zu erkennen und zu übertragen. Hierzu werden die modifizierten Anteile mit einer MD4-Prüfsumme erkannt. Dies reduziert die zu übertragende Datenmenge erheblich und erlaubt eine sehr einfache und schnelle Verteilung der Dateien.

rsync ist Bestandteil der so genannten *r-tools*. Diese Werkzeuge (*rsh*, *rlogin*, *rcp* und *rexec*) wurden entwickelt, um die Administration vieler Rechner zu vereinfachen. Hierzu erlauben diese Werkzeuge mit geeigneten Konfigurationsdateien die Anmeldung auf entfernten Rechnern, ohne eine erneute Eingabe von Kennworten durchzuführen. Die Konfigurationsdateien (*rhosts*) enthalten dazu Vertrauensstellungen. Alle *r-tools* werden heute als unsicher angesehen, da die komplette Übertragung in Klartext erfolgt und die Vertrauensstellungen keinerlei Authentifizierung erfordern.

Die *r-tools* wurden daher weitgehend durch die Secure Shell (*ssh* und *scp*) ersetzt. Diese Werkzeuge erlauben die Authentifizierung von Rechnern und Benutzern mit starken öffentlichen Schlüsseln. Dennoch kann für den Einsatz in Skripten die automatische Anmeldung auf einem anderen Rechner ohne die interaktive Eingabe eines Kennwortes ermöglicht werden.

rsync ist in der Lage, die Synchronisation auch basierend auf der Secure Shell durchzuführen. Die Authentifizierung und die Übertragung der Dateien auf die entfernten Rechner erfolgt dann stark verschlüsselt.



Exkurs: Authentifizierung bei der Secure Shell

In diesem kleinen Exkurs soll kurz die Funktionsweise der Authentifizierung bei der Secure Shell vorgestellt werden.

Wenn ein Client eine SSH-Verbindung aufbauen möchte, so müssen im Vorfeld bestimmte Schlüssel erzeugt und ausgetauscht worden sein.

Der SSH-Server (Version 2) erzeugt bei seinem Start ein DSA-Schlüsselpaar. Dieses identifiziert den Server eindeutig. Der Server legt diese Schlüssel im Verzeichnis `/etc/ssh` als `ssh_host_dsa_key.pub` (öffentlicher Schlüssel) und `ssh_host_dsa_key` (privater Schlüssel) ab. Der öffentliche Schlüssel muss nun allen Clients bekannt gemacht werden. Dazu kann er auf eine Diskette kopiert und bei den entsprechenden Benutzern, die sich mit dem SSH-Server verbinden wollen, in der Datei `~/.ssh/known_hosts2` abgelegt werden (2 für Protokoll/Version 2). Diese Datei enthält alle öffentlichen SSH-Server-Schlüssel, mit denen sich der Benutzer verbinden möchte.

Zusätzlich erzeugt der Benutzer ein DSA-Schlüsselpaar. Die Erzeugung dieser Schlüssel erfolgt bei der *OpenSSH* mit dem Befehl

```
# ssh-keygen -t dsa
```

Dieser Befehl legt die Schlüssel üblicherweise unter `~/.ssh/id_dsa.pub` (öffentlicher Schlüssel) und `~/.ssh/id_dsa` (privater Schlüssel) ab. Der private Schlüssel kann zusätzlich mit einer Passphrase geschützt werden. Dies erhöht die Sicherheit, da ansonsten der private Schlüssel in Klartext gespeichert wird und sich zum Beispiel auch auf jeder Sicherung dieses Rechners im Klartext befindet. Der öffentliche Schlüssel muss nun auf allen Servern, auf denen sich der Benutzer anmelden darf, an die Datei `authorized_keys2` des Benutzers, als der die Anmeldung erfolgen soll, angehängt werden. Die Datei `authorized_keys2` eines Benutzers Otto enthält also alle öffentlichen Schlüssel aller anderen Benutzer, die sich als Otto anmelden dürfen. Der Benutzer Otto kann also selbst entscheiden, welche weiteren Benutzer sich unter seinem Namen anmelden dürfen, da er die Datei `authorized_keys2` verwaltet.

Bei der Authentifizierung verlangt der Client nun zunächst die Authentifizierung des Servers. Hierzu sendet er eine Herausforderung (challenge) an den Server. Dabei handelt es sich um eine sehr große Zufallszahl. Der Server muss nun diese Zahl verschlüsseln und zurücksenden. Besitzt der Client den öffentlichen Schlüssel des Servers und wurden die Schlüssel auf dem Server nicht verändert, so kann der Client die korrekte Verschlüsselung durch die Entschlüsselung prüfen. Erhält er dabei nicht das erwartete Ergebnis, so wurden ent-

weder die Pakete beim Transport modifiziert, die Schlüssel auf dem Server verändert oder es handelt sich nicht um den echten Server, sondern um einen Man-in-the-Middle-Angriff.

Anschließend verlangt der Server die Authentifizierung des Clients. Hierbei sendet der Server dem Client die Herausforderung, die nun vom Client verschlüsselt wird. Der Server überprüft auf die gleiche Art die Korrektheit durch die Entschlüsselung.

Die Verwendung von Zufallszahlen als Herausforderung verhindert einen Angriff, bei dem der Angreifer den Verkehr abhört und anschließend selbst eine Anmeldung mit zuvor abgehörten Paketen versucht. Der Angreifer wird jedes Mal eine neue zuvor nicht gesehene Zufallszahl angeboten bekommen.

Wurde der private Schlüssel des Benutzers mit einer Passphrase gesichert, so besteht keine Gefahr, wenn eine dritte Person Zugang zu der Datei erhält, in der dieser Schlüssel abgespeichert wurde. Leider erfordert dies jedoch bei jeder Verbindung die Eingabe der Passphrase.

Um die Sicherheit der Schlüssel mit einer Passphrase zu gewährleisten, dennoch aber eine bequeme Benutzung der Secure Shell auch mit Scripts zu ermöglichen, wurde der *ssh-agent* erzeugt. Die Schlüssel werden weiterhin mit einer Passphrase auf dem Dateisystem geschützt. Für den Gebrauch werden sie einmal mit dem Befehl `ssh-add` geladen und die Passphrase eingegeben. Anschließend liegen sie im Arbeitsspeicher entschlüsselt vor. Ein Einbrecher kann weiterhin nur die durch eine Passphrase geschützten Schlüssel entwenden. Sicherlich ist er in der Lage, die im Arbeitsspeicher vorhandenen Schlüssel zu nutzen; dies jedoch nur so lange, wie der *ssh-agent* läuft. Nach einem Neustart sind die Informationen verloren. Ein Auslesen aus dem Arbeitsspeicher ist sehr aufwändig.

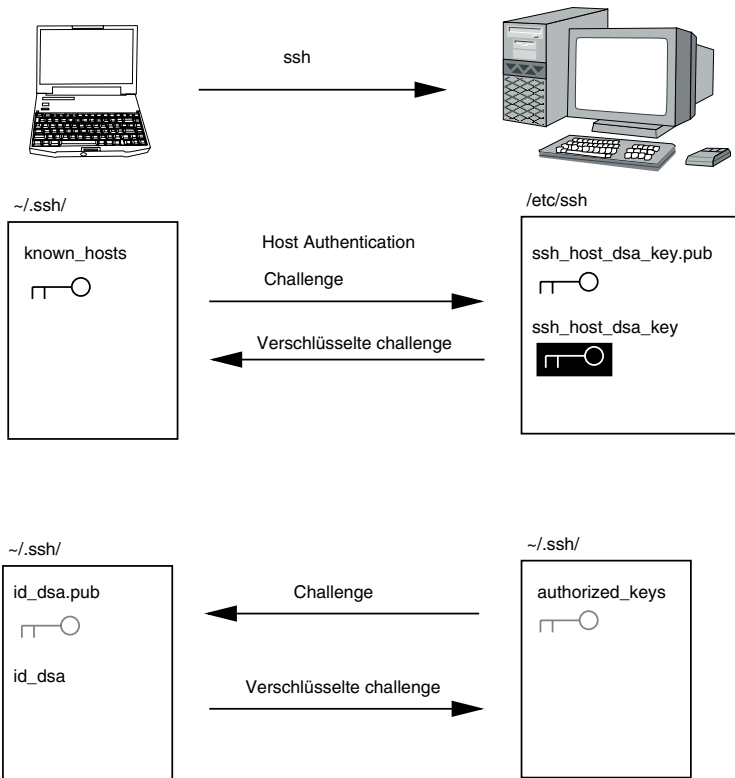


Abbildung 14.1 SSH-Authentifizierung

Um nun die Verteilung der Dateien mit der Kombination `rsync` und `ssh` durchzuführen, muss zunächst auf dem zentralen Verwaltungsrechner ein Benutzer erzeugt werden, der diese Arbeiten durchführen soll. Anschließend sollte dieser Benutzer entsprechende DSA-Schlüssel erzeugen und mit einer Passphrase sichern.

```
# useradd distributor
# su - distributor
$ ssh-keygen -t dsa
```

Der entsprechende öffentliche Schlüssel wird anschließend auf allen mit Tripwire überwachenden Rechnern als autorisierter Schlüssel für eine Anmeldung als `root` in der Datei `/root/.ssh/authorized_keys2` eingetragen.

Wurden zuvor die zu verteilenden Konfigurationsdateien und Tripwire-Schlüssel im Verzeichnis `/home/distributor` angelegt, so kann die Verteilung erfolgen mit dem Script:

```
#!/bin/bash
```

```

for remote in squid dns www relay
do
  echo "Bearbeite Rechner: $remote"
  rsync -e ssh -v /home/distributor/tw.cfg \
              /home/distributor/tw.pol \
              /home/distributor/site.key \
              /home/distributor/$remote-local.key \
              root@$remote:/etc/tripwire
done

```

Dieses Script wird nun auf alle in der *for*-Schleife angegebenen Rechnernamen verteilt. Dabei wird bei jeder Verbindung die Passphrase für den privaten Schlüssel angefordert. So kann verhindert werden, dass durch einen Einbruch auf dem zentralen Verwaltungsrechner die Schlüssel kompromittiert werden.

14.4 Erzeugung und Überprüfung der Datenbanken mit ssh

Nachdem die Konfigurationsdateien und die Tripwire-Schlüssel verteilt wurden, müssen zunächst die Tripwire-Datenbanken initialisiert werden.

Die Initialisierung der Datenbanken kann ebenfalls aus der Ferne mit der Secure Shell durchgeführt werden. Die Datenbanken sollten anschließend auch auf der zentralen Tripwire-Managementstation gesichert werden. Hierzu kann das folgende Script genutzt werden:

```

#!/bin/bash

for remote in squid dns www relay
do
  echo "Erzeuge Tripwire Datenbank auf Rechner: $remote"
  ssh root@$remote tripwire --init
  scp root@$remote:/var/lib/tripwire/$remote.twd \
      /var/lib/tripwire/$remote.twd
done

```

Dieses Script kann nicht unbeaufsichtigt ablaufen, da es sowohl die Passphrase für den privaten SSH-Schlüssel benötigt als auch die für den lokalen Tripwire-Schlüssel des Zielrechners. Die Abfrage der SSH-Passphrase lässt sich bei Einsatz des *ssh-agent* vermeiden.

Sobald die Datenbanken erzeugt wurden, kann die Integrität der zu überwachenden Rechner geprüft werden. Dies erfolgt sinnvollerweise ebenfalls zentral gesteuert. So kann ihre erfolgreiche Ausführung direkt zentral kontrolliert werden.

Für die Überprüfung der Integrität wird auf den zu überwachenden Rechnern ein weiterer Benutzer *tripwire* angelegt. Dieser Benutzer erhält das Recht, mit *root*-Rech-

ten den Befehl *tripwire --check* auszuführen. Dieses Recht erhält er bei Verwendung des Kommandos `sudo`.

Um den Benutzer *tripwire* anzulegen, führen Sie bitte auf den zu überwachenden Rechnern den Befehl `useradd tripwire` aus oder verwenden Sie auf dem zentralen Verwaltungsrechner folgendes Script:

```
#!/bin/bash

for remote in squid dns www relay
do
    echo "Erzeuge Tripwire Benutzer auf Rechner: $remote"
    ssh root@$remote useradd tripwire
done
```

Anschließend muss die `sudo`-Konfigurationsdatei `/etc/sudoers` angepasst werden, indem folgende Zeile hinzugefügt wird:

```
tripwire ALL = (root) NOPASSWD: /usr/sbin/tripwire --check
```

Dieser Eintrag in der Datei `/etc/sudoers` erlaubt dem Benutzer *tripwire* den Aufruf von `/usr/sbin/tripwire --check` ohne Eingabe eines Kennwortes mit `root`-Rechten. Dies ist jedoch der einzige Befehl, den *tripwire* als `root` ausführen darf.

Sinnvollerweise wird nun mit *ssh-keygen* ein zweites Schlüsselpaar erzeugt, welches entweder keine Passphrase aufweist oder dessen Passphrase mit dem *ssh-agent* verwaltet wird:

```
[root@kermit root]# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa): /root/.ssh/tripwire
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/tripwire.
Your public key has been saved in /root/.ssh/tripwire.pub.
The key fingerprint is:
cc:df:00:b1:52:52:2b:29:c7:59:58:9a:c0:1d:b0:ee root@kermit.spenneberg.de
```

Der öffentliche Schlüssel *tripwire.pub* wird auf den zu verwaltenden Rechnern in der Datei `/home/tripwire/.ssh/authorized_keys2` abgelegt. Damit ist es nun `root` möglich, mit diesem zweiten Schlüssel sich als *tripwire* auf den entfernten Rechnern anzumelden. Die Überprüfung der Datenbanken kann nun mit folgendem Script erfolgen:

```
#!/bin/bash

for remote in squid dns www relay
do
    echo "Überprüfe Tripwire Datenbank auf Rechner: $remote"
```

```
ssh -i /root/.ssh/tripwire tripwire@$remote sudo tripwire --check
done
```

Die Option `-i /root/.ssh/tripwire` verwendet statt des Standardschlüssels den angegebenen Schlüssel für die Anmeldung. So erfolgt die Anmeldung mit dem Schlüssel `tripwire` als der Benutzer `tripwire`, der anschließend nur den einen Befehl `tripwire --check` ausführt. Dieses Script sollte nun auf dem zentralen Verwaltungsrechner regelmäßig ausgeführt werden. Dazu wird es sinnvollerweise in Cron integriert. Der Aufruf kann stündlich bis täglich erfolgen. Dies hängt von der zu überprüfenden Datenmenge, der Leistungsfähigkeit der Rechner und dem persönlichen Sicherheitsbedürfnis bzw. der Sicherheitsrichtlinie des Unternehmens ab. Bei einem sehr häufigen Aufruf sollte darauf geachtet werden, dass die Aufrufe sich nicht überschneiden!

14.5 Zentrale Auswertung

Wenn der Tripwire-Integritätstest der Rechner zentral gesteuert wird, ist es auch sinnvoll, die Auswertung zentral vorzunehmen und die Berichte zentral abzuspeichern. Dazu sollten die Berichte direkt nach ihrer Erzeugung auf den zentralen Rechner übertragen werden. Dort können sie dann weiter analysiert werden und die Berichte ausgedruckt werden.

Am sinnvollsten werden die Berichte direkt nach ihrer Erzeugung übertragen. So besteht die geringste Gefahr einer Modifikation der Berichte durch einen Einbrecher. Hierzu wird geeigneterweise das Script, welches die Dateien erzeugt, ergänzt:

```
#!/bin/bash

for remote in squid dns www relay
do
    echo "Überprüfe Tripwire Datenbank auf Rechner: $remote"
    ssh -i /root/.ssh/tripwire tripwire@$remote sudo tripwire --check
    scp -i /root/.ssh/tripwire \
        tripwire@$remote:/var/lib/tripwire/report/$remote.twr \
        /var/lib/tripwire/reports/$remote/$remote-$(date +%Y%m%d-%H).twr
done
```

Mit dem an die Überprüfung anschließenden Befehl `scp` wird die gerade erzeugte Berichtsdatei `$remote.twr` auf die zentrale Verwaltungsstation kopiert. Dort wird der Bericht abgelegt unter `/var/lib/tripwire/reports`. Hier wurde für jeden Rechner ein Verzeichnis entsprechend seinem Namen angelegt und der Bericht in diesem Verzeichnis mit Datum und Stunde abgespeichert. Wird die Überprüfung häufiger als einmal pro Stunde aufgerufen, so ist zusätzlich die Angabe der Minute erforderlich.

Die Ausgabe des Berichtes und die Versendung kann nun mit dem Befehl `twprint` erfolgen. Hierzu kann ebenfalls ein Script verwendet werden, das eine Zusammenfassung sämtlicher Berichte anfertigt:

```
#!/bin/bash

DETAIL=0 # Detailgrad 0-5

for remote in squid dns www relay
do
  echo "Erzeuge Bericht für Rechner: $remote"
  twprint -m r -t $DETAIL -r \
    /var/lib/tripwire/reports/$remote/$remote-$(date +%Y%m%d-%H).twr
done | mail tripwire@nohup.info -s "Berichtszusammenfassung"
```

Wurden Unterschiede beim Tripwire-Bericht gemeldet, so können diese auch auf der zentralen Tripwire-Managementstation zur Aktualisierung der Datenbank verwendet werden. Hierzu muss der Befehl `tripwire --update` unter Angabe aller benötigten Dateien aufgerufen werden. Dies erfolgt ebenfalls am besten mit einem Script. Das folgende Script erwartet die Angabe des Rechners, dessen Datenbank aktualisiert werden soll, und wählt automatisch den letzten Bericht aus:

```
#!/bin/bash

remote=$(1:?Fehler: Bitte als $0 rechner aufrufen)

bericht=$(ls -t /var/lib/tripwire/reports/$remote/ | head -1)

echo "Aktualisiere Datenbank für Rechner: $remote"
echo "Verwende Bericht: $bericht"
echo "Fortfahren? <ENTER>"
tripwire --update --cfgfile /home/distributor/tw.cfg \
  --polfile /home/distributor/tw.pol \
  --database /var/lib/tripwire/$remote.twd \
  --local-keyfile /home/distributor/$remote-local.key \
  --site-keyfile /home/distributor/site.key \
  --twrfile $/var/lib/tripwire/reports/$remote/$bericht

echo "Soll die Datenbank direkt wieder auf den Rechner $remote übertragen
↳ werden?"
echo "ja? <ENTER> nein? <Strg-C>"
rsync -e ssh /var/lib/tripwire/$remote.twd
↳ root@$remote:/var/lib/tripwire/$remote.twd
echo "Fertig."
```

Das letzte Script erlaubt es, die Datenbank offline zu aktualisieren und anschließend auf den betroffenen Rechner zurückzuspielen. Dies erfolgt wieder mit dem Befehl `rsync` und der Secure Shell. Dies reduziert die zu übertragende Datenmenge und beschleunigt den Transfer.

14.6 Zusammenfassung

Die zentrale Verwaltung des Intrusion-Detection-Systems Tripwire lohnt sich bereits ab einer geringen Anzahl von Installationen. Selbst bei zwei Systemen kann es sinnvoll sein. Hierzu sind auch kommerzielle Lösungen wie zum Beispiel der Tripwire-Manager erhältlich, der zusätzlich eine sehr beeindruckende grafische Oberfläche für die Administration bietet. Die in diesem Kapitel vorgestellten Techniken und Werkzeuge ermöglichen jedoch bereits eine sehr sichere und einfache Fernadministration einer beliebigen Anzahl von Tripwire-Installationen. Zusätzliche Sicherheitspunkte wie die Sicherung der Datenbanken und Berichte kommen als weitere Pluspunkte hinzu. Die gezeigten Bash Shell-Scripts zeigen recht eindrucksvoll, dass viele Aufgaben durch intelligenten Einsatz von Linux-Hausmitteln gelöst werden können. Wenn das Interface der Shell-Scripts zu einfach und bedienungsunfreundlich erscheint, besteht auch die Möglichkeit, die Funktionalität in PHP (oder Perl) einzubetten und über einen Webserver mit Browser zu steuern.