

HOWTO zu "tripwire"

(C) 2006 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
OSTC GmbH, <http://www.ostc.de>
\$Id: tripwire-HOWTO.txt,v 1.9 2011-08-25 15:15:21 tsbirn Exp \$

Dieses Dokument beschreibt die Absicherung eines Linux-Systems mittels tripwire.

Inhaltsverzeichnis

- 1) Einführung
- 2) Übersicht zum Ablauf einer Tripwire-Konfiguration und -Anwendung
 - 2.1) Sicherheit
 - 2.2) Eigenschaften
 - 2.3) Installation und Grund-Konfiguration
 - 2.4) Richtlinien (Policies) definieren
- 3) Durchführung einer Konfiguration und mehrerer Prüfungen
- 4) man-Pages

1) Einführung

Tripwire ist ein "Host Based Intrusion Detection System" (IDS) oder besser ein "System Integrity Verifier". Es erstellt eine Momentaufnahme über den Zustand eines Systems anhand von "Richtlinien" (Policies). Diese wählen Verzeichnissen und Dateien des Dateisystems eines Rechners aus und zeichnen bestimmte Eigenschaften von ihnen (Größe, Besitzer, Zugriffrechte, Alter, ...) in einer Datenbank auf.

Die "Richtlinien" (Policies) müssen sorgfältig definiert werden, da später nur genau diese Eigenschaften geprüft werden. Zu strenge/laxe oder zu viele/wenige Vorgaben führen zur Unbrauchbarkeit der generierten Datenbank.

Weiterhin sollte sich ein per Tripwire abzusicherndes System im "sauberen Zustand" befinden, damit die Datenbank nicht den Zustand eines Systems im bereits kompromittierten Zustand aufzeichnet (die Datenbank wäre dann wertlos):

- * Gerade frisch installiert
- * Installations-Daten aus vertrauenswürdiger Quelle (CD, DVD, Signatur)
- * Noch keine Netzwerkverbindung eingerichtet / bestanden

Später kann Tripwire den AKTUELLEN Dateisystem-Zustand gegen den in der Datenbank abgelegten prüfen. Bei Abweichungen generiert es einen Bericht (Report), aufgrund dessen der Administrator entscheiden kann, von welchem Typ die Abweichung ist:

- * Gewollt
- * Unbefugt

D.h. es erkennt einen Angriff "indirekt", nämlich NICHT online, sondern erst später an seinen Auswirkungen auf das Dateisystem.

Werden Abweichungen festgestellt, die durch Aktualisierung oder Aufspielen von Software-Paketen bedingt sind, können diese Änderungen als neuer Ist-Zustand in die Datenbank aufgenommen werden, damit diese Änderungen nicht ständig "reported" werden (spart Zeit gegenüber Neu-Generierung und erlaubt eine feingranulare Kontrolle).

Die für Tripwire notwendigen ASCII-Konfigurationsdateien werden nach ihrer Definition in ein binäres Format umgewandelt und mit einem "Site Key" signiert (kryptografisch gesichert), das bringt Performanz und Sicherheit.

Die von Tripwire generierte Datenbank hat ebenfalls ein binäres Format und wird durch einen "Local Key" signiert (kryptografisch gesichert), das bringt Performanz und Sicherheit.

Die beiden Keys (Schlüssel) müssen einmal ganz am Anfang der Einrichtung von Tripwire mit "twadmin" erzeugt werden. Sie sollten mit einer "Passphrase" gesichert werden, damit sie nicht im Klartext auf dem Dateisystem liegen. Die Trennung dieser beiden Schlüssel erlaubt eine Trennung der Aufgaben:

- * Eine Person definiert die zu prüfenden Richtlinien
- * Jemand anderes führt die eigentliche Prüfung des Dateisystems durch

2) Übersicht zum Ablauf einer Tripwire-Konfiguration und -Anwendung

```

+-----+
| E | 0 | Rechnernamen ermitteln |

```

I	1	Lokal Key "HOST-local.key" generieren (mit Site Passphrase)
N	2	Site Key "site.key" generieren (mit Local Passphrase)

R		
I	3	ASCII-Konfigurations-Datei "twcfg.txt" erstellen (Editor)
C	4	Binäre Konfigurations-Datei "tw.cfg" generieren (Site Passphrase)

H		
T	5	ASCII-Richtlinien-Datei "twpol.txt" erstellen (Editor)
E	6	Binäre Richtlinien-Datei "twpol.pol" generieren (Site Passphrase)

N		
	7	Binäre Datenbank "HOST.twd" generieren (Local Passphrase)

	8	Durch Prüfen Bericht generieren

A		
N	9	Änderung am Dateisystem vornehmen
W		Prüfen
E		Akzeptierte Änderungen in DB nachtragen (Editor, Local Passphrase)
N		Prüfen

D		
E	10	Täglich per Cronjob einen Bericht generieren lassen
N	11	Geänderte Richtlinien in Datenbank einpflegen

2.1) Sicherheit

Die Klartext-Versionen der beiden Konfigurations-Dateien "twcfg.txt" und "twpol.txt" sollten nach der Generierung der binären Variante an einen sicheren Ort verschoben werden, damit Angreifer aus den darin stehenden Informationen keinen Nutzen ziehen können (z.B. nicht überwacht Verzeichnis erkennen).

Weiterhin sollte eine Kopie der generierten Datenbank "HOST.twd" auf ein Read-Only-Medium gemacht werden, damit der Angreifer sie nicht löschen kann.

Tripwire kann sogenannte "Rootkits" (z.B. lrk3, lrk4, lrk5) entdecken, die einige wichtige Systembefehle gegen eigene Versionen austauschen. Bei "Kernelbasierten Rootkits" (z.B. Knark), die Systemaufrufe austauschen, hilft Tripwire allerdings nicht mehr, da der Kern eine andere Struktur und einen anderen Inhalt des Dateisystems vorspiegeln kann, als in Wirklichkeit vorliegt.

Hier helfen dann andere Tools wie "kstat" zur Analyse des Arbeitsspeichers. Sollte auch der Kern durch "Kernel-Memory-Patching" unterwandert sein (z.B. KIS), dann hilft ein System wie "LIDS" (Linux Intrusion Detection System) weiter.

2.2) Eigenschaften

Von Tripwire kann pro Datei/Verzeichnis eine beliebige Auswahl folgender Eigenschaften ("Properties") überprüft werden (wenn z.B. die Datei wachsen darf, wird die Größe nicht geprüft):

Code	Begriff	Eigenschaft
-	remove	Folgende Eigenschaften ignorieren
+	add	Folgende Eigenschaften prüfen

a	access	Zugriffsdatum
b	blocks	Anzahl Datenblöcke
c	create/modify	Inode-Änderungsdatum
d	device	Disk Device Nummer
g	group	Besitzer-Gruppe
i	inode	Inode-Nummer
l	length	Datei darf wachsen (Verzeichnis?)
m	modify	Änderungsdatum
n	number	Anzahl Hard-Links
p	permission	Zugriffsrechte und Dateimodus
r	major/minor	Device Nummer
s	size	Dateigröße
t	type	Dateityp
u	user	Besitzer

An Prüfsummen-Typen sind möglich (sortiert nach steigendem Aufwand und zunehmender Sicherheit, mind. 2 Prüfsummen sollten generiert werden):

Code	Eigenschaft
C	CRC-32 Hashwert
M	MD5 Hashwert
S	SHA Hashwert
H	Haval Signatur

In der Datei "twpol.txt" werden über "Richtlinien" (Regeln, Policies) beliebig viele Dateien und Verzeichnisse mit den zu prüfenden Eigenschaften verknüpft.

Eine Richtlinie kann mit einem "Namen" und einer "Priorität" (Wichtigkeitsgrad) versehen werden. Prüfungen können dann auf Richtlinien mit bestimmten Namen oder Prioritäten beschränkt werden.

Häufig benötigte Kombinationen sind über vordefinierte Variable verfügbar:

```
ReadOnly    = +pinugtsdbmCM-rlacSH    # Für viele verfügbar, aber nur lesbar
Dynamic     = +pinugtd-srlbamcCMSH    # Dynamik erlaubt
Growing     = +pinugtdl-srbamcCMSH    # Soll nur wachsen
Device      = +pugsdr-intlbamcCMSH    # Gerät (soll nicht geöffnet werden)
IgnoreAll   = -pinugtsdrlbamcCMSH    # Nur Vorhandensein/Abwesenheit prüfen
IgnoreNone  = +pinugtsdrbamcCMSH-1   # Startpunkt für eigene Regeln
```

2.3) Installation und Grund-Konfiguration

Tripwire wird bei den meisten Distributionen mitgeliefert. Prüfung, ob Tripwire installiert ist durch (q=query, a=all):

```
rpm -qa | grep tripwire      # -> tripwire-2.3.1-186
dpkg --list | grep tripwire  # -> tripwire 2.3.1.2.0-11
```

Alle von Tripwire installierten Dateien auflisten (q=query):

```
rpm -q --list tripwire
dpkg --listfiles tripwire
```

Ergibt folgende Verzeichnisse bzw. Dateien in folgenden Verzeichnissen:

```
/etc/tripwire/*
/usr/sbin/*
/usr/lib/tripwire/databases
/usr/share/doc/packages/tripwire/*
/usr/share/man/*
/usr/share/lintian/*
/var/lib/tripwire
/var/lib/tripwire/report
```

Für Tripwire müssen folgende Dateien verwaltet werden:

Typ	Zweck + Dateien + man-Pages
Key	Schlüssel zur Absicherung der Datenbank /etc/tripwire/site.key /etc/tripwire/HOST-local.key
Configuration	System-spezifische Information /etc/tripwire/twcfg.txt -> /etc/tripwire/tw.cfg man twconfig
Policy	Richtlinien /etc/tripwire/twpol.txt -> /etc/tripwire/twpol.pol man twpolicy
Database	Generierte Datenbank /var/lib/HOST.twd man tripwire
Report	Generierte Berichte /var/lib/tripwire/report/HOST-DATE.twr man twprint

Nach der Installation von Tripwire entsteht ein Konfigurations-Verzeichnis

```
/etc/tripwire
```

mit den 2 Konfigurationsdateien

```
twcfg.txt      # Grundkonfiguration (Ort von Verz. + Dateien)
twpol.txt      # Richtlinien (Systemdaten + Eigenschaften)
```

Die Datei "twcfg.txt" muss normalerweise nicht geändert werden, sie enthält die Pfade für die restlichen von Tripwire benötigten oder angelegten Dateien:

```
POLFILE        = /etc/tripwire/twpol.pol
DBFILE         = /var/lib/tripwire/$(HOSTNAME).twd
REPORTFILE     = /var/lib/tripwire/report/$(HOSTNAME)-$(DATE).twr
SITEKEYFILE    = /etc/tripwire/site.key
LOCALKEYFILE   = /etc/tripwire/$(HOSTNAME)-local.key
```

2.4) Richtlinien (Policies) definieren

 Richtlinien haben folgende Form:

```
OBJEKT -> EIGENSCHAFTEN; # Strichpunkt NICHT vergessen!
```

Soll eine Datei oder ein Verzeichnis NICHT geprüft werden, ist eine "Stop-Regel" anzugeben:

```
!OBJEKT; # Strichpunkt NICHT vergessen!
```

In OBJEKT werden die zu prüfenden Dateien und Verzeichnisse aufgelistet, pro Objekt darf NUR EINE Richtlinie vorgegeben werden (wird geprüft). Objekte müssen ABSOLUTE PFADE zu Dateien und Verzeichnissen sein, z.B.:

```
/etc
/usr/bin
/sbin/init
```

In EIGENSCHAFTEN werden die weiter oben aufgelisteten Kennbuchstaben (Codes) für Eigenschaften und Prüfsummen-Typ aufgezählt. Häufig benötigte Kombinationen sind über vordefinierte Variable verfügbar:

```
ReadOnly    = +pinugtsdbmCM-rlacSH # Für viele verfügbar, aber nur lesbar
Dynamic     = +pinugd-srlbamcCMSH # Dynamik erlaubt
Growing     = +pinugtdl-srbamcCMSH # Soll nur wachsen
Device      = +pugsdr-intlbamcCMSH # Gerät (soll nicht geöffnet werden)
IgnoreAll   = -pinugtsdrlbamcCMSH # Nur Vorhandensein/Abwesenheit prüfen
IgnoreNone  = +pinugtsdrbamcCMSH-1 # Startpunkt für eigene Regeln
```

Eigene Variablen für Objekte und Eigenschaften können jederzeit definiert und anschließend eingesetzt werden:

```
VAR = WERT; # Definition (Strichpunkt NICHT vergessen!)
... $(VAR) ... # Verwendung ("$" und "(...)" NICHT vergessen!)
```

Beispiel für Variablendefinition (GROSS/kleinschreibung zählt):

```
param1 = +CMSH; # Variable "param1" definieren
dir1 = /etc/inet; # 1. Variable "dir1" definieren
DIR1 = /etc/init.d; # 2. Variable "DIR1" definieren
```

Beispiel für Variablenverwendung:

```
$(dir1) -> +tbamc; # Links eine Variable
/etc/inet -> $(param1); # Rechts eine Variable
$(DIR1) -> $(param1); # Links und rechts eine Variable
```

Beispiele für Richtlinien (Strichpunkt nicht vergessen!):

```
# Ganzen Dateibaum "/bin" prüfen (Vorhandensein + Readonly + ...)
/bin -> $(ReadOnly);

# Ganzen Dateibaum "/etc" prüfen (nur Vorhandensein!)
/etc -> $(IgnoreAll);

# AUSNAHME: Datei "/etc/hostname.hme0" vollständig prüfen
# (ausser access date und major/minor-device number)
/etc/hostname.hme0 -> $(IgnoreNone) -ar;

# AUSNAHME: Datei "/etc/passwd" genauer prüfen
/etc/passwd -> $(Growing);

# AUSNAHME: Dateibaum "/etc/init.d" + "/etc/rc.d" und
# Datei "/etc/mnttab" NICHT prüfen
!/etc/init.d;
!/etc/rc.d;
!/etc/mnttab;
```

Richtlinien können auf 2 Arten auch mit "Attributen" versehen werden, die Name, Wichtigkeit, Verschachtelung und Wirkung festlegen.

* Für eine einzelne Richtlinie (Strichpunkt NICHT vergessen!):

```
OBJEKT -> EIGENSCHAFTEN (ATTRIBUT = WERT, ...);
```

* Für Gruppen von Richtlinien (Strichpunkt NICHT vergessen!):

```
(ATTRIBUT = WERT, ...) {
  OBJEKT -> EIGENSCHAFTEN;
  OBJEKT -> EIGENSCHAFTEN;
  ...
}
```

}

Folgende 4 Attribute sind möglich:

Attribut	Bedeutung
emailto	eMail-Empfänger bei verletzter Regel
rulename	Bericht-Auswahl + Bericht-Output
severity	Bericht-Auswahl 0..1000000 (Default 0)
recurse	Bis zu welcher Verzeichnistiefe untersuchen true (-1) = Komplett (Default) false (0) = Nur Verzeichniseintrag selbst 1..1000000 = Rekursionstiefe

Beispiele für Richtlinien mit Attributen:

```
/usr/lib -> $(ReadOnly) (emailto = admin@foo.com, severity = 80);

(emailto = admin@foo.com, severity = 80)
{
  /bin      -> $(ReadOnly);
  /sbin    -> $(ReadOnly);
  /lib     -> $(ReadOnly);
  /usr/bin -> $(ReadOnly);
  /usr/sbin-> $(ReadOnly);
  /usr/lib -> $(ReadOnly);
}
```

Musterbeispiele für Richtlinien-Dateien findet man hier:

```
/usr/share/doc/packages/tripwire/policyguide.txt
/usr/share/doc/packages/tripwire/twpol.txt
```

3) Durchführung einer Konfiguration und mehrerer Prüfungen

Vorgehen gemäß folgender Datei (zur Geschwindigkeitssteigerung und Vermeidung von Tippfehlern die Befehle mit der Maus aus dieser Datei kopieren!):

```
/usr/share/doc/packages/tripwire/README.SuSE
```

0. Rechnernamen ermitteln:

```
echo $HOSTNAME      # -> z.B. "linux"
hostname            # -> z.B. "linux"
```

1. Lokalen Key erzeugen (1. Passphrase eingeben und merken!) und anzeigen (Binärformat):

```
twadmin --generate-keys -L /etc/tripwire/linux-local.key
ls -l /etc/tripwire/linux-local.key
```

2. Site Key erzeugen (2. Passphrase eingeben und merken!) und anzeigen (Binärformat):

```
twadmin --generate-keys -S /etc/tripwire/site.key
ls -l /etc/tripwire/site.key
```

3. Konfigurations-Datei anpassen (normalerweise NICHT notwendig!):

```
vi /etc/tripwire/tw.cfg
```

4. Konfigurations-Datei in binäre signierte Form umwandeln (die vorher verwendete Passphrase des Site Keys eingeben!) und anzeigen (signiertes Binärformat):

```
twadmin --create-cfgfile -S /etc/tripwire/site.key /etc/tripwire/twcfg.txt
ls -l /etc/tripwire/tw.cfg
```

5. Richtliniendatei erstellen (vorher aus der mitgelieferten Musterdatei kopieren):

```
cp /usr/share/doc/packages/tripwire/twpol.txt /etc/tripwire/twpol.txt
vi /etc/tripwire/twpol.txt
```

Einfaches Beispiel für "twpol.txt" (eigenes Heimatverzeichnis absichern):

```
/home/tom -> $(IgnoreNone);      # Strichpunkt NICHT vergessen!:

(rulename = FirstRule, severity = 0) {
  /home/tom/cat -> $(IgnoreNone);  # Strichpunkt NICHT vergessen!:
```

- ```
}
}
6. Richtlinien-Datei in binäre signierte Form umwandeln (die vorher verwendete
Passphrase des Site Key eingeben!) und anzeigen (signiertes Binärformat):

twadmin --create-polfile -S /etc/tripwire/site.key /etc/tripwire/twpol.txt
ls -l /etc/tripwire/twpol.pol

7. Gemäß Richtlinien-Datei aus dem Dateisystem eine binäre, signierte
Tripwire-Datenbank mit den ausgewählten Eigenschaften der ausgewählten
Verzeichnisse und Dateien generieren (die vorher verwendete Passphrase des
Local Key eingeben) und anzeigen:

tripwire --init
ls -l /var/lib/tripwire/linux.twd

8. Aktuellen Systemzustand gegen Tripwire-Datenbank prüfen (gibt ASCII-Report
auf Bildschirm aus, erzeugt binären Report in "/var/lib/tripwire/report"):

tripwire --check
ls -l /var/lib/tripwire/report # -> linux-20050218-103957.twr

Erzeugte binäre Reportdatei nochmal in ausführlicher ASCII-Form ausgeben:

twprint --print-report -r /var/lib/tripwire/report/linux-20050218-103957.twr

Einschränkungen der zu prüfenden Regeln anhand Regelname oder Severity:

tripwire --check --rule-name "FirstRule"
tripwire --check --severity "100"

9a. Mehrere Änderungen am Heimatverzeichnis vornehmen:

cd /home/tom
mkdir VERZ
touch VERZ/DATEI
touch .profile
echo "echo HiHi..." >> .bashrc

9b. Aktuellen Systemzustand nochmal gegen Tripwire-Datenbank prüfen (gibt
ASCII-Report auf dem Bildschirm aus und erzeugt binären Report in
"/var/lib/tripwire/report"):

tripwire --check
ls -l /var/lib/tripwire/report # -> linux-20050218-143957.twr

Erzeugte binäre Reportdatei nochmal in ausführlicher ASCII-Form ausgeben:

twprint --print-report -r /var/lib/tripwire/report/linux-20050218-143957.twr

9c. Anhand eines erzeugten Berichts (ist als Parameter anzugeben) die
Tripwire-Datenbank updaten. Dazu wird der Bericht mit "[x]" vor jeder
Änderung im Editor "vim" angezeigt (Pfad zum Editor vorher in Variable
"VISUAL" bekannt geben). Durch Löschen des "x" wird die Änderung in der
Datenbank nicht eingetragen ("[X/x]" => "[]/[]"):

export VISUAL=/bin/vim
tripwire --update -r /var/lib/tripwire/report/linux-20050203-143957.twr

9d. Aktuellen Systemzustand nochmal gegen Tripwire-Datenbank prüfen (gibt
ASCII-Report auf dem Bildschirm aus und erzeugt binären Report in
"/var/lib/tripwire/report"):

tripwire --check
ls -l /var/lib/tripwire/report # -> linux-20050218-143957.twr

Die akzeptierten Änderungen sollten nun nicht mehr berichtet werden.

10. Tägliche Läufe des Tripwire-Reporting als "cronjob" einrichten und den
Bericht per Mail an USER@HOST versenden:

crontab -e
MIN STD TAG MON WOTAG KMDO
22 2 * * * /usr/sbin/tripwire --check 2>&1 | mail -s "Tripwire" USER@HOST

11. Modifikationen der Richtlinien müssen in der ASCII-Richtliniendatei
vorgenommen werden. Dann ist die Tripwire-Datenbank anzupassen und die
binäre Richtlinien-Datei neu zu erzeugen ("low/high"):
```
- ```
vi /etc/tripwire/twpol.txt
tripwire --update-policy --secure-mode low /etc/tripwire/twpol.txt
```

Dabei werden Verletzungen der Richtlinien berichtet (beim Standard "--secure-mode low" wird dies nicht gemacht!).
AM SCHLUSS wird automatisch die Binärform der Richtlinien-Datei erzeugt!

4) man-Pages

```
-----  
man twintro           # Einführung in Tripwire  
man twfiles           # Beschreibung der Tripwire-Dateien  
man twconfig          # /etc/tripwire/twcfg.txt  
man twpolicy          # /etc/tripwire/twpol.txt  
man tripwire          # Beschreibung tripwire-Kommando  
man twadmin           # Beschreibung twadmin-Kommando  
man twprint           # Beschreibung twprint-Kommando  
man siggen            # Beschreibung siggen-Kommando
```