HOWTO zu "tripwire"

(C) 2006-2013 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
OSTC Open Source Training and Consulting GmbH
http://www.ostc.de

$Id: tripwire-HOWTO.txt,v 1.19 2019/11/26 19:37:07 tsbirn Exp $

Dieses Dokument beschreibt die Absicherung eines Linux-Systems über das
HIDS (Host Intrusion Detection System) "Tripwire" ("Stolperdraht").

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
2.5) Direktiven
3) Durchführung einer Konfiguration und mehrerer Prüfungen
4) man-Pages
5) Literaturverweise

---

1) Einführung

Tripwire (Stolperdraht) ist ein "Host Intrusion Detection System" (HIDS) oder besser ein "System Integrity Verifier". Es erstellt eine Momentaufnahme über den Zustand eines (Datei)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 Spezialdatenbank 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.

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

  * Frisch installiert
  * Installations-Daten aus vertrauenswürdiger Quelle (CD, DVD, Signatur)
  * Noch keine Netzwerkverbindung eingerichtet/verwendet

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

  * Gewollt (z.B. Update, Konfiguration, SW-Inst.) --> Neu in Datenbank aufnehmen
  * Unbefugt (Fehler, Angriff) --> Prüfen ob gefährlich

D.h. es verhindert keinen Angriff, sondern erkennt einen Angriff "indirekt", nämlich NICHT online, sondern erst zu einem späteren Zeitpunkt (offline) an seinen Auswirkungen auf das Dateisystem.

Werden Abweichungen festgestellt, die auf Grund von Umkonfiguration, Aktualisierung, Aufspielen oder Entfernen von Software-Paketen entstanden sind, können diese Änderungen als neuer Ist-Zustand zur vorhandenen Datenbank hinzugefügt werden, damit sie 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 Sicherheit und Performanz.

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

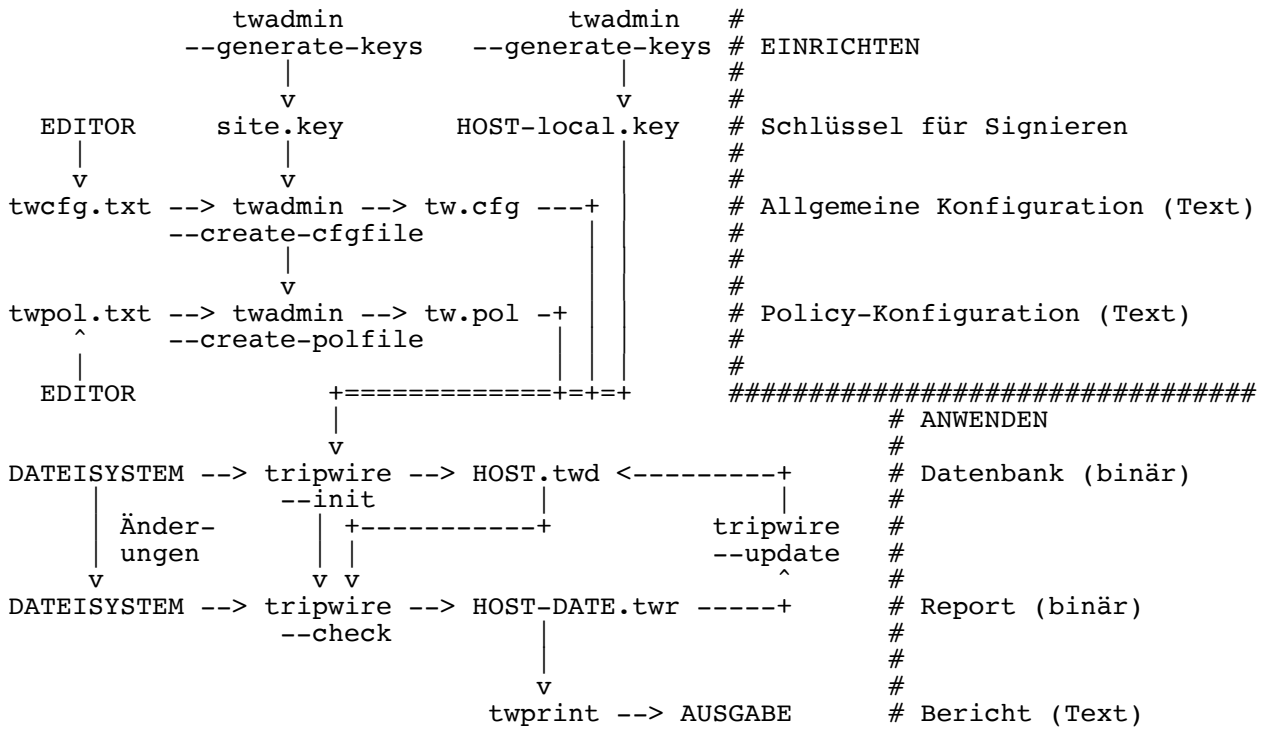
Die beiden Keys (Schlüssel) müssen 1x ganz am Anfang während der Einrichtung

von Tripwire mit "twadmin" erzeugt werden. Sie sollten mit je einer "Passphrase" gesichert werden, damit sie nicht im Klartext im Dateisystem liegen. Die Trennung der 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

Folgende Zusammenhänge zwischen den Kommandos (inkl. Optionen) und den Dateien gibt es:



Der Ablauf der Absicherung eines Systems durch Tripwire lässt sich folgendermaßen übersichtlich darstellen. In Abschnitt 3) werden zu jedem Schritt 0-11 die dabei verwendeten Befehle beschrieben.

E	0	Rechnernamen ermitteln ("hostname")
I	1	Host Local Key "HOST-local.key" generieren (mit Local Passphrase)
N	2	Site Key "site.key" generieren (mit Site Passphrase)
R	3	Konfigurations-Datei "twcfg.txt" erstellen (ASCII, Editor)
I	4	Konfigurations-Datei "tw.cfg" generieren (binär, Site Passphrase)
C	5	Richtlinien-Datei "twpol.txt" erstellen (ASCII, Editor)
H	6	Richtlinien-Datei "twl.pol" generieren (binär, Site Passphrase)
T	7	Binäre Datenbank "HOST.twd" generieren (binär, Local Passphrase)
E	8	Durch Vergleich System <-> Datenbank einen Prüfbericht generieren
A	9	Änderung werden am (Datei)System vorgenommen
N		Prüfbericht generieren --> Unterschiede
W		Akzeptierte Änderungen in DB nachtragen (Editor, Local Passphrase)
E		Prüfbericht generieren --> Unterschiede (sollten keine mehr sein)
N		
D	10	Täglich per Cronjob Prüfbericht generieren lassen
E	11	Neue/geänderte Richtlinien in DB einpflegen (Site Passphrase)
N		

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 überwachtes 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 (darf z.B. eine Datei wachsen, sollte die Größe nicht geprüft werden):

Code	Begriff	Eigenschaft
a	access	Zugriffsdatum
b	blocks	Anzahl Datenblöcke
c	change	Inode-Änderungsdatum
d	device	Gerätenummer auf der Datei-Inode liegt
g	group	Besitzer-Gruppe (GID)
i	inode	Inode-Nummer
l	length	Datei darf wachsen (Verzeichnis?)
m	modify	Inhalt-Änderungsdatum
n	number	Anzahl Hard-Links
p	permission	Zugriffsrechte und Dateimodus
r	major/minor	Device Nummer (nur für Geräte)
s	size	Dateigröße
t	type	Dateityp (File, Verz., Symlink,...)
u	user	Besitzer (UID)

Durch Auflisten der einbuchstabigen Codes und Voranstellen von "+" und "-" werden sie folgendermaßen zu MASKEN (Kombination von zu prüfenden und zu ignorierenden Eigenschaften) kombiniert:

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

Fängt eine Maske nicht mit "+" oder "-" an, dann gilt "+". Fehlt eine Eigenschaft in einer Maske, dann wird sie nicht geprüft (äquivalent "-" vor der Eigenschaft). D.h alle folgenden Masken sind identisch:

```
+p+n                  # permission + number
pn                    # (analog)
+pn-g                 # (analog)
+pn-iugtsdrlbamcCMSH  # (analog)
```

Weiterhin kann in der Maske festgelegt werden, welche der möglichen Prüfsummen generiert werden sollen (sortiert nach steigendem Aufwand und zunehmender Sicherheit, mind. zwei Prüfsummen sollten generiert werden):

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

Häufig benötigte Masken ("Kombinationen" von Eigenschaften) sind über

vordefinierte Variable verfügbar:

Name	Definition	Bedeutung
ReadOnly	+pinugtsdbmCM-rlacSH	Nur lesbar
Dynamic	+pinugtd-srlbamcCMSH	Dynamik erlaubt (Wachsen+Schrumpfen)
Growing	+pinugtdl-srbamcCMSH	Darf nur wachsen
Device	+pugsdr-intlbamcCMSH	Gerät (von Tripwire nicht zu öffnen)
IgnoreAll	-pinugtsdrlbamcCMSH	Nur Vorhandensein/Abwesenheit prüfen
IgnoreNone	+pinugtsdrbamcCMSH-l	Startpunkt für eigene Regeln

Diese Kombinationen dienen als Ausgangspunkt für abweichende Masken durch Anhängen von Eigenschaften mit "+" und "-":

```
$(IgnoreNone) -ar   # --> Alles außer Zugriffsdatum + Device-Nummer prüfen
```

In der Datei "twpol.txt" werden über "Richtlinien" (Regeln, Policies) Dateien und Verzeichnisse mit den zu prüfenden Eigenschaften verknüpft. Für jede Datei/jedes Verz. kann nur EINE Richtlinie vorgegeben werden. Bei Verzeichnissen können für Teile ihres Inneren abweichende Richtlinien (AUSNAHME) vorgegeben werden.

```
/bin        -> $(ReadOnly);    # Verzeichnis
/etc        -> $(IgnoreAll);   # Verzeichnis
/etc/passwd -> $(Growing);     # Datei in vorigem Verzeichnis
```

Soll eine Datei oder ein Verzeichnis NICHT geprüft werden, so ist sie mit einem "!" zu versehen.

```
!/etc/init.d;
!/etc/rc.d;
!/etc/mnttab;
```

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

```
/bin -> $(ReadOnly) (rulename = "Executables", severity = 0);
```

Jede Richtlinie ist mit Strichpunkt ";" abzuschließen

2.3) Installation und Grund-Konfiguration

Tripwire wird bei den meisten Distributionen mitgeliefert. Installieren von Tripwire durch (i=install):

```
rpm -i tripwire               # SuSE
apt-get install tripwire      # Debian
```

Prüfung, ob Tripwire installiert ist durch (q=query, a=all):

```
rpm -qa | grep tripwire       # SuSE   --> tripwire-2.3.1-186
dpkg --list | grep tripwire   # Debian --> 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:

Pfad	Bedeutung
/etc/tripwire/*	Konfiguration
/usr/sbin/*	Programme
/usr/lib/tripwire/databases/*	Datenbank
/usr/share/doc/tripwire/*	Dokumentation
/usr/share/doc/tripwire/examples/*	Beispiel-Konfiguration
/usr/share/man/man8/*	Manualpages
/var/lib/tripwire/*	Datenbank
/var/lib/tripwire/report/*	Berichte

Für Tripwire müssen folgende Dateien verwaltet werden:

Typ + man-Page	Zweck + Dateien
Key (man twadmin)	Schlüssel zur Absicherung der Datenbank /etc/tripwire/site.key /etc/tripwire/HOST-local.key
Configuration (man twconfig)	System-spezifische Information /etc/tripwire/twcfg.txt --> /etc/tripwire/tw.cfg
Policy (man twpolicy)	Richtlinien /etc/tripwire/twpol.txt --> /etc/tripwire/tw.pol
Database (man tripwire)	Generierte Datenbank /var/lib/tripwire/HOST.twd
Report (man twprint)	Generierte Berichte /var/lib/tripwire/report/HOST-DATE.twr
Backup (man twfiles)	Backup neu generierter Dateien Endung ".bak"

Datei-Extensions:

Ext	Bedeutung
.bak	Backup
.cfg	Configuration
.key	Schlüssel
.pol	Policy
.twd	Tripwire Database
.twr	Tripwire Report

Während der Installation von Tripwire entsteht ein Konfigurations-Verzeichnis

```
/etc/tripwire
```

mit den beiden 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. Die folgenden fünf Angaben sind notwendig:

```
POLFILE      = /etc/tripwire/tw.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
```

Die folgenden Angaben dürfen fehlen (Standardwerte angegeben):

```
ROOT                    = /usr/tripwire
EDITOR                  = /bin/vi                   # /usr/bin/vim
TEMPDIRECTORY           = /tmp

LATEPROMPTING           = false
LOOSEDIRECTORYCHECKING  = false
SYSLOGREPORTING         = true
REPORTLEVEL             = 3                         # 0..4

GLOBALEMAIL             = none                      # tb@ostc.de
MAILMETHOD              = SENDMAIL                  # SMTP
SMTPHOST                = mail.domain.com           # localhost
SMTPPORT                = 25
```

```
  | MAILPROGRAM            = /usr/lib/sendmail -oi -t                    |
  | EMAILREPORTLEVEL       = 3                                           | # 0..4
  | MAILNOVIOLATIONS       = true                                        |
  +-------------------------------------------------------------+
```

2.4) Richtlinien (Policies) definieren

Kommentare werden durch "#" eingeleitet und gelten bis zum Zeilenende.
Sie können auf einer Zeile für sich stehen oder nach einer Richtlinie.

```
  # Kommentar
  ... # Kommentar
```

Richtlinien haben folgende Form (ihre Reihenfolge ist nicht relevant):

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

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

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

Als OBJEKT wird eine zu prüfende Datei oder ein Verzeichnis aufgelistet, pro
Objekt darf NUR EINE Richtlinie vorgegeben werden (wird geprüft). Wird ein
Verzeichnis angegeben, dann werden alle Objekte ab diesem Startpunkt geprüft.
Objekte müssen ABSOLUTE PFADE zu Dateien und Verzeichnissen sein, z.B.:

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

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

Name	Definition	Bedeutung
ReadOnly	+pinugtsdbmCM-rlacSH	Nur lesbar
Dynamic	+pinugtd-srlbamcCMSH	Dynamik erlaubt (Wachsen+Schrumpfen)
Growing	+pinugtdl-srbamcCMSH	Darf nur wachsen
Device	+pugsdr-intlbamcCMSH	Gerät (von Tripwire nicht zu öffnen)
IgnoreAll	-pinugtsdrlbamcCMSH	Nur Vorhandensein/Abwesenheit prüfen
IgnoreNone	+pinugtsdrbamcCMSH-l	Startpunkt für eigene Regeln

Eigene Variablen für Objekte und Eigenschaften können jederzeit definiert und
anschließend benutzt werden (GROSS/kleinschreibung der Variablennamen zählt,
der Name darf Buchstabe, Ziffern, Unterstrich und "+-@:." enthalten):

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

Vordefiniert sind bereits folgende Variablen:

```
  HOSTNAME   # Rechnername
  DATE       # Datum + Uhrzeit in der Form YYYYmmdd-HHMMSS
```

Beispiele für Variablendefinition:

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

Beispiele für Variablenverwendung:

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

Beispiele für Richtlinien (Strichpunkt nicht vergessen!):

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

```
  |                                                                    |
  | # Ganzen Dateibaum "/etc" prüfen (nur Vorhandensein!)              |
  | # AUSNAHMEN folgen noch                                            |
  | /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 zusätzlich mit "Attributen" versehen werden, die Name, Wichtigkeit, Verzeichnistiefe und Mailempfänger festlegen.

a) Für einzelne Richtlinie (Strichpunkt NICHT vergessen!):

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

b) Für Gruppe von Richtlinien (Strichpunkt NICHT vergessen!):

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

Folgende Attributangaben zu einer Richtinie sind möglich:

Attribut	Bedeutung
emailto	eMail-Empfänger bei verletzter Regel
rulename	Zur Bericht-Auswahl + im Bericht-Output (Benennung von Regeln)
severity	Zur Bericht-Auswahl 0..1000000 (Default 0) (mind. notwendige Wichtigkeit)
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 = tripwire@foo.com, severity = 100)
  {
      /bin     ->  $(ReadOnly);
      /sbin    ->  $(ReadOnly);
      /lib     ->  $(ReadOnly);
      /usr/bin ->  $(ReadOnly);
      /usr/sbin->  $(ReadOnly);
  }
```

Musterbeispiele für Richtlinien-Dateien findet man hier:

```
  /usr/share/doc/tripwire/examples/policyguide.txt.gz
  /usr/share/doc/tripwire/examples/twpol.txt.gz
```

2.5) Direktiven

Direktiven erlauben eine bedingte Interpretation der Policy-Datei und einige Diagnose- und Debugging-Operationen, sie haben folgende allgemeine Form

```
  @@DIRNAME [ARGS]    # Allgemeine Syntax
```

Folgende Directiven sind erlaubt:

Direktive	Bedeutung
@@ifhost HOST...	Bedingte Interpretation abhängig vom Host (|| = oder)
@@else	"
@@endif	"
@@section NAME	Abschnitt der Policy-Datei (FS, GLOBAL, NTFS, NTREG)
@@print "STRING"	Text auf Standard-Ausgabe ausgeben
@@error "STRING"	Text auf Standard-Ausgabe ausgeben und Abbruch
@@end	Logisches Dateiende (Rest der Datei wird ignoriert)

Hinweise: In einer @@ifhost-Direktive darf eine Liste von Hostnamen getrennt durch "||" angegeben werden. @@ifhost-Direktiven dürfen verschachtelt werden. "FS" ist Standard bei "@@section". "NTFS" und "NTREG" haben nur Bedeutung auf Windows NT Systemen, auf UNIX-Systemen werden bei diesen beiden Angabeng alle folgenden Zeilen bis zur nächsten @@section-Direktive übersprungen.

Beispiele:

```
  @@section GLOBAL               # Abschnitt
                                 #
  @@ifhost plymouth              #
      /bin -> $(ReadOnly);       # Regel nur für Host "plymouth" geprüft
  @@endif                        #
                                 #
  @@section FS                   # Abschnitt
                                 #
  @@ifhost exeter || charlton    #
      /usr/bin -> +pinug;        # Regel nur für Hosts exeter + charlton gültig
  @@else                         #
      /usr/bin -> +pinugsmC;     # Regel für alle anderen Hosts gültig
  @@endif                        #
                                 #
  @@print "Ok"                   # Ausgabe
  @@error "Fehler"               # Ausgabe + Abbruch
                                 #
  @@end                          # Rest der Datei wird ignoriert
```

3) Durchführung einer Konfiguration und mehrerer Prüfungen

Hier nun der konkrete Ablauf zur Absicherung und Prüfung eines Systems gemäß den Schritten 0-11 in der Tabelle von Abschnitt 2) (zur Steigerung der Geschwindigkeit und Vermeidung von Tippfehlern die Befehle mit der Maus kopieren!):

0. Rechnernamen ermitteln:

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

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

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

2. Site Key erzeugen (Site 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:

```
vi /etc/tripwire/twcfg.txt   # Normalerweise NICHT notwendig!
```

4. Konfigurations-Datei in binäre signierte Form umwandeln (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!
| }                                     |
|                                       |
| !/home/tom/bin;                       | # Strichpunkt NICHT vergessen!
+---------------------------------------+
```

6. Richtlinien-Datei in binäre signierte Form umwandeln (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/tw.pol
```

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

```
tripwire --init
ls -l /var/lib/tripwire/HOST.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    # --> HOST-20050218-103957.twr
```

Erzeugte binäre Reportdatei nochmal in ausführlicher ASCII-Form ausgeben:

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

Einschränkungen der zu prüfenden Regeln anhand Regelname oder Severity:

```
tripwire --check --rule-name "FirstRule"   # Nur mit diesem Namen
tripwire --check --severity "100"          # Mind. diese Wichtigkeit
```

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    # --> HOST-20050218-143957.twr
```

Erzeugte binäre Reportdatei nochmal in ausführlicher ASCII-Form ausgeben:

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

9c. Anhand eines erzeugten Berichts (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 "EDITOR/VISUAL" bekannt geben). Durch Löschen des "x" wird die Änderung nicht in der Datenbank eingetragen ("[X/x]" => "[]/[ ]"):

```
export VISUAL=/bin/vim
tripwire --update -r /var/lib/tripwire/report/HOST-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    # --> HOST-20050218-143957.twr
```

    Die akzeptierten Änderungen sollten nun nicht mehr berichtet werden.

10. Tägliche Läufe um 02:22 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 und AM SCHLUSS wird
    automatisch die Binärform der Richtlinien-Datei erzeugt!
    (letzteres wird beim Standardwert "--secure-mode high" nicht gemacht!).

4) man-Pages

man-Page	Inhalt
tripwire(8)	Beschreibung tripwire-Kommando
twadmin(8)	Beschreibung twadmin-Kommando
twprint(8)	Beschreibung twprint-Kommando
siggen(8)	Beschreibung siggen-Kommando
twintro(5)	Einführung in Tripwire
twfiles(5)	Beschreibung der Tripwire-Dateien
twconfig(5)	Beschreibung /etc/tripwire/twcfg.txt
twpolicy(5)	Beschreibung /etc/tripwire/twpol.txt

* Das Kommando "siggen" zeigt die tripwire-Hashwerte (CRC-32, MD5, SHA, Haval)
  für eine oder mehrere Dateien an.

5) Literaturverweise

http://sourceforge.net/projects/tripwire/   Tripwire-Projekt
http://www.tripwire.org/   Tripwire-Projekt
http://www.tripwire.com/   Tripwire (kommerzielle Version)