Tripwire
host-based intrusion detection system

Virenchecker (Attack-Signature-Scanner) benötigen immer eine aktuelle Bibliothek mit den neuesten Virensignaturen.

diesen Makel haben Filesystem Integrity Checker nicht:
- sie konservieren einen Systemzustand, der als sicher gilt (Referenz-Datenbank im single-user-mode)
- bösartiger Code kann durch einfachen Vergleich mit diesem Zustand gefunden werden (Integritätstest zum Aufspüren von Inkonsistenzen)
- nach Neu- oder Deinstallationen sollte die Referenz-Datenbank aktualisiert werden (Wartung)
- Tests sollten automatisch in regelmäßigen Zeitabständen als Cron-Job stattfinden

weiter nutzbar:
- Ursachenerkennung bei Versagen einer Anwendung
- zur Entfernung installierter Software
- zur Beweissicherung

statisches System: sobald neue Software aufgespielt wird, muss die Konfiguration angepasst werden → nicht für Testsysteme geeignet!

Regeln der Tripwire Policy

Syntax:

[!|=] Object_name -> property_mask;

object_name	absoluter Pfadname zu einer Datei oder einem Verzeichnis (rekursiv, nicht dateisystemübergreifend)
property_mask	zu prüfende oder zu ignorierende Eigenschaften eines Objekts
'->'	Trenner zwischen Objekt und Eigenschaftsmaske, eingeschlossen von Whitespaces
;	Abschluss mit einem Semikolon

Beispiele

```
# definiert Tw-Aktivitäten für den ganzen Verzeichnisbaum /bin
/bin   ->   $(ReadOnly);

# Tw überwacht alle Eigenschaften von hostname.hme0.
/etc/hostname.hme0    ->    $(IgnoreNone) -ar;

# Scannen des gesamten Verzeichnisses /etc unter Nutzung der mask1, ausser
# der Datei /etc/passwd, die mit mask2 gescannt wird.
/etc        ->  $(mask1);
/etc/passwd ->  $(mask2);
```

Tripwire hält vordefinierte Auswahlmasken, so genannte Templates bereit. Auch Kombinationen aus Templates und Select-flags wie N-a oder E+7 sind erlaubt.

Auswahl der Aktionen für Objekte ist möglich anhand von:

Operatoren	= ! (! entfernt Eintrag und = entfernt Inhalt des Eintrags aus der Testliste)
Select-Flags	p i n u g s a m c t d l r b
Templates	R L N E > Device

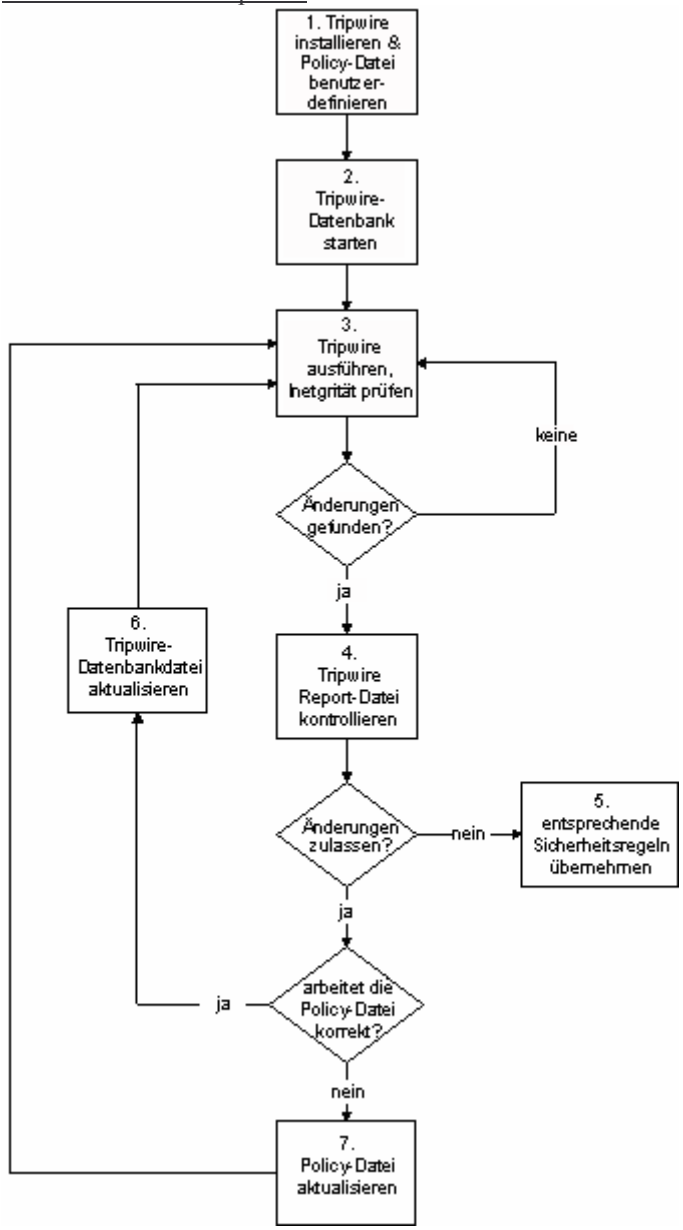
Beispiele.:

= /var/spool/mail L	# überwacht Inodes, nicht den Inhalt (resourcensparend)
	# auch sinnvoll für /tmp
!/dev	# von der Überwachung ausgeschlossen
+ug-a	# es werden Benutzer- und Gruppenkennung des Eigentümers,
	# nicht aber der Zeitpunkt des letzten Zugriffs überwacht
	# Achtung: ASR betrachtet Undefiniertes grundsätzlich als selektiert
+pinugsmc123456789-a	# identisch mit +ug-a
-a	# identisch mit +ug-a
E+ug	# Benutzer- und Gruppenkennung (Kombination aus Templates und Select-flags)
N-a	
E+7	
R-12+8	# guter Kompromiss in Hinblick auf Sicherheit und Datendurchsatz

Gewissheit über die Integrität eines Objekts bringt nur die direkte Vermessung der Datenzonen durch eine wirksame Signaturfunktion. (sehr sicher sind Algorithmen wie SHA und Haval)
Da jeder Funktion ein eigenes Select-flag spendiert wurde, kann der Administrator bei der Konfiguration sehr flexibel auf spezielle Anforderungen reagieren. Die ergeben sich aus der Bedeutung des Objekts, der verfügbaren Rechenleistung und dem individuellen Anspruch an die Systemsicherheit.

Eine zentrale Konfigurationsdatei im Netz
ein und dieselbe Konfigurationsdatei kann auf mehreren Rechnern mit unterschiedlicher Architektur gleichzeitig verwendet werden. Tripwire besitzt dafür einen einstufigen Präprozessor, der spezielle Schlüsselwörter wie @@include, @@ifhost und @@define interpretiert.
Es ist beispielsweise denkbar, die Konfigurationsdatei auf einem einzigen Rechner vorzuhalten und den anderen Rechnern nur bei Bedarf zur Verfügung zu stellen. Existierende Konfigurationsdateien könnten vorher zu einer einzigen verschmolzen werden. Das hat natürlich nur Sinn, wenn eine Manipulation der Umgebungsvariablen des anfragenden Rechners ausgeschlossen werden kann.

Der Gebrauch von Tripwire:



Die Ext2-Select-flags der ASR und ihre Bedeutung		
Select-flag	**Bericht**	**Bedeutung**
p	st_mode	Zugriffsrechte und Ausführungsmodi (SUID-, SGID-Bit (!) und Text-Bit)
i	st_ino	Nummer der I-Node: Die I-Node-Nummer eines Objekts wird durch normale Schreib-/Leseoperationen nicht verändert. Findet sich eine solche Inkonsistenz im Integritätsbericht, liegt die Vermutung nahe, dass das betreffende Objekt gelöscht und durch eine Fälschung mit gleichem Namen ersetzt wurde.
n	st_nlink	Anzahl der Hardlinks oder Unterverzeichnisse: Ein spezielles Feld der I-Node, der *Links count*, gibt im Fall eines Verzeichnisses die Anzahl der zugehörigen Unterverzeichnisse, im Fall einer Datei die Anzahl der mit der I-Node assoziierten Links an. In diesem Fall erhöht sich der Zähler immer dann, wenn ein Hardlink auf die zugehörigen Datenzonen hergestellt wird. (Beispiel: Wird mit *ln /etc/passwd /home/hacky* ein Hardlink auf */etc/passwd* erzeugt, so wird der entsprechende Zähler in der I-Node von */etc/passwd* um eins heraufgesetzt. Beim nachfolgenden Integritätstest würde die Datei also als *changed* ausgewiesen.)
u	st_uid	Benutzer-ID: Benutzer- und Gruppen-ID geben natürlich vorzügliche Zielscheiben für Angriffe aller Art ab.
g	st_gid	Gruppen-ID
s	st_size	Dateigröße: Ein ganz brauchbarer Indikator, zumal es nicht immer leicht sein dürfte, eine Konfigurationsdatei so abzuändern, dass die Dateigröße dabei erhalten bleibt und trotzdem die gewünschte Wirkung erzielt wird.
a	st_atime	Datum des letzten Zugriffs: Bereits das Einlesen einer Datei genügt, um diesen sensiblen Eintrag in der zugehörigen I-Node zu aktualisieren. Das betreffende Select-flag in Kombination mit einer Signatur-Überwachung einzusetzen macht daher wenig Sinn, denn zur Berechnung der Signatur muss die betreffende Datei natürlich gelesen werden. Der *access timestamp* lässt sich mit *ls -l - -time=atime* sichtbar machen.
m	st_mtime	Zeitpunkt der letzten Veränderung: Dieses Feld wird nur dann aktualisiert, wenn die betreffende Datei modifiziert und erneut gesichert wurde. Der *modification timestamp* ist jedem Linux-Anwender von Inhaltsverzeichnissen her bekannt, die mittels *dir* oder *vdir* erstellt wurden.
c	st_ctime	Datum der letzten Statusänderung, also des letzten Schreibzugriffs auf die I-Node: Eine Statusänderung erfolgt etwa beim Ändern der Zugriffsrechte für eine Datei. Der *inode timestamp* kann mit *ls -l - -time=ctime* abgerufen werden.
t (2.2.1)	Object Type	Dateityp (Datei, Verzeichnis, symbolic Link)
d (2.2.1)	Device Number	Partitionstyp: Partitionen werden beim Einrichten mit einer speziellen Kennzahl versehen, die Aufschluss über die Art der Formatierung gibt (magische Zahl). Das betreffende Select-flag stellt sicher, dass neben anderen Merkmalen auch die Kennzahl der Partition in der Referenz-Datenbank vermerkt wird, von der die I-Node des betreffenden Objekts stammt.
l (2.2.1)	Size	Logfile zeigt an, dass sich der Umfang der betreffenden Datei im regulären Betrieb nur vergrößern kann. Im Gegensatz zu *s*, das jede Veränderung der Dateigröße moniert, erfolgt hier nur dann eine Meldung, wenn ein Schwund festgestellt wird. Ein typischer Kandidat wäre zum Beispiel */var/log/messages*. Die ASR stellt diese Funktionalität nur in Verbindung mit anderen Select-flags als Template (*>) zur Verfügung.*
r (2.2.1)	File Device Number	Hauptgerätenummer: Diese Eigenschaft ist nur für Gerätedateien erklärt und bezeichnet hier die Nummer des Garätetreibers, der zur betreffenden I-Node gehört. Wenn das */dev*-Verzeichnis mittels *ls -l /dev* gelistet wird, sind statt der Dateigröße die Hauptgerätenummer (und evtl. existierende Untergerätenummern) zu sehen.
b (2.2.1)	Blocks	Blockcount: Anzahl der Datenblöcke, die durch die Zonenzeiger der I-Node belegt werden. Die Größe eines Ext2fs-Blocks wird beim Einrichten der Partition vorgegeben (typisch 1024 Bit).

Tabelle 2: Das kryptographische Arsenal von Tripwire				
flag	**Algorithmus**	**Durchsatz in MByte/s**	**Sicherheitsklasse**	**Erläuterungen**
1	MD5 (auch 2.2.1)	7,2	*****	Der von Krypto-Papst Ronald Rivest entwickelte Message-Digest-5-Algorithmus sollte durch sein eher konservatives Design bekannte Schwächen des Vorgängermodells MD4 beseitigen. Die wichtigsten Änderungen betrafen die Rundenzahl (vier statt bisher drei) und die Zahl der additiven Konstanten (eine eigene für jeden der 64 Teilschritte). Die Modifikation ging zwar zu Lasten der Arbeitsgeschwindigkeit, schien den Algorithmus aber gegen analytisch gestützte Angriffe deutlich resistenter zu machen. Dazu die positive Beurteilung zweier angesehener Kryptographen aus dem Jahre '94, die einen groben Eindruck von der Leistungsfähigkeit einer guten Signaturfunktion vermitteln kann: "Van Oorschot and Wiener have considered a brute-force search for collisions in hash functions, and they estimate a collision search machine designed specifically for MD5 (costing $10 million in 1994) could find a collision for MD5 in 24 days on average." Im Licht neuerer Erkenntnisse aus dem Bereich der Kryptographie scheint diese Einschätzung leicht revisionsbedürftig. Zwar gelang es bislang nicht, die Wirksamkeit der Hash-Funktion selbst in Frage zu stellen, doch konnten - wie schon vorher bei MD4 - Kollisionen für die Compression Function (eine wesentliche Teilstruktur der Hash-Funktion) gefunden werden. (Wird ausführlich in einer späteren Folge behandelt.) MD5 ist derzeit (noch) der meistgenutzte Hash-Algorithmus, seine Zukunft sieht indes düster aus. Führende Kryptographen bescheinigen künftigen Attacken ausgesprochen gute Erfolgschancen.
2	Snefru (R)	1,4	****	"The ideal pyramid was eventually built by Snefru's successor, Khufu, and the first - the Great Pyramid at Giza - was the finest and most successful." Der von Ralf Merkle am Xerox Palo Alto Research Center (PARC) erdachte Algorithmus konnte im Ansehen nicht ganz an seinen berühmten Namensvetter heranreichen. Bereits im April '90 gelang es einem ehrgeizigen Studenten, die bis dato beliebte zweistufige Version zu entthronen und dafür eine Erfolgsprämie von 1000 Dollar zu kassieren. Das PARC empfiehlt jetzt schon die achtstufige Variante! [2] Da bisher aber noch jeder Versuch scheiterte, die hier eingesetzte vierstufige Version mit 128-Bit-Signaturformat zu überlisten, dürfte sich die Sicherheitsleistung noch in akzeptablem Rahmen bewegen. Ein echter Nachteil ist allerdings der vergleichsweise niedrige Datendurchsatz.
3	CRC-32 (auch 2.2.1)	9,3	**	Siehe Eräuterungen zu CRC-16.
4	CRC-16	16,2	*	Diese beiden robusten und schnellen Algorithmen sind eigentlich zur Erkennung von Hardwarebedingten Übertragungsfehlern gedacht. Die einfachste Variante einer solchen Prüfsummen Funktion wird durch sukzessives XOR-Verknüpfen aller Wörter einer Nachricht realisiert. Bereits die Signaturgröße von gerade mal 16 und 32 Bit verbietet einen Einsatz bei großen oder wichtigen Dateien. Da eine gefälschte Datei jedoch neben der passenden Signatur auch noch die entsprechende Funktionalität mitbringen muss, ist ein Einsatz bei weniger kritischen Objekten durchaus zu erwägen.
5	MD4	14,4	***	Wurde bereits ´90 eingeführt und war seither wegen seiner Schnelligkeit auf RISC-Prozessoren sehr beliebt. Erst '98 folgte die Ernüchterung. Eine leicht abgewandelte Version erwies sich als umkehrbar. MD4 gilt heute als widerlegt und sollte daher zum Schutz wichtiger Objekte nicht mehr herangezogen werden. (Kollisionen für MD4 lassen sich auf einem handelsüblichen PC innerhalb weniger Sekunden künstlich erzeugen! Das macht auf eindrucksvolle Weise die Relevanz dieser Betrachtung deutlich.)
6	MD2	0,3	****	Ungewöhnlich langsam, da als einziger für altertümliche 8-Bit-Prozessoren ausgelegt, während MD4 und MD5 volle 32 Bit, also die Wortgröße der meisten heutigen Prozessoren, ausschöpfen können! Obwohl MD2 der älteste der drei Message-Digest-Algorithmen von RSA ist, steht seine Wirksamkeit bis heute außer Frage. Die einzige Erkenntnis kryptanalytischer Natur betrifft eine leicht abgewandelte Version. Kollisionen ließen sich nämlich erst dann künstlich herstellen, wenn beim so genannten Padding (wird ausführlich in einer späteren Folge behandelt) auf das Einfügen der Nachrichtenlänge verzichtet wurde.
7	SHA (auch	5,4	*****	Der Secure-Hash-Algorithm des NIST [3] ist - wie die meisten Hash-Algorithmen - strukturell mit MD4 verwandt. ´94 wurde er durch seinen offiziellen Nachfolger SHA-1

	2.2.1)			abgelöst,der eine undokumentierte Schwachstelle korrigieren sollte, um die sich noch heute zahlreiche Gerüchte ranken. Die große 160-Bit-Signatur dürfte SHA trotzdem zu einer guten Wahl machen, auch für sicherheitskritische Objekte. Die NASA gibt diesem Algorithmus in der haus eigenen Tripwire-Installation den Vorzug vor dem ursprünglich populäreren Snefru! [4] (Die hartnäckigste Mutmaßung geht von einer absichtlichen Implantierung durch die National Security Agency (NSA) aus. Eine geheime Schwachstelle, die den späteren Zugriff auf fremdes Datenmaterial ermöglichen könnte, würde zumindest ein plausibles Motiv abgeben. Allerdings nur, so lange die Schwachstelle auch geheim bleibt und nicht durch übereifrige zivile Kryptographen aufgedeckt wird. Viele Beobachter sehen hier den Grund für die unvermittelte Ablösung von SHA. Eine Hypothese, der ich mich angesichts des noch dürftigen Kenntnisstands in der rasch wachsenden Theorie der Hash-Funktionen nur ungern anschließen möchte. TSS jedenfalls scheint diese Ansicht zu teilen, denn SHA findet sich in unveränderter Form auch in der aktuellen Version 2.2.1.)
8	Haval (auch 2.2.1)	10,7	****	Wurde 1992 an der Universität von Wollongong von Yuliang Zheng [5] kreiert. Als einziger weist Haval sowohl eine variable Signaturgröße (128, 160, 192, 224 oder 256 Bit) als auch eine variable Anzahl an Arbeitsschritten (drei, vier oder fünf) auf. (Die Nachricht wird hier in Blöcke von 1024 Bit aufgeteilt, die dann in jeweils drei, vier oder fünf Runden durch die Compression Function bearbeitet werden.) Damit sind insgesamt 15 unterschiedliche Varianten des Algorithmus für praktische Anwendungen verfügbar. In der Academic Source Release kommt die vierstufige Variante mit 128-Bit-Signaturformat zum Einsatz. Meine Einschätzung in puncto Sicherheit darf möglicherweise nach oben korrigiert werden. Der unkonventionelle Aufbau ist ein Glückstreffer, weil er den Algorithmus damit gegen herkömmliche Attacken, die fast ausnahmslos auf MD4-Methoden fußen, immun macht.

Um die Konfiguration zu vereinfachen, wurden verschiedene Analysemuster zu Templates zusammengefasst:

Tabelle 3: Die Templates der ASR		
Template	**Definition**	**Verwendung**
R	+pinugsm12-ac3456789	**(R)ead-only**: Dateien, die zwar allgemein verfügbar sind, aber nur gelesen werden dürfen (Standard).
L	+pinug-sacm123456789	**(L)og file**: Benutzerverzeichnisse und Dateien, die ständiger Veränderung unterworfen sind.
N	+pinugsamc123456789	**Ignore (N)othing**: Nichts ignorieren. Diese Auswahlmaske eignet sich auch gut als Ausgangspunkt für benutzereigene Definitionen.
E	-pinugsamc123456789	**Ignore (E)verything**: Für Bestandsaufnahmen. Es werden nur hinzugefügte oder gelöschte Objekte ausgewiesen.
>	+pinug-samc123456789	Growing file: Dateien, die im Umfang stetig wachsen, aber nicht schrumpfen dürfen.
Device (2.2.1)	+pugsdr-intlbamcCMSH	Dateien die Tripwire beim Integritätstest nicht öffnen darf. Darunter fallen zum Beispiel alle Gerätedateien.

Preprozessoren unter Tripwire
Argument Beschreibung
@@ifhost hostname Wahr, wenn hostname mit uname –n oder hostname übereinstimmt.
Wenn der Wert wahr zurückgegeben wird, werden alle
Anweisungen bis zum nächsten `@@endif` oder `@@else` ausgeführt.
@@ifnhost hostname Wahr, wenn hostname mit `uname -n` oder hostname nicht übereinstimmt.
Wenn der Wert wahr zurückgegeben wird, werden alle
Anweisungen bis zum nächsten `@@endif` oder `@@else` ausgeführt.
@@else Wird ausgeführt, falls `@@ifhost`, `@@ifnhost`, `@@ifdef` oder
`@@ifndef` den Wert falsch zurückgeben
@@ifdef Variable Gibt den Wert `true` zurück, wenn die Variable definiert ist
@@ifndef Variable Gibt den Wert `true` zurück, wenn die Variable nicht definiert ist
@@endif Schließt das `@@if*` Statement
@@define Variable
String
Weist der Variable einen String zu, falls kein String angegeben
wird, so wird der Null-String zugewiesen
@@undef Variable Variable ist nicht weiter definiert
@@include Pfadname Zieht die Datei, die unter dem angegebenen Pfadnamen liegt, mit
ein. Diese Datei muss dieselbe Konfigurationssyntax haben.
@@Variable Weist den String, der in Variable gespeichert ist zu.
@@{Variable} Selbe Beschreibung wie das zuvor aufgeführte Argument
|| Logisches ODER
&& Logisches UND

noch zu setzende Variablen des Musterskripts von SUSE 9.2 aus /usr/share/doc/packages/tripwire/twpol.exe

```
@@section GLOBAL
TWROOT="/usr";
TWBIN="/usr/sbin";
TWPOL="/etc/tripwire";
TWDB="/var/lib/tripwire";
TWSKEY="/etc/tripwire";
TWLKEY="/etc/tripwire";
TWREPORT="/var/lib/tripwire/report";
HOSTNAME="wie auch immer";
```

```
# einfaches Tripwire config-file
#
/     R     # Alle Objekte unter / werden ueberwacht.
/usr  R     # Eintrag erforderlich, falls zweiter Partition zugeordnet.
/boot R     # Ebenso, da eigene Partitionen gesondert aufgefuehrt werden.
!/dev       # Uninteressant!
=/tmp       # Nur Verzeichnis, nicht aber den Inhalt ueberwachen.
=/proc            # Auch beim Prozessdateisystem ausreichend.
=/home            # Privat!
/etc/ppp/pap-secrets    R-m   # Zeitstempel unwichtig, da haeufiger Zugriff.
/var/log    L     # Logfiles.
/var/log/messages >     # Monoton wachsende Datei.

# @@include fuegt zur Laufzeit externen Text in tw.config ein. Saemtliche Host-
# spezifischen Eigenheiten koennten in einer separaten Datei beschrieben sein.
@@include /root/tw.host-special

# Hier kommt eine variable Auswahlmaske @@var zum Einsatz, deren jeweilige
# Bedeutung mit der Kommandozeilenoption -Dvar=... vorgegeben werden kann.
# Beim Integritaetstest oder Update muss stets die gleiche Option gewaehlt
# werden - wie bei der Initialisierung! Das Gegenstueck zu -D existiert auch.
# Mit -Uvar kann eine in tw.config formulierte Definition wieder aufgehoben
# werden. (Falls @@var in der Kommandozeile nicht spezifiziert ist, wird
# hier gleich E gesetzt.):
@@ifndef var
@@ define var     E
@@endif
/opt  @@var

# Das Makro @@ifhost ist das wohl komfortabelste Werkzeug zur Anpassung
# an unterschiedliche Rechnerarchitekturen. Im Beispiel wird erreicht, dass
# ein und derselbe Bereich des Dateisystems von Tripwire - je nach Rechner -
# unterschiedlich behandelt wird. (Dazu muss allerdings die Umgebungsvariable
# HOSTNAME, die zur Laufzeit ausgewertet wird, korrekt gesetzt sein.):
@ifhost babyboy.mammabaer.org || babygirl.mammabaer.org
@ define TEMPLATE_S   N
@else
@ define TEMPLATE_S   E
@endif
/var/Honigtopf         @@TEMPLATE_S
# Natuerlich nur bei Baerenkindern relevant!

# Mit @@define kann der Inhalt auch strukturiert werden. Komplexe Konfigura-
# tionsdateien lassen sich damit wesentlich uebersichtlicher gestalten:
@@define private  E
@@define critical R-12+78
@@define secret   N-a
/home/maria @@private
/home/paul  @@private
/root @@critical
/sbin  @@critical
/etc/inetd.conf   @@critical
/etc/hosts.allow  @@critical
/root/banking-details   @@secret
```