EDV-Paranoia unter Linux

Alexander Schreiber (als@informatik.tu-chemnitz.de)

1998-11-13

Security is applied paranoia.

Inhaltsverzeichnis

1 Netzwerksicherheit

1.1 Allgemeine Netzwerksicherheit

Viele Linux-Distributionen haben per default relativ offene Konfigurationen. Eine der ersten Aktionen nach der Installation sollte deshalb das Überarbeiten der Netzwerk-Konfiguration sein :

- Entfernen nicht benötigter Dienste aus `/etc/inetd.conf` :

  Standardmäßig sind dort einige Dienste aktiviert die man besser deaktivieren sollte. Was soll zum Beispiel ein offener POP3 oder IMAP Port auf einem Rechner der nicht dedizierter Mailhost ist ? Und die r-Dienste (rsh, rcp) sollte man tunlichst deaktivieren da sie zu große Sicherheitslöcher darstellen.

- Benutzen der tcp-wrapper

  Sofern dies noch nicht der Fall ist (bei den aktuellen Distributionen *ist* es mittlerweile der Fall) sollte man unbedingt die tcp-wrapper aktivieren. Neben dem Logging aller akzeptierten Verbindungen erlauben sie außerdem eine einfach host-basierte Zugriffssteuerung.

1.2 telnet vs ssh

Diese beiden Dienste bieten Shell-Zugriff auf remote-Rechner. Dabei ist telnet ein Klartext-Protokoll, das bedeutet :

- Authentisierung im Klartext - Login und Passwort gehen *im Klartext* über das Netz. Jeder der einen sniffer hat kann dadurch Passworte „fischen“.
- Sitzungsdaten im Klartext - somit kann jeder die Verbindung mitlesen

Im Gegensatz dazu benutzt ssh kryptographische Protokolle um diese Sicherheitsprobleme zu vermeiden :

- kryptographische Authentisierung - es wird eine sichere Authentisierung ohne Klartextübertragung durchgeführt,
- verschlüsselte Sitzungsdaten - die Sitzungsdaten werden vollständig verschlüsselt, ein Mitschneiden der Verbindung ist somit nutzlos.

1.3 ktcpd-strobemasker

Der ktcpd-strobemasker-Patch ist ein Linux-Kernelpatch welcher sehr nützliche Sicherheits-Features hinzufügt :

- erweitertes TCP-Logging : jede angenommene TCP-Verbindung wird geloggt mit :
  - Source-IP,
  - Source-Port,
  - Destination-IP,
  - Destination-Port,
  - PID, UID und Name des lokal annehmenden Prozesses
- Portscan-Erkennung und (passive) Abwehr : bei Erkennung eines Portscans wird der scannende Host für eine konfigurierbare Zeit ignoriert
- Portscan-Schutz : ein so gepatcher Host sieht bei einem Portscan aus wie ein Mac : keine erkennbaren offenen Ports trotz laufender Dienste und problemlos möglicher Verbindungen,
- Portscan-Logging : es gibt bei diversen Portscannern sogenannte „Stealth-Modi“ bei denen ein Portscan von gescannten Host normalerweise nicht erkannt wird - mit diesem Patch wird auch das geloggt.

1.4 Arpwatch

Das Tools arpwatch überwacht die Zuordnung zwischen Ethernet- und IP-Adressen. Beim Einsatz lernt es alle neuen IP-Ethernet Zuordnungen, trägt diese in eine Liste ein und informiert root via Mail über jeden neuen Host. Sobald sich eine bekannte Kombination ändert wird root via Mail gewarnt. Zudem werden diese und weitere Äuffälligkeiten auch im syslog vermerkt.

1.5 IPChains als Überwachungstool

Man kann IPChains auch sehr gut zur Netzüberwachung einsetzen. Dazu schaltet an des entsprechende Interface in den promiscous mode (wenn es das nicht schon ist), definiert entsprechende loggende Überwachungsregeln (ohne Target, nur mit Log-Option) für verdächtige IP-Pakete (z.B. Back Orifice) und findet dann entsprechende Einträge im Syslog vor.

2 Firewalls

2.1 Was sind Firewalls ?

Als Firewall bezeichnet man einen Rechner der durch geeignete Abschirmtechniken eine Sicherheitssperre zwischen zwei Netzwerken implementiert. Typischerweise sind das auf der inneren Seite in Intranet (z.B. einer Firma) und auf der Aussenseite das Internet. Durch den Firewall wird dabei grundsätzlich der Zugriff von außen nach innen und oft auch von innen nach außen aufgrund bestimmter Regeln beschränkt und überwacht. Dabei ist es natürlich wichtig das jeder Datenverkehr zwischen internem und externem Netz den Firewall passieren muß, da er ansonsten auch umgangen werden kann.

2.2 Was bringt mir ein Firewall ?

Der Einsatz eines Firewall hat meist mehrere Gründe. Die typischen Aufgaben eines Firewalls sind:

- Schutz des internen Netzes vor Angriffen von außen

  Ein potentieller Angreifer der es auf einen Rechner hinter dem Firewall abgesehen hat muß zuerst den Firewall durchdringen. Dieser ist aber normalerweise gut gesichert und auch mit entsprechenden Warnsystemen ausgestattet um einen unerkannten Einbruch maximal zu erschweren.

- Verhinderung unerwünschten Traffics

  Mit den Sperrfähigkeiten eines Firewalls kann unerwünscher Traffic (zum Beispiel in einer Firma IRC (Internet Relay Chat)) in beide Richtungen wirkungsvoll verhindert werden.

- Einsparung von Bandbreite durch erzwungenen Proxy-Zugriff

- Monitoring

  Da sämtlicher Traffic den Firewall passieren muß eignet er sich natürlich auch optimal zum Traffic-Monitoring. Dabei werden sowohl aufgebaute als auch abgelehnte Verbindungen registriert.

- ...

2.3 Firewall-Architekturen

2.3.1 Packet Filtering

Ein Packet Filterung Firewall filtert auf Paket-Ebene, d.h. er entscheidet anhand von :

- IP-Protokollfamilie (ICMP, UDP, TCP),
- Quell- und Zieladresse,
- Quel- und Zielport bei TCP und UDP oder Typecode bei ICMP,
- Eingangs- und Ausganginterface

ob ein Paket weitergeleitet wird oder nicht. In Linux ist seit 1.3.x der Code für Packet Filtering im Kernel - der IP firewalling code, der in Zukunft vom IPChains Code abgelöst wird.

2.3.2 Application Level Gateway

Ein Application Level Gateway setzt auf der Ebene der High-Level Protokolle (HTTP, FTP, ...) an. Dabei wird der Gateway als Proxy zwischen Client und eigentlichen Server geschaltet. Der Gateway nimmt die Anfragen des Client entgegen und reicht sie entsprechend seiner Konfiguration an den eigentlichen Server weiter - oder auch nicht.

2.3.3 Kombinierte Systeme

Da beide System alleine für sich im allgemeinen nicht gerade das Optimum darstellen werden meist kombinierte System verwendet.

2.4 Aufbau eines Firewall-Systems

Grundsätzlich gilt :

- *Absolute Sicherheit gibt es nicht.*

  Ein Firewall verbessert die Sicherheit, aber er ist kein Allheilmittel.

- *Sicherheit hat ihren Preis.*

  Die Sicherung eines System ist immer mit Aufwand (materiell, personell, zeitlich, ..) verbunden.

- *Sicherheit ist selten bequem.*

  Wer sicher sein will muß im allgemeinen Abstriche an der Bequemlichkeit machen weil manche Dinge zwar bequem aber auch zugleich unsicher sind.

Grundsätzlich muß man beim Aufbau eines Firewall-Systems immer Aufwand und Nutzen gegeneinander abwägen : Wieviel Sicherheit brauche/will ich ?

2.4.1 Allgemeine Grundprinzipien

Unabhängig vom verwendeten Firewalling-Konzept gelten einige Grundregeln bei allen Konzepten:

- *keine User auf dem Firewall*

  Auf dem Firewall sollten auf keinen Fall User-Logins existieren sondern lediglich ein Wartungsaccount. Dies vermeidet schon eine ganze Ladung Probleme.

- *nur minimale Dienste auf dem Firewall*

  Auf dem Firewall sollten nur die allernotwendigsten Dienste laufen. Effektiv sollte man bis auf ssh praktisch alle Dienste abklemmen. Und auch diese ssh sollte nur Verbindungen aus dem inneren Netz akzeptieren. Jeder laufende Dienst auf einem Firewall ist ein potentieller Angriffspunkt.

- *aktuellen Snapshot des Firewall-Systems vorhalten*

  Falls doch ein Einbruch auf den Firewall erfolgen sollte so kann man damit den aktuellen Zustand des Firewalls sichern (Band, ...) um in später in Ruhe analysieren zu können. Anschließend plättet man die gesamte Installation (Filesysteme neu anlegen) und spielt das Firewall-System von obigem Snapshot zurück. Dies reduziert im worst-case die Ausfallzeiten des Firewalls drastisch.

2.4.2 Systeme mit DMZ

Dies ist die aufwendige Variante. Dabei befindet sich zwischen 2 Firewall-Systemen eine sog. DMZ (de-militarisierte Zone) in der sich die öffentlich erreichbaren Dienste (Webserver, ...) befinden :

```
Internet - Firewall - DMZ - Firewall - Intranet
```

Dabei befinden sich in der DMZ alle nach außen exponierten Systeme (WWW-Server, FTP-Server, Gateways, ...). Diese Maschinen werden von den Maschinen des inneren Netzes als grundsätzlich nicht vertrauenswürdig eingestuft - es könnte ja sein daß jemand in diese Maschinen eingebrochen ist. Die Idee hinter der DMZ ist es, einem potentiellen Angreifer zum einen mehrere Barrieren in den Weg zu legen und zum anderen die sensitiven Systeme so weit wie möglich aus seiner Reichweite zu entfernen.

Bis ein Angreifer es in eine Maschine im internen Netz geschafft hat muß er somit mindestens 2 Firewalls und ggf. weitere Maschinen „knacken“ - und dabei sollte er *einige* Alarme auslösen.

Die Maschinen in der DMZ sollten natürlich auch auf sicher konfiguriert sein, d.h. es sollten (neben dem sshd für den Admin) nur die Dienste darauf laufen die wirklich benötigt werden - je weniger offene Dienste, desto weniger potentielle Angriffspunkte. Weiterhin sollten dort auch keine User-Logins existieren sofern das nicht absolut unvermeidbar ist.

Auch hier gilt das die für Maschinen Snapshots existieren sollten die im Falle einer Kompromittierung eine schnelle Wiederherstellung der Funktion erlauben. Eine genaue Analyse kann man später in Ruhe anhand des vor der Wiederherstellung gemachten Abzuges der Maschine machen.

2.4.3 Firewall-Policies

Grundsätzlich sollte das Prinzip gelten : es ist alles verboten was nicht explizit erlaubt ist. Es sollten auf dem nur die Verbindungen durch den Firewall erlaubt werden die man als hinreichend sicher betrachten kann und die auch als notwendig erachtet werden, zum Beispiel bei incoming :

- ssh,
- http (wenn Webserver hinter Firewall),
- ftp (wenn FTP-Server hinter Firewall),
- smtp,
- evtl. weitere Dienste

Mit Ausnahme von ssh sollten diese Verbindungen jedoch nur zu den für diese Dienste vorgesehen Maschinen erlaubt werden. Bei den outgoing Verbindungen kann man entweder den offenen Ansatz verfolgen und prinzipiell alles erlauben oder aber man bleibt auf der sicherheitsbewußten Seite und erlaubt nur bestimmte Verbindungen. So ist es zum Beispiel durchaus üblich lediglich outgoing http zu erlauben und auch daß nur über einen Proxy.

Während TCP relativ gut zu verfolgen ist da es sich um ein verbindungsorientiertes Protokoll handelt ist dies bei UDP praktisch nicht möglich da bei diesem paketorientierten Protokoll keine Verbindungen aufgebaut werden. Es ist deshalb durchaus üblich (und sinnvoll) UDP entweder bis auf Port 53 (DNS) oder auch vollständig beidseitig zu sperren.

Auch wenn dies von einigen Firewall-Admins gemacht wird so sollte man ICMP durch den Firewall entweder überhaupt nicht oder zumindest nicht vollständig sperren. Es sollten zumindest „fragmentation needed“ und „unreachable“ zugelassen werden. Gerade ersteres wird dringend für die MTU-Discovery benötigt ohne die die Datenrate katastrophal niedrig werden kann.

2.5 Verbergen der internen Struktur

Ein Firewall sollte grundsätzlich - soweit möglich - die interne Struktur des von ihm geschützten Netzes nach außen verbergen. Wenn diese Struktur (benutze IP-Adressbereiche, gültige IP-Adressen, IP-Adressen von Maschinen mit bestimmter Bedeutung, ...) bekannt ist lassen sich bereits detailliertere Angriffsvorbereitungen planen. Normalerweise wird durch das Sperren der meisten incoming TCP und UDP Pakete die interne Struktur recht gut verborgen, aber es gibt eine als **Firewalking** bezeichnete Technologie[1] die es bei den meisten Netzen ohne NAT erlaubt ein Mapping der internen Struktur durchzuführen.

2.5.1 Firewalking

Diese Technologie basiert im wesentlichen auf einer Analyse von IP-Antwortpaketen ähnlich dem Utility `traceroute`. Mit Firewalking ist es möglich zum einen die offenen Ports eines Firewall-Systems zu finden, zum anderen kann man damit recht gut das Netzwerk dahinter ausmappen um Routern und normale Maschinen zu finden.

2.5.2 NAT oder masquerading Firewalls

Man kann viele dieser Angriffe von vornherein erheblich erschweren wenn der Firewall NAT[2] macht. In der einfachsten Variante, der m-zu-1 Umsetzung werden alle Verbindungen aus dem internen Netz masqueradiert, sie scheinen alle vom Firewall auszugehen. Die internen Hosts sind dann auch nicht von außen erreichbar (`ping, traceroute`).

3 Verschlüsselte Virtual Private Networks

Virtual Private Networks (VPN) sind eine Technologie die über ein existierendes Netzwerk als eine weitere Schicht ein virtuelles Netzwerk legt. Typischer Einsatzzweck ist ein einheitliches Netz einer verteilten Organisation (z.B. Firma mit verschiedenen miteinander vernetzten Niederlassungen). Durch IP-IP-Tunneling ist es möglich ein VPN auf andere existierende Netzwerke wie z.B. das Internet aufzubauen. Der Einsatz von Verschlüsselung kann die Sicherheit eines solchen Netzes drastisch erhöhen, da ein Analysieren des Traffics nicht mehr möglich ist.

Für Linux gibt es u.a. das **CIPE**-Paket. CIPE steht für **C**ryptographic **IP E**ncapsulation. CIPE tunnelt IP-Pakete in UDP-Paketen und baut damit Punkt-zu-Punkt Verbindungen auf. Dabei hat jede Verbindung ihren eigen Link-Key der nur den beiden Verbindungsendpunkten bekannt sein darf. Die Implementation besteht aus einem Kernelmodul das sich um das Handhabung der IP-Pakete kümmert und einem Daemon - vergleichbar dem PPP Daemon - für den Verbindungsauf- und abbau zuständig ist.

[1] David Goldsmith, Michael Schiffman : „Firewalking“, October 1998

[2] Network Adress Translation

4 Systemsicherheit allgemein

4.1 Logging

Eine sehr wichtiges Element in der Sicherheit eines Systems ist das Logging. Grundsätzlich sollten alle loggenden Dienste mit dem maximal akeptablen Loglevel laufen, das produziert zwar sehr viele Log-Einträge, aber das ist besser als im Ernstfall nicht zu wissen ob ein bestimmtes logbares Ereignis eingetreten oder nicht.

4.1.1 Wohin mit den Logdaten ?

Ich empfehle dazu folgendes Vorgehen : alle Logmeldungen werden zentral in `/var/log/messages` abgelegt, ebenfalls eine volle Kopie geht auf `/dev/tty12` (evtl. nicht bei öffentlich zugänglichen Maschinen) und schließlich wird alles auch noch auf einen dedizierten Loghost geschrieben.

4.1.2 Schutz der Logs vor Angreifern

Eine Standardprozedur bei einem Einbruch in ein System ist die Manipulation der Systemlogfiles. Da diese aber wiederum bei der Erkennung und Bekämpfung eines Systemeinbruchs eine wichtige Rolle spielen sollten sie vor Veränderung durch einen Angreifer sicher geschützt werden. Welche Möglichkeiten gibt es dazu ?

- Schreiben der Logs auf WORM (Write Once Read Many),

  Eine sicherlich sehr sichere Methode, allerdings nicht nur etwas wenig einfach zu handhaben sondern auch nicht ganz billig.

- Schreiben auf Hardcopy-Device (Drucker)

  Praktisch nicht sinnvoll - wer will schon Hunderte von Seiten an Log-Ausdrucken von Hand analysieren. Dazu u.U. enormer Papierverbrauch ...

- dedizierter gesicherter Logarchiv-Rechner am Loghost

  Die meiner Meinung nach günstigste Lösung. Sie soll im folgenden weiter erläutert werden.

Für die Variante des dedizierten Logarchiv-Rechners brauchen wir nur sehr wenig :

- ein alter Rechner (80386 mit 8 MB sollte reichen),
- eine hinreichend große Platte (muß skeletales Linux (50 MB) plus anfallende Logdaten aufnehmen können),
- ein serielles Kabel zwischen Loghost und Logarchiv-Rechner.

Die Grundidee ist relativ simpel : der syslogd auf dem Loghost schreibt auf die serielle Schnittstelle an der Logarchiv-Rechner hängt. Auf dem Logarchiv-Rechner läuft ein Prozess der die Daten von der seriellen Schnittstelle liest und in ein File schreibt welches ggf. regelmässig rotiert wird. Der Logarchiv-Rechner darf natürlich kein (aktives) Netzwerk-Interface haben. Damit haben wir eine Lösung die unsere Logfiles vor nachträglicher Veränderung durch einen Angreifer schützt da dazu physischer Zugang zum Logarchiv-Rechner notwendig ist. Die einzig verbleibende Angriffsmöglichkeit ist eine Flooding Attack - d.h. Logmessages schneller erzeugen als sie über die serielle Leitung auf den Logarchiv-rechner übertragen werden können. Aber da kann man mit entsprechenden Tools diese Situation erkennen und ggf. Alarm auslösen.

4.1.3 Auswertung der Logs

Jetzt liegen jede Menge Logdaten vor - was macht man nun damit ? Nun, zum einen ist ein großer Teil der Logdaten im normalen Betrieb eher weniger interessant - er wird es u.U. erst bei der Analyse eines Einbruchs, zum anderen ist es schlichtweg unmöglich die Logs zu lesen und so nach Auffälligkeiten zu suchen. Es wird eine automatische Auswertung der Logfiles gebraucht die im regulären Betrieb lediglich ein paar Statistiken liefert und auf alles hinweist was evtl. auffällig ist. Dazu gibt es jede Menge Tools - und wenn das nicht reicht schreibt selber etwas geeignetes. Die Grundidee ist eigentlich immer das man Muster spezifiziert nach denen in den Logfiles gesucht werden soll. Aus den so ausgegrabenen Daten kann man dann Reports und Statistiken bauen (lassen) die einen Überblick über den Zustand des Systems geben sollten und zugleich auf Äuffälligkeiten hinweisen müssen.

4.2 Tripwire

Tripwire ist ein Tool zur Überwachung der logischen Filesystemintegrität. Es kann Veränderungen an existierenden Dateien (sowohl Inodedaten als auch Dateinhalt), wachsende Dateien (Logfiles), verschwundene und neue Dateien erkennen. Damit ist Tripwire ein sehr wertvolles Tool zur Erkennung von Veränderungen die auf einen Systemeinbruch hindeuten. Zu finden ist es unter

`http://www.visualcomputing.com/tripwire`

Tripwire legt dazu nach einem Konfiguratiosnfile eine Datenbank mit dem aktuellen Zustand des Filesystems an und vergleicht diese dann auf Anforderung mit dem Ist-Zustand. Eventuelle Differenzen werden berichtet. Sehr wichtig beim Einsatz von Tripwire ist die Tatsache daß sich die Datenbank auf einem read-only medium befinden muß da sie sonst zu einfach von einem Angreifer manipuliert werden kann.

4.3 secure-linux

Der secure-linux Kernel-Patch ändert das Verhalten des Linux-Kernels an einigen sicherheitsrelevanten Stellen :

- nicht-ausführbarer User-Stack :

  Der Stack wird nichtausführbar, damit werden Bufferoverflow-Exploits (eine Standard-Angriffstechnik) wesentlich schwieriger.

- beschränkte Links in `/tmp` :

  Es ist damit für User unmöglich in `/tmp` Hardlinks zu Files zu erzeugen die ihnen nicht gehören. Dies ist eigentlich sogar Soll-Verhalten da man eine solche Datei nicht wieder löschen kann. Daran gekoppelt ist ein Logging für versuchte Exploits. Betrifft `+t` Verzeichnisse.

- beschränkte Pipes in `/tmp` :

  Verbietet das Schreiben in Pipes die dem User (außer root) nicht gehören in `+t` Verzeichnissen. Erschwert datenfälschende Angriffe (gcc-Exploit).

- beschränktes `/proc` :

  Nicht-root User können nur ihre eigenen Prozesse sehen und können keine aktiven Netzwerkverbindungen sehen - es sei den sie gehören einer speziellen Gruppe an.

Zu finden ist dieser Patch unter :

`http://www.false.com/security/linux/`

4.4 Immunix StackGuard Compiler

Der StackGuard Compiler versucht das Problem der buffer overflows dadurch zu bekämpfen indem an den relevanten Stellen auf dem Stack Kontrollworte, sogenannte „canaries“ (Kanarienvögel) untergebracht werden. Beim Rücksprung aus einer Funktion werden diese canaries überprüft und wenn festgestellt wird daß sich der Wert geändert hat erfolgt ein Eintrag in das Syslog über einen versuchten buffer overflow und der Prozess terminiert. Um diese Technologie anzuwenden muß man lediglich die relevanten Programme mit dem StackGuard Compiler neu kompilieren. Der Performance-Verlust der durch den canary-check verursacht wird hält sich in tolerablen Grenzen - Sicherheit kostet eben.

4.5 SUID-Binaries

Eine beliebte Einbruchsstelle sind Bugs in SUID root Binaries, typischerweise buffer overflows. Gegen das Problem der buffer overflows gibt es wie oben erwähnt mehrere Ansatze - trotzdem sollte man das Problem noch früher angehen :

- Systemweite Suche nach SUID root Binaries, überprüfen ob es ein legitimes Binary ist (und kein Backdoor), ob das SUID-Bit wirklich benötigt wird und dem Binary gegebenfalls das SUID-Bit wegnehmen oder es löschen,
- alle Filesysteme in die User schreiben können mit `nosuid` mounten und wenn man schon mal dabei ist kann man auch gleich `nodev` setzen - Device-Files haben in `$HOME` nichts verloren.