Abwehr von SSH-Bruteforce-Angriffen

Sicherheitstage SS 2006

Hergen Harnisch

harnisch@rrzn.uni-hannover.de

14.06.2006

Bruteforce-Angriffe

Zugriffsbeschränkung:

- User
- Quell-IP
- Keys

Versuchsbeschränkung:

- PAM-Module
- IP-Tables

Sonstiges

Nachtrag PAM-Cracklib

Log-Auszug

SSH-Port-Scan

```
Apr 23 12:33:43 HOST sshd[4476]:
  Did not receive identification string from 140.115.35.XXX
Apr 23 13:39:48 HOST sshd[6560]:
  Did not receive identification string from 161.53.202.XXX
```

Erfolgloser Login-Versuch

```
Feb 23 13:35:19 HOST sshd[14196]:
  error: PAM: Authentication failure for root from HOST
```

Log-Auszug: User durchprobieren

```
Feb 14 05:03:08 HOST sshd[16639]: Invalid user adelbert from 81.31.26.XXX
Feb 14 05:03:09 HOST sshd[16647]: Invalid user adrian from 81.31.26.XXX
Feb 14 05:03:09 HOST sshd[16654]: Invalid user albert from 81.31.26.XXX
Feb 14 05:03:10 HOST sshd[16662]: Invalid user albrecht from 81.31.26.XXX
Feb 14 05:03:10 HOST sshd[16670]: Invalid user alexander from
81.31.26.XXX
Feb 14 05:03:11 HOST sshd[16678]: Invalid user alfred from 81.31.26.XXX
Feb 14 05:03:11 HOST sshd[16686]: Invalid user ali from 81.31.26.XXX
Feb 14 05:03:12 HOST sshd[16695]: Invalid user andreas from 81.31.26.XXX
Feb 14 05:03:12 HOST sshd[16699]: Invalid user anton from 81.31.26.XXX
Feb 14 05:03:13 HOST sshd[16707]: Invalid user arnim from 81.31.26.XXX
Feb 14 05:03:13 HOST sshd[16712]: Invalid user arno from 81.31.26.XXX
Feb 14 05:03:14 HOST sshd[16719]: Invalid user august from 81.31.26.XXX
Feb 14 05:03:14 HOST sshd[16725]: Invalid user axel from 81.31.26.XXX
Feb 14 05:03:14 HOST sshd[16731]: Invalid user bastian from 81.31.26.XXX
Feb 14 05:03:15 HOST sshd[16735]: Invalid user benedikt from 81.31.26.XXX

Feb 14 05:03:15 HOST sshd[16741]: Invalid user benjamin from 81.31.26.XXX
```

Log-Auszug: Passwörter durchprobieren

```
Apr 19 06:31:23 HOST sshd[14167]: Invalid user guest from 125.181.36.XXX
Apr 19 06:31:26 HOST sshd[14170]: Invalid user guest from 125.181.36.XXX
Apr 19 06:31:29 HOST sshd[14172]: Invalid user guest from 125.181.36.XXX
Apr 19 06:31:32 HOST sshd[14174]: Invalid user guest from 125.181.36.XXX
Apr 19 06:31:34 HOST sshd[14176]: Invalid user guest from 125.181.36.XXX
Apr 19 06:31:37 HOST sshd[14178]: Invalid user guest from 125.181.36.XXX
Apr 19 06:31:40 HOST sshd[14180]: Invalid user guest from 125.181.36.XXX
...
Apr 23 10:52:47 HOST sshd[29349]: Invalid user jim from 125.181.36.XXX
Apr 23 10:52:50 HOST sshd[29351]: Invalid user jim from 125.181.36.XXX
Apr 23 10:52:53 HOST sshd[29353]: Invalid user jim from 125.181.36.XXX
Apr 23 10:52:56 HOST sshd[29356]: Invalid user jim from 125.181.36.XXX
Apr 23 10:53:02 HOST sshd[29358]: Invalid user jim from 125.181.36.XXX
Apr 23 10:53:06 HOST sshd[29360]: Invalid user jim from 125.181.36.XXX
Apr 23 10:53:09 HOST sshd[29362]: Invalid user jim from 125.181.36.XXX
```

Durchprobieren von Username/Passwort-Kombinationen am SSH-Dienst:

- übliche Usernames: `root`, `guest`, `test`, `mueller`
- teilweise schnell, teilweise langsam & kontinuierlich
- teilweise eine Quell-IP, teilweise verteilt (Bot-Netz)
- Wörterbuch-Attacken

- kein Implementationsfehler von SSH
- bekanntes Problem von Passwort-Authentifizierung

Maßnahmen

- Reduktion auf nötige User
  Alt-Accounts löschen,
  nicht jeder benötigt Fernzugriff
- letzten Login anzeigen, Logins aufzeichnen
- Remote-Login nur mit personalisierten Accounts
  keine Funktionsaccounts wie `root`, erleichtert Nachvollziehbarkeit & Passwort-Änderung
- User zu guten Passwörtern anhalten, besser zwingen

Umsetzung

z.T. technisch (s.u.), aber auch: Organisation, Aufklärung, ...

Umsetzung User-Maßnahmen

Einträge in `/etc/ssh/sshd_config`

- `AllowUsers` / `DenyUsers`:
  einzelne User zulassen oder ausschließen
- `AllowGroups` / `DenyGroups`:
  Mitglieder von Gruppen zulassen oder ausschließen
  z.B. Gruppe `sshusr` ($\neq$ `ssh`) anlegen, analog zu `audio`, `floppy`
- `PermitRootLogin no`
- `PrintLastLog yes`

Sonstiges

- gute Passwörter durch regelmäßige Crack-Versuche, PAM
- Zugangsrestriktionen auch über PAM möglich

R|R|Z|N

Maßnahme

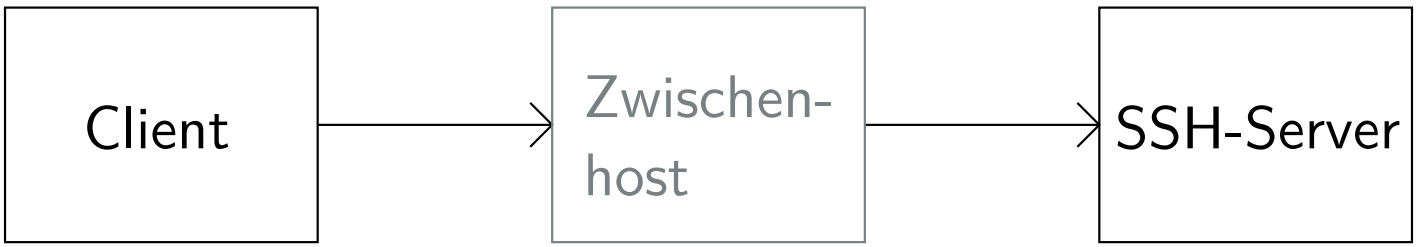
Einschränkung der zugelassenen Systeme, z.B. nur `130.75.*`.

Vorteile

- wirkungsvoll gegen Scans von außerhalb
- schnell umsetzbar

Nachteile

Umgehung durch legitime Nutzer meist durch Zwischenhosts:



Problem: Passwort-Eingabe auf evt. unsicherem Zwischenhost

R|R|Z|N

Maßnahme

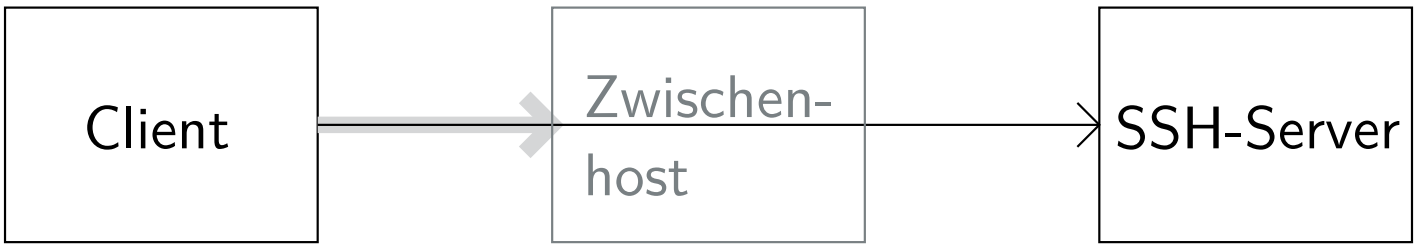
Einschränkung der zugelassenen Systeme, z.B. nur `130.75.*`.

Vorteile

- wirkungsvoll gegen Scans von außerhalb
- schnell umsetzbar

Nachteile

Umgehung durch legitime Nutzer meist durch Zwischenhosts:



Tunnelung wird meist sowieso nicht genutzt

R|R|Z|N

Umsetzung

TCP-Wrapper

- SSH mit TCP-Wrapper-Support übersetzen, meist Standard
- nur erlaubt, wenn in `hosts.allow` oder nicht in `hosts.deny`
  - `/etc/hosts.allow`:
    ```
    sshd: 130.75.0.0/16 127.0.0.1/32
    ```
  - `/etc/hosts.deny`:
    ```
    sshd: ALL
    ```
- komplizierte Bedingungen mit `*`, Listen, `EXCEPT` möglich

IP-Tables

zusätzliche Absicherung über IP-Tables

R|R|Z|N

Maßnahme

Bruteforce-Angriffe nutzen Passwort-Authentifizierung, daher Verzicht auf Passwörter.

Umsetzung

Verwendung alternativer Authentifizierungen

- Public-/Private-Key-Paare ⟵ *Mittel der Wahl!*
- PKI & Zertifikate ⟵ wenige Clients/Server
- Einmal-PW: S/Key, OTP; Tokens ⟵ aufwändig bzw. teuer
- (keine Nutzerauthentifizierung: hostbasiert) ⟵ unsicher

Public-/Private-Keys

- Prinzip entspricht den Keys in einer PKI
- Key-Paar: geheimer Private-Key, publizierbarer Public-Key
- Private-Key nicht ratbar, da lang & komplex
- Private-Key ist geheim zu halten,
  mit Passphrase zu schützen
- Public-Key dient der Überprüfung des Private-Keys
- Key-Paare in SSH:
  - Host-Keys: meist automatisch generiert,
    Kenntnis schützt vor Man-in-the-Middle
  - User-Keys: zu generieren, ggf. mehrere Paare,
    authentifiziert User

R|R|Z|N

User-Key-Paar

User-Einstellung

- auf Client: Schlüsselgenerierung mit `ssh-keygen -t dsa`
- Public-Key auf Server kopieren, dort anhängen an `~/.ssh/authorized_keys`
- auf Client in `~/.ssh/config` evt.
  `Host` *Servername oder -IP*
  `User` *Username auf Server*
  `IdentityFile ~/.ssh/`*Private-Key-File*

Server-Einstellung

in `/etc/ssh/sshd_config`:

- alle zwingen: `PasswortAuthentication no`
- mindestens `root`: `PermitRootLogin without-password`

Beispiel Key-Dateien

Private-Key ~/.ssh/id_dsa

```
---BEGIN DSA PRIVATE KEY---
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,3FF18C4F42C69C6B

3ZO8sRiiDsWOAdP2wYWQlC9MPadHgwf2mqGEzb30wFIPcRb+7U4cxbqw/7IoZFcj
...
n7k6mhTz5lI8mdSLWjS94OcdmAprRnSa
---END DSA PRIVATE KEY---
```

Public-Key ~/.ssh/id_dsa.pub

```
ssh-dss AAAAB3N ...fu4VCjT  User@Host
```

Versuchsbeschränkung:

Ziel

Anzahl möglicher Rateversuche reduzieren / begrenzen

erreichen mit

Timeouts nach fehlgeschlagenen Versuchen

bei SSH

SSH bietet `MaxAuthTries` in `/etc/ssh/sshd_config`:

- begrenzt Versuchszahl je Verbindung
- aber erneute Verbindung sofort möglich
  $\longrightarrow$ praktisch wirkungslos

R|R|Z|N

PAM

- Pluggable Authentication Modules:
  flexible User-Authentifizierung
  verschiedene Methoden, für verschiedene Dienste, auch PW ändern
- *Dienste:* Programme, die PAM nutzen
- *Module:* für verschiedene Methoden; aber auch
  verschiedene Funktionen (*Modultyp*):

auth	eigentliche Nutzer-Authentifizierung
account	Authorization (z.B. Zeitbeschränkung)
session	Sitzungseinstellungen (z.B. automatisches Mounten)
password	Passwort-Aktualisierung

PAM-Nutzung durch SSH

aktivieren in `/etc/ssh/sshd_config`: `UsePAM yes`

Versuchsbeschränkung: PAM-Module

Speicherorte

- Module in `/lib/security/`
- Konfiguration in `/etc/pam.d/*` (eher nicht `/etc/pam.conf`)

Konfiguration

- Zeilen der Form
  `Modul-Typ Kontroll-Flag Modul-Datei Modul-Argumente`
- in Reihenfolge Abarbeitung aller Zeilen des benötigten Typs, meist werden mehrere Module aufgerufen; einzelne Module melden Erfolg/Fehlschlag
- Kontroll-Flags: `required`, `requisite`, `sufficient`, `optional` und `inlude` (bald neue Syntax, feiner)

z.B. /etc/pam.d/su

```
1 auth     sufficient     pam_rootok.so
2 session     required      pam_env.so readenv=1
3 session     optional      pam_mail.so nopen
4 @include common-auth
5 @include common-account
6 @include common-session
```

in 1 root darf ohne weitere Prüfung rein

in 2 Umgebungsvariablen beim Login setzen (erweitert)

in 3 bei su keine "New MailMeldung

4–6 ansonsten wie normales Login

R|R|Z|N

pam_tally

- meist Standardumfang PAM
- Funktionsweise:
  1. zählt Fehlversuche pro Account (egal woher)
  2. sperrt nach gewisser Anzahl ⟵ DoS-Angriff leicht
  3. setzt Anzahl nach gewisser Zeit zurück
- Anzahlabfrage & -Rücksetzung mit `pam_tally`-Kommando

Aktivierung in `/etc/pam.d/ssh(d)`

```
1 account    required    pam_tally.so deny=3 reset \
                         unlock_time=300 no_magic_root per_user
2 auth    required    pam_tally.so no_magic_root
```

⟶ wg. DoS (und Timeout/Reset-Verhalten) nicht empfehlenswert

pam_abl (auto blacklisting)

- meist nicht Standardumfang, nachzuinstallieren;
  für Debian als libpam-abl schon länger in Vorbereitung
- Funktionsweise:
  1. Logging von Fehlversuchen je Host und/oder je User
  2. sperrt nach konfigurierbaren Regeln (sehr flexibel)
  3. hebt Sperrung nach gewisser Zeit auf
- Statusabfrage & Rücksetzung mit pam_abl-Kommando

Aktivierung

- in /etc/pam.d/ssh(d) vor eigentlicher Authentifizierung:

```
auth    required    /lib/security/pam_abl.so \
                        config=/etc/security/pam_abl.conf
```

- eigentliche Konfiguration in /etc/security/pam_abl.conf

/etc/security/pam_abl.conf

```
1 host_db=/var/lib/abl/hosts.db
2 user_db=/var/lig/abl/users.db
3 host_purge=2d
4 user_purge=12h
5 host_rule=*:10/1h
6 user_rule=!root:5/1h,20/1d
```

1,2 Speicherort von Fehlzugriffen (bzw. keine Aufzeichnung)

3,4 Aufzeichnungslänge

5 alle Hosts ab 10 Fehlversuchen in der letzten Stunde

6 außer root: Sperrung ab 5 pro Stunde oder 20 pro Tag

pam_abl-Regeln

- `*:10/1h`
  alle User bzw. Hosts bei 10 pro Stunde
- `*:10/1h,20/1d`
  alle User bzw. Hosts bei 10 pro Stunde *oder* 20 pro Tag
- `root:10/1h`
  gilt nur für User root
- `!root:10/1h`
  gilt für alle User außer root
- `*/sshd:10/1h`
  gilt für alle User/Hosts, aber nur für den SSH-Dienst
- `*:10/1h root/sshd:5/1h`
  alle ab 10/Stunde, root bei ssh schon ab 5/Stunde

leider nicht alle Details dokumentiert (z.B. `!root/sshd`)

R|R|Z|N

Netfilter/IP-Tables

IP-Tables ermöglicht nicht nur Beschränkung

- der zugelassenen Quell-IPs

sondern auch

- der Gesamtzahl aktiver Verbindungen,
- der Häufigkeit von Verbindungsaufbauten

und das auch je Dienst (Port) und Quell-IP.

$\longrightarrow$ mehr dazu morgen in Netfilter/IP-Tables

Port $\neq$ 22

- schützt vor simplen Scanversuchen
- security by obscurity
- stört auch legitimen Nutzer
- Firewall-Regeln müssen Port zulassen, auch auf Nutzerseite

rechtliche Absicherung

„Belehrung“ mit `banner` in `/etc/ssh/sshd_config`

z.B. Verweis auf StGB:

Datenausspähen §202a, Datenveränderung §303a, Computersabotage §303b

Aktivierung in /etc/pam.d/common-password

```
password    required    pam_cracklib.so \
                                retry=3 minlen=6 difok=3
```

Debian: schon drin, aber nach Installation auskommentiert;
Installation von `libpam-cracklib` notwendig

Suse: benutzt stattdessen `pam_pwcheck`

- zusätzliche Passwort-Prüfung („required“),
  eigentliche Änderung mit üblichem Modul (`pam_unix`)
- Optionen:
  - `retry`: 3 Eingabeversuche vor Abbruch
  - `minlen`: minimale Passwortlänge
  - `difok`: mind. 3 Buchstaben gegenüber altem PW neu