fail2ban: SSH automatisch schützen

16.11.2026, ergänzt 17.11.2026 · @Sebastian Ebner · Quelle: KI Werkzeug Cloude

> Für Ubuntu 24.04 und Debian 12/13. Platzhalter: `ADRESSE` (eine IP-Adresse), `NETZ` (z. B. `203.0.113.0/24`), `THEMA` (ein ntfy-Thema). Alle Befehle als root.

1. Was fail2ban macht

Jeder Server mit offenem SSH-Port wird rund um die Uhr von automatischen Programmen angegriffen, die Passwörter durchprobieren. fail2ban liest die Anmeldeprotokolle mit, erkennt wiederholte Fehlversuche und **sperrt die Absender-Adresse automatisch** in der Firewall.

Begriff	Bedeutung
Filter	Suchmuster, das Fehlversuche in einem Protokoll erkennt (z. B. `filter.d/sshd.conf`)
Jail	Verbindung aus Filter, Protokollquelle und Regeln (wie viele Versuche, wie lange Sperre), z. B. `[sshd]`
Action	Was beim Sperren passiert: Firewall-Regel setzen, optional Mail oder Benachrichtigung (`action.d/…`)

fail2ban sperrt nur die Dienste, für die ein Jail aktiv ist, hier SSH. WireGuard, Webserver und Container bleiben unberührt.

2. Installation und Grundeinrichtung

```
apt install -y fail2ban
nano /etc/fail2ban/jail.local
```

Grundkonfiguration in `/etc/fail2ban/jail.local`:

```
[DEFAULT]
bantime  = 1d
findtime = 10m
maxretry = 5
bantime.increment = true
bantime.maxtime   = 4w
ignoreip = 127.0.0.1/8 ::1

[sshd]
enabled = true
backend = systemd
mode    = aggressive
port    = ssh
```

```
systemctl enable --now fail2ban
fail2ban-client status sshd
```

Teil	Bedeutung
`enable`	fail2ban startet künftig bei jedem Hochfahren des Servers automatisch
`--now`	fail2ban startet sofort, ohne Neustart

Der Befehl wird nur **einmal** gebraucht. Danach läuft fail2ban dauerhaft, auch nach einem Neustart des Servers.

Wichtig: Eigene Einstellungen immer in `jail.local` (bzw. `jail.d/*.conf`) schreiben, nie in `jail.conf`. `jail.conf` wird bei Updates überschrieben, `jail.local` nicht.

---

3. Die Grundeinstellungen

Einstellung	Standard	Bedeutung	Beispiel
`bantime`	`10m`	Sperrdauer. Einheiten: `s`, `m`, `h`, `d`, `w`. `-1` = dauerhaft	`bantime = 1d`
`findtime`	`10m`	Zeitfenster, in dem die Fehlversuche gezählt werden	`findtime = 10m`
`maxretry`	`5`	Anzahl Fehlversuche innerhalb von `findtime` bis zur Sperre	`maxretry = 5`
`ignoreip`	`127.0.0.1/8 ::1`	Adressen, Netze oder Hostnamen, die **nie** gesperrt werden (Leerzeichen getrennt)	`ignoreip = 127.0.0.1/8 ::1 NETZ`
`ignoreself`	`true`	eigene Adressen des Servers nie sperren	`ignoreself = true`
`enabled`	`false`	Jail ein- oder ausschalten	`enabled = true`
`port`	je Jail	Port(s), die gesperrt werden. Bei geändertem SSH-Port anpassen	`port = 2222`

Faustregel: Strenger wird es mit kleinerem `maxretry`, größerem `findtime` und längerem `bantime`. Zu streng sperrt man sich bei Tippfehlern selbst aus.

---

4. Wiederholungstäter länger sperren

Einstellung	Bedeutung
`bantime.increment = true`	Jede weitere Sperre derselben Adresse dauert länger (Standard: 1×, 2×, 4×, 8× … der `bantime`)
`bantime.maxtime = 4w`	Obergrenze, über die die Sperre nicht hinauswächst
`bantime.factor = 1`	Faktor für das Wachstum. Größer = schnelleres Anwachsen
`bantime.multipliers = 1 5 30 60 300 720 1440 2880`	Eigene Stufen statt Formel (Vielfache der `bantime`)
`bantime.formula = …`	eigene Berechnungsformel (nur für Fortgeschrittene)
`bantime.rndtime = 300`	zufällige Zusatzzeit in Sekunden, damit Botnetze das Ende der Sperre nicht genau berechnen können
`bantime.overalljails = true`	Wiederholung wird über **alle** Jails gezählt, nicht nur im aktuellen

Zusätzlich: Jail `recidive`. Es liest das eigene Protokoll von fail2ban und sperrt Adressen, die in kurzer Zeit mehrfach gesperrt wurden, für lange Zeit auf **allen** Ports:

```
[recidive]
enabled   = true
logpath   = /var/log/fail2ban.log
banaction = %(banaction_allports)s
bantime   = 1w
findtime  = 1d
maxretry  = 3
```

5. Langsame Angreifer: zweites SSH-Jail

Die Regel „5 Versuche in 10 Minuten“ stoppt schnelle Angriffe. Viele Botnetze probieren aber absichtlich **langsam**: z. B. ein Passwort alle 20 bis 30 Minuten, jedes Mal mit einem anderen Benutzernamen und oft parallel auf vielen Servern. So bleiben sie dauerhaft unter der Grenze und werden nie gesperrt.

Dagegen hilft ein **zweites Jail** mit demselben SSH-Filter, aber einem langen Zählzeitraum. Unter `[sshd]` in `jail.local` anfügen:

```
[sshd-langsam]
enabled  = true
filter   = sshd[mode=aggressive]
backend  = systemd
port     = ssh
findtime = 1d
maxretry = 10
bantime  = 1w
```

Zeile	Bedeutung
`[sshd-langsam]`	frei gewählter Name des zweiten Jails
`filter = sshd[mode=aggressive]`	derselbe SSH-Filter wie bei `[sshd]`. Muss hier angegeben werden, weil fail2ban den Filter sonst über den Jail-Namen sucht (`sshd-langsam.conf` gibt es nicht)
`findtime = 1d`	zählt die Fehlversuche über 24 Stunden
`maxretry = 10`	10 Fehlversuche in 24 Stunden führen zur Sperre
`bantime = 1w`	Sperre für eine Woche

Beide Jails arbeiten gleichzeitig: `[sshd]` für schnelle, `[sshd-langsam]` für langsame Angreifer. `ignoreip` und `bantime.increment` aus `[DEFAULT]` gelten automatisch für beide.

```
fail2ban-client -t
systemctl restart fail2ban
fail2ban-client status
fail2ban-client status sshd-langsam
```

`status` muss jetzt `Jail list: sshd, sshd-langsam` zeigen. Beim Start liest das Jail das Journal der letzten 24 Stunden nach und sperrt passende Adressen sofort.

Eigene Erfahrung: Auf zwei Servern gingen mit der Grundeinrichtung die Passwortversuche um rund 95 % zurück. Übrig blieb ein Botnetz, das auf beiden Servern im gleichen Takt etwa alle 26 Minuten einen Versuch machte. Das zweite Jail sperrte direkt beim Start **46 Adressen je Server**, davon 36 auf beiden Servern gleich.

Grenze: Adressen, die nur wenige Male am Tag probieren, erfasst keine Zählregel. Ganz wirkungslos werden Passwort-Angriffe nur mit Anmeldung per SSH-Schlüssel (siehe Ende).

---

6. Was gezählt wird: `mode` beim SSH-Filter

Modus	Was zählt als Fehlversuch
`normal` (Standard)	falsches Passwort, unbekannter Benutzer, gesperrter Benutzer
`ddos`	Verbindungen, die vor der Anmeldung abbrechen oder keinen gültigen SSH-Austausch machen (typisch für Scanner)
`extra`	weitere Auffälligkeiten, z. B. keine passenden Anmeldeverfahren
`aggressive`	alles zusammen: normal, ddos und extra

Achtung bei `aggressive`: Auch eigene Verbindungen, die vor der Anmeldung abbrechen, zählen, z. B. wenn eine neue SSH-App den unbekannten Server-Schlüssel ablehnt. Dann eigene Adressen in `ignoreip` eintragen oder `mode = normal` wählen.

---

7. Wo fail2ban liest: `backend` und `logpath`

Einstellung	Bedeutung
`backend = systemd`	liest aus dem systemd-Journal. Empfohlen für Ubuntu 24.04 und Debian 12/13
`backend = auto`	wählt selbst (pyinotify, dann polling) und liest Dateien aus `logpath`
`backend = polling`	prüft Dateien regelmäßig, ohne Zusatzpakete
`logpath = /var/log/auth.log`	Protokolldatei, nur bei dateibasiertem Backend nötig
`usedns = warn`	Umgang mit Hostnamen in Protokollen (`yes`, `warn`, `no`, `raw`)
`maxmatches = 5`	wie viele erkannte Zeilen pro Adresse gespeichert werden

---

8. Wie gesperrt wird: `banaction`

Einstellung	Bedeutung
`banaction = nftables`	Sperre über nftables (in neueren Debian-/Ubuntu-Paketen Standard)
`banaction = iptables-multiport`	Sperre über iptables, nur die Ports des Jails
`banaction = ufw`	Sperre über die Firewall ufw (nur sinnvoll, wenn ufw aktiv ist)
`banaction_allports = nftables[type=allports]`	Variante, die **alle** Ports sperrt (z. B. für `recidive`)
`protocol = tcp`	Protokoll der Sperre
`allowipv6 = auto`	IPv6 unterstützen (`auto`, `yes`, `no`)

Welche Sperrregel aktiv ist, zeigt: `nft list ruleset | grep -A5 f2b` bzw. `iptables -L -n | grep f2b`.

---

9. Benachrichtigungen

Per Mail (braucht einen eingerichteten Mailversand):

```
[DEFAULT]
destemail = name@example.org
sender    = fail2ban@example.org
mta       = sendmail
action    = %(action_mw)s
```

Kurzform	Wirkung
`%(action_)s`	nur sperren (Standard)
`%(action_mw)s`	sperren und Mail mit Whois-Daten
`%(action_mwl)s`	sperren und Mail mit Whois-Daten und den Protokollzeilen

Per ntfy (Beispiel, eigene Action-Datei `/etc/fail2ban/action.d/ntfy.conf`):

```
[Definition]
actionban   = curl -s -H "Title: fail2ban <name>" -H "Tags: no_entry" -d "Gesperrt: <ip> (<fail
ures> Versuche)" https://ntfy.sh/THEMA
actionunban =
```

Einbinden im Jail: `action = %(action_)s` und in der nächsten Zeile `ntfy`. Bei vielen Angriffen entstehen viele Meldungen, deshalb eher nur für `recidive` verwenden. Alternativ gibt es die mitgelieferte Action `apprise` für viele Benachrichtigungsdienste.

---

10. Weitere Jails (Beispiele)

Jail	Schützt
`recidive`	Wiederholungstäter über alle Dienste
`nginx-http-auth`, `apache-auth`	passwortgeschützte Webseiten
`postfix`, `dovecot`	Mailserver
`proftpd`, `pure-ftpd`, `vsftpd`	FTP-Server

Ein eigenes Jail mit beliebigem Namen kann einen vorhandenen Filter nutzen (`filter = …`), siehe Abschnitt 5. Alle verfügbaren Jails stehen in `/etc/fail2ban/jail.conf`, die Filter in `/etc/fail2ban/filter.d/`. Einschalten jeweils mit `enabled = true` im eigenen Abschnitt von `jail.local`.

---

11. Allgemeine Einstellungen des Dienstes (`fail2ban.local`)

Einstellung	Standard	Bedeutung
`loglevel`	`INFO`	Ausführlichkeit des eigenen Protokolls
`logtarget`	`/var/log/fail2ban.log`	Ziel des eigenen Protokolls
`dbfile`	`/var/lib/fail2ban/fail2ban.sqlite3`	Datenbank der Sperren (nötig für `bantime.increment`)

Einstellung	Standard	Bedeutung
`dbpurgeage`	`1d`	wie lange alte Sperren in der Datenbank bleiben. Für `bantime.increment` größer wählen, z. B. `dbpurgeage = 30d`

Diese Werte gehören in `/etc/fail2ban/fail2ban.local` unter `[Definition]`.

12. Bedienung

Zweck	Befehl
Läuft fail2ban? (muss „active (running)“ zeigen)	`systemctl status fail2ban`
Komplett neu starten	`systemctl restart fail2ban`
Vorübergehend anhalten	`systemctl stop fail2ban`
Autostart abschalten	`systemctl disable fail2ban`
Übersicht aller Jails	`fail2ban-client status`
Status und gesperrte Adressen eines Jails	`fail2ban-client status sshd`
Status des Jails für langsame Angreifer	`fail2ban-client status sshd-langsam`
Adresse in einem bestimmten Jail freigeben	`fail2ban-client set sshd-langsam unbanip ADRESSE`
Alle gesperrten Adressen	`fail2ban-client banned`
Adresse von Hand sperren	`fail2ban-client set sshd banip ADRESSE`
Adresse freigeben	`fail2ban-client set sshd unbanip ADRESSE`
Alle Sperren aufheben	`fail2ban-client unban --all`
Nach Änderung an `jail.local` übernehmen	`fail2ban-client reload`
Konfiguration prüfen	`fail2ban-client -t`
Filter gegen ein Protokoll testen	`fail2ban-regex /var/log/auth.log /etc/fail2ban/filter.d/sshd.conf`
Filter gegen das Journal testen	`fail2ban-regex systemd-journal 'sshd[mode=aggressive]'`
Eigenes Protokoll	`tail -50 /var/log/fail2ban.log`

13. Fehlerquellen

Problem	Ursache	Lösung
Selbst ausgesperrt	Mehrfach falsches Passwort von eigener Adresse	über anderen Zugang (anderes Netz, Konsole im Kundenpanel) anmelden und `unbanip`
Es wird nichts gesperrt	falsches `backend`, Filter findet keine Zeilen	`backend = systemd` setzen, mit `fail2ban-regex` testen
Dienst startet nicht	Tippfehler in `jail.local`	`fail2ban-client -t` zeigt die fehlerhafte Zeile
Jail fehlt in `fail2ban-client status`	Modus falsch geschrieben, z. B. `aggressiv` statt `aggressive`	Schreibweise korrigieren, gültig sind nur `normal`, `ddos`, `extra`, `aggressive`

Problem	Ursache	Lösung
Trotz fail2ban täglich viele Versuche	langsame Angreifer bleiben unter `maxretry` in `findtime`	zweites Jail wie in Abschnitt 5
Freigegebene Adresse bleibt gesperrt	Adresse ist zusätzlich im zweiten Jail gesperrt	`fail2ban-client banned` prüfen und in jedem Jail `unbanip`
Sperre wirkt nicht	anderer SSH-Port oder Firewall-Konflikt	`port` anpassen, `banaction` zur verwendeten Firewall wählen
Mobilfunk: dauernd neue Adressen	Anbieter teilt Adressen (CGNAT)	Netz des Anbieters in `ignoreip` aufnehmen oder sorgfältig tippen
`bantime.increment` wirkt kaum	`dbpurgeage` zu kurz	`dbpurgeage` vergrößern

Notzugang: Vor dem Einschalten sicherstellen, dass man notfalls über die Konsole des Anbieters an den Server kommt.

Noch wirksamer als jede Sperre: Anmeldung per SSH-Schlüssel statt Passwort. Dann sind Passwort-Angriffe wirkungslos. fail2ban ist der einfachere Schritt und reduziert die Angriffe stark.

Quellen

1. fail2ban, offizielles Projekt (GitHub): https://github.com/fail2ban/fail2ban
2. fail2ban, Hauptkonfiguration jail.conf mit Erklärung aller Optionen: https://github.com/fail2ban/fail2ban/blob/master/config/jail.conf
3. fail2ban, SSH-Filter sshd.conf (mode: normal, ddos, extra, aggressive): https://github.com/fail2ban/fail2ban/blob/master/config/filter.d/sshd.conf
4. fail2ban, Dienstkonfiguration fail2ban.conf (loglevel, dbfile, dbpurgeage): https://github.com/fail2ban/fail2ban/blob/master/config/fail2ban.conf
5. fail2ban, Debian-/Ubuntu-Pfade und Standard-banaction (paths-debian.conf): https://github.com/fail2ban/fail2ban/blob/master/config/paths-debian.conf
6. fail2ban, mitgelieferte Actions (nftables, iptables, ufw, apprise, Mail): https://github.com/fail2ban/fail2ban/tree/master/config/action.d
7. fail2ban, Wiki: https://github.com/fail2ban/fail2ban/wiki
8. jail.conf(5), Debian Manpages: https://manpages.debian.org/trixie/fail2ban/jail.conf.5.en.html
9. jail.conf(5), Arch Linux Manpages: https://man.archlinux.org/man/jail.conf.5
10. fail2ban-regex(1), Debian Manpages (Filter testen, auch gegen systemd-journal): https://dyn.manpages.debian.org/testing/fail2ban/fail2ban-regex.1.en.html
11. fail2ban-regex(1), Arch Linux Manpages: https://man.archlinux.org/man/fail2ban-regex.1
12. fail2ban-client(1), Debian Manpages: https://manpages.debian.org/trixie/fail2ban/fail2ban-client.1.en.html
13. Debian-Quellpaket fail2ban 1.1.0, jail.conf: https://sources.debian.org/src/fail2ban/1.1.0-9/config/jail.conf
14. ntfy, Nachrichten senden (Title, Tags, Priority): https://docs.ntfy.sh/publish/
15. Eigene Einrichtung und Tests von Sebastian Ebner auf zwei V-Servern (Ubuntu 24.04 mit LXD, WireGuard, EOTK), 16. und 17. November 2026