11.2 Firewall-Regeln und Regelwerke

Firewall-Regeln bestimmen, welcher Netzwerkverkehr erlaubt oder blockiert wird.

Eine Firewall-Regel beantwortet im Kern diese Fragen:

```
Wer möchte kommunizieren?
Mit wem?
Über welches Protokoll?
Über welchen Port?
In welche Richtung?
Aus welcher Zone?
In welche Zone?
Soll der Verkehr erlaubt oder blockiert werden?
```

Merksatz:

```
Firewall-Regeln steuern,
wer mit wem über welchen Dienst kommunizieren darf.
```

Warum sind Firewall-Regeln so wichtig?

Eine Firewall ist nur so gut wie ihr Regelwerk.

Wenn Regeln zu offen sind, entsteht ein Sicherheitsrisiko.

Wenn Regeln zu streng oder falsch sind, funktionieren Dienste nicht.

Beispiele:

```
Webserver ist nicht erreichbar,
weil TCP 443 blockiert wird.

Datenbank ist zu offen erreichbar,
weil zu viele Quellen erlaubt sind.

Adminzugang ist aus dem Internet erreichbar,
weil eine alte Testregel vergessen wurde.
```

Merksatz:

> Firewall-Regeln müssen sicher und funktionsfähig sein.

Bestandteile einer Firewall-Regel

Eine typische Firewall-Regel enthält:

- Aktion
- Quelle
- Ziel
- Dienst
- Protokoll
- Port
- Richtung
- Zone oder Interface
- Kommentar
- Logging
- Zeitplan
- Priorität oder Reihenfolge

Beispiel:

Aktion:
Erlauben

Quelle:
Admin-Netz

Ziel:
Webserver in DMZ

Dienst:
SSH TCP 22

Richtung:
LAN → DMZ

Merksatz:

Eine Firewall-Regel braucht Quelle,
Ziel,
Dienst
und Aktion.

Aktion

Die Aktion sagt, was mit passendem Verkehr passieren soll.

Typische Aktionen:

Aktion	Bedeutung
Allow	erlauben
Deny	blockieren
Drop	still verwerfen
Reject	aktiv ablehnen
Log	protokollieren
NAT	übersetzen oder weiterleiten

Merksatz:

Aktion = Was macht die Firewall mit dem Verkehr?

Quelle

Die Quelle beschreibt, woher der Verkehr kommt.

Beispiele:

- einzelne IP-Adresse
- Subnetz
- Sicherheitszone
- Interface
- Benutzergruppe
- VPN-Gruppe
- Hostobjekt

Beispiel:

```
Quelle:
192.168.10.0/24
```

bedeutet:

```
alle Geräte aus diesem Subnetz.
```

Merksatz:

```
Quelle = Wer startet die Verbindung?
```

Ziel

Das Ziel beschreibt, wohin der Verkehr gehen soll.

Beispiele:

```
- einzelner Server
- Subnetz
- DMZ-Zone
- Internet
- Servergruppe
- Datenbankserver
- Reverse Proxy
```

Beispiel:

```
Ziel:
192.168.20.10
```

bedeutet:

```
genau dieser Zielhost.
```

Merksatz:

```
Ziel = Welches System soll erreicht werden?
```

Dienst

Der Dienst beschreibt, welche Anwendung oder welcher Port verwendet wird.

Beispiele:

```
HTTP
HTTPS
DNS
SSH
RDP
SMB
SMTP
LDAP
NTP
```

Ein Dienstobjekt besteht meist aus:

```
Protokoll
plus
Port
```

Beispiel:

```
HTTPS:
TCP 443
```

Merksatz:

```
Dienst = Protokoll und Port des gewünschten Verkehrs.
```

Protokoll

Das Protokoll beschreibt, ob zum Beispiel TCP, UDP oder ICMP verwendet wird.

Beispiele:

Protokoll	Beispiel

TCP	HTTP, HTTPS, SSH, SMB
UDP	DNS, DHCP, NTP, SNMP
ICMP	Ping, Fehlermeldungen
ESP	IPsec
GRE	bestimmte Tunnel

Merksatz:

> Port allein reicht nicht,
> das Transportprotokoll muss auch stimmen.

Port

Der Port gehört zur Transportschicht.

Beispiele:

Dienst	Port
HTTP	TCP 80
HTTPS	TCP 443
SSH	TCP 22
RDP	TCP 3389
DNS	UDP/TCP 53
NTP	UDP 123
SMB	TCP 445

Wichtig:

> TCP 53 und UDP 53 sind nicht dasselbe.
> TCP 443 und UDP 443 sind nicht dasselbe.

Merksatz:

> Port immer zusammen mit TCP oder UDP betrachten.

Richtung

Die Richtung beschreibt, von welcher Zone oder welchem Interface der Verkehr wohin geht.

Beispiele:

```
LAN → WAN
WAN → DMZ
DMZ → LAN
VPN → Servernetz
Gastnetz → Internet
Managementnetz → Switches
```

Viele Fehler entstehen, weil Regeln in der falschen Richtung erstellt werden.

Merksatz:

```
Firewall-Regeln sind richtungsabhängig.
```

Zone

Eine Zone ist ein Sicherheitsbereich.

Beispiele:

```
- LAN
- WAN
- DMZ
- Gastnetz
- Servernetz
- Managementnetz
- VPN
- IoT-Netz
```

Regeln werden häufig von Zone zu Zone definiert.

Beispiel:

```
LAN → WAN:
HTTPS erlauben
```

```
WAN → LAN:
blockieren
```

Merksatz:

```
Zonen helfen,
Netzbereiche mit unterschiedlichem Schutzbedarf zu trennen.
```

Interface

Ein Interface ist eine konkrete Netzwerkschnittstelle der Firewall.

Beispiele:

```
WAN-Port
LAN-Port
DMZ-Port
VLAN-Interface
VPN-Interface
```

Manche Firewalls arbeiten stärker mit Interfaces, andere stärker mit Zonen.

Merksatz:

```
Interface = konkrete Schnittstelle,
Zone = logischer Sicherheitsbereich.
```

Regelreihenfolge

Viele Firewalls prüfen Regeln von oben nach unten.

Die erste passende Regel entscheidet.

Beispiel:

```
Regel 1:
Blockiere LAN → Server TCP 443

Regel 2:
```

```
Erlaube LAN → Server TCP 443
```

Ergebnis:

```
Verkehr wird blockiert,
weil Regel 1 zuerst passt.
```

Merksatz:

```
Die erste passende Regel gewinnt.
```

Spezifisch vor allgemein

Spezifische Regeln sollten vor allgemeinen Regeln stehen.

Beispiel richtig:

```
Regel 1:
Erlaube Admin-PC → Server SSH

Regel 2:
Blockiere restliches SSH
```

Beispiel falsch:

```
Regel 1:
Blockiere restliches SSH

Regel 2:
Erlaube Admin-PC → Server SSH
```

Im falschen Beispiel wird die Erlaubnis nie erreicht.

Merksatz:

```
Spezifische Regeln vor allgemeine Regeln setzen.
```

Explizite Regeln

Explizite Regeln sind bewusst angelegte Regeln.

Beispiel:

```
Erlaube Webserver → Datenbank TCP 5432
```

Diese Regel beschreibt genau:

```
wer
wohin
über welchen Dienst
kommunizieren darf.
```

Merksatz:

```
Explizite Regeln sind gezielte Erlaubnisse oder Verbote.
```

Implizite Regel

Viele Firewalls haben am Ende eine implizite Regel.

Diese lautet häufig sinngemäß:

```
Alles blockieren,
was nicht vorher erlaubt wurde.
```

Das nennt man oft:

```
implicit deny
```

Diese Regel ist nicht immer sichtbar, wirkt aber trotzdem.

Merksatz:

```
Am Ende steht oft ein unsichtbares „alles blockieren“.
```

Default Deny

Default Deny bedeutet:

```
Standardmäßig ist alles verboten,
außer es wurde ausdrücklich erlaubt.
```

Das ist sicherer, weil neue oder vergessene Dienste nicht automatisch erreichbar sind.

Beispiel:

```
LAN → WAN nur HTTP, HTTPS und DNS erlauben.
Alles andere blockieren.
```

Merksatz:

```
Default Deny ist ein wichtiges Sicherheitsprinzip.
```

Default Allow

Default Allow bedeutet:

```
Standardmäßig ist alles erlaubt,
außer es wurde ausdrücklich blockiert.
```

Das ist einfacher, aber riskanter.

Problem:

```
Unbekannte Dienste können unbeabsichtigt erreichbar sein.
```

Merksatz:

```
Default Allow ist bequem,
aber oft unsicherer.
```

Allow-Regel

Eine Allow-Regel erlaubt bestimmten Verkehr.

Beispiel:

```
Erlaube
Quelle: LAN
Ziel: Internet
Dienst: HTTPS TCP 443
```

Bedeutung:

```
Clients im LAN dürfen HTTPS-Verbindungen ins Internet aufbauen.
```

Merksatz:

```
Allow-Regel öffnet gezielt Kommunikation.
```

Deny-Regel

Eine Deny-Regel blockiert bestimmten Verkehr.

Beispiel:

```
Blockiere
Quelle: Gastnetz
Ziel: LAN
Dienst: any
```

Bedeutung:

```
Gäste dürfen nicht auf interne Systeme zugreifen.
```

Merksatz:

```
Deny-Regel verhindert bestimmte Kommunikation.
```

Drop-Regel

Drop bedeutet:

```
Paket wird still verworfen.
```

Der Absender bekommt keine direkte Antwort.

Vorteil:

```
weniger Informationen für Angreifer
```

Nachteil:

```
Fehlersuche kann schwieriger sein,
weil Verbindungen einfach timeouten.
```

Merksatz:

```
Drop = still verwerfen.
```

Reject-Regel

Reject bedeutet:

```
Paket wird aktiv abgelehnt.
```

Der Absender erhält eine Rückmeldung.

Vorteil:

```
Fehlersuche oft klarer
```

Nachteil:

```
Absender erfährt,
dass ein System oder Filter aktiv antwortet.
```

Merksatz:

```
Reject = aktiv ablehnen.
```

Logging bei Firewall-Regeln

Logging bedeutet:

```
Treffer einer Regel werden protokolliert.
```

Logs können zeigen:

```
- Quelle
- Ziel
- Port
- Protokoll
- Zeit
- Aktion
- Regelname
- Zone
- Interface
```

Logging ist wichtig für:

```
- Fehlersuche
- Sicherheitsanalyse
- Nachvollziehbarkeit
- Angriffserkennung
```

Merksatz:

```
Ohne Firewall-Logs ist Fehlersuche oft blind.
```

Nicht jede Regel sollte alles loggen

Zu viel Logging kann problematisch sein.

Nachteile:

```
- sehr große Logmengen
- wichtige Ereignisse gehen unter
- Speicherverbrauch
```

- Performancebelastung
- unübersichtliche Analyse

Sinnvoll:

wichtige erlaubte Zugriffe loggen
wichtige Blockierungen loggen
Standardrauschen begrenzen

Merksatz:

Logging gezielt einsetzen.

Objekte in Firewall-Regeln

Viele Firewalls nutzen Objekte.

Beispiele:

Hostobjekt:
webserver01 = 192.168.20.10

Netzwerkobjekt:
LAN = 192.168.10.0/24

Dienstobjekt:
HTTPS = TCP 443

Vorteil:

Regeln werden lesbarer und leichter pflegbar.

Merksatz:

Objekte machen Firewall-Regeln übersichtlicher.

Adressobjekte

Adressobjekte beschreiben IP-Adressen oder Netze.

Beispiele:

```
Admin-PC = 192.168.10.50
Webserver = 192.168.20.10
Servernetz = 192.168.20.0/24
Gastnetz = 192.168.50.0/24
```

Merksatz:

```
Adressobjekte benennen IP-Adressen und Netze.
```

Dienstobjekte

Dienstobjekte beschreiben Ports und Protokolle.

Beispiele:

```
HTTPS = TCP 443
DNS = UDP/TCP 53
SSH = TCP 22
NTP = UDP 123
```

Merksatz:

```
Dienstobjekte benennen Protokoll-Port-Kombinationen.
```

Gruppenobjekte

Objekte können zu Gruppen zusammengefasst werden.

Beispiel:

```
Webdienste:
HTTP
HTTPS
```

Oder:

```
Admin-Hosts:
Admin-PC-1
Admin-PC-2
Bastion-Host
```

Vorteil:

```
weniger doppelte Regeln
bessere Übersicht
```

Merksatz:

```
Gruppenobjekte vereinfachen Regelwerke.
```

Any

Any bedeutet:

```
beliebig
```

Beispiele:

```
Quelle: any
Ziel: any
Dienst: any
```

Any kann sinnvoll sein, ist aber gefährlich, wenn es zu großzügig verwendet wird.

Besonders riskant:

```
Quelle any
Ziel any
Dienst any
Aktion allow
```

Merksatz:

```
Any sparsam verwenden.
```

Any-Any-Regel

Eine Any-Any-Regel erlaubt sehr viel.

Beispiel:

```
Quelle: any
Ziel: any
Dienst: any
Aktion: allow
```

Das ist meistens ein Sicherheitsproblem.

Solche Regeln sollten vermieden, dokumentiert oder zeitlich begrenzt werden.

Merksatz:

```
Any-Any ist fast immer zu offen.
```

Temporäre Regeln

Temporäre Firewall-Regeln entstehen oft für Tests oder Störungen.

Problem:

```
Sie werden später vergessen.
```

Deshalb sollten temporäre Regeln enthalten:

```
- Zweck
- Verantwortlicher
- Ablaufdatum
- Ticketnummer
- Kommentar
- Review-Datum
```

Merksatz:

```
Temporäre Regeln brauchen Ablaufdatum.
```

Kommentare in Firewall-Regeln

Kommentare helfen, den Zweck einer Regel zu verstehen.

Ein guter Kommentar enthält:

```
- Warum existiert die Regel?
- Wer hat sie angefordert?
- Welcher Dienst wird benötigt?
- Gibt es ein Ticket?
- Ist sie dauerhaft oder temporär?
```

Schlecht:

```
Test
```

Besser:

```
Ticket 1234: Webserver DMZ darf Datenbank TCP 5432 erreichen.
```

Merksatz:

```
Firewall-Regeln ohne Kommentar werden später unklar.
```

Regeln nach Diensten planen

Regeln sollten vom benötigten Dienst aus geplant werden.

Beispiel:

```
Anwendung braucht Datenbankzugriff.
```

Dann fragt man:

```
Welche Quelle?
Welches Ziel?
Welcher Port?
Welches Protokoll?
Welche Richtung?
Dauerhaft oder temporär?
Muss es geloggt werden?
```

Merksatz:

```
Erst Bedarf klären,
dann Regel erstellen.
```

Regel zu breit

Eine zu breite Regel erlaubt mehr als nötig.

Beispiel schlecht:

```
LAN → Servernetz any erlauben
```

Besser:

```
Anwendungsserver → Datenbank TCP 5432 erlauben
```

Warum?

```
Wenn ein Client kompromittiert wird,
kann er bei breiten Regeln mehr Ziele erreichen.
```

Merksatz:

```
Breite Regeln erhöhen die Angriffsfläche.
```

Regel zu eng

Eine zu enge oder falsche Regel kann Dienste stören.

Beispiel:

```
DNS nur UDP 53 erlaubt,
aber TCP 53 wird für bestimmte Antworten benötigt.
```

Oder:

```
FTP-Port 21 erlaubt,
aber passive Datenports fehlen.
```

Merksatz:

```
Regeln müssen sicher,
aber technisch vollständig sein.
```

Rückverkehr

Bei Stateful Firewalls wird Rückverkehr oft automatisch erlaubt, wenn er zu einer bestehenden Verbindung gehört.

Beispiel:

```
Client → Webserver TCP 443 erlaubt
```

Antwort:

```
Webserver → Client gehört zur bestehenden Verbindung
und wird erlaubt.
```

Bei stateless Regeln muss Rückverkehr oft separat betrachtet werden.

Merksatz:

```
Stateful Firewalls erleichtern Rückverkehr.
```

Asymmetrischer Rückweg

Eine Stateful Firewall erwartet, dass Hin- und Rückverkehr über dieselbe Firewall laufen.

Wenn der Rückweg anders läuft, kennt die Firewall die Verbindung nicht.

Folge:

```
Antwortverkehr kann blockiert werden.
```

Merksatz:

```
Stateful Firewalls brauchen passenden Hin- und Rückweg.
```

Regelrichtung bei Client-Server-Kommunikation

Bei Client-Server-Kommunikation startet meistens der Client die Verbindung.

Beispiel:

```
Client im LAN ruft Webserver in DMZ auf.
```

Regel:

```
Quelle: LAN-Client
Ziel: DMZ-Webserver
Dienst: HTTPS TCP 443
```

Nicht umgekehrt.

Merksatz:

```
Regelrichtung nach Verbindungsaufbau bestimmen.
```

Quellport und Zielport

Bei Client-Server-Verbindungen ist der Zielport meist der bekannte Dienstport.

Beispiel:

```
Client-Quellport:
zufällig hoher Port
```

```
Server-Zielport:
TCP 443
```

Firewall-Regeln beziehen sich meist auf den Zielport des Dienstes.

Merksatz:

```
Standardport ist meistens Zielport beim Server.
```

Ephemeral Ports

Ephemeral Ports sind kurzlebige hohe Ports, die Clients für ausgehende Verbindungen verwenden.

Beispiel:

```
Client 192.168.10.50:53124
→ Webserver 203.0.113.10:443
```

Der Quellport 53124 ist ein temporärer Clientport.

Merksatz:

```
Clients nutzen temporäre Quellports.
```

ICMP in Firewall-Regeln

ICMP ist wichtig für Diagnose und Netzwerkfunktion.

Beispiele:

```
- Ping
- Ziel nicht erreichbar
- Fragmentierung nötig
- Time Exceeded bei Traceroute
```

ICMP komplett zu blockieren kann Fehlersuche erschweren und bestimmte Netzwerkfunktionen stören.

Merksatz:

```
ICMP ist mehr als nur Ping.
```

Ping erlauben oder blockieren?

Ping kann bei Fehlersuche helfen.

Aber:

```
Ping offen zeigt,
dass ein System erreichbar ist.
```

In vielen Umgebungen wird Ping intern erlaubt, extern aber eingeschränkt.

Wichtig ist eine bewusste Entscheidung.

Merksatz:

```
ICMP bewusst regeln,
nicht blind alles blockieren.
```

DNS-Regeln

DNS braucht je nach Fall UDP und TCP 53.

Typische Regel:

```
Clients → interner DNS-Server UDP/TCP 53 erlauben
```

Nicht ideal:

```
Alle Clients → beliebige externe DNS-Server erlauben
```

Warum?

```
Interne DNS-Kontrolle,
Filterung
und Protokollierung können umgangen werden.
```

Merksatz:

```
DNS gezielt zu erlaubten Resolvern steuern.
```

HTTP- und HTTPS-Regeln

Typische ausgehende Webregel:

```
Clients → Internet TCP 80 und TCP 443 erlauben
```

Sicherer:

```
über Proxy steuern
oder
Zielkategorien beschränken
```

Wichtig:

```
TCP 443 erlaubt nicht automatisch sichere Anwendung,
weil auch unerwünschter Verkehr über 443 laufen kann.
```

Merksatz:

```
HTTPS-Port offen heißt nicht automatisch sicherer Inhalt.
```

Administrationsregeln

Administrative Zugriffe sollten besonders eingeschränkt werden.

Beispiele:

```
SSH nur aus Managementnetz
RDP nur über VPN
Firewall-Webinterface nur von Admin-PCs
Datenbankport nicht für Clients öffnen
Hypervisor-Management getrennt halten
```

Merksatz:

```
Adminzugriffe nur aus Managementbereichen erlauben.
```

Datenbankregeln

Datenbanken sollten nicht allgemein erreichbar sein.

Beispiel:

```
Webserver → Datenbank TCP 5432 erlauben
```

Aber nicht:

```
gesamtes LAN → Datenbank any erlauben
```

Datenbankzugriff ist besonders sensibel.

Merksatz:

```
Datenbanken nur für notwendige Systeme öffnen.
```

DMZ-Regeln

In einer DMZ gilt:

```
Internet → DMZ:
nur öffentliche Dienste erlauben

DMZ → LAN:
nur zwingend notwendige Verbindungen erlauben

LAN → DMZ:
Administration nur gezielt erlauben
```

Beispiel:

```
Internet → Reverse Proxy TCP 443 erlauben
Reverse Proxy → interner Webdienst TCP 8080 erlauben
Reverse Proxy → gesamtes LAN any blockieren
```

Merksatz:

```
DMZ-Regeln müssen besonders eng sein.
```

Gastnetz-Regeln

Typische Gastnetzregeln:

```
Gastnetz → Internet:
HTTP/HTTPS/DNS erlauben

Gastnetz → LAN:
blockieren

Gastnetz → Managementnetz:
blockieren

Gastnetz → Servernetz:
blockieren
```

Merksatz:

```
Gäste bekommen Internet,
aber keinen internen Zugriff.
```

IoT-Regeln

IoT-Geräte sollten eingeschränkt werden.

Beispiele:

```
IoT → Internet nur notwendige Dienste
IoT → LAN blockieren
```

```
Admin-Netz → IoT-Verwaltung erlauben
IoT → DNS nur zu internem Resolver
IoT → NTP nur zu freigegebenem Zeitserver
```

Merksatz:

```
IoT-Geräte nur kontrolliert kommunizieren lassen.
```

VPN-Regeln

VPN-Zugriff sollte nicht automatisch alles erlauben.

Besser:

```
VPN-Gruppe Support → bestimmte Server RDP
VPN-Gruppe Admin → Managementnetz SSH
VPN-Gruppe Extern → nur benötigte Anwendung
```

Merksatz:

```
VPN-Zugriff nach Rolle und Bedarf einschränken.
```

Regelprüfung bei Störung

Wenn ein Dienst nicht funktioniert, prüft man bei der Firewall:

```
- kommt Verkehr an?
- welche Regel greift?
- wird erlaubt oder blockiert?
- stimmt Quelle?
- stimmt Ziel?
- stimmt Port?
- stimmt Protokoll?
- stimmt Richtung?
- greift vorher eine Blockregel?
- ist NAT zusätzlich nötig?
- gibt es Rückverkehr?
```

Merksatz:

```
Bei Störung immer die tatsächlich getroffene Regel prüfen.
```

Hit Count

Viele Firewalls zeigen, wie oft eine Regel getroffen wurde.

Das nennt man oft:

```
Hit Count
```

Nützlich für:

```
- aktive Regeln erkennen
- ungenutzte Regeln finden
- Regelprüfung
- Aufräumen
- Fehlersuche
```

Aber:

```
Kein Hit Count heißt nicht immer,
dass Regel sicher gelöscht werden kann.
```

Merksatz:

```
Hit Count hilft,
ersetzt aber keine fachliche Prüfung.
```

Shadowed Rule

Eine shadowed rule ist eine Regel, die nie erreicht wird, weil eine vorherige Regel den Verkehr bereits behandelt.

Beispiel:

```
Regel 1:
Blockiere LAN → Server any

Regel 2:
Erlaube LAN → Server HTTPS
```

Regel 2 wird nie wirken, wenn Regel 1 vorher greift.

Merksatz:

```
Shadowed Rule = Regel wird durch frühere Regel verdeckt.
```

Redundante Regel

Eine redundante Regel ist überflüssig, weil eine andere Regel denselben Verkehr bereits behandelt.

Problem:

```
Regelwerk wird unübersichtlich.
```

Merksatz:

```
Redundante Regeln erschweren Wartung und Fehlersuche.
```

Regelkonflikt

Ein Regelkonflikt entsteht, wenn Regeln widersprüchlich oder unklar wirken.

Beispiel:

```
Eine Regel erlaubt Zugriff.
Eine andere blockiert scheinbar denselben Zugriff.
```

Entscheidend ist dann:

```
Reihenfolge
Genauigkeit
```

Zonen
Objekte
tatsächlicher Treffer

Merksatz:

Bei Konflikten zählt die Regel,
die tatsächlich zuerst passt.

Firewall-Änderungsprozess

Firewall-Änderungen sollten kontrolliert ablaufen.

Typischer Ablauf:

1. Anforderung stellen.
2. Zweck beschreiben.
3. Quelle, Ziel und Dienst angeben.
4. Risiko bewerten.
5. Änderung genehmigen.
6. Regel umsetzen.
7. Funktion testen.
8. Dokumentation aktualisieren.
9. Regel später überprüfen.

Merksatz:

Firewall-Änderungen brauchen Kontrolle und Dokumentation.

Vor einer Regeländerung klären

Vor einer Firewall-Regeländerung sollte klar sein:

Wer braucht Zugriff?
Von wo nach wo?
Welcher Dienst?
Welcher Port?
TCP oder UDP?

```
Dauerhaft oder temporär?
Muss NAT eingerichtet werden?
Gibt es Sicherheitsrisiken?
Gibt es eine bessere Alternative?
Wer ist verantwortlich?
```

Merksatz:

```
Keine Firewall-Regel ohne klaren Zweck.
```

Nach einer Regeländerung testen

Nach Änderung testet man:

```
funktioniert der Dienst?
ist nur gewünschter Zugriff möglich?
sind unerwünschte Zugriffe blockiert?
erscheinen Logs korrekt?
greift die richtige Regel?
ist NAT korrekt?
funktioniert Rückverkehr?
ist Dokumentation aktualisiert?
```

Merksatz:

```
Firewall-Regel nach Änderung immer testen.
```

Regelbereinigung

Regelwerke sollten regelmäßig bereinigt werden.

Dabei prüft man:

```
- alte Regeln
- temporäre Regeln
- ungenutzte Regeln
- Any-Any-Regeln
- redundante Regeln
```

- shadowed rules
- fehlende Kommentare
- zu breite Freigaben
- nicht mehr benötigte Dienste

Merksatz:

Firewall-Regelwerke altern und müssen gepflegt werden.

Typische Fehler bei Firewall-Regeln

Typische Fehler sind:

- falsche Richtung
- falsche Quelle
- falsches Ziel
- falscher Port
- TCP statt UDP oder umgekehrt
- falsche Zone
- falsche Regelreihenfolge
- NAT vergessen
- Rückweg vergessen
- DNS-Name zeigt auf anderes Ziel
- Objekt falsch gepflegt
- temporäre Regel vergessen
- zu breite Any-Regel

Merksatz:

Firewall-Fehler sind oft Detailfehler.

Beispiel: Webserver in DMZ freigeben

Anforderung:

Externe Benutzer sollen Webserver in DMZ per HTTPS erreichen.

Benötigt:

Quelle:
Internet

Ziel:
Webserver oder Reverse Proxy in DMZ

Dienst:
HTTPS TCP 443

Richtung:
WAN → DMZ

NAT:
öffentliche IP auf DMZ-Ziel weiterleiten

Firewall:
TCP 443 erlauben

Merksatz:

Externer Webdienst braucht NAT und passende Firewall-Regel.

Beispiel: Adminzugriff auf DMZ-Server

Anforderung:

Administrator soll per SSH auf DMZ-Server zugreifen.

Besser:

Quelle:
Admin-Netz oder Bastion Host

Ziel:
DMZ-Server

Dienst:
SSH TCP 22

Richtung:
Management → DMZ

Nicht gut:

Internet → DMZ-Server SSH erlauben

Merksatz:

Administration nicht aus beliebigen Quellen erlauben.

Beispiel: Webserver zur Datenbank

Anforderung:

Webserver braucht Datenbankzugriff.

Regel:

Quelle:
Webserver

Ziel:
Datenbankserver

Dienst:
passender Datenbankport

Richtung:
DMZ oder App-Netz → DB-Netz

Wichtig:

Nicht gesamtes Netz zur Datenbank erlauben.

Merksatz:

Datenbankzugriff nur von notwendigen Systemen erlauben.

Beispiel: DNS im Clientnetz

Anforderung:

Clients sollen Namen auflösen.

Regel:

Quelle:
Clientnetz

Ziel:
interner DNS-Server

Dienst:
DNS UDP/TCP 53

Optional:

externe DNS-Server für Clients blockieren,
damit DNS zentral kontrolliert bleibt.

Merksatz:

Clients sollten definierte DNS-Resolver nutzen.

Beispiel: Gastnetz

Anforderung:

Gäste brauchen Internet,
aber keinen Zugriff ins LAN.

Regeln:

```
Gastnetz → Internet HTTPS erlauben
Gastnetz → Internet DNS erlauben
Gastnetz → LAN blockieren
Gastnetz → Managementnetz blockieren
Gastnetz → Servernetz blockieren
```

Merksatz:

```
Gastnetz strikt vom internen Netz trennen.
```

Was Firewall-Regelwerke nicht lösen

Firewall-Regeln lösen nicht automatisch:

```
- falsche Anwendungskonfiguration
- schwache Passwörter
- fehlende Updates
- unsichere Dienste
- fehlerhafte DNS-Einträge
- falsche Zertifikate
- schlechte Berechtigungen
- fehlendes Monitoring
- fehlende Backups
```

Merksatz:

```
Firewall-Regeln sind wichtig,
ersetzen aber kein Gesamtsicherheitskonzept.
```

Typische IHK-Fragen

In AP1 und AP2 kann zum Beispiel gefragt werden:

```
- Welche Bestandteile hat eine Firewall-Regel?
- Warum ist die Regelreihenfolge wichtig?
- Was bedeutet Default Deny?
- Was ist der Unterschied zwischen Allow, Drop und Reject?
```

- Warum sind Any-Any-Regeln gefährlich?
- Warum sollten spezifische Regeln vor allgemeinen Regeln stehen?
- Was ist eine implizite Deny-Regel?
- Was ist ein Dienstobjekt?
- Was ist ein Adressobjekt?
- Was ist eine shadowed rule?
- Warum müssen Firewall-Regeln dokumentiert werden?
- Warum sind temporäre Regeln riskant?
- Wie prüft man eine Firewall-Regel bei einer Störung?
- Warum reicht eine NAT-Regel allein nicht aus?
- Warum sollte VPN-Zugriff nicht automatisch alles erlauben?

Typische Prüfungsfallen

Firewall-Regeln sind richtungsabhängig.

Die erste passende Regel entscheidet.

Spezifisch vor allgemein.

Port immer mit Protokoll betrachten.

TCP 53 ist nicht UDP 53.

TCP 443 ist nicht UDP 443.

Any bedeutet beliebig.

Any-Any ist meist zu offen.

Default Deny ist sicherer als Default Allow.

Drop und Reject sind nicht dasselbe.

Logging ist wichtig,
aber zu viel Logging kann unübersichtlich werden.

NAT ersetzt keine Firewall-Regel.

Firewall-Regel ersetzt keine NAT-Regel.

Rückverkehr bei Stateful Firewalls beachten.

Asymmetrisches Routing kann Stateful Firewalls stören.

Temporäre Regeln brauchen Ablaufdatum.

Regeln ohne Kommentar werden später problematisch.

VPN-Zugriff muss ebenfalls eingeschränkt werden.

DNS-Regeln sollten UDP und TCP beachten.

Wichtige Begriffe kurz erklärt

Begriff	Kurze Erklärung
Firewall-Regel	Bedingung mit Aktion für Netzwerkverkehr
Aktion	erlauben, blockieren, verwerfen oder ablehnen
Quelle	System oder Netz, von dem Verkehr ausgeht
Ziel	System oder Netz, das erreicht werden soll
Dienst	Protokoll-Port-Kombination
Richtung	Verkehrsrichtung zwischen Zonen oder Interfaces
Zone	logischer Sicherheitsbereich
Interface	konkrete Netzwerkschnittstelle
Allow	Verkehr erlauben
Deny	Verkehr blockieren
Drop	Paket still verwerfen
Reject	Paket aktiv ablehnen
Default Deny	alles blockieren, außer erlaubt
Any	beliebige Quelle, beliebiges Ziel oder beliebiger Dienst
Any-Any	sehr breite Regel mit beliebig zu beliebig
Hit Count	Anzahl der Regel-Treffer
Shadowed Rule	verdeckte Regel, die nie erreicht wird
Redundante Regel	überflüssige doppelte Regel

Begriff	Kurze Erklärung
Dienstobjekt	benannter Port mit Protokoll
Adressobjekt	benannte IP-Adresse oder Netz
Regelreview	regelmäßige Prüfung des Regelwerks

IHK-sichere Kurzformulierung

Firewall-Regeln legen fest, welcher Netzwerkverkehr erlaubt oder blockiert wird. Eine Regel besteht typischerweise aus Aktion, Quelle, Ziel, Dienst, Protokoll, Port, Richtung und Zone. Viele Firewalls arbeiten Regeln von oben nach unten ab, sodass die erste passende Regel entscheidet. Deshalb müssen spezifische Regeln vor allgemeinen Regeln stehen. Das Prinzip Default Deny bedeutet, dass alles blockiert wird, was nicht ausdrücklich erlaubt wurde. Firewall-Regeln sollten möglichst eng gefasst, dokumentiert, regelmäßig überprüft und nach Änderungen getestet werden. NAT-Regeln und Firewall-Regeln sind zu unterscheiden: NAT übersetzt Adressen oder Ports, die Firewall erlaubt oder blockiert den Verkehr.

Merksätze

Firewall-Regel = Quelle,
Ziel,
Dienst
und Aktion.

Aktion entscheidet,
was mit Verkehr passiert.

Quelle = wer startet Verbindung.

Ziel = wer soll erreicht werden.

Dienst = Protokoll plus Port.

Port immer mit TCP oder UDP betrachten.

Richtung ist entscheidend.

Zone = Sicherheitsbereich.

Interface = konkrete Schnittstelle.

Die erste passende Regel gewinnt.

Spezifisch vor allgemein.

Default Deny ist sicherer.

Allow erlaubt.

Deny blockiert.

Drop schweigt.

Reject antwortet.

Any sparsam verwenden.

Any-Any vermeiden.

NAT ersetzt keine Firewall-Regel.

Firewall-Regel ersetzt keine NAT-Regel.

Stateful Firewall kennt Rückverkehr.

Asymmetrischer Rückweg kann stören.

DNS braucht oft UDP und TCP 53.

Adminzugriffe nur aus Managementnetzen erlauben.

Datenbanken nur für notwendige Systeme öffnen.

VPN-Zugriff nicht automatisch alles erlauben.

Temporäre Regeln brauchen Ablaufdatum.

Kommentare erklären den Zweck.

Hit Count hilft beim Prüfen.

Shadowed Rules finden und bereinigen.

Firewall-Regeln regelmäßig überprüfen.