UFW Firewall (Uncomplicated Firewall)

UFW (oder Uncomplicated Firewall) ist eine vereinfachte Firewall-Verwaltungsschnittstelle, die die Komplexität von Paketfilterungstechnologie auf niedriger Ebene wie iptables und nftables versteckt. Wenn Sie mit dem Sichern Ihres Netzwerks beginnen möchten und Sie nicht sicher sind, welches Tool Sie verwenden sollen, könnte UFW die richtige Wahl für Sie sein.

- Installation
- Bedienung

Installation

Beschreibung:

UFW (oder Uncomplicated Firewall) ist eine vereinfachte Firewall-Verwaltungsschnittstelle, die die Komplexität von Paketfilterungstechnologie auf niedriger Ebene wie `iptables` und `nftables` versteckt. Wenn Sie mit dem Sichern Ihres Netzwerks beginnen möchten und Sie nicht sicher sind, welches Tool Sie verwenden sollen, könnte UFW die richtige Wahl für Sie sein.

Installation:

```
apt install ufw
```

Das war alles.
Standardmäßig blockt die Firewall alles nach der Installation.
Die Firewall ist nach der Installation aber noch nicht aktiv.

ipv6 aktivieren:

Möchte man auch ipv6 nutzen müssen wir dies aktivieren

```
nano /etc/default/ufw
```

Dort nach IPV6= suchen und den Wert wenn er nicht auf yes steht auf yes ändern.
Wenn nachher die Firewall aktiviert wird, werden die Regeln auch für ipv6 angewandt

Die Konfiguration/Bedienung kommt im nächsten Kapitel:

Bedienung

Bedienung

Beschreibung:

Wie schon erwähnt ist die Firewall so eingestellt das Sie alles blockt was eingehend ist. Alles was ausgehend ist wird zugelassen, würde man sie schon einschalten, wäre gar keine Kommunikation rein mehr möglich. Ohne ein ipmi, ausgesperrt
Gott sei Dank ist die Firewall direkt nach der Installation noch nicht aktiv.
Hier kommen einige Fallbeispiele wie man die Firewall konfigurieren kann.
Ist man fertig mit der Konfiguration durch, kann durch ein simples

```
ufw enable
```

die Firewall eingeschaltet werden.

Werkseinstellungen:

Sollte man nicht mehr sicher sein, was alles vergeben wurde oder man die Firewall einfach nur zurücksetzten will, Firewall ausschalten die Standard regeln wiederherstellen, eigene Regeln neu setzten, Firewall wieder aktivieren

Firewall ausschalten:

```
ufw disable
```

Nun die Standard Regens setzten:

```
ufw default deny incoming
ufw default allow outgoing
```

Jetzt noch eigene definieren, weil alles eingehende geblockt wird, also auch kein ssh möglich. Dieses müssen wir freischalten, bevor wir aktivieren und noch viele andere Dienste die wir vielleicht nutzen.
Wenn wir sie nicht nutzen, brauchen diese logischerweise auch nicht freigeschaltet werden.

Status anzeigen:

```
ufw status verbose
```

Ausgabe:

```
Status: inactive
```

Wenn die Firewall an ist werden hier die Regeln aufgelistet, so kann man auch schnell Prüfen ist die Firewall an oder aus.

Zulassen von ssh Verbindungen:

```
ufw allow ssh
```

Ausgabe, hier sieht man das die für ipv6 und ipv4 angewandt wurden::

```
Rules updated
Rules updated (v6)
```

Dadurch werden Firewall-Regeln erstellt, die alle Verbindungen an Port `22` zulassen; das ist der Port, an dem der SSH-Daemon standardmäßig lauscht. UFW weiß, was Port `allow ssh` bedeutet, da dies in der Datei `/etc/services` als Dienst aufgeführt wird.

Wir können die äquivalente Regel jedoch auch schreiben, indem wir den Port anstelle des Dienstnamens angeben. Dieser Befehl funktioniert zum Beispiel genauso wie oben

```
ufw allow 22
```

Ausgabe, hier sieht man das die für ipv6 und ipv4 angewandt wurden:

```
fw allow 22
Rules updated
Rules updated (v6)
```

Nun die Firewall anschalten:

```
ufw enable
```

Ausgabe, Frage:

```
Command may disrupt existing ssh connections. Proceed with operation (y|n)?
```

Da wir gerade die Ausnahme für ssh hinzugefügt haben, sollten wir nicht getrennt werden und kann mit y bestätigt werden

Nun können wir uns die Regelns anzeigen lassen

```
ufw status verbose
```

Ausgabe, wie wir sehen default deny incoming, allow outgoing, hier explicit 22 enabled für incoming:

```
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22                         ALLOW IN    Anywhere
22/tcp                     ALLOW IN    Anywhere
22 (v6)                    ALLOW IN    Anywhere (v6)
22/tcp (v6)                ALLOW IN    Anywhere (v6)
```

Zulassen Portbereiche und Protokolltyp UDP/TCP:

```
ufw allow 6000:6007/tcp
ufw allow 6000:6007/udp
```

Zulassen eingehender Verbindung von einer bestimmten IP (Source):

```
ufw allow from 203.0.113.4
```

Man kann aber auch festlegen das diese ip auch nur zu einem Bestimmten Port darf und somit nicht alles darf wie im Beisiel darüber. Somit darf die ip ssh nutzen, aber zum beispiel kein http. Beim oberen Beispiel hätte auch http funktioniert, weil für diese ip alles offen wäre. Jetzt da wir den Port mit angeben haben. Kann die IP auch nur mit Port 22 verbinden

```
ufw allow from 203.0.113.4 to any port 22
```

Zulassen von Subnetzen:

```
ufw allow from 203.0.113.0/24
```

Auch hier wieder mit to any port kann explizit der Port festgelegt werden.

```
ufw allow from 203.0.113.0/24 to any port 22
```

Verschiedenen Netzwerkschnittstellen:

Man hat zwei Netzwerkkarten, möchte aber nur auf Netzwerkkarte eth0 ssh erlauben auf eth1 nicht. Das machen wir mit allow in on gefolgt von der Netzwerkkarte und dann to any port Portnummer

```
ufw allow in on eth0 to any port 22
```

Ablehnen von Verbindungen:

```
ufw deny 80
```

Oder wenn alle eingehenden Verbindungen von einer IP geblockt werden sollen

```
ufw deny from 203.0.113.4
```

Oder diese ip nur auf diesen Port nicht zugreifen darf.

```
ufw deny from 203.0.113.4 to any port 80
```

Regeln löschen

Zu wissen, wie man Firewall-Regeln löscht, ist genauso wichtig wie zu wissen, wie man sie erstellt. Es gibt zwei Wege, um anzugeben, welche Regeln gelöscht werden sollen: anhand der Regelnummer oder der tatsächlichen Regel (ähnlich wie beim Angeben der Regeln im Rahmen der Erstellung). Wir beginnen mit der Methode **Löschen anhand von Regelnummer**, da sie einfacher ist.

Nach Regelnummer:

Wenn Sie die Regelnummer verwenden, um Firewall-Regeln zu löschen, wird eine Liste Ihrer Firewall-Regeln angezeigt. Der UFW-Statusbefehl hat eine Option, um neben jeder Regel eine Nummer anzuzeigen, wie hier gezeigt:

```
ufw status numbered
```

Ausgabe:

```
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22                         ALLOW IN    Anywhere
[ 2] 22/tcp                     ALLOW IN    Anywhere
[ 3] 22 (v6)                    ALLOW IN    Anywhere (v6)
[ 4] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
```

Nun können wir die regel mit der Nummer die wir nicht mehr haben wollen löschen, als Beispiel Nr 2

```
ufw delete 2
```

Nach tatsächlicher regel, kann ich persönlich nur empfehlen

```
ufw delete allow 80
```

Wir löschen die regel genauso wie sie angelegt wurde nur eben mit delete davor.
Dies löscht alle ipv4 und ipv6 regeln auf diesen Port.

Zurücksetzten, wenn Standardrichtlinien nicht geändert wurden.:

Es gibt auch einen Schnellbefehl, alle regeln zu löschen und die Firewall zu deaktivieren um neu anzufangen:

```
ufw reset
```

Ausgehende Verbindungen blocken:

Erstmal auch die Standardregel alle ausgehenden Verbindungen blocken anwenden:

```
ufw default deny outgoing
```

Nun die Verbindung zum Beispiel zu einer IP erlauben, indem diese freigeschaltet wird.

```
ufw allow out to 11.22.33.44
```

Nun können einzelne Ports ausgehend erlaubt werden:

```
ufw allow out to any port 80
```

Oder auf eine einzelne Netzwerkkarte bezogen

```
ufw allow out on eth0 to any port 80
```

Um das ganze schnell wieder rückgängig zu machen und alle ausgehenden Verbindungen wieder zuzulassen

```
ufw default allow outgoing
```

Besonderheit Docker:

Da Docker auch regeln in die iptbales ballert, allerdings diese Vorrang for der ufw haben, müssen wir hier ein bisschen nachhelfen:

wir müssen dazu folgende Datei editieren und am ende einfügen. (weiter unten gibst eins Script)

```
/etc/ufw/after.rules
```

Inhalt :

```
# BEGIN UFW AND DOCKER
*filter
:ufw-user-forward - [0:0]
:ufw-docker-logging-deny - [0:0]
:DOCKER-USER - [0:0]
-A DOCKER-USER -j ufw-user-forward

-A DOCKER-USER -j RETURN -s 10.0.0.0/8
-A DOCKER-USER -j RETURN -s 172.16.0.0/12
-A DOCKER-USER -j RETURN -s 192.168.0.0/16

-A DOCKER-USER -p udp -m udp --sport 53 --dport 1024:65535 -j RETURN
```

```
-A DOCKER-USER -j ufw-docker-logging-deny -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -d 192.168.0.0/16
-A DOCKER-USER -j ufw-docker-logging-deny -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -d 10.0.0.0/8
-A DOCKER-USER -j ufw-docker-logging-deny -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -d 172.16.0.0/12
-A DOCKER-USER -j ufw-docker-logging-deny -p udp -m udp --dport 0:32767 -d 192.168.0.0/16
-A DOCKER-USER -j ufw-docker-logging-deny -p udp -m udp --dport 0:32767 -d 10.0.0.0/8
-A DOCKER-USER -j ufw-docker-logging-deny -p udp -m udp --dport 0:32767 -d 172.16.0.0/12

-A DOCKER-USER -j RETURN

-A ufw-docker-logging-deny -m limit --limit 3/min --limit-burst 10 -j LOG --log-prefix "[UFW DOCKER BLOCK] "
-A ufw-docker-logging-deny -j DROP

COMMIT
# END UFW AND DOCKER
```

Mit einem kleinen Script kann man erledigen lassen

```
cd /root
wget -O /usr/local/bin/ufw-docker https://github.com/chaifeng/ufw-docker/raw/master/ufw-docker
chmod +x /usr/local/bin/ufw-docker
ufw-docker install
systemctl restart ufw
```

Nun muss man auch für Docker explizit die Ports/Routing freigeben, da das aber gerouteter traffic ist, sehen die Befehle ein bisschen anders aus.

Vorallem gillt hier der lokale Port also der im Container.
Beispiel Port 8080:80
Der Port 8080 ist sowieso offen weil die von Docker schon angelegt wurden in iptables.
Wir müssen jetzt eine route anlegen das der port 80 wieder geöffnet wird.
Da das ändern after.rules allen traffic in das lokales Docker Netz blockt.
Container mit Port 80, werdendurch regeln geblockt.
Internen Docker Traffic funktioniert natürlich weiterhin..

Also um den Traffic auf intern im Container Port 80 zu öffnen/routen unteren Befehl anwenden.
Aber Hier noch ein wichtiger Hinweis:

Hinweis: Sollten mehere Container auf Port 80 laufen werden diese mit Eingschlossen. Bedeutet setzt ich das unten eth0 to any port 80. Sind alle container von außen nicht mehr erreichbar sondern nur über eth0. Es gibt aber Scenarien da möchte man, das Container A Public ist und Container B nur privat.
Dann muss man in dem Port Teil des Dockers Containers, den Container explizit an die IP-

Adresse der interne Netzwerkkarte lauschen lassen.

Auszug Docker compose Ports mit expliziter IP

```
  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    container_name: phpmyadmin
    restart: always
      ports:
        - "172.0.2.2:8080:80"                    # phpMyAdmin
```

Der eigentliche Befehl

```
ufw route allow 80
```

Allerdings kann von allen Interfaces wie eth1 für public und eth0 für privates Netz zugegriffen werden.
Hinter Port 8080:80 verbirgt sich ein phpmyadmin.
Wir wollen aber nicht das dieser im Public netzt eth1 also Internet sichtbar ist

```
ufw route allow in on eht0 to any port 80
```

Damit erlauben wir nur Port 80 routing ins docker Netz über die eth0 Netzwerkkarte.
Es sei denn wir haben wie im Hinweis oberes Scenario, dann geben wir nicht in der ufw die Netzwerkkarte explizit an, sondern schon in der Docker Compose Datei.

Logging:

Die log Datei liegt unter

```
  /var/log/ufw.log
```