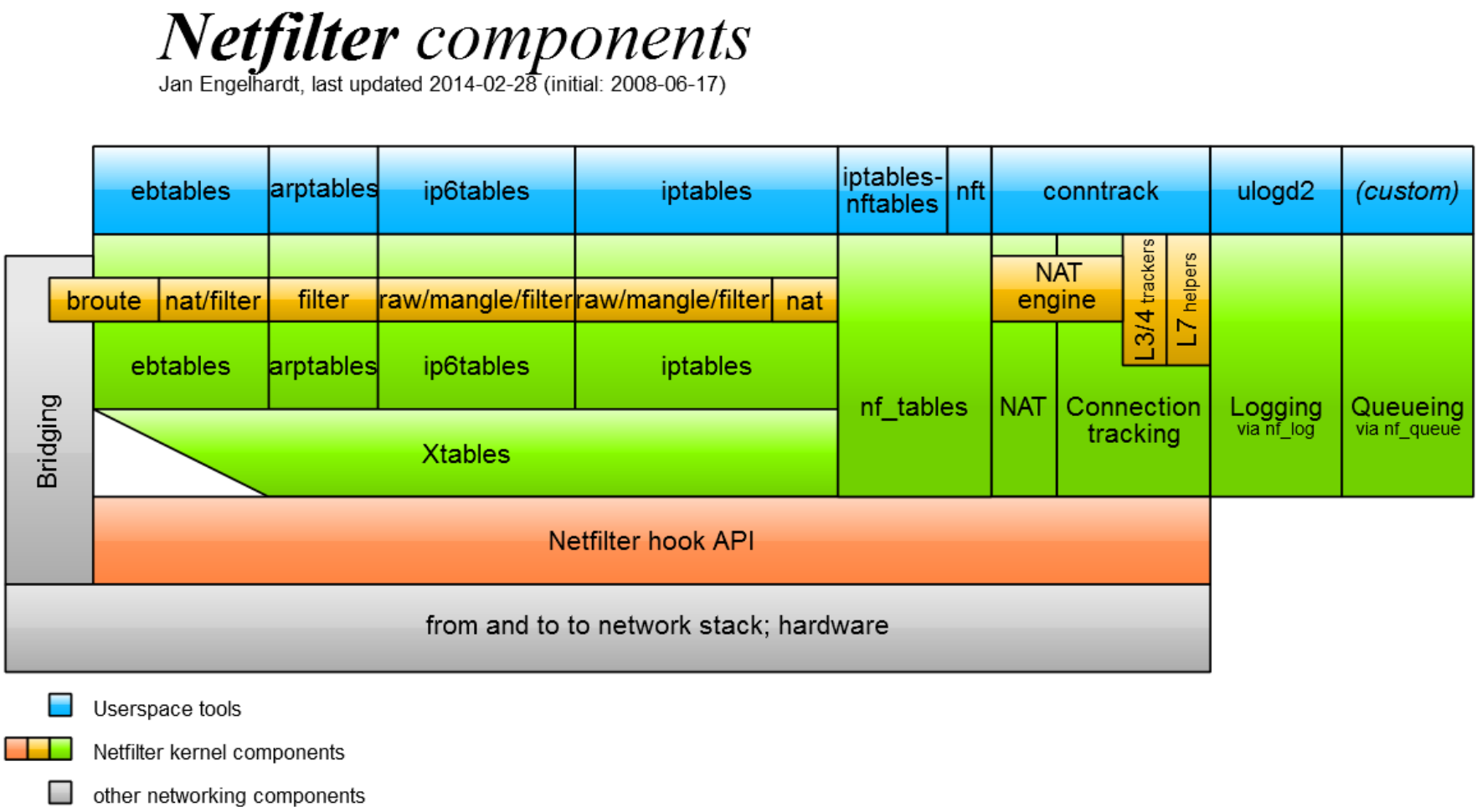
# Module FWL
## (Firewall)


IUT Béziers, dépt. R&T © 2014-2018
http://www.borelly.net/
Christophe.BORELLY@umontpellier.fr

# Généralités

- Netfilter
  - Framework de filtrage de paquets
- Depuis le noyau LINUX 2.4
- Basé sur des « tables » auxquelles on applique des « règles » sur différentes « chaînes ».
  - Table filter : INPUT, FORWARD et OUTPUT
  - Table nat : PREROUTING, OUTPUT et POSTROUTING
  - Autres tables mangle, raw, ...

# Composants de Netfilter



# Actions possibles

- **ACCEPT**
- **DROP** (détruit le paquet)
- **REJECT** (Informe en plus l'émetteur)
- **RETURN** (stoppe l'analyse de la chaîne)
- **LOG** (Sauvegarde une trace dans les logs)
- ...
- Exemple (-A ou --append et -j ou --jump) :

```
iptables -A INPUT -i lo -j ACCEPT
```

# Règles par défaut

- Politique par défaut :
  - `iptables -P INPUT ACCEPT`
  - `iptables -P FORWARD ACCEPT`
  - `iptables -P OUTPUT ACCEPT`
- Effacement des règles :
  - `iptables -F INPUT`
  - `iptables -F FORWARD`
  - `iptables -F OUTPUT`

# Règles de base

- ICMP : `-p icmp --icmp-type echo-request`
- HTTP : `-p tcp --dport 80`
- DNS : `-p udp --sport 53`
- Adresses IP :
  - `-d 192.168.1.5`
  - `-s 172.31.0.0/16`
- Limitation du nombre de paquets :
  - `-m limit --limit 30/m`

# Affichage des règles

```
# iptables -nvL --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     all  --  *      lo      0.0.0.0/0            0.0.0.0/0
# iptables -nvL -t nat --line-numbers
```

- Effacement (-D ou --delete)

  `iptables -D INPUT 1`

- Insertion (-I ou --insert)

# Suivi de connexions

- Connection Tracking (conntrack -L)
- Différents états :
  - NEW, ESTABLISHED, RELATED (ftp), INVALID…
- Exemples :
  - `-m state --state NEW`
  - `-m state --state ESTABLISHED,RELATED`

# Chaînes personnelles

- Nouvelle chaîne (--new-chain) :
  - `iptables -N cb`
- Effacement de la chaîne (--delete-chain) :
  - `iptables -X cb`
- Exemple d'utilisation :
  - `iptables -A INPUT -m state --state NEW -j cb`
  - `iptables -A cb -p tcp --dport 22 -j ACCEPT`

# Table NAT

- Cibles possibles :
  - **DNAT** (serveur)

```
iptables -t nat -A PREROUTING -p tcp -d 1.2.3.4
 --dport 80 -j DNAT --to-destination 192.168.1.1
```

  - **SNAT** (clients)

```
iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT
 --to-source 1.2.3.155-1.2.3.160
```

  - **MASQUERADE** (Idem SNAT, mais automatique)

```
iptables -t nat -A POSTROUTING -p tcp -o eth0
 -j MASQUERADE
```

  - **REDIRECT**

```
iptables -t nat -A PREROUTING -p tcp --dport 80
 -j REDIRECT --to-ports 8080
```

# Table MANGLE

- Modification du paquet :
  - **TOS** (Type Of Service, sur 8 bits)

```
iptables -t mangle -A PREROUTING -p tcp --dport 22
 -j TOS --set-tos 0x10
```

  - **TTL** (Time To Live)
  - **MARK** (Permet de « marquer » les paquets, cf. `tc` – Traffic Control)

```
iptables -t mangle -A PREROUTING -p tcp --dport 22
 -j MARK --set-mark 2
```

  - **SECMARK** (Idem mais pour un contexte de sécurité)
  - **CONNSECMARK** (Copie de contexte de sécurité)

# Sauvegarde et restauration

- Utilise moins de ressources qu'un script bash à la restauration s'il y a beaucoup de règles.
- Sauvegarde :
  - `iptables-save > /etc/iptables/rules.v4`
- Restauration :
  - `iptables-restore < /etc/iptables/rules.v4`

# Références

- http://www.netfilter.org/
- http://www.inetdoc.net/guides/iptables-tutorial/
- man iptables
- man iptables-extensions