Network Security 19: Firewalls Redux

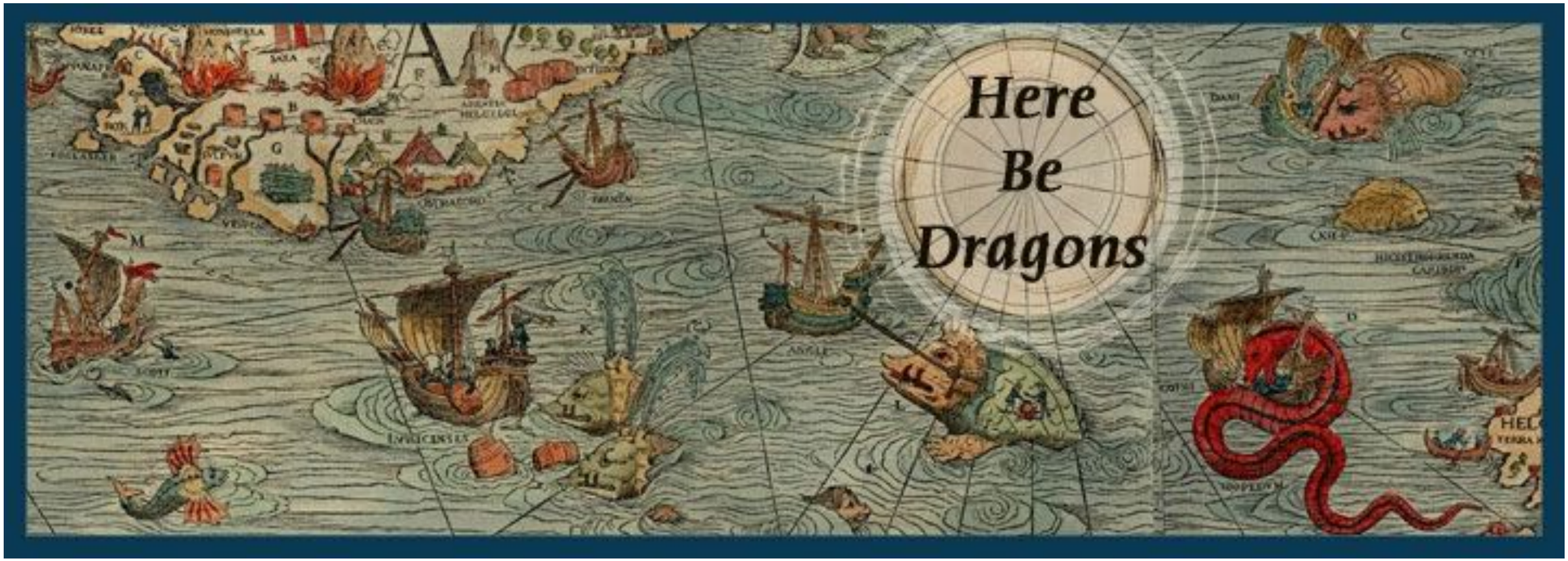
i.g.batten@bham.ac.uk

Rationale

- We skipped quickly through firewalls (ipf and iptables) without talking about syntax and operation.

- Let’s talk about syntax and operation.

- And learn about the chaos of real firewall configurations in anything other than fully change-controlled secure environments


Here
Be
Dragons

iptables

- Let’s dissect a real firewall on a real machine
- offsite6.batten.eu.org: a server which runs Cyrus replication, a few other services, and has some IPsec tunnels
- Only one set of rules: the INPUT table, called for packets headed up the stack to local processes (machine only has one interface)
- No NAT, so no mangle rules, just filter rules

iptables

- A sequence of rules are checked for each packet as it arrives, and any matching actions are taken.

- The table is maintained by using iptables -I (upper case i) (to insert rules) or iptables -A (to append rules)

The gory details

```
iptables -P INPUT DROP
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -N ossec
iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED
   -j ACCEPT
iptables -A INPUT -i eth0 -p icmp -j ACCEPT
iptables -A INPUT -i tun0 -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 23 -j DROP
iptables -A INPUT -p esp -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --dport 500 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --dport 4500 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --sport 17500 --dport 17500
   -j DROP
iptables -A INPUT -i eth0 -p udp -m udp --dport 67:68 -j DROP
iptables -A INPUT -s 128.204.195.0/24 -i eth0 -p udp -m udp --dport
   138 -j DROP
iptables -A INPUT -i eth0 -p udp -m udp --dport 53 -j ACCEPT
iptables -A INPUT -s 85.233.160.0/24 -i eth0 -p tcp -m tcp --dport
   25 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --dport 123 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 443 -m state --state
   NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 80 -m state --state
   NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 53 -m state --state
   NEW,ESTABLISHED -j ACCEPT
```

```
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 3306 -m policy --dir
   in --pol ipsec -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 147.188.192.0/24 -i eth0 -p tcp -m state
   --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 104.161.79.123/32 -p tcp -m tcp --dport 22 -m
   state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 149.255.32.0/24 -p tcp -m tcp --dport 22 -m
   state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 46.28.203.0/24 -p tcp -m tcp --dport 22 -m
   state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 5.101.172.0/24 -p tcp -m tcp --dport 22 -m
   state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 69.65.45.0/24 -p tcp -m tcp --dport 22 -m
   state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.17.28.0/24 -p tcp -m tcp --dport 22 -m
   state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 147.188.0.0/16 -p tcp -m tcp --dport 22 -m
   state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.2.79.220/32 -p tcp -m tcp --dport 22 -m
   state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.187.150.208/28 -p tcp -m tcp --dport 22 -m
   state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 104.161.79.123/32 -p udp -m udp --dport 161
   -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -j
   REJECT --reject-with icmp-port-unreachable
.
```

```
iptables -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN
   -m hashlimit --hashlimit-upto 1/min --hashlimit-burst 1
   --hashlimit-mode srcip,dstport --hashlimit-name loglimit
   --hashlimit-srcmask 24 -j LOG
iptables -A INPUT -p tcp -m tcp --dport 22 --tcp-flags
   FIN,SYN,RST,ACK SYN -j REJECT --reject-with
   icmp-port-unreachable
iptables -A INPUT -m hashlimit --hashlimit-upto 1/min
   --hashlimit-burst 1 --hashlimit-mode srcip,dstport
   --hashlimit-name loglimit --hashlimit-srcmask 24 -j LOG
```

•

Policy

```
iptables -P INPUT DROP
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
```

This is the policy for what happens if you arrive at the end of the table and haven't made a decision. Default drop INPUT, default pass FORWARD and OUTPUT (machine is configured to not forward)

Extra Tables

```
iptables –N ossec
iptables –A INPUT –i eth0 –j ossec
```

This is a chain that will be used to file packets against a dynamic list of bad sites maintained by the OSSEC HIDS.

OSSEC

-s: source

-j: jump

```
iptables -A ossec -s 182.100.67.0/24 -j DROP
iptables -A ossec -s 182.118.53.0/24 -j DROP
iptables -A ossec -s 222.161.4.0/24 -j DROP
iptables -A ossec -s 58.218.213.0/24 -j DROP
iptables -A ossec -s 61.240.144.0/24 -j DROP
```

This is a list of the networks that have been making continuous SSH login attempts to my machines. After a while, I distribute the firewall list around my whole network and block the whole /24 for a day

End of a user-created chain?

- If you haven't executed a jump to a target like DROP or ACCEPT, the flow of control returns to where the the chain was invoked
-

Initial Acceptance

-m: module

```
iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

—ctstate: connection state

The “conntrack” module has to be used explicitly with Linux firewalls (with ipf, anything whose state was kept with “keep state” is automatically accepted).

Initial Acceptance

```
iptables -A INPUT -i eth0 -p esp -j ACCEPT
iptables -A INPUT -i eth0 -p icmp -j ACCEPT
iptables -A INPUT -i tun0 -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
```

-i: interface

-p: protocol

- ESP is encrypted IPsec traffic, and cannot be processed by the firewall until it has been decoded
- tun0 is an old VPN interface where I trust the destination (no longer in use)
- Accepting everything on loopback protects against stupid mistakes

Protocols

```
iptables -A INPUT -i eth0 -p udp -m udp --dport 500 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --dport 4500 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -m udp --sport 17500 --dport 17500 -j DROP
iptables -A INPUT -i eth0 -p udp -m udp --dport 67:68 -j DROP
iptables -A INPUT -s 128.204.195.0/24 -i eth0 -p udp -m udp --dport 138 -j DROP
iptables -A INPUT -i eth0 -p udp -m udp --dport 53 -j ACCEPT
```

—dport: destination port

—sport: source port

- 500: IKE
- 4500: IKE with NAT
- 17500: Shamefully, I can’t remember
- 67 and 68: dhcp/bootp
- 138: NetBIOS (background noise)
- 53: DNS

Protocols

```
iptables -A INPUT -s 147.188.0.0/16 -p tcp -m state --state NEW -m tcp --dport 3000 -j ACCEPT
iptables -A INPUT -s 81.2.79.220/32 -p tcp -m tcp --dport 3000 -m state --state NEW -j ACCEPT
iptables -A INPUT -s 85.233.160.0/24 -i eth0 -p tcp -m tcp --dport 25 -m state \
      -state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.2.79.220/32 -p udp -m state --state NEW -m udp --dport 9996 -j ACCEPT
```

- -m invokes a module, and all the following arguments are passed to that module up until the next -m. So you can use multiple modules on the same packet. Here we see a packet being checked for being the start of a new flow, and being to port 3000, where I was running a non-standard web server for NTOP.

- Netflow data for NTOP travels over port 9996 UDP.
- Permit email from one particular network only (the "ESTABLISHED" does nothing: can you think why?)

More Protocols

```
iptables -A INPUT -i eth0 -p udp -m udp --dport 123 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 53 -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 81.2.79.220/32 -i eth0 -p tcp -m tcp --dport 4242 \
    -m state --state NEW,ESTABLISHED -j ACCEPT
```

Again, ESTABLISHED does nothing

NTP, http/s, ssh, DNS over TCP, marking undergraduate exercises

Slow logging of attacks

```
iptables -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN \
     -m hashlimit --hashlimit-upto 1/min --hashlimit-burst 1 \
    -hashlimit-mode srcip,dstport --hashlimit-name loglimit \
    -hashlimit-srcmask 24 -j LOG
```

```
iptables -A INPUT -m hashlimit --hashlimit-upto 1/min --hashlimit-burst 1 \
    --hashlimit-mode srcip,dstport --hashlimit-name loglimit \
    --hashlimit-srcmask 24 -j LOG
```

—tcp-flags looks at the first set of flags, and matches if any of the second set of flags are set; here we are looking for first packet on connection hashlimit module maintains a hashtable of addresses and ports, and rate-limits based on new entries

IPsec Policy

```
iptables -A INPUT -i eth0 -p tcp -m tcp --dport 3306 \
    -m policy --dir in --pol IPsec \
-m state --state NEW,ESTABLISHED -j ACCEPT
```

Accept MySQL, but only over IPsec

“Friendly Machines”

```
iptables -A INPUT -s 147.188.192.0/24 -i eth0 -p tcp -m state
    \-state NEW,ESTABLISHED -j ACCEPT
iptables -A INPUT -s 147.188.192.0/24 -p tcp -m state
     \-state NEW,ESTABLISHED -m tcp --dport 161 -j ACCEPT
iptables -A INPUT -s 147.188.192.0/24 -i eth0 -p udp -m udp --dport 161 -j ACCEPT
```

All TCP from bham-cs server network
All SNMP (tcp and udp) from same

ipf

- Here’s a more complex scenario
- mail.batten.eu.org, my primary mail server
- Used to be a global zone with shared IP, hence all the /30 netmasks
- Now a local zone

The file (1)

```
block in quick from pool/100 to any

# pass outbound traffic and replies to it
pass out quick on mailprod0 proto tcp from 147.188.192.248/30 to any flags S/SA keep state
pass out quick on mailprod0 proto udp from 147.188.192.248/30 to any port = 123
pass out quick on mailprod0 proto udp from 147.188.192.248/30 to any keep state
pass out quick on mailprod0 proto icmp from 147.188.192.248/30 to any keep state

pass in quick on mailprod0 proto ipip from 81.2.79.220 to any
pass out quick on mailprod0 proto ipip from any to 81.2.79.220

pass in quick on ip.tun0
pass out quick on ip.tun0
pass in quick on log1
pass out quick on log1

# shut down stray TCP connections and block all other traffic

block return-rst out quick proto tcp all
block out quick all

# zap rfc1918 and so on: no quick on pass because these are first-pass rules

pass in on mailprod0 all head 100
block in quick from 192.168.0.0/16 to any group 100
block in quick from 172.16.0.0/12 to any group 100
block in quick from 10.0.0.0/8 to any group 100
block in quick from 127.0.0.0/8 to any group 100
block in quick from 169.254.0.0/16 to any group 100
block in quick from 224.0.0.0/3 to any group 100
```

The file (2)

```
# block broadcast and multicast

block in quick from any to 147.188.192.255/32
block in quick from any to 224.0.0.0/3

# accept SMTP connections from known MXes

block return-rst in log first level local1.info quick \
    on mailprod0 proto tcp from any to 147.188.192.248/30 \
    port = 25 flags S/SA head 101
pass in quick from 147.188.128.54/32 to any keep state group 101
pass in quick from 147.188.128.127/32 to any keep state group 101
pass in quick from 147.188.128.129/32 to any keep state group 101
pass in quick from 147.188.128.219/32 to any keep state group 101
pass in quick from 147.188.128.221/32 to any keep state group 101
pass in quick from 147.188.192.250/32 to any keep state group 101
pass in quick from 147.188.192.249/32 to any keep state group 101

# generally acceptable protocols; block quick because there is no further TCP
# processing

# imaps, submission, domain, imap, http, https, ssh

block return-rst in log first level local1.info quick \
    on mailprod0 proto tcp from any to 147.188.192.248/30 \
    flags S/SA head 102
# 5877 is obsolete clone of 587,
# 993 is obsolete imaps prior to use of STARTTLS verb
# block return-rst in quick from any to any port = 993 keep state group 102
block return-rst in quick from any to any port = 5877 keep state group 102
pass in quick from any to any port = 587 keep state group 102
pass in quick from any to any port = 993 keep state group 102
pass in quick from any to any port = 53 keep state group 102
pass in quick from any to any port = 143 keep state group 102
pass in quick from any to any port = 80 keep state group 102
pass in quick from any to any port = 443 keep state group 102
pass in quick from any to any port = 22 keep state group 102
pass in quick from 128.204.195.144 to any port = 2010 keep state group 102
pass in quick from 128.204.195.144 to any port = 2007 keep state group 102
pass in quick from 128.204.195.144 to any port = 2009 keep state group 102
pass in quick from 128.204.195.144 to any port = 2003 keep state group 102
```

The file (3)

```
# and some UDP we need; again block quick as there is no further UDP processing
# ntp and domain
pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 123
pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 53 keep state
pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 1514 keep state
block return-icmp in quick on mailprod0 proto udp from any to 147.188.192.248/30
block in quick on mailprod0 proto udp all

# management machine can ping us

pass in quick on mailprod0 proto icmp \
    from 147.188.130.98/32 to 147.188.192.248/30 \
    icmp-type echo keep state

# backstop: send a reset for non-S/SA TCP traffic, discard all of it, with no logging
# (stray S/SA frames have been caught and logged higher up)
# logging for everything else

block return-rst in quick proto tcp all
block in log level local1.info quick all
```

“quick”

- ipf logic is to keep on making matches, recording the action that is required

- When it reaches the end of the rules, it takes the action from the last match

- quick means “done, take this action now”

OSSEC, again

```
block in quick from pool/100 to any
```

ipf “pools” are lists of addresses that are matched together in one rule

Used in this case to record the long-term bad sites and block them

Outbound Rules

```
# pass outbound traffic and replies to it
pass out quick on mailprod0 proto tcp from 147.188.192.248/30 to any flags S/SA keep state
pass out quick on mailprod0 proto udp from 147.188.192.248/30 to any port = 123
pass out quick on mailprod0 proto udp from 147.188.192.248/30 to any keep state
pass out quick on mailprod0 proto icmp from 147.188.192.248/30 to any keep state

pass in quick on mailprod0 proto ipip from 81.2.79.220 to any
pass out quick on mailprod0 proto ipip from any to 81.2.79.220
```

S/SA = Syn is Set, Ack is not set

keep state = accept anything that looks like a reply

ipip is a tunnel

Internal Networks

```
pass in quick on ip.tun0
pass out quick on ip.tun0
pass in quick on log1
pass out quick on log1
```

ip.tun0: result of de-encapsulating ipip
log1: private network into another zone

```
log1: flags=100001000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,PHYSRUNNING> mtu 9000 index 4
        inet 192.168.106.101 netmask ffffff00 broadcast 192.168.106.255
```

Default Block Out

```
# shut down stray TCP connections and block all other
# traffic

block return-rst out quick proto tcp all
block out quick all
```

Any packets exiting the machine “out of state” generate an RST back to the calling process (can you think why?)

Drop everything else

Block illegal packets

```
# zap rfc1918 and so on: no quick on pass because these are first-pass rules

pass in on mailprod0 all head 100
block in quick from 192.168.0.0/16 to any group 100
block in quick from 172.16.0.0/12 to any group 100
block in quick from 10.0.0.0/8 to any group 100
block in quick from 127.0.0.0/8 to any group 100
block in quick from 169.254.0.0/16 to any group 100
block in quick from 224.0.0.0/3 to any group 100
```

Drop packets on mailprod0 which appear to come from places that they shouldn’t be coming from

Multicast and Broadcast

```
# block broadcast and multicast

block in quick from any to 147.188.192.255/32
block in quick from any to 224.0.0.0/3
```

I’m on a shared network, but don’t need to listen to any particular protocols to operate

Multicast isn’t being used and can be dangerous

Mail from known relays

```
# accept SMTP connections from known MXes

block return-rst in log first level local1.info quick \
    on mailprod0 proto tcp from any to 147.188.192.248/30 \
    port = 25 flags S/SA head 101
pass in quick from 147.188.128.54/32 to any keep state group 101
pass in quick from 147.188.128.127/32 to any keep state group 101
pass in quick from 147.188.128.129/32 to any keep state group 101
pass in quick from 147.188.128.219/32 to any keep state group 101
pass in quick from 147.188.128.221/32 to any keep state group 101
pass in quick from 147.188.192.250/32 to any keep state group 101
pass in quick from 147.188.192.249/32 to any keep state group 101
```

Other TCP

```
block return-rst in log first level local1.info quick \
    on mailprod0 proto tcp from any to 147.188.192.248/30 \
    flags S/SA head 102
# 5877 is obsolete clone of 587,
# 993 is obsolete imaps prior to use of STARTTLS verb
# block return-rst in quick from any to any port = 993 keep state group 102
block return-rst in quick from any to any port = 5877 keep state group 102
pass in quick from any to any port = 587 keep state group 102
pass in quick from any to any port = 993 keep state group 102
pass in quick from any to any port = 53 keep state group 102
pass in quick from any to any port = 143 keep state group 102
pass in quick from any to any port = 80 keep state group 102
pass in quick from any to any port = 443 keep state group 102
pass in quick from any to any port = 22 keep state group 102
pass in quick from 128.204.195.144 to any port = 2010 keep state group 102
pass in quick from 128.204.195.144 to any port = 2007 keep state group 102
pass in quick from 128.204.195.144 to any port = 2009 keep state group 102
pass in quick from 128.204.195.144 to any port = 2003 keep state group 102
```

Look at SYN packets, match against list of valid ports (and in some cases sources)

Other UDP

```
# and some UDP we need; again block quick as there is no further UDP processing

# ntp and domain

pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 123
pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 53 keep state
pass in quick on mailprod0 proto udp from any to 147.188.192.248/30 port = 1514 keep state
block return-icmp in quick on mailprod0 proto udp from any to 147.188.192.248/30
block in quick on mailprod0 proto udp all
```

In 2015 NTP firewall should be tighter, but in this case it is legacy: mail.batten.eu.org is now a zone and Solaris zones don’t have their own clock, so don’t run NTP.

“return-icmp” sends an ICMP port unreachable.

University management machine (may be legacy)

```
# management machine can ping us

pass in quick on mailprod0 proto icmp \
   from 147.188.130.98/32 to 147.188.192.248/30 \
   icmp-type echo keep state
```

Passes ICMP echo requests (the active side of a PING) from one particular machine, and gets ready to send a response

Default Deny In

```
block return-rst in quick proto tcp all
block in log level local1.info quick all
```

All that is logged is obscure protocols that have been dropped, mostly for debugging purposes

Summary

- Firewalls suffer from bitrot: they accumulate “stuff” which you don’t understand but don’t dare remove
- Their syntax is usually complex, and the effects subtle and hard to debug
- Tendency to fiddle with a working one, rather than start from scratch