# Getting “freki” With Your Honeypot

Jonathan P. Camp -- Intelecy

# Outline

- Motivation
- iptables and Netfilter
- Freki
- Demo

# Motivation

<story time>

# This doesn’t work

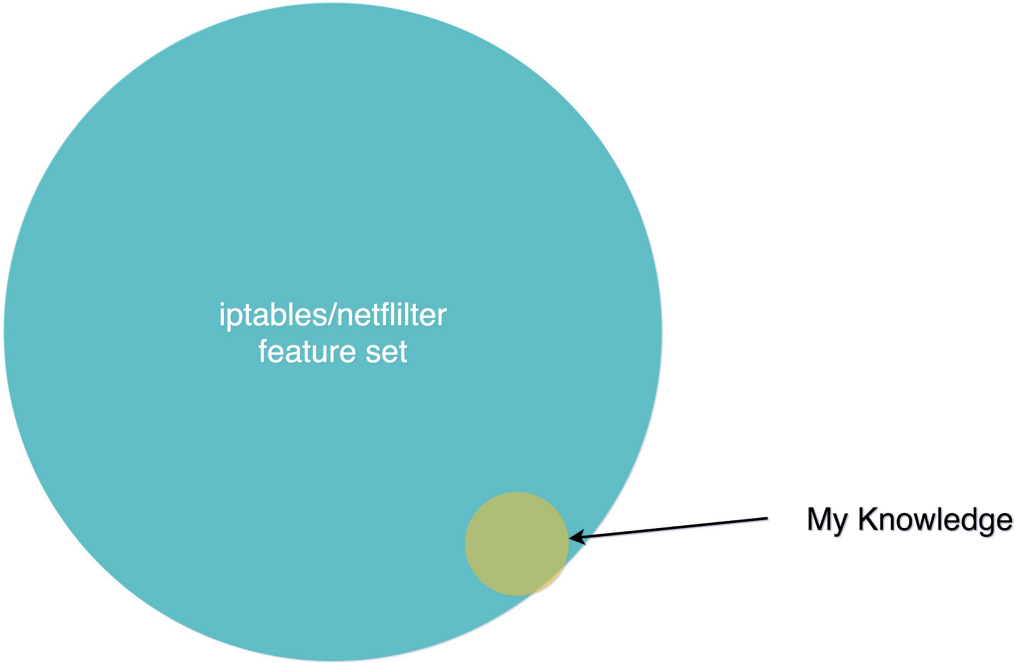
```go
func server() {
    for i := 1; i < 65536; i++ {
        go func(port int) {
            ln, _ := net.Listen("tcp", fmt.Sprintf(":%d", port))
            for {
                conn, _ := ln.Accept()
                go handleConnection(conn)
            }
        }(i)
    }
}
```

# What I'm looking for

```go
func myHandler(conn net.Conn) {
    // do something with the new connection
}
```

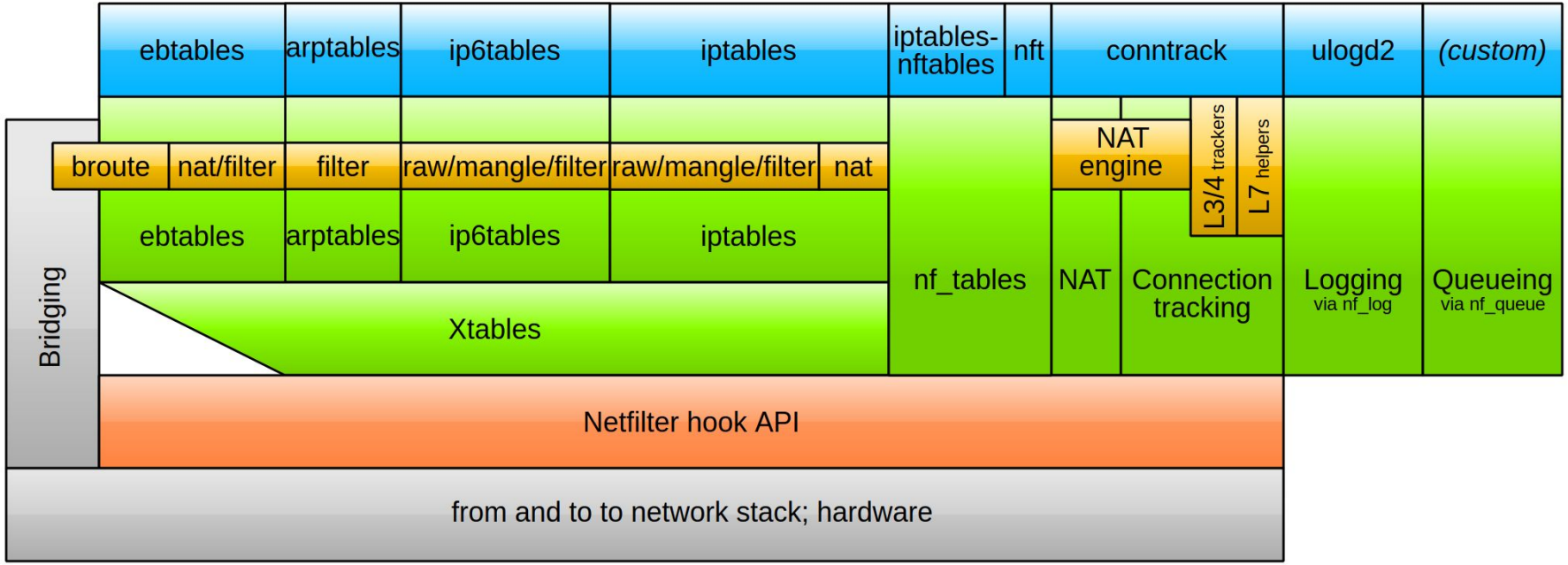
# Enter iptables/netfilter/nfqueue

# Caveat



# *Netfilter components*

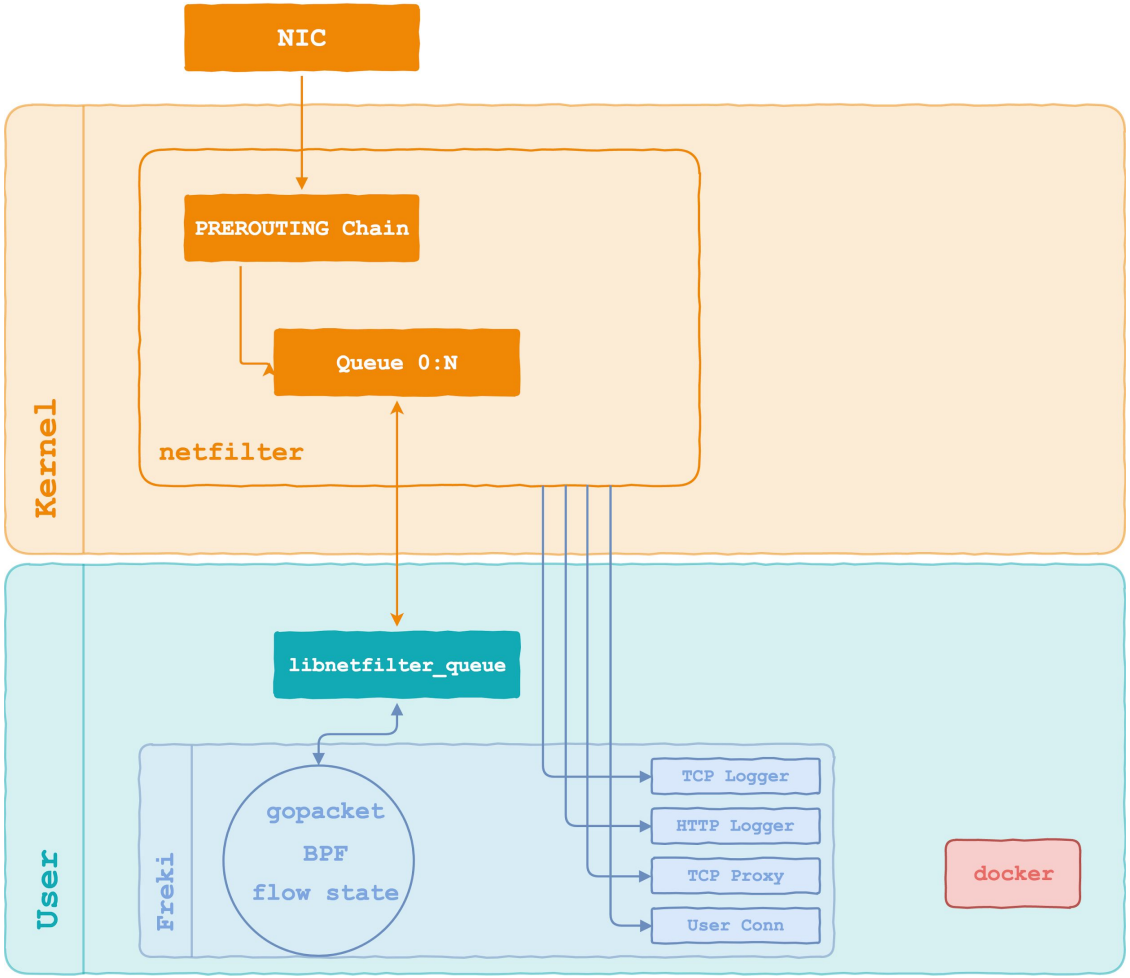
Jan Engelhardt, last updated 2014-02-28 (initial: 2008-06-17)




Userspace tools

Netfilter kernel components

other networking components


NIC
PREROUTING Chain
Queue 0:N
netfilter
Kernel
libnetfilter_queue
Freki
gopacket
BPF
flow state
TCP Logger
HTTP Logger
TCP Proxy
User Conn
docker
User

# Freki Features

- Runs in user-space
- Matchers based on BPF
- Handlers:
  - passthrough
  - drop
  - rewrite
  - proxy
  - log TCP|HTTP
  - conn_handler (library only)
- Proxy Targets:
  - TCP (i.e. `tcp://portquiz.net:666`)
  - Docker (i.e. `docker://redis:6379`)

# rules.yaml

```yaml
version: 1
rules:
  - match: tcp dst port 22 and src host 1.2.3.4
    type: passthrough
  - match: tcp dst port 10022
    type: rewrite
    target: 22
  - match: tcp dst port 379
    type: proxy
    target: docker://redis:6379
  - match: tcp dst port 666
    type: proxy
    target: tcp://portquiz.net:666
  - match: tcp port 80 or tcp port 8080
    type: log_http
  - match: tcp portrange 5000-5010
    type: drop
  - match: tcp portrange 7000-8000
    type: conn_handler
    target: echo
  - match: tcp port 8888
    type: drop
  - match: tcp
    type: log_tcp
  - match:
    type: passthrough
```

# Echo server implementation

```go
func echo(conn net.Conn, md *freki.Metadata) error {
    const timeout = time.Second * 5
    defer conn.Close()

    log.Printf("[echo    ] new conn: %v —> %d", conn.RemoteAddr(), md.TargetPort)

    b := bufio.NewReader(conn)

    for {
        conn.SetDeadline(time.Now().Add(timeout))
        line, err := b.ReadBytes('\n')
        if err ≠ nil {
            break
        }
        fmt.Fprintf(conn, "hello on: %d\n%s", md.TargetPort, line)
    }

    return nil
}
```

# Demos

#1 -- echo server

#2 -- redis on all ports

#3 -- port.party

# Back to `corppki`...

```
INFO[180605] [prt.prty] request: 223.91.252.22 corppki - [2019-02-24T16:52:02Z] "GET
/crl/mswww(6).crl HTTP/1.1" 200 2 "" "Microsoft-CryptoAPI/10.0" 80

INFO[180607] [prt.prty] request: 223.91.252.22 corppki - [2019-02-24T16:52:04Z] "GET
/crl/MSIT%20Machine%20Auth%20CA%202(1).crl HTTP/1.1" 200 2 "" "Microsoft-CryptoAPI/10.0" 80

INFO[188060] [prt.prty] request: 78.30.2.215 corppki - [2019-02-24T18:56:17Z] "GET
/crl/mswww(6).crl HTTP/1.1" 200 2 "" "Microsoft-CryptoAPI/6.1" 80
```

# crt.sh

```
X509v3 CRL Distribution Points:

    Full Name:
      URI:http://mscrl.microsoft.com/pki/mscorp/crl/Microsoft%20Secure%20Server%20Authority(8).crl
      URI:http://crl.microsoft.com/pki/mscorp/crl/Microsoft%20Secure%20Server%20Authority(8).crl
      URI:http://corppki/crl/Microsoft%20Secure%20Server%20Authority(8).crl

Authority Information Access:
    CA Issuers - URI:http://www.microsoft.com/pki/mscorp/Microsoft%20Secure%20Server%20Authority(8).crt
    CA Issuers - URI:http://corppki/aia/Microsoft%20Secure%20Server%20Authority(8).crt
```

# Thank you!

GitHub: https://github.com/kung-foo/freki/

NFQUEUE: https://home.regit.org/netfilter-en/using-nfqueue-and-libnetfilter_queue/

https://port.party:1337