How We Built Magic Transit
Agenda

● Who are we
● Who is Cloudflare
● What is Magic Transit
● Designing The Product
● How Magic Transit Works
● Questions
Who Are We?
Who is Cloudflare?
“Helping build a better internet”
But How???

- GLOBAL Anycast network
- Provide security & performance services:
  - DDoS
  - DNS
  - Spectrum
  - CDN, WAF, Workers
  - ...

{erich, conjones}@cloudflare.com
HTTP Request

Proxied HTTP Request

Customer Servers

{erich, conjones}@cloudflare.com
Customer Servers

HTTP Request

Proxied HTTP Request

Attack Traffic

{erich, conjones}@cloudflare.com
HTTP Request → Customer Servers

Proxy HTTP Request → Proxied HTTP Request

Attack Traffic → Cloudflare

{erich, conjones}@cloudflare.com
Customers to Cloudflare:

Can you do this with all my traffic?
But why?

- Consolidation of vendors
- Network functions in the network (not the boxes)
- Proxying is not always the answer
Cloudflare to customers:

- Much of this operates at L3 already
- We have a big network
- We’ll build you a Magic Transit
What is Magic Transit?
Magic Transit

- Network functions as a service
- L3 DDoS scrubbing
- Firewall

By:
- Existing DDoS solution
- Advertises customer prefixes on BGP
IP Packets

GRE Packets

IP Packets (DSR)

Customer Router

{erich, conjones}@cloudflare.com
Designing the product

Proof of concept
Will it even work?

- Does GRE actually work that way?
- Does it scale?
- How do we fit it in our platform?
A Cloudflare POP
Challenges

- The metal is a router
- Uniform servers
- Customer isolation
- Security
- Timeline + skillset
Technology options

- VRFs
- XDP
- Namespaces
VRF

Pros:
- Routing isolation
- Stays at L3

Cons:
- Lacking expertise
- Unclear iptables interaction
XDP

Pros:
- Bypass a lot of processing
- Lots of expertise
- “Just Code”

Cons:
- It’s a lot of code
- Time investment
Namespaces

Pros:
● Isolation
● Expertise
● Bash!

Cons:
● Lots of veths
● Complex setup
And the winner is...
Namespaces

- Proof of concept in a few weeks
- Surprisingly small shell script
- Dataplane performance is pretty good
Namespaces

- Proof of concept in a few weeks
- Surprisingly small shell script
- **Dataplane performance is pretty good**

We can use this for the product!
Designing the product

The actual product
One namespace per customer

metal

cust_ns
Configuration daemon

Root NS network configs

Configurator: Conduit

cust_ns

metal
Configurator: Conduit

- Daemon written in Go
- Uses QS as config distribution
- Can crash without (big) consequence
Behind The Curtain Of Magic Transit

Shuffling packets from eyeballs to origins

{erich, conjones}@cloudflare.com
How Packets Get To The Origin
Packets Arrive At The Edge
Magic Happens
Origin Responds to Eyeball
An Edge Metal

metal
One Namespace Per Customer

metal
cust_ns
Packets Arrive

metal

eth0

cust_ns
A Lil XDP

metal

eth0

XDPD

cust_ns
Marking Packets

```
metal
  | iptables
  |  pre-routing
eth0
  | XDPD
```

```
cust_ns
```
Route Tables with an S

metal
  └─ iptables
      │   pre-routing
eth0
  │   XDPD
  │   └─ GRE_table
      │       main
      └─ cust_ns
Getting Traffic Into The Namespace

- metal
- eth0
  - XDPD
  - iptables
    - pre-routing
  - GRE_table
    - main
    - cust_veth
    - cust_veth
veth IPs...aaand its gone
Connection tracking with half the connection, with AF_XDP
Your Traffic, Your FW Rules

metal

iptables
pre-routing

eth0
XDPD

GRE_table
main

cust_veth
flowtrackd

nftables
pre-routing

cust_veth

cust_ns
Dummy nft rule + conntrack

ethic, conjones}@cloudflare.com
Dummy nft rule + notrack
Route Tables With an S, The Second, pt 1

metal

iptables
pre-routing

GRE_table
main

cust_veth
flowtrackd

cust_veth

nftables
pre-routing

cust_table
main

cust_ns

eth0

XDPD
Route Tables With an S, The Second, pt 2
Wrapped With A Bow

diagram showing network components and table structures
Devices, Devices, Devices

ECMP and Route Priorities
Devices, Devices, Devices

MAGIC-3

- metal
  - iptables (pre-routing)
  - GRE_table (main)
  - cust_veth
  - flowtrackd

- cust_ns
  - nftables (pre-routing)
  - cust_table (main)
  - cust_table_alt

- eth0
  - XDPD

- gre1

{erich, conjones}@cloudflare.com
Focusing On One Device

- metal
  - iptables pre-routing
- eth0
  - XDPD
- GRE_table
  - main
- cust_veth
- cust_veth
- flowtrackd
- cust_table
  - cust_table_alt
  - main
- gre1
- cust_ns
- {erich, conjones}@cloudflare.com
Default Route

- `metal`
  - `iptables` (pre-routing)
  - `GRE_table`
    - main
  - `cust_veth`
  - `flowtrackd`
  - `nftables` (pre-routing)
  - `cust_table`
    - `cust_table_alt`
    - main
  - `gre1`

- `eth0`
- `XDPD`
Outgoing Rules

- metal
  - `iptables pre-routing`
  - `GRE_table main`
  - `cust_veth`
  - `flowtrackd`

- cust_ns
  - `nftables pre-routing`
  - `cust_veth`
  - `nftables post-routing`
  - `cust_table cust_table_alt main`
  - `gre1`
Traverse The veth, again

- metal
  - iptables pre-routing
  - GRE_table main
  - cust_veth
  - flowtrackd

- eth0
  - XDPD

- cust_veth
  - cust_veth
  - nftables post-routing

- cust_ns
  - nftables pre-routing
  - cust_table cust_table_alt main
  - gre1

{erich, conjones}@cloudflare.com
NIC Again

eth0
- XDPD
- iptables post-routing

GRE_table
main

cust_veth

flowtrackd

nftables post-routing

cust_table
cust_table_alt
main

gre1

nftables pre-routing

cust_ns

metal

iptables pre-routing

{erich, conjones}@cloudflare.com
shipit!
Whoops! We bricked a metal!

*Slow-loop/Fast-loop Config Updates*
<todo>quippy gre-ping title</todo>
Open Questions

- tx/rx checksum offloading
- Disappearing IPs
- ARP?
Questions?

{erich, conjones}@cloudflare.com