IPSEC AND IKE TUTORIAL

NetDev0x12, Montreal July 2018

Presented by:
Paul Wouters, Red Hat
Sowmini Varadhan, Oracle

libreswan

The Libreswan Project

An Internet Key Exchange ("IKE") daemon for IPsec

- Enterprise VPN solutions based on IPsec
- Opportunistic IPsec:
  Encrypt the Internet

- Certifications
  FIPS, Common Criteria, USGv6, TAHI
- Contribute to IETF Standards
  IKE and IPsec

- See appendix slides for SWAN history

What is the IPsec Suite

IKE + IPsec = VPN

IKE (USERLAND)

ISAKMP, IKE SA, PHASE 1

UDP PORT 500 AND 4500

- Command Channel
- Peer authentication
- Connection parameter negotiation
- IPsec symmetric key generation
- Communicates to kernel

- IKE itself is encrypted
- IKE does not encrypt the IP traffic

IPsec (KERNEL)

IPsec SA, CHILD SA, PHASE 2

PROTOCOL 50 AND 51

- Data Channel
  - Encapsulated Security Payload (ESP) IP packet encryption
  - Authenticated Header (AH)
  - ESPinUDP (for NAT)

- Tunnel Mode (IP in IP)
- Transport Mode

Linux IPsec Implementation

XFRM/NETKEY interaction with userland

1. IPsec in the kernel has policies (SPD) and states (SAD)
   - Packets matching policies without a linked state cause ACQUIREs to userland
   - Packets matching policies with a linked state causes encryption/decryption
2. Userland (libreswan) opens netlink socket to the kernel
   - Request to receive ACQUIREs and inserts “trap” policies
3. Userland (libreswan) processes ACQUIREs
   - Perform IKE negotiation with remote peer
   - Send IPsec policy and encryption/authentication keys to the kernel via netlink
4. Kernel processes netlink messages
   - Install received crypto keys in state, link crypto state to policy
   - If TCP triggered, send out previously cached packet

Internet Key Exchange (IKE protocol)

Designed by the Internet Engineering Task Force (IETF)

- 1998: RFC 2409 IKE version 1 (Proposed Standard)
- 2005: RFC 4306 IKE version 2 (Proposed Standard)
- 2006: RFC 4718 IKEv2 Clarifications and Implementation Guidelines
- 2010: RFC 5996 IKE version 2 (Proposed Standard)
- 2015: RFC 7296 IKE version 2 (Internet Standard)

- IKEv1: Main Mode negotiation: 3 round trips to establish IKE SA
- IKEv1: Aggressive Mode negotiation: 1.5 round trip for IKE SA
    - initiator ID not protected by DH, vulnerable when using weak PreSharedKey
- IKEv1: Quick Mode: one round trip for each new IPsec SA

- IKEv2: Initial Exchange: 2 round trips to establish IKE and 1 IPsec SA
- IKEv2: Create Child Exchange: one round trip for each further IPsec SA

The IKE Protocol

General mechanism

1) Perform Diffie-Hellman Key Exchange to start a private communication channel
   - Diffie-Hellman groups, encryption and integrity algorithms for IKE
2) Peers authenticate each other, resulting in an IKE Security Association (IKE SA)
   - X.509, raw pubic key, PreSharedKey (PSK), EAP, user/password, etc.
3) Exchange information about supported and desired features
   - Including NAT detection, Dead Peer Detection, Session Resumption, etc
4) Negotiate one or more IPsec Security Associations
   - IPsec encryption and integrity algorithms and symmetric keys
   - Tunnel Mode (IPIP) or Transport Mode
   - Source and Destination network/port ranges (if Tunnel Mode)

IKEv1

Don't start new deployments, only use when remote endpoint insists

- Decline sped up around with Windows 10 and iPhone iOS8 around 2014/2015
- Android is the last bastion without IKEv2 (bug the developers!)
- The IKEv1 protocol (and the ESP protocol) have never been broken
    - IKEv1 Aggressive Mode + weak PreSharedKeys allows offline brute force attacks
- Old deployments, so typically use ancient configuration parameters:
    - AES_XCBC or even 3DES instead of AES_GCM or CHACHA20POLY1306
    - HMAC-MD5 or HMAC-SHA1 instead of HMAC-SHA2 or AEAD
    - Diffie-Hellman group 2 (MODP1024) or group 5 (MODP1536) instead of group 14 (MODP2048), ECP groups or Curve25519
    - PreSharedKeys instead of public key or X.509 based authentication
    - Prescient enough to have Post Quantum protection (but at expense of awkward error reporting)
- Some improvements not universally supported (NAT, DPD, fragmentation)

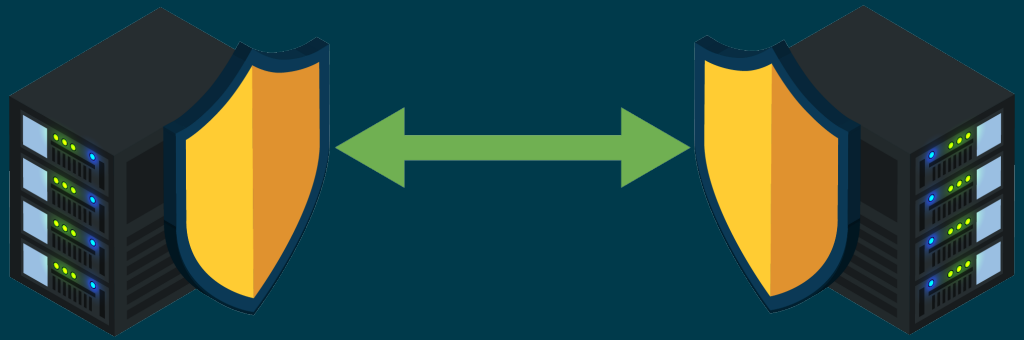
IKEv2

Universally supported except Android (whatsup google devs?)

- Used for 5G, voLTE, Wi-Fi Calling on every phone.
- Used by Xbox One
- The protocol has (still) never been broken
- IKEv2 has less roundtrips to establish, session resumption, MOBIKE migration
- Latest crypto suites (AES_GCM, CHACHA20POLY1306, Curve25519, AES_CCM for IoT)
- NAT support, DPD/liveness, fragmentation, resumption, more AUTH methods
- Postquantum PresharedKey (PPK) support while awaiting postquantum algorithms
- Better retransmission protocol (only initiator retransmits)
- Better error reporting (bad AUTH no longer leads to undecryptable message)
- Anti-DDOS support via COOKIES, IKEv2 REDIRECT
- Improved Configuration Payloads (client/server) such as narrowing, DNS, DNSSEC
- Use IKEv2 for all NEW deployments (Android can do IKEv2 via strongSwan app)

Typical Host to Host VPN

Tunnel Mode or Transport Mode. Tunnel Mode is preferred when there are NATs



Host to Host VPNs

Create an identity for your IPsec gateway

- Raw RSA
  - ipsec newhostkey
  - ipsec showhostkey --list
  - ipsec showhostkey –left –ckaid XXXXXXX
- X.509 Certificates via PKCS#12 (grab one from netdev.nohats.ca)
  - ipsec import */dir/path/pwouters.p12*
  - *certutil -L -d sql:/etc/ipsec.d*
- PreSharedKey (we won't be using these, because you shouldn't start a bad habit)
  - echo “YourID TheirID :PSK “SomeStrongSecret” > */etc/ipsec.d/somename.secret*

Host to Host VPN

IPsec is peer to peer, not client to server. Hence no source/dest but left/right

```
# /etc/ipsec.d/pwouters.conf
conn example1
     left=%defaultroute
     leftid=@pwouters
     leftrsasigkey=0sXXXXXXXX[…]
     right=TheirIPorHostname
     rightid=@TheirID
     authby=rsasig
     rightrsasigkey=0xYYYYYY[…]
     #auto=start, ondemand, add
     auto=add
     ikev2=insist
```

Host to Host VPNs

Establish the VPN connection

```
# systemctl enable ipsec
# systemctl start ipsec
# ipsec auto --add example1
002 added connection description "vpn.nohats.ca"
# ipsec auto --up example1
002 "example1"[1] 193.110.157.152 #2: initiating v2 parent SA
134 "example1"[1] 193.110.157.152 #3: STATE_PARENT_I2: sent v2I2, expected v2R2 {auth=IKEv2
cipher=aes_gcm_16_256 integ=n/a prf=sha2_512 group=DH19}
002 "example1"[1] 193.110.157.152 #3: IKEv2 mode peer ID is ID_FQDN: '@TheirID'
003 "example1"[1] 193.110.157.152 #3: Authenticated using RSA
002 "example1"[1] 193.110.157.152 #3: negotiated connection [192.168.1.1-192.168.1.1:0-65535 0] ->
[193.110.157.152-193.110.157.152:0-65535 0]
004 "example1"[1] 193.110.157.152 #3: STATE_V2_IPSEC_I: IPsec SA established tunnel mode {ESP/
NAT=>0x0f2ba66b <0xfd4bd38d xfrm=AES_GCM_16_256-NONE NATOA=none NATD=193.110.157.152:4500
DPD=active}
```

Host to Host VPNs

Verify the VPN connection is up

```
# ping -c4 TheirIP

# ipsec whack --trafficstatus
006 #2: "example1", type=ESP, add_time=1234567890, inBytes=168, outBytes=168, id='@TheirID'

# ip xfrm state
# ip xfrm policy

# ipsec status | grep example1

# tcpdump -n esp or port 500 or port 4500 -i eth0
```

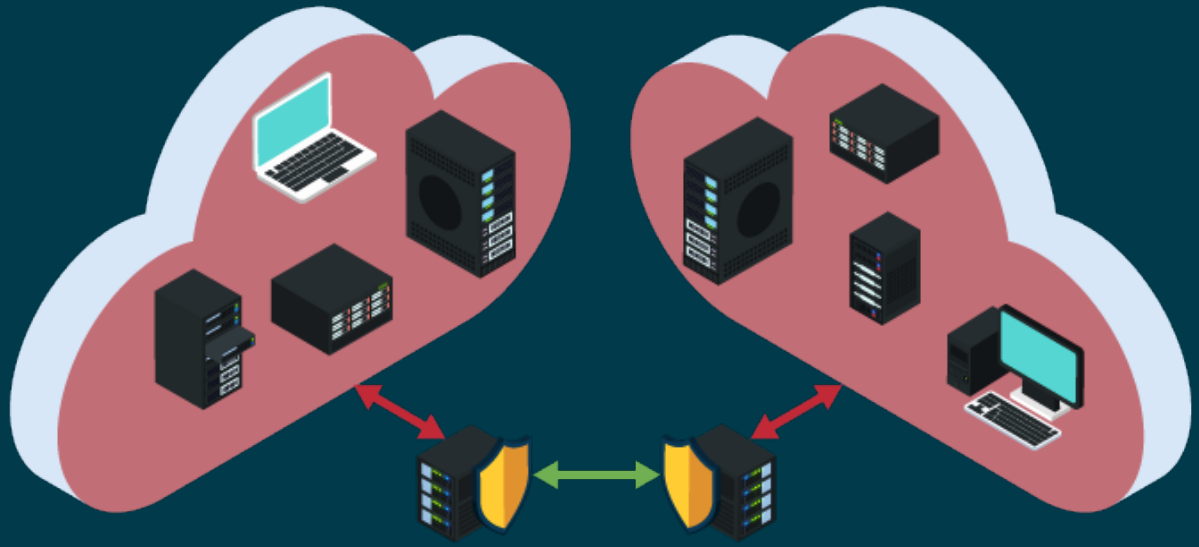
Linux XFRM / NETKEY Kernel State and Policy

```
# ip xfrm policy
src 10.3.230.191/32 dst 10.0.0.0/8
        dir out priority 666 ptype main
        tmpl src 76.10.157.68 dst 209.132.183.55
                proto esp reqid 16413 mode tunnel
src 0.0.0.0/0 dst 10.3.230.191/32
        dir fwd priority 666 ptype main
        tmpl src 209.132.183.55 dst 76.10.157.68
                proto esp reqid 16413 mode tunnel
src 0.0.0.0/0 dst 10.3.230.191/32
        dir in priority 666 ptype main
        tmpl src 209.132.183.55 dst 76.10.157.68
                proto esp reqid 16413 mode tunnel

# ip xfrm state
src 209.132.183.55 dst 76.10.157.68
        proto esp spi 0x605ad2be reqid 16413 mode tunnel
        auth-trunc hmac(sha1) 0x4b7e46cdee9c27588a1a75f6846073cea 96
        enc cbc(aes) 0x11ddc9080945111087e81f9ebda5aacb7612c78af1895
src 76.10.157.68 dst 209.132.183.55
        proto esp spi 0x8ca00de3 reqid 16413 mode tunnel
        auth-trunc hmac(sha1) 0x1119585d334a88e023134a100eca6b09f 96
        enc cbc(aes) 0x310b852b9cbaf2cace7979c1aeb5df4b32eb418c5c300
```

Typical Site to Site VPN

Individual networks are unencrypted, only the interconnect is encrypted



Site to Site VPN

Create a connection, eg */etc/ipsec.d/yourname.conf*

```
conn example1
     left=MyIP
     leftid=@pwouters
     leftrsasigkey=0sXXXXXXXX[…]
     leftsubnet=13.0.1.0/24
     right=TheirIPorHostname
     rightid=@TheirID
     rightrsasigkey=0xYYYYYY[…]
     rightsubnet=13.0.2.0/24
     authby=rsasig
     auto=start
     ikev2=insist
```

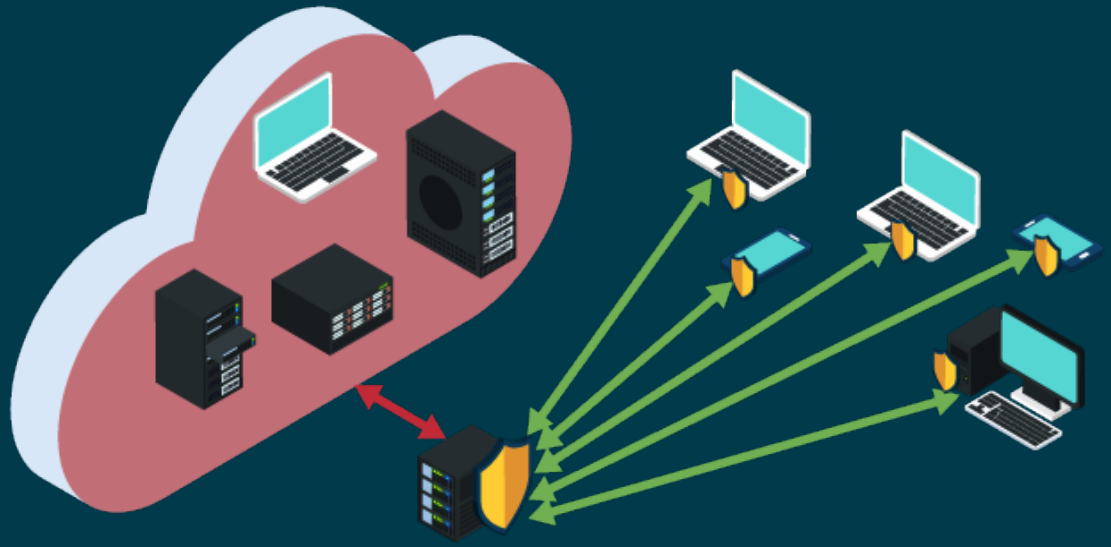
Multiple Sites to Sites VPN

Create separate conn's for IPv4 and IPv6 subnets, 4in6, 6in4

```
conn example1
     left=MyIP
     leftid=@pwouters
     leftrsasigkey=0sXXXXXXXX[…]
     leftsubnets={13.0.1.0/24, 192.168.13.0/24, ...}
     right=TheirIPorHostname
     rightid=@TheirID
     rightrsasigkey=0xYYYYYY[…]
     rightsubnet=13.0.2.0/24, 192.168.243.0/24, ...}
     authby=rsasig
     auto=start
     ikev2=insist
```

Typical Remote Access VPN

End device to site network access point encrypted – LAN still unencrypted



Remote Access VPN Client

You can download a certificate and this conf file at netdev.nohats.ca

```
conn netdev.nohats.ca
     left=%defaultroute
     leftcert=pwouters.nohats.ca
     leftid=%fromcert
     leftrsasigkey=%cert
     leftmodecfgclient=yes
     leftsubnet=0.0.0.0/0
     right=netdev.nohats.ca
     rightid=@netdev.nohats.ca
     rightrsasigkey=%cert
     rightsubnet=0.0.0.0/0
     narrowing=yes
     ikev2=insist
     mobike=yes
     auto=add
```

Remote Access VPN Client

Start the VPN and test it by browsing http://ip.libreswan.org/

```
root@thinkpad:/etc/ipsec.d# ipsec auto --up netdev.nohats.ca
002 "netdev.nohats.ca"[1] 193.110.157.152 #5: initiating v2 parent SA
133 "netdev.nohats.ca"[1] 193.110.157.152 #5: STATE_PARENT_I1: sent v2I1, expected v2R1
134 "netdev.nohats.ca"[1] 193.110.157.152 #6: STATE_PARENT_I2: sent v2I2, expected v2R2 {auth=IKEv2
cipher=aes_gcm_16_256 integ=n/a prf=sha2_512 group=DH19}
002 "netdev.nohats.ca"[1] 193.110.157.152 #6: certificate verified OK:
E=info@nohats.ca,CN=netdev.nohats.ca,OU=Clients,O=NetDev,L=Toronto,ST=Ontario,C=CA
002 "netdev.nohats.ca"[1] 193.110.157.152 #6: IKEv2 mode peer ID is ID_FQDN: '@netdev.nohats.ca'
003 "netdev.nohats.ca"[1] 193.110.157.152 #6: Authenticated using RSA
002 "netdev.nohats.ca"[1] 193.110.157.152 #6: received INTERNAL_IP4_ADDRESS 100.64.13.3
002 "netdev.nohats.ca"[1] 193.110.157.152 #6: received INTERNAL_IP4_DNS 193.110.157.123
005 "netdev.nohats.ca"[1] 193.110.157.152 #6: Received INTERNAL_DNS_DOMAIN: nohats.ca
002 "netdev.nohats.ca"[1] 193.110.157.152 #6: up-client output: updating resolvconf
002 "netdev.nohats.ca"[1] 193.110.157.152 #6: negotiated connection
    [100.64.13.3-100.64.13.3:0-65535 0] -> [0.0.0.0-255.255.255.255:0-65535 0]
004 "netdev.nohats.ca"[1] 193.110.157.152 #6: STATE_V2_IPSEC_I: IPsec SA established tunnel mode
{ESP/NAT=>0x21d767b3 <0xf37c0db7 xfrm=AES_GCM_16_256-NONE NATOA=none NATD=193.110.157.152:4500
DPD=passive}
```

Remote Access VPN Server

You can download the server config file at netdev.nohats.ca

```
conn netdev.nohats.ca
        left=193.110.157.152
        leftcert=netdev.nohats.ca
        leftsendcert=always
        leftid=@netdev.nohats.ca
        leftsubnet=0.0.0.0/0
        rightaddresspool=100.64.13.2-100.64.13.254
        right=%any
        rightid=%fromcert
        rightca=%same
        modecfgdns=100.64.13.1, 8.8.8.8
        modecfgdomains="nohats.ca, secret.redhat.com"
        modecfgpull=yes
        mobike=yes
        authby=rsasig
        ikev2=insist
        auto=add
        rekey=no
```

Opportunistic IPsec Deployment

mesh network encryption (Enterprise network, but also Internet wide)

- Enterprise / cloud wide
  - host to host encryption without hub-spoke
  - Adding a node should not require reconfiguration of other nodes
  - X.509 certificate or DNS(SEC) based PKI
  - Packet or DNS triggered
- Internet wide
  - X.509 certificate (LetsEncrypt) or DNS(SEC) based PKI
  - Initiator remains anonymous (Like SSL/TLS)
  - Support for initiator behind NAT
  - Packet or DNS triggered

Libreswan – Group Policies

Group files in /etc/ipsec.d/policies/* list network CIDRs to match

```
/etc/ipsec.d/policies/block              Drop all packets
/etc/ipsec.d/policies/clear              Only allow cleartext
/etc/ipsec.d/policies/clear-or-private   Default clear, allow crypto
/etc/ipsec.d/policies/private            Mandate crypto, hard fail
/etc/ipsec.d/policies/private-or-clear   Attempt crypto, allow clear
```

```
# cat /etc/ipsec.d/policies/private-or-clear
193.110.157.0/24
193.111.228.0/24
# cat /etc/ipsec.d/policies/private
10.0.0.0/8
192.168.0.0/16
```

Enterprise Cloud Mesh Encryption

Configuration for mandated mutual certificate based authentication

```
For example add 10.0.0.0/8 to /etc/ipsec.d/policies/private

# install localcertificate: ipsec import node1.example.com.p12
# /etc/ipsec.d/YourCloud.conf

conn private
        left=%defaultroute
        leftid=%fromcert
        # our certificate
        leftcert=node1.example.com
        right=%opportunisticgroup
        rightid=%fromcert
        # their certificate transmitted via IKE
        rightca=%same
        ikev2=insist
        authby=rsasig
        failureshunt=drop
        negotiationshunt=hold
        auto=ondemand
```

Optional Opportunistic IPsec

Configuration for optional anonymous IPsec

```
For example add 0.0.0.0/0 to /etc/ipsec.d/policies/private-or-clear

conn private-or-clear
        left=%defaultroute
        leftid=%null
        rightid=%null
        right=%opportunisticgroup
        authby=null
        ikev2=insist
        failureshunt=passthrough
        negotiationshunt=passthrough
        # to not leak during IKE negotiation, use
        # negotiationshunt=hold
        auto=ondemand
        # clear-or-private uses auto=add
```

Opportunistic IPsec using public keys from DNSSEC

with anonymous clients, and support for NAT (so basically TLS but for all IP traffic)

```
# Install the unbound DNS server and libreswan 3.25 or newer
# Verify the ipsec module options in the installed unbound.conf with the version
#    located at oe.libreswan.org/unbound.conf
# Install the libreswan configuration for Opportunistic IPsec in /etc/ipsec.d
wget oe.libreswan.org/oe-dnssec.conf -O /etc/ipsec.d
# Ensure the unbound daemon can send commands to libreswan
ipsec restart
chmod 777 /run/pluto/pluto.ctl
# debian users will need to get a fixed _unbound-hook.py from oe.libreswan.org
systemctl start unbound
# edit /etc/resolv.conf and only list 127.0.0.1 as name server
# Now let's trigger Opportunistic IPsec
ping oe.libreswan.org
# confirm it worked
ipsec whack –trafficstatus
# You can also browser to oe.libreswan.org, it will tell you if IPsec protection is active for you
```

draft-antony-ipsecme-oppo-nat

Eliminating the IP address conflicts caused by NAT

```
193.110.15.131  Remote Opportunistic IPsec server
192.168.2.45      Opportunistic Client pre-NAT IP address
100.64.0.2      IP address from IPsec server address pool
# ip xfrm pol
src 100.64.0.2/32 dst 193.110.157.131/32
          dir out priority 2080 ptype main
          tmpl src 192.1.2.45 dst 193.110.157.131
                       proto esp reqid 16389 mode tunnel
src 193.110.157.131/32 dst 100.64.0.2/32
          dir fwd priority 2080 ptype main
          tmpl src 193.110.157.131 dst 192.1.2.45
                       proto esp reqid 16389 mode tunnel
src 193.110.157.131/32 dst 100.64.0.2/32
          dir in priority 2080 ptype main
          tmpl src 193.110.157.131 dst 192.1.2.45
                       proto esp reqid 16389 mode tunnel
src 192.168.2.45/32 dst 193.110.157.131/32
          dir out priority 2080 ptype main
          tmpl src 192.1.2.45 dst 193.110.157.131
                       proto esp reqid 16389 mode tunnel
```

draft-antony-ipsecme-oppo-nat

Eliminating the IP address conflicts caused by NAT

```
193.110.15.131  Remote Opportunistic IPsec server
192.168.2.45    Opportunistic Client pre-NAT IP address
100.64.0.1      Client IP address assigned y Opportunistic Ipsec server

# iptables -t nat -L -n

Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       all  --  193.110.157.131      100.64.0.1 \
                      policy match dir in pol ipsec to:192.168.2.45

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
SNAT       all  --  0.0.0.0/0            193.110.157.131 \
                      policy match dir out pol ipsec to:100.64.0.1


Basically: NAT within the IPsec subsystem
```

Bonus slides (aka, we probably don't have more time)

LetsEncrypt Certificates

Preparing libreswan to use LetsEncrypt certificates

```
mkdir letsencrypt ; cd letsencrypt
wget https://letsencrypt.org/certs/lets-encrypt-x4-cross-signed.pem
wget https://letsencrypt.org/certs/lets-encrypt-x3-cross-signed.pem
wget https://letsencrypt.org/certs/isrgrootx1.pem
# https://www.identrust.com/certificates/trustid/root-download-x3.html
wget https://nohats.ca/LE/identrust-x3.pem
yum install libreswan
ipsec initnss
certutil -A -i lets-encrypt-x3-cross-signed.pem -n lets-encrypt-x3 \ -
   -t CT,, -d sql:/etc/ipsec.d
certutil -A -i lets-encrypt-x4-cross-signed.pem -n lets-encrypt-x4 \
   -t CT,, -d sql:/etc/ipsec.d
certutil -A -i isrgrootx1.pem -n isrgrootx1 -t CT,, -d \
   sql:/etc/ipsec.d
certutil -A -i identrust-x3.pem -n identrust-x3 -t CT,, -d \
   sql:/etc/ipsec.d
```

Letsencrypt Client for IPsec

Anonymous client to authenticated server

```
cd /etc/ipsec.d
wget https://nohats.ca/LE/oe-letsencrypt-client.conf
echo "193.110.157.131/32" >> /etc/ipsec.d/policies/private-or-clear
(if adventurous, echo ”0.0.0.0/0" to private-or-clear)

ping letsencrypt.libreswan.org
PING letsencrypt.libreswan.org (193.110.157.131) 56(84) bytes of data.
64 bytes from letsencrypt.libreswan.org (193.110.157.131): icmp_seq=2 ttl=64 time=96.2 ms
64 bytes from letsencrypt.libreswan.org (193.110.157.131): icmp_seq=3 ttl=64 time=96
.7 ms

ipsec whack --trafficstatus
006 #2: "private-or-clear#193.110.157.131/32"[1] 100.64.0.2/32=== ...193.110.157.131, type=ESP,
add_time=1471926595, inBytes=252, outBytes=252, id='CN=letsencrypt.libreswan.org',
lease=100.64.0.2/32
```

LetsEncrypt Server for IPsec

/etc/ipsec.d/oe-letsencrypt-client.conf

```
conn private-or-clear
        left=%defaultroute
        leftid=%null
        leftauth=null
        leftmodecfgclient=yes
        leftcat=yes
        #
        rightid=%fromcert
        rightrsasigkey=%cert
        rightauth=rsasig
        right=%opportunisticgroup
        rightmodecfgclient=yes
        #
        narrowing=yes
        negotiationshunt=hold
        failureshunt=passthrough
        ikev2=insist
        auto=ondemand
```

LetsEncrypt Server for IPsec

Authenticated server for anonymous clients

```
# Install LetsEncrypt CA certs as documented on earlier slide

yum install letsencrypt
letsencrypt certonly -d yourserver.example.com

cd /etc/letsencrypt/live/yourserver.example.com
openssl pkcs12 -export -in cert.pem -inkey privkey.pem \
    -out yourserver.example.com.p12 -name yourserver.example.com \
    -CAfile chain.pem -certfile chain.pem -caname lets-encrypt-x3
ipsec import letsencrypt.libreswan.org.p12
cd /etc/ipsec.d
wget https://nohats.ca/LE/oe-letsencrypt-server.conf
echo "0.0.0.0/0" >> /etc/ipsec.d/policies/clear-or-private
ipsec restart
(for opportunistic server plus client, add 0.0.0.0 to private-or-clear)
```

LetsEncrypt Server for IPsec

/etc/ipsec.d/oe-letsencrypt-server.conf

```
conn clear-or-private
        leftid=%fromcert
        leftrsasigkey=%cert
        # your LetsEncrypt certificate
        leftcert=yourserver.example.com
        leftauth=rsasig
        left=%defaultroute
        leftaddresspool=100.64.0.1-100.64.255.254
        leftmodecfgclient=yes
        rightid=%null
        rightauth=null
        right=%opportunisticgroup
        negotiationshunt=passthrough
        failureshunt=passthrough
        ikev2=insist
        sendca=issuer
        auto=add
```

Features Planned

- Native kernel support for IPsec-NAT
- New libreswan library for
    - DBUS API for add, remove, status
    - Fake getsockopt() similar to tcpcrypt ?
    - Fake setsockopt() to require authenticated encryption
    - Get new socket options into kernel :-)

Appendix

History:

The FreeS/WAN Project (1996-2003)

“My project for 1996 was to secure 5% of the Internet traffic against passive wiretapping”

– John Gilmore

- Opportunistic Encryption
- Enable encryption between any two nodes without pre-configuration

History:

The FreeS/WAN Project (1996-2003)

- S/WAN stands for “Secure Wide Area Network” (trademarked by RSA Inc.)
  - The term “Virtual Private Network” (VPN) became popular instead
- Predates OpenSSL
  - Used some SSLeay code (with special permission of Eric Andrew Young)
- Predates the Internet key Exchange (“IKEv1” – RFC 2409) published in 1998
- Predates CryptoAPI (kerneli.org started around 2002)
- Predates United States export laws for cryptography
  - 1995 – 1999: Bernstien v. United States on “crypto is free speech”
  - 1996: Allow export of 56-bit crypto (RC4) with key recovery backdoor
  - 1999: Allow export of 56-bit crypto (DES, RC4) without backdoor, and 1024-bit RSA
- Predates DNSSEC and most of the CA industry

History:

The FreeS/WAN Project (1996-2003)

Where FreeS/WAN succeeded

- Supported Linux 2.0 and up
- Became the gold standard of IKE and IPsec
- Strong deployment in the Enterprise

History:

The FreeS/WAN Project (1996-2003)

Where FreeS/WAN failed

- IKEv1 protocol specification took 3-years
- Developers could not be United States citizens (Oh Canada…)
- Packet triggered events relied on IP address
  - And no real access to the reverse DNS in-addr.arpa
- The Internet became reliant on NAT
  - And no universal deployment of IPv6 to obsolete it
- Required mutual authentication is problematic (SSL got it right)
- Unauthenticated encryption rejected as unsafe, confusing for enduser
- Secure DNS needed for key distribution took 15-years
  - Root zone finally signed in 2010
  - FreeS/WAN kicked out DNSSEC KEY/SIG records, back to TXT
- 2001: Echelon spying network exposed – no one cared

History:

The Openswan Project (2003-2011)

Ex-employees and volunteers forked FreeS/WAN

- John Gilmore gives up on OE
- Americans can submit code

- Enterprise deployments just work
- Support native IPsec (XFRM/NETKEY)
- Use optional NSS crypto library
- Hardware acceleration support (OCF and native)
- Initial rough IKEv2 implementation

- Opportunistic Encryption mothballed
- 2012: A lawsuit requires that the project renames itself

History:

The Libreswan Project (2011-ongoing)

The Great Overhaul

- Only use NSS crypto library
- IKEv2
- Crypto suites update
- Modern network timers
- Event loops
- Cleanup codebase to support FIPS, CAVP, Common Criteria
- Cloud support

- Revisit Opportunistic Encryption

History:

The strongSwan Project (2003-present)

Andreas Steffan forked FreeS/WAN and added his X.509 code

- Strong focus on the Enterprise world
- No Opportunistic IPsec support
- Early adopter of IKEv2
- Contains no more old freeswan/openswan code

- libreswan uses strongswan to perform interoperability testing which requires two independently written implementations
    - http://testing.libreswan.org/

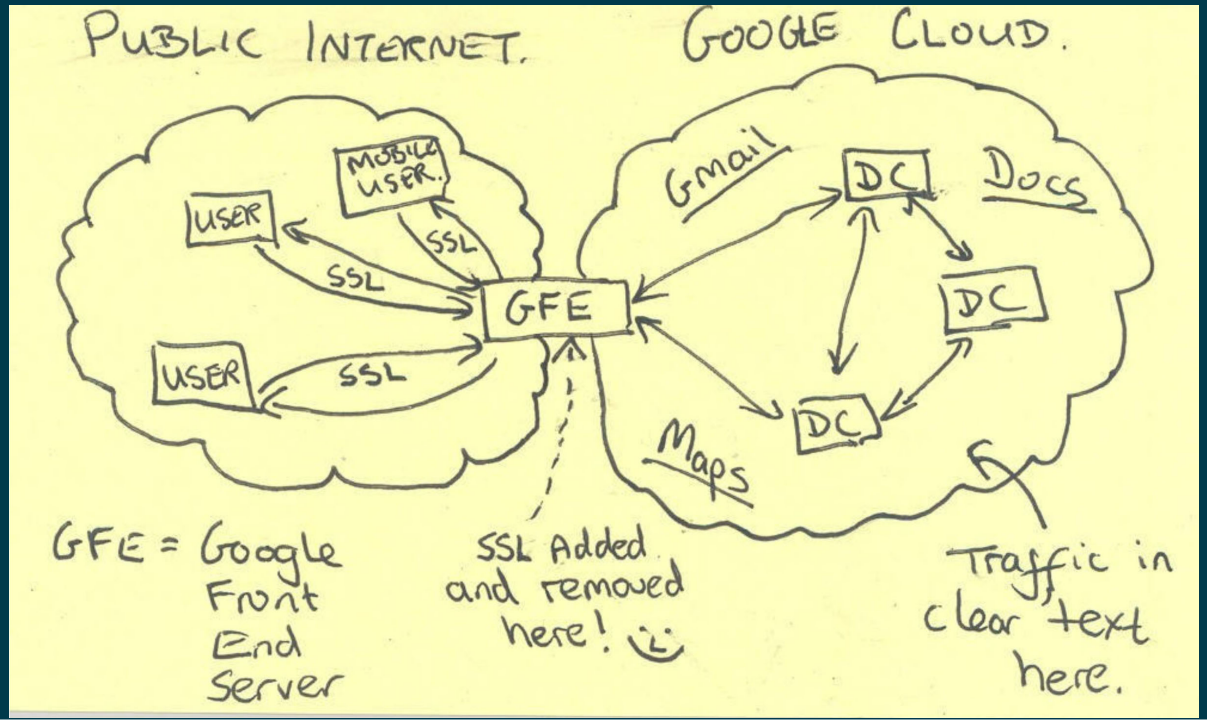
History:
Revisiting Opportunistic Encryption

- IKEv2 allows asymmetric AUTH like SSL/TLS
- IKEv2 allows assigning IP addresses natively
- Linux conntrack vastly improved
- Linux XFRM/NETKEY vastly improved (TCP packet caching)
- DNSSEC expected to go on the end node
- Unbound DNS server with DNSSEC-trigger
- Allows DNS based triggers for Opportunistic Encryption

- DNSSEC triggers to replace the reverse DNS in-addr.arpa for ID AUTH
- Linux conntrack and IKEv2 addresspool to resolve NAT problem

- If people only realized they want ambiguous encryption.....

History:
Edward Snowden (2013)



History:

The crypto rush

- Encryption is more important now
- Enterprises must encrypt:
    - Cloud instances
    - Data centres
    - MPLS, fibre
    - Transit cables
- Puppet / ansible does not scale for mesh encryption
- IPsec mesh encryption needed configuration modification on all nodes!

- Opportunistic encryption is cloud encryption
- Opportunistic encryption is internet encryption

History:
IETF Response

Internet Engineering Task Force steps up encryption

- RFC 7258 “Pervasive Monitoring Is an Attack” (May 2014)

  “Pervasive monitoring is a technical attack that should be mitigated in the design of IETF protocols, where possible.”

- RFC 7435 “Opportunistic Security” (Dec 2014)

  “Protocol designs based on Opportunistic Security use encryption even when authentication is not available, and use authentication when possible, thereby removing barriers to the widespread use of encryption on the Internet.”

History:
Opportunistic IPsec Work at IETF

- RFC 7619 “NULL Authentication for IKEv2” (Aug 2015)
  - IKEv2 (2008) already allowed asymmetrical authentication
  - Allow Anonymous client to Authenticated Server
  - Allow Anonymous to Anonymous
- draft-antony-ipsecme-oppo-nat (Mar 2015)
  - NAT-Traversal support for Opportunistic IPsec