THE LIBRESWAN PROJECT

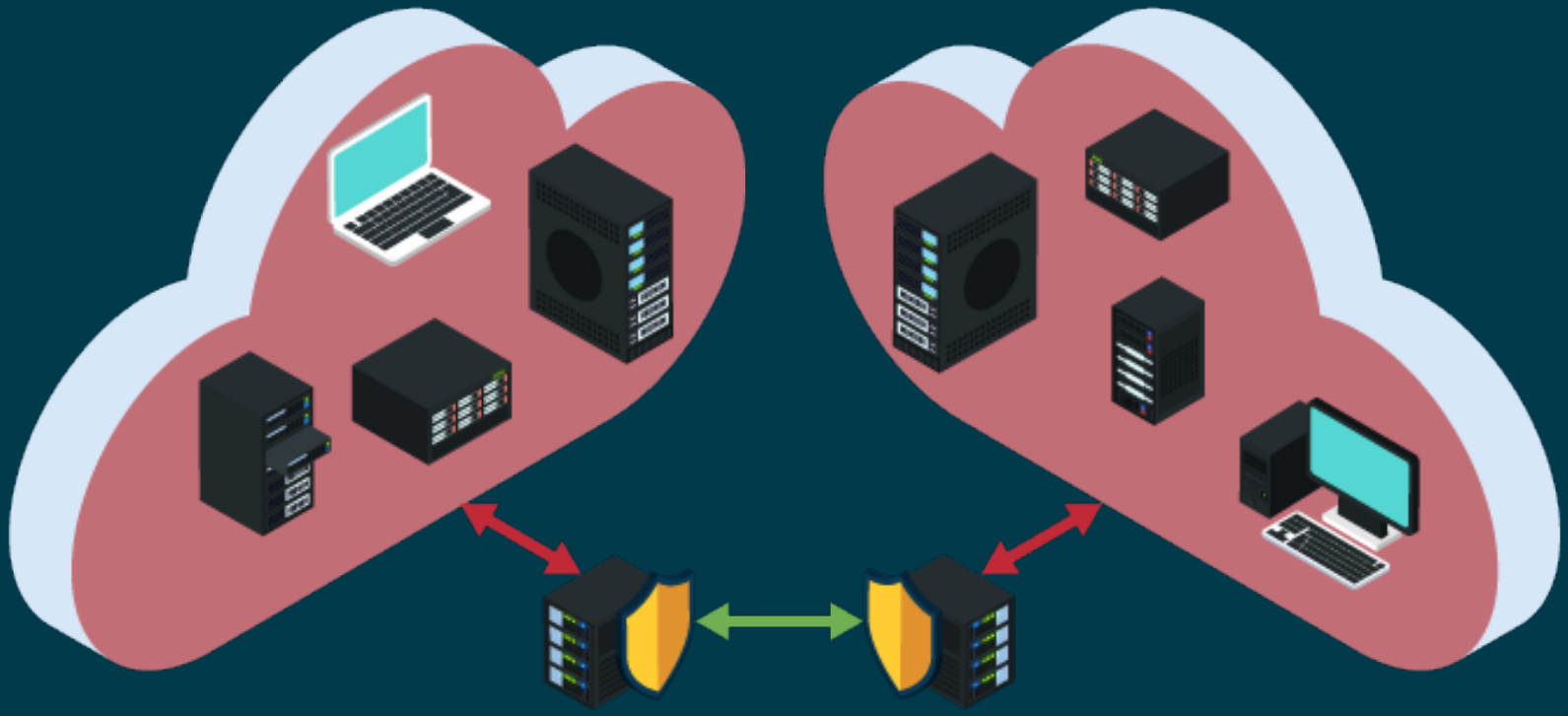
An Internet Key Exchange (“IKE”) daemon for IPsec

- Enterprise IPsec based VPN solution
- Make encryption the default mode of communication

- Certifications (FIPS, Common Criteria, USGv6, etc.)
- Contributing to IETF Standards for IKE and IPsec

TYPICAL SITE TO SITE VPN

Individual networks are unencryped, only the interconnect is encrypted



TYPICAL REMOTE ACCESS VPN

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

“OPPORTUNISTIC ENCRYPTION”

- “Try to setup IPsec to everyone”
- It failed to be deployed widely:
  - Packet trigger based needs to map to some kind of identity
  - IKE/IPsec had only mutual authentication, mobile users could not easily get an identity and publish it.
  - Used reverse DNS zone (in-addr.arpa) which no one controlled
  - DNSSEC deployment needed for secure use of DNS
  - NATs breaks everything
  - Users didn't care too much (until Snowden)

“OPPORTUNISTIC IPSEC”

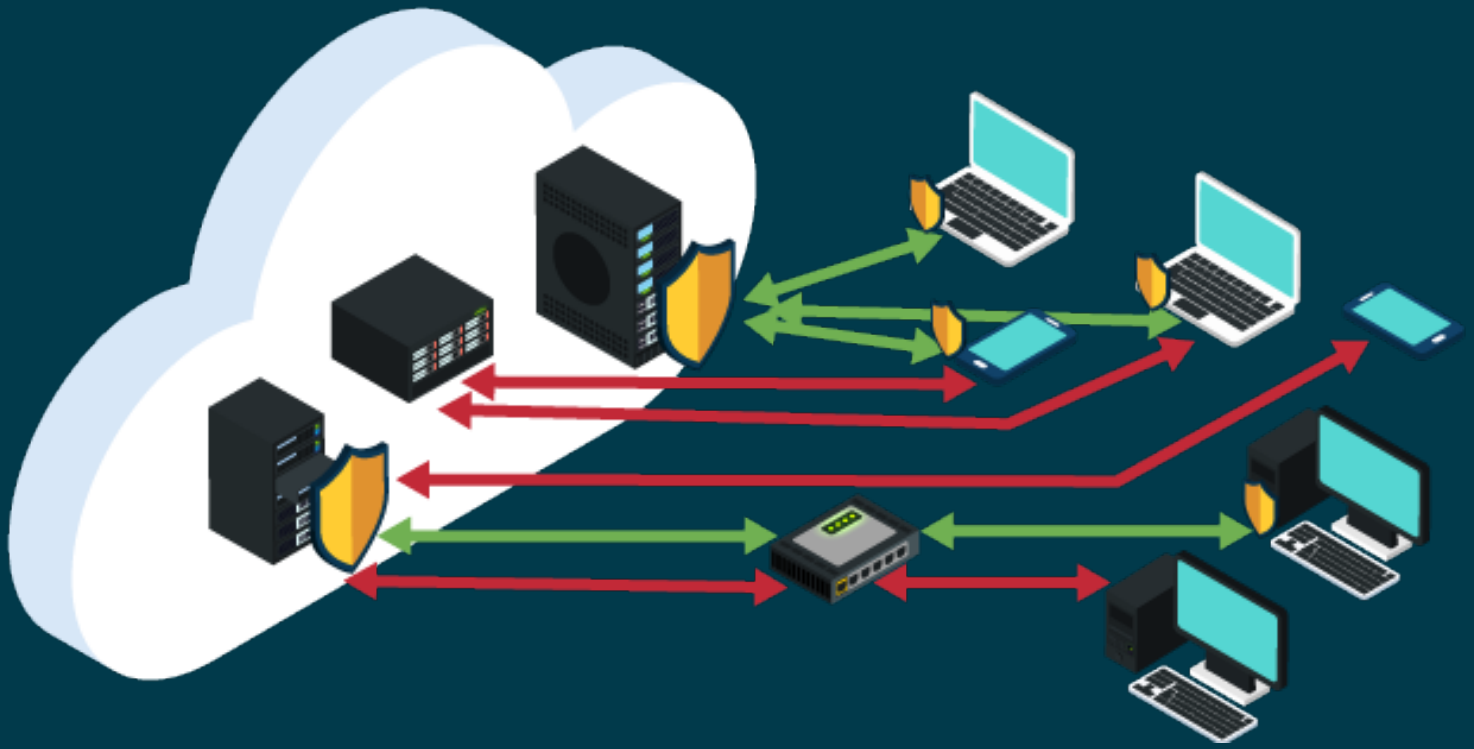
- Term used to mean “any packet trigger based IPsc”
  - enterprise mesh encryption
  - Internet wide

NULL AUTHENTICATION FOR IKEV2 (2015)

- IKEv2 (2005) already allowed asymmetrical authentication
- We needed Anonymous client to Authenticated Server
- We wanted Anonymous to Anonymous (passive attack protection)
- Makes IPsec work like TLS

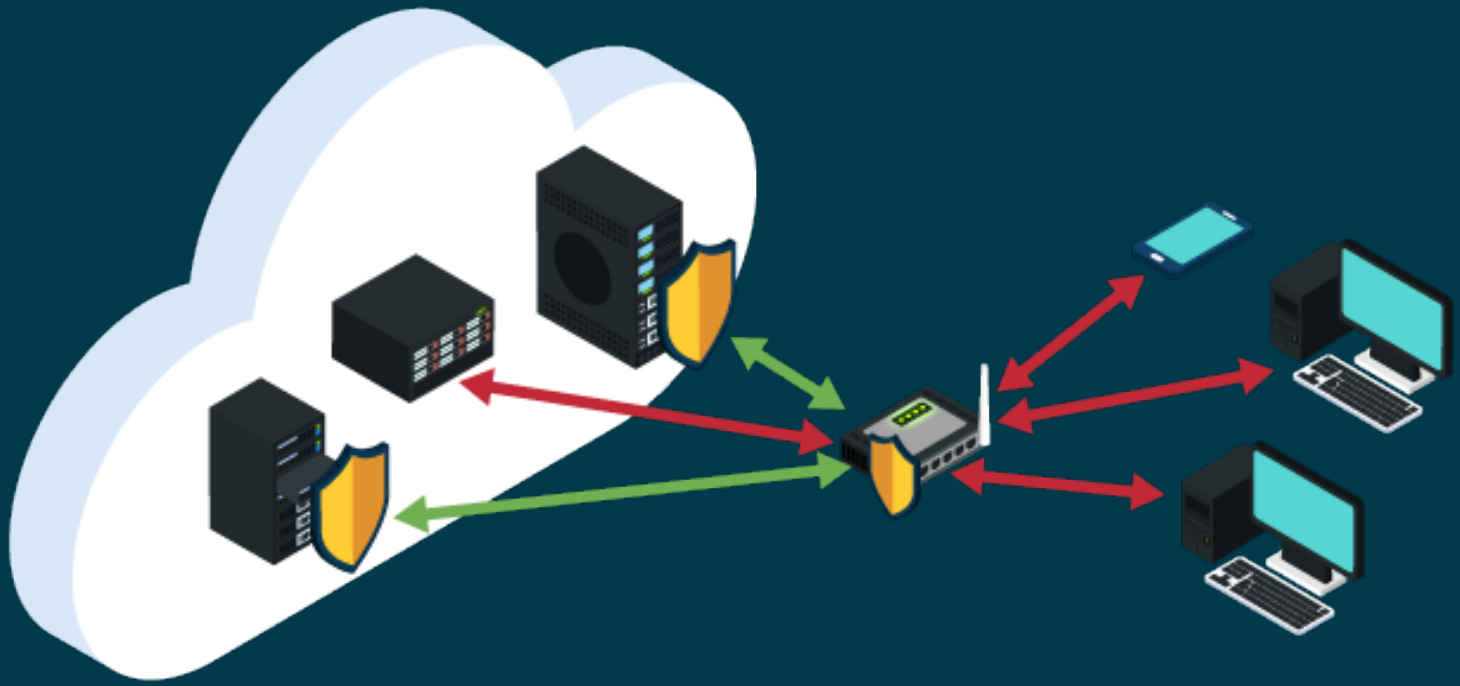
OPPORTUNISTIC IPSEC DEPLOYMENT

End-to-end encryption using IPsec



OPPORTUNISTIC IPSEC GATEWAY

Use a Linux gateway to protect devices not able to run opportunistic



LIBRESWAN – GROUP POLICIES

Group files in /etc/ipsec.d/policies/*.conf 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
        leftauth=null
        leftid=%null
        rightauth=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
```

UNBOUND DNS IPSEC MODULE

Use DNS based public keys for IPsec authentication

1. Unbound DNS server IPsec module
   - When looking up A/AAAA records, also lookup IPSECKEY records
   - If no IPSECKEY records:
     - return A/AAAA answers
   - If IPSECKEY record found:
     - give DNS QNAME, IPSECKEY, TTL, A/AAAA records to IKE
     - libreswan initiates IKE and establishes IPSEC tunnel
       - Server authenticated against IPSECKEY record
       - Client uses AUTH-NULL and remains anonymous
       - On failure, returns error, causes DNS ServFail error
     - return A/AAAA answers to application (and cache)

UNBOUND CONFIGURATION

/etc/unbound/unbound.conf

```
server:
        # [...]

        module-config: "ipsecmod validator iterator"

        # libreswan enables this on demand via unbound-control
        ipsecmod-enabled: no

        ipsecmod-hook:/usr/libexec/ipsec/_unbound-hook

        # When enabled unbound will reply with SERVFAIL if the return value of
        # the ipsecmod-hook is not 0.
        # ipsecmod-strict: no
        #
        # Maximum time to live (TTL) for cached A/AAAA records with IPSECKEY.
        # ipsecmod-max-ttl: 3600
        #
        # Reply with A/AAAA even if the relevant IPSECKEY is bogus. Mainly used for
        # testing.
        # ipsecmod-ignore-bogus: no
        #
        # Domains for which ipsecmod will be triggered. If not defined (default)
        # all domains are treated as being whitelisted.
        # ipsecmod-whitelist: "libreswan.org"
        # ipsecmod-whitelist: "nlnetlabs.nl"
```

A “NAT” LAYER INSIDE IPSEC

Obtained IP address (for tunnel mode) only lives inside IPsec

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

A “NAT” LAYER INSIDE IPSEC

use iptables to NAT to the IP address assigned via IKE

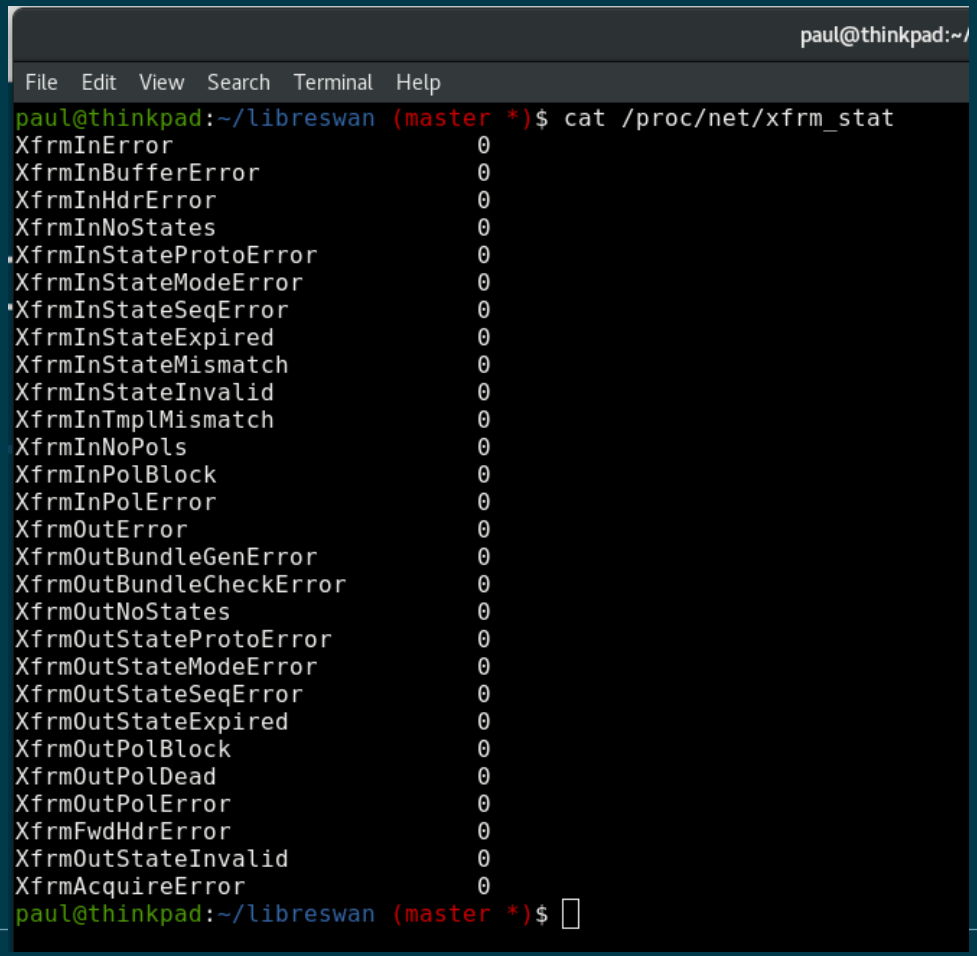
```
193.110.15.131  Remote Opportunistic IPsec server
192.168.2.45      Opportunistic Client pre-NAT IP address
100.64.0.1      IP addres from IPsec server addresspool

# 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
```

IPSEC ISSUES FOR HUMAN BEING

1. XFRM without interfaces is too hard for firewall admins to configure rules
2. XFRM + tcpdump = madness
3. NAT + IPsec = foot bullet
4. IPsec MTU issues / workaround is hard (TCPMSS, clamping)
5. XFRM for hub-spoke tunnel kills lan traffic (10.0.0.0/8 ↔ 10.0.0.0/24)
6. XFRM + DSL/LAN (one interface) + rp_filter = martians
7. IPsec SA flags are undocumented: noecn, decap-dscp, nopmtudisc, esn **wildrecv, icmp, af-unspec, align4**
8. ip xfrm monitor throws error for XFRM_MIGRATE messages
9. Using /proc values is dangerous / undefined / unknown
   - */proc/sys/net/core/xfrm_acq_expires (linked to get_newspi())*
   - */proc/sys/net/core/xfrm_larval_drop (linked to packet caching)*
   - */proc/*sys/net/core/xfrm_aevent_etime / aevent_rseqth (?)

IPSEC ISSUES FOR HUMAN BEING

Errors in */proc/net/xfrm_stat*



redhat. | libreswan

WISH LIST FROM YOUR IKE DEV CUSTOMERS

- ESPoverTCP support RFC 8229
- ESPoverTLS support RFC 8229
- Linux not compliant with IKEv2/ESP NAT requirements:

  ”If Network Address Translation Traversal (NAT-T) is supported (that is, if NAT_DETECTION_*_IP payloads were exchanged during IKE_SA_INIT), **all devices MUST be able to receive and process both UDP-encapsulated ESP and non-UDP-encapsulated ESP packets at any time**. Either side can decide whether or not to use UDP encapsulation for ESP irrespective of the choice made by the other side. However, if a NAT is detected, both devices MUST use UDP encapsulation for ESP.”

WISH LIST FROM YOUR IKE DEV CUSTOMERS

- socket options to set/get IPsec policy name
- named sockets in general
- socket options to close/error on socket mandating IPsec
- INVALID_SPI ACQUIRES (fast crash recovery)
- Don't “Destination unreachable” when there is no default route or there is a unreachable route. For example two machines without default route:

  10.0.0.0/8 – 1.2.3.4/24 -- 1.2.3.5/24 – 192.168.0.0/24

WISH LIST FROM YOUR IKE DEV CUSTOMERS

- Populate From Packet (PFP) support
  - Send new ACQUIRE's for the same policy with different protoports
- Diet ESP
  - https://tools.ietf.org/html/draft-mglt-ipsecme-diet-esp-05
- Implicit IV
  - https://datatracker.ietf.org/doc/draft-ietf-ipsecme-implicit-iv
- ESP PMTU
  - https://datatracker.ietf.org/meeting/101/materials/slides-101-ipsecme-packetization-layer-path-mtu-discovery-01

WISH LIST FROM YOUR IKE DEV CUSTOMERS (FIPS VERSION)

- Censoring private key material from ip xfrm state
  - Only show key after setting some /proc or /sys option
  - Don't allow this option in FIPS mode
- Who should enforce byte limits (eg 3des to 2^16)
  - If kernel forces it, userland admin cannot make mistake :)

WISH LIST FROM YOUR IKE DEV CUSTOMERS

- instance/container can't load kernel modules
- PFKEY for ESP crypto discovery **lies** to us – we manually fix up data
- Is there a “modern” (non-PFKEY) API to ask for possible ESP/AH xforms ?
- We don't want to be the managers of kernel modules
- We would like all of this to autoload like other kernel modules

WE DON'T WANT TO DO THIS STUFF

```
    # load the most common ciphers/algo's
    # padlock must load before aes module - though does not exist on newer
    # kernels
    # padlock-aes must load before padlock-sha for some reason
    ${MODPROBE} padlock 2>/dev/null
    ${MODPROBE} padlock-aes 2>/dev/null
    ${MODPROBE} padlock-sha 2>/dev/null
    # load the most common ciphers/algo's
    # aes-x86_64 has higher priority in via crypto api
    # kernel directory does not match uname -m on x86_64 :(
    modules=$(ls /lib/modules/$(uname -r)/kernel/arch/*/crypto/* 2>/dev/null)
    modules="aesni-intel aes-x86_64 geode-aes aes aes_generic des sha512 \
        sha256 md5 cbc xcbc ecb twofish blowfish serpent ccm gcm ctr cts \
        deflate cast5 cast6 lzo sha256_generic sha512_generic camellia \
        cmac chacha20poly1305 ${modules}"
    for module in ${modules}
    do
        module=$(basename ${module} | sed "s/\.ko$//")
        # echo -n "${module} " >&2
        ${MODPROBE} ${module} 2>/dev/null
    done
}
```

```
if [ -f /proc/modules ]; then
    # load all NETKEY modules
    for mod in ipcomp6 xfrm_ipcomp ipcomp xfrm6_tunnel xfrm6_mode_tunnel \
        xfrm6_mode_beet xfrm6_mode_ro xfrm6_mode_transport \
        xfrm4_mode_transport xfrm4_mode_tunnel xfrm4_tunnel \
        xfrm4_mode_beet esp4 esp6 ah4 ah6 af_key ip_vti
    do
        # echo -n "${mod} " >&2
        ${MODPROBE} ${mod} 2>/dev/null
    done

    # xfrm_user is the old name for xfrm4_tunnel - backwards compatibility
    ${MODPROBE} xfrm_user 2>/dev/null
```

IPSEC ISSUES FOR IKE DEVELOPERS

1. Can we delete the larval acquire state ? Should we ?

2. xfrm.h is needed by userland. Sometimes a newer copy than available on the system (eg for XFRM_OFFLOAD_*)

3. USE_XFRM_HEADER_COPY=false|true

   xfrm.h drags in various kernel-only structs, conflicting based on include file ordering of netinet/in.h and linux/in6.h

   - USE_GLIBC_KERN_FLIP_HEADERS=false|true