IPsec
IPsec

- “Layer 3” security association
- Idea: don’t change applications, just OS
- vs SSL/SSH: don’t change OS, only change application. They run on top of layer 4 (TCP/UDP)
Layer 3 vs layer 4

- layer 3 technically superior
- Rogue packet problem
  - TCP doesn’t participate in crypto, so attacker can inject bogus packet, no way for TCP to recover
- easier to do outboard hardware processing (since with IPsec, each
But if don’t change apps

• API just says which IP address talking to
• So all the fancy PKI, names, etc… No authentication of anything but IP address
• What you get (similar to firewalls between)
  – hides traffic from eavesdroppers
  – can do packet filtering like firewalls do
  – if authentication based on IP address, then OK
IP Security (IPSEC)

- Encryption and integrity protection for IP packets (layer 3)
- Pieces include:
  - AH and ESP
  - ISAKMP/IKE (create SA: mutual authentication, session key establishment)
  - Endless documents about specific encryption formats, etc
AH / ESP

- extra header between layers 3 and 4 (IP and TCP) to give dest enough info to identify “security association”
- AH does integrity only - includes source and destination IP addresses, ...
- ESP does encryption and integrity protection
# ESP

## Encapsulating Security Payload

<table>
<thead>
<tr>
<th>IP Header</th>
<th>Next Header = ‘50’ (ESP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP Header</td>
<td></td>
</tr>
<tr>
<td>Encrypted Payload</td>
<td>TCP = 6, UDP = 17, ESP = 50, IP = 4</td>
</tr>
<tr>
<td>Encrypted Padding</td>
<td></td>
</tr>
<tr>
<td>Pad Len</td>
<td>NXT</td>
</tr>
<tr>
<td>MIC</td>
<td>Over ESP Header, Encrypted Payload/Pad/Padlen/NXT</td>
</tr>
<tr>
<td>Session ID</td>
<td></td>
</tr>
<tr>
<td>Sequence #</td>
<td>IV (size alg-depndnt)</td>
</tr>
</tbody>
</table>
AH

• Looks kind of like IPv6 extension header
• IPv6 has length in units of 8-octet chunks
• But AH length in units of 4-octet chunks
• Because of AH, IP spec specifies, for each field, “mutable”, “immutable”, or “immutable but predictable”, and flag options
• Note MIC before data
Why AH?

• AH and ESP designed by different groups. AH designers were IPv6 supporters
• AH looks more like IPv6
• AH also protects “immutable” fields in IP header.
• Originally, ESP just encryption
• Encryption without integrity has flaws
Why AH, con’t

• Then integrity protection added to ESP.
• Excuses for keeping AH
  – protects IP header (nobody has a credible security reason why, and ESP-tunnel can too.
  – Makes NAT harder, which pleases IPv6 fans
  – with AH, firewalls and routers that want to look at layer 4 info (like ports) know it’s not encrypted. With ESP, can’t tell from packet
Why Not AH?

- IPSEC way too complex.
- Layer 4 info should be hidden from routers, firewalls, and can’t be integrity protected en route anyway
- You could peek inside ESP and almost always tell if it’s encrypted or not. A flag might be nice (reserved SPIs would work)
IPSEC Key Exchange Contenders

- Photuris: Signed Diffie Hellman, stateless cookies, optional hiding endpoint ids
- SKIP: Diffie-Hellman public keys, so if you know someone’s public key $g^B$, you automatically know a shared secret $g^{AB}$. Each msg starts with per-msg key S encrypted with $g^{AB}$
- And the winner was...
ISAKMP

• Internet Security Association and Key Management Protocol
• Gift to the IETF from NSA
• A “framework”, not a protocol. Complex encodings. Flexible yet constraining.
• Two “phases”. Phase 1 expensive, establishes a session key with which to negotiate multiple phase 2 sessions
Why Two Phases

• We argued for removing this, but people wanted it for:
  – firewalls creating lots of VPNs for lots of customers…they feel safer if different SAs
  – different QOS, since might travel at different speeds, sequence numbers get far apart
  – makes rekeying faster
  – different SAs with different security properties
IKE

• IKE authors tried to fit academic papers (SKEME, OAKLEY) into ISAKMP
• Mostly a rewriting of ISAKMP, but not self-contained. Uses ISAKMP
• Since both so badly written, hasn’t gotten thorough review
• Really 3+ specs (ISAKMP, IKE, DOI)
• Plus a few more (NAT traversal, etc.)
Imagine 150 pages of this!

- While Oakley defines “modes”, ISAKMP defines “phases”. The relationship between the two is very straightforward and IKE presents different exchanges as modes which operate in one of two phases. —RFC 2409
IKE

• Two phases, like ISAKMP
• Phase 1 has two “modes”: aggressive, and main
• Main has 6 messages, aggressive has 3
• Main does more, sometimes, like hiding endpoint identifiers
• Phase 2 known as “quick mode”
IKE Key types

• For each of main and aggressive, protocols are defined for each of the following key types:
  – pre-shared secret key
  – public signature keys
  – public encryption keys (old crufty way)
  – public encryption keys (new improved method)
General Idea of Aggressive-Mode

Alice  

I’m Alice, $g^A \mod p$, nonce$_A$

Bob  

I’m Bob, $g^B \mod p$, proof I’m Bob, nonce$_B$

proof I’m Alice
General Idea of Main-Mode

Alice:
- Crypto suites I support
- Crypto suites I choose
- $g^A \mod p$, nonce$_A$
- $g^B \mod p$, nonce$_B$

Bob:
- {“Alice”, proof I’m Alice} key variant-dependent
- {“Bob”, proof I’m Bob}
General idea of “quick mode”

IKE-SA, Y, traffic, SPI_A, \([g^A \mod p]\)

IKE-SA, Y, traffic, SPI_B, \([g^B \mod p]\)

IKE-SA, Y, ack
IKEv1 vs IKEv2

- IKEv1:
  - 9 msgs (ID hiding) or 6 to set up IPsec SA
  - 8 different protocols
  - lots of networking issues

- What we did for IKEv2:
  - one protocol, 4 messages, ID hiding
  - cleaned it up and simplified it a lot
  - wrote the spec in English
General idea of IKEv2

Alice

g^A \mod p, \text{nonce}_A

Bob

g^B \mod p, \text{nonce}_B

{“Alice”, proof I’m Alice}g^{AB} \mod p

{“Bob”, proof I’m Bpb}g^{AB} \mod p
Of the IKEv1 Variants

• The one required variant (main mode, pre-shared secret keys): nobody uses it
  – ID transmitted, encrypted with key which is a function of pre-shared key! Can’t decrypt unless you can guess who you’re talking to!
  – So ID=IP Address. Useless for “road warrier”

• Instead, the most commonly used one is aggressive mode, pre-shared secret keys
Idiosyncrasies of other modes

- public encryption keys, must know public key of other side before it sends cert. Original doesn’t even allow sending cert.
- original public encryption keys: separately encrypt fields with other side’s public key (requires separate private key ops too). Undefined if field (e.g. name) bigger than an RSA block.
Main-Mode-preshared key $S$

Alice: crypto suites I support

<table>
<thead>
<tr>
<th>g^A \mod p, nonce_A</th>
</tr>
</thead>
<tbody>
<tr>
<td>g^B \mod p, nonce_B</td>
</tr>
</tbody>
</table>

Bob: crypto suites I choose

{“Alice”, proof I’m Alice} $f(S, g^{AB})$

{“Bob”, proof I’m Bob} $f(S, g^{AB})$
Original Main, Encryption Keys

Alice  
crypto suites I support

Bob  
crypto suites I choose

\[ g^A \mod p, \{\text{nonce}_A}\}Bob, \{“Alice”}\}Bob \]

\[ g^B \mod p, \{\text{nonce}_B\}Alice, \{“Bob”\}Alice \]

\{proof I’m Alice\} K=f(nonces, D-H key)

\{proof I’m Bob\} K
Revised Main, Encryption Keys

Alice  \[ \text{crypto suites I support} \]
\[ K_A = h(\text{nonce}_A, \text{cookie}_A) \]
\[ \{\text{nonce}_A\} \text{Bob, } \{g^A \mod p, \text{“Alice”}, \text{cert}\} \]

Bob  \[ \text{crypto suites I choose} \]
\[ K_B = h(\text{nonce}_B, \text{cookie}_B) \]
\[ \{\text{ nonce}_B\} \text{Alice, } \{g^B \mod p, \text{“Bob”}\} \]
\[ \{\text{proof I’m Alice}\} \text{ } K = f(\text{nonces, cookies, D-H key}) \]
\[ \{\text{proof I’m Bob}\} \text{ } K \]
IKEv1 vs the version number

- version number: doesn’t say what to do if version number bigger… “SHOULD reject”. So can’t (in theory) count on v1 throwing away v2.

- from our book:

  “ISAKMP doesn’t exactly say you reject it if the version is larger than yours. It says you SHOULD reject it. So implementations are free to ignore the version number, but perhaps feel a little guilty about it”
Version #, cont’d

- 8 bit field
- 4 bits major, 4 bits minor
- What major/minor should be: ignore minor
- But ISAKMP says “SHOULD reject if major larger than yours, or if the same, if minor larger than yours”
- So just like 8 bit field, but more complicated, and more likely to run out of numbers
Stateless Cookies

- Photuris came up with stateless cookies so Alice proves she can receive from her claimed IP address before Bob devotes any state or significant computation
- ISAKMP has fields called “cookies”, but lost the ability to be stateless
- Could have been stateless by copying info from msg 1 into msg 3
- Except they’re required to be unique...
IKEv1 “cookies”

- The pair of cookies (initiator, responder) form the SA identifier

- possible to have “cookie collision”
  - Alice initiates to Bob, choosing A
  - Carol initiates to Alice, choosing A
  - Alice responds to Carol, choosing B
  - Bob might choose B
  - result: two connections Alice is involve in defined by (A,B)
Negotiating Crypto

- negotiate: encryption, hash, auth method, Diffie-Hellman group
- If n of one work with any of k of another, exponential explosion, since have to specify each combination separately
- very complex encoding
- from our book:
  
  “Assembly of SA payload requires great peace of mind”…apologies to Pirsig
IKEv2

- First simplified the a la carte negotiation, and allowed “any of these together with any of those” and allowed Bob to choose
- Then proposed suites, WG enthusiastic
- We changed the spec
- Then WG demanded we change it back..
Reflection Attacks

- same key in both directions of IKE SA
- Same SPI: (originator, responder) cookies
- Message ID: random number instead of sequence number
- IKEv2: message ID is sequence number, different key in each direction, although (SPIi, SPIo), has flag saying what direction packet is going
Unprotected fields

- IKEv1 based on SKEME, theoretical paper, which said hash of this and that
- The fields left out of the hash could be exploited by attackers
- Much more straightforward to just integrity check entire message
- IKEv2: started out simple, but...you know, committees...all fields covered though
Traffic Restrictions

• IPsec policy: Traffic between these sets of IP adds, and protocol types, and ports, must have this sort of cryptographic protection
• Creating SA, specify “traffic selectors”
• IKEv1: Initiator proposes. Responder (if has more restrictive policy) can just say “no”
• IKEv2: allowed responder to narrow or say “single pair”
Lost messages

• IKEv1 kind of didn’t say
  – had “commit bit”, defined incomprehensibly and almost oppositely in ISAKMP and IKE
    • ISAKMP: for Bob to tell Alice to wait for his ack
    • IKE: for Bob to tell Alice to send an ack

• IKEv2: all messages request/response, and requester keeps sending until it gets a response
Whose identity hidden?

- With signature keys and active attacker, whoever reveals identity first might be tricked into revealing identity
- Can avoid this (in theory) with pre-shared keys or public encryption keys (but have to know who you’re talking to!)
- We first argued Alice’s ID more important to hide
- Then decided polling attack easier to mount (if IPsec truly peer-to-peer)
Fragmentation Attack

• IKE runs on top of UDP
• Things that run on top of TCP can adjust to avoid IP fragmentation
• Once SA established, there’s state and again, can adjust to avoid IP fragmentation
• But message 3 contains certs and can be really large. If that’s where cookie is returned, have fragmentation attack
IKEv2 with stateless cookies, 4-msgs

Alice          Bob

$g^A \mod p$, crypto proposal

$g^B \mod p$, cookie=C, crypto

C, {“Alice”, proof I’m Alice} $g^{AB} \mod p$,
repeat other info from msgs 1 and 2

{“Bob”, proof I’m Bob} $g^{AB} \mod p$
So, did optional pre-round trip

- If Bob isn’t under attack, 4-msgs
- If he wants a cookie, he’ll refuse message 1 that doesn’t contain a cookie (and give a cookie to return)
- This (among other advantages) allowed easy protection against fragmentation attack
IKEv2 with stateless cookies, 4/6-msgs

Alice: $g^A \mod p$, crypto proposal

Bob: send cookie = C

Alice: C, $g^A \mod p$, crypto proposal

Bob: $g^B \mod p$, crypto accepted

Alice: {"Alice", proof I’m Alice} $g^{AB} \mod p$

Bob: {"Bob", proof I’m Bob} $g^{AB} \mod p$
Fragmentation Defense

- With 4/6, msgs short until cookie verified
- IKE pass hint to reassembly code “this IP address preferred”
- Preferable to leave it on preferred list for minimum time
- Other defenses?
Original JFK

Alice

$g^A \mod p, N_A$

Bob

“Bob”, $[g^B \mod p]Bob$, cookie=C, crypto, $N_B$

C, {“Alice”, proof I’m Alice}$g^{AB} \mod p$

repeat other info from msgs 1 and 2

[signature on nonces, etc.]Bob
JFK

- Hides initiator’s identity from active attacker, doesn’t hide responder’s at all
- Bob chooses crypto, Alice gets no say
- extra signature
Other stuff added into IKEv2

- NAT Traversal
- Legacy authentication
- Acquiring an address
NAT

Keeps mapping g/K. Overwrites source address on outgoing, overwrites destination address on incoming
NAT Issues

- All sorts of IP protocols violate layering
- TCP/UDP:
  - “pseudoheader”: checksum computed on addresses from IP header, and TCP header
  - So NAT box must fudge TCP/UDP checksum
- FTP
  - sends addresses as text string “144.27.8.95
  - NAT must look inside FTP data to change address
  - Worse yet! if changes TCP byte numbering: NAT must keep track and fix TCP ack, and msg #’s!
NAT/NAPT/ESP

- NAT: reassigns IP address
- NAPT: if not enough IP addresses to hand out, uses layer 4 port, maybe all IP=Q
- But can’t do that with ESP
- So, if see and ESP packet from A for B, with SPI=x, remembers that, reassigns IP address to Q
- If see packet from B for Q, assume it’s for A
- This sort of works
But it gets funnier!

- IKEv1 specifies IKE must be on port 500 (both source and destination)
- Would have been simpler if it said send to 500, but accept from any port
- So NAPT's map based on (Ci, Cr)
- But it starts out as (Ci, 0)
- If two nodes start IKE connections (Ci, 0) to Bob simultaneously, a problem
IKEv2 vs NAT

• NAT detection (by putting original IP addresses inside IKE message and comparing with header)

• So, you’re behind a NAT…
  – Switch to port 4500
  – don’t insist you receive from 4500
  – Encapsulate child-SA in UDP as well, also port 4500, so IKE packets start with 0.0.0.0