Network security: IPsec

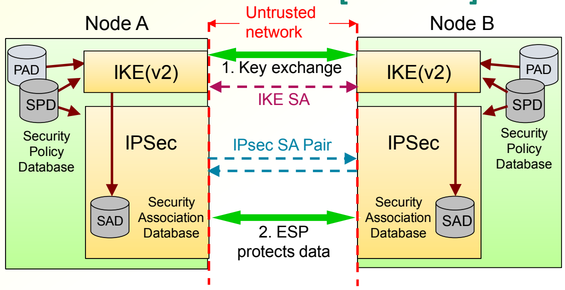
Tuomas Aura, Microsoft Research, UK

IPsec architecture and protocols

Internet protocol security (IPsec)

- Network-layer security protocol
  - Protects IP packets between two hosts or gateways
  - Transparent to transport layer and applications
  - IP addresses used to as host identifiers
- Two steps:
  1. IKE creates security associations
  2. ESP session protocol protects data
- Specified by Internet Engineering Task Force (IETF)
  - Original goal: encryption and authentication layer that will replace all others
  - Sales point for IPv6; now also in IPv4

IPsec architecture [RFC4301]



- Security associations (SA) created by IKE, used by IPsec ESP
- Security policy guides SA creation and selection for use
- IPsec is part of the IP layer in the OS kernel; IKE is a user-space service (daemon)

Internet Key Exchange (IKE)

- IKE(v1) [RFC 2407, 2408, 2409]
  - Framework for authenticated key-exchange protocols, typically with Diffie-Hellman
  - Multiple authentication methods: certificates, pre-shared key, Kerberos
  - Two phases: Main Mode (MM) creates an ISAKMP SA (i.e. IKE SA) and Quick Mode (QM) creates IPsec SAs
  - Main mode (identity-protection mode) and optimized aggressive mode
  - Interoperability problems: too complex to implement and test all modes; specification incomplete
- IKEv2 [RFC 4306]
  - Redesign of IKE: less modes and messages, simpler to implement
  - Initial exchanges create the IKE SA and the first IPsec SA
  - CREATE_CHILD_SA exchange create further IPsec SAs
  - EAP authentication for extensions
- Works over UDP port 500

Internet Key Exchange (IKEv2)

Initial exchanges:

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,B), SAr1, KEr, Nr, [CERTREQ]
3. I → R: HDR(A,B), SK { IDi, [CERT,] [CERTREQ,] [IDr,] AUTHi, SAi2, TSi, TSr }
4. R → I: HDR(A,B), SK { IDr, [CERT,] AUTHr, SAr2, TSi, TSr }

A, B = SPI values that identity the protocol run and the created IKE SA
Nx = nonces
SAx1 = offered and chosen algorithms, DH group
KEx = Diffie-Hellman public key
IDx, CERT = identity, certificate
AUTHr = $Sign_I$ (Message 1, Nr, h(SK, IDi))
AUTHr = $Sign_R$ (Message 2, Ni, h(SK, IDr))
SK = h(Ni, Nr, g^{xy}) — a bit simplified, 6 keys are derived from this
SK { … } = E_{SK}(…, MAC_{SK}(…)) — MAC and encrypt
SAx2, TSx = parameters for the first IPsec SA (algorithms, SPIs, traffic selectors)
CERTREQ = recognized root CAs (or other trust roots)

Internet Key Exchange (IKEv2)

Initial exchanges:

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,B), SAr1, KEr, Nr, [CERTREQ]
3. I → R: HDR(A,B), SK { IDi, [CERT,] [CERTREQ,] [IDr,] AUTHi, SAi2, TSi, TSr }
4. R → I: HDR(A,B), SK { IDr, [CERT,] AUTHr, SAr2, TSi, TSr }

A, B = SPI values that identity the protocol run and
Nx = nonces
SAx1 = offered and chosen algorithms, DH group
KEx = Diffie-Hellman public key
IDx, CERT = identity, certificate
AUTHi = Sign_I (Message 1, Nr, h(SK, IDi))
AUTHr = Sign_R (Message 2, Ni, h(SK, IDr))
SK = h(Ni, Nr, g^{xy}) — a bit simplified, 6 keys are der
SK { … } = E_{SK}(…, MAC_{SK}(…)) — MAC and encrypt
SAx2, TSx = parameters for the first IPsec SA (algor
CERTREQ = recognized root CAs (or other trust roo

Secret session key?
Fresh session key?
Mutual authentication?
Entity authentication?
Key confirmation?
Protection of long-term secrets?
Forward secrecy?
Contributory?
Non-repudiation?
Integrity of negotiation?
DoS protection?
Identity protection?

IKEv2 with a cookie exchange

- Responder may respond to the initial message by sending a cookie
- Goal: prevent DOS attacks from a spoofed IP address

1. I → R: HDR(A,0), SAi1, KEi, Ni
2. R → I: HDR(A,0), N(COOKIE) // R stores no state
3. I → R: HDR(A,0), N(COOKIE), SAi1, KEi, Ni
4. R → I: HDR(A,B), SAr1, KEr, Nr, [CERTREQ] // R creates a state
5. I → R: HDR(A,B), SK{ IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr }
6. R → I: HDR(A,B), E_{SK} (IDr, [CERT,] AUTH, SAr2, TSi, TSr)

How to bake a good cookie? For example:
COOKIE = h($N_{R\text{-periodic}}$, IP addr of I, IP addr of R) where $N_{R\text{-periodic}}$ is a periodically changing secret random value know only by the responder R

Security Associations (SA)

- One IKE SA for each pair of nodes
  - Stores the master key SK = h(Ni, Nr, g^{xy}) for creating IPsec SAs
- At least one IPsec SA pair for each pair of nodes
  - Stores the negotiated session protocol, encryption and authentication algorithms, keys and other session parameters
  - Stores the algorithm state
  - IPsec SAs always come in pairs, one in each direction
- SAs are identified by a 32-bit security parameter index (SPI) [RFC4301]
  - For unicast traffic, the destination node selects an SPI value that is unique to that destination
- Node stores SAs in a security association database (SAD)

10

Session protocol

- Encapsulated Security Payload (ESP) [RFC 4303]
  - Encryption and/or MAC for each packet
  - Optional replay prevention with sequence numbers
  - Protects the IP payload (= transport-layer PDU) only
- ESP with encryption only is insecure
- Deprecated: Authentication Header (AH)
  - Do not use for new applications
  - Authentication only
  - Protects some IP header fields

11

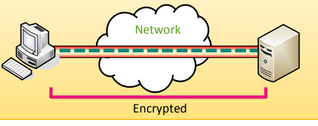
Session protocol modes

- Transport mode:
  - Host-to-host security
  - ESP header added between the original IP header and payload
- Tunnel mode:
  - Typically used for tunnels between security gateways to create a VPN
  - Entire original IP packet encapsulated in a new IP header plus ESP header
- In practice, IPsec is mainly used in tunnel mode
- Proposed BEET mode:
  - Like tunnel mode but inner IP header not sent explicitly
  - Transport-mode headers but tunnel mode semantics

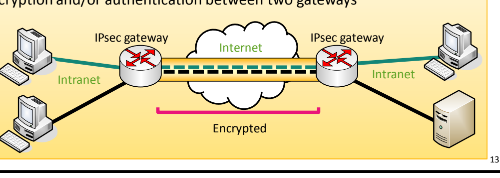
12

Session protocol modes

Transport mode
Encryption and/or authentication from end host to end host



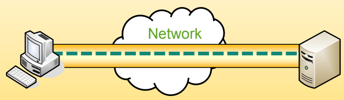
Tunnel mode
Encryption and/or authentication between two gateways



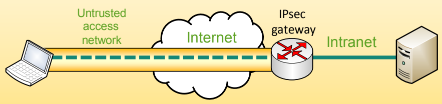
13

Using tunnel mode with hosts

Tunnel mode - between end hosts (equivalent to transport mode)



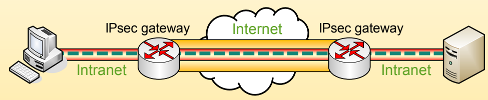
Tunnel mode - between a host and a gateway

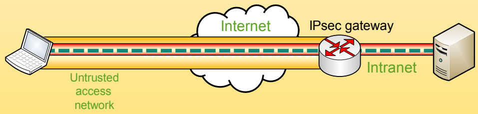


14

Nested protection

Nested tunnel and transport mode

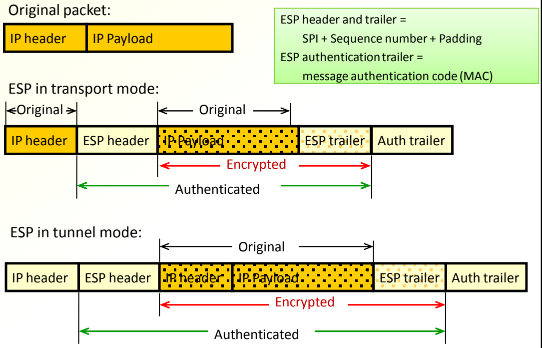




15

ESP packet format (1)

Original packet:



16

ESP packet format (2)

- ESP packets in a more abstract notation
- Transport mode headers:

  IP(src host, dst host)|ESP|payload
- Tunnel mode headers:

  IP(src gw, dst gw)|ESP|IP(src host,dst host)|payload

17

IPsec databases

- Security association database (SAD)
  - Contains the dynamic protection state
- Security policy database (SPD)
  - Contains the static security policy
  - Usually set by system administrators (e.g. Windows group policy), although some protocols and applications make dynamic changes
- Peer authorization database (PAD)
  - Needed in IKE for mapping between authenticated names and IP addresses
  - Conceptual; not implemented as an actual database
- Additionally, the IKE service stores IKE SAs:
  - Master secret created with Diffie-Hellman
  - Used for instantiating IPsec SAs

(Note: our description of SDP differs somewhat from RFC4301 and is probably closer to most implementations)

18

Security policy database (SPD)

- Specifies the static security policy
- Multi-homed nodes have a separate SPD for each network interface
- Maps inbound and outbound packets to actions
  - SPD = linearly ordered list of policies
  - Policy = selectors + action
  - The first policy with matching selectors applies to each packet
- Policy selectors:
  - Local and remote IP address
  - Transport protocol (TCP, UDP, ICMP)
  - Source and destination ports
- Actions: BYPASS (allow), DISCARD (block), or PROTECT
  - PROTECT specifies also the session protocol and algorithms
  - Packet is mapped to a suitable SA
  - If the SA does not exist, IKE is triggered to create one
  - SPD stores pointers to previously created SA

19

Security association database (SAD)

- Contains the dynamic encryption and authentication state
- IPsec SAs always come in pairs: inbound and outbound
- SAD is keyed by SPI (for unicast packets)
- SAs are typically created by IKE but may also be configured manually, e.g. for fixed VPN tunnels
- Each SAD entry contains also the policy selector values that were used when creating it

20

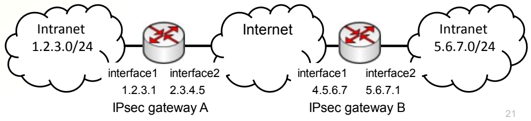
Gateway SPD/SAD example

SPD of gateway A, interface 2

Protocol	Local IP	Port	Remote IP	Port	Action	Comment
UDP	2.3.4.5	500	4.5.6.7	500	BYPASS	IKE
*	1.2.3.0/24	*	5.6.7.0/24	*	ESP tunnel to 4.5.6.7	Protect VPN traffic
*	*	*	*	*	BYPASS	All other peers

SAD of gateway 1

SPI	SPD selector values	Protocol	Algorithms, keys and algorithm state
spi1	UDP,1.2.3.0/24,5.6.7.0/24	ESP tunnel from 4.5.6.7	...
spi2	–	ESP tunnel to 4.5.6.7	...



21

Host SPD example

- SPD for host 1.2.3.101 in intranet 1.2.3.0/24, connecting to server 1.2.4.10 in DMZ 1.2.4.0/24 and to the Internet

Protocol	Local IP	Port	Remote IP	Port	Action	Comment
UDP	1.2.3.101	500	*	500	BYPASS	IKE
ICMP	1.2.3.101	*	*	*	BYPASS	Error messages
*	1.2.3.101	*	1.2.3.0/24	*	PROTECT: ESP in transport-mode	Encrypt intranet traffic
TCP	1.2.3.101	*	1.2.4.10	80	PROTECT: ESP in transport-mode	Encrypt to server
TCP	1.2.3.101	*	1.2.4.10	443	BYPASS	TLS: avoid double encryption
*	1.2.3.101	*	1.2.4.0/24	*	DISCARD	Others in DMZ
*	1.2.3.101	*	*	*	BYPASS	Internet

- What is the danger of bypassing TLS traffic (line 5)?
- What is the danger of bypassing outbound ICMP (line 2)?
- Note that both IPsec endpoints must have matching policies

22

IPsec policy implementation differences

- Historically, IPsec and firewalls have different models of the network:
  - Firewall is a packet filter: which packets to drop?
  - IPsec sits between the secure and insecure areas (host and network at IPsec hosts, intranet and Internet at IPsec gateways) and encrypts packets that leave the secure side

  The models, however, can be unified
- In some IPsec implementations, the policy is specified in terms of source and destination addresses (like a typical firewall policy), instead of local and remote addresses

  → mirror flag is shorthand notation to indicates that the policy applies also with the source and destination reversed

Mirror	Protocol	Source IP	Port	Destination IP	Port	Action	Comment
yes	UDP	2.3.4.5	500	4.5.6.7	500	BYPASS	IKE
yes	*	1.2.3.0/24	*	5.6.7.0/24	*	ESP tunnel to 4.5.6.7	Protect VPN traffic
yes	*	*	*	*	*	BYPASS	All other peers

23

Outbound packet processing

- Processing outbound packets:
  1. For each outbound packet, IPsec finds the first matching policy in the security policy database (SDP)
  2. If the policy requires protection, IPsec maps the packet to the right security association (SA) in the SA database (SAD)
  3. If no SA exists, IPsec invokes the IKE service to create a new SA pair
  4. While waiting for the IPsec SA, at most one outbound packet (often TCP SYN) is buffered in the kernel
  5. When the SA exists, the packet is encrypted and a MAC added

24

Inbound packet processing

- Processing inbound IPsec packets:
  1. IPsec looks up the inbound SA in SAD based on the SPI
  2. IPsec processes the packet with the SA, i.e. verifies the MAC and decrypts
  3. IPsec compares the packet with the selector values that were used when creating this SAD entry. For tunnel-mode packets, the comparison is done with the inner IP header
- Processing of inbound non-IPsec packets:
  - IPsec finds the first matching policy in the SPD and checks that the action is BYPASS
  - If the action is not BYPASS, the packet is dropped
- In Windows, it is possible to allow the first inbound packet (often TCP SYN) to bypass IPsec. The outbound response will trigger IKE
  - Helps in gradual deployment of host-to-host IPsec

25

Some problems with IPsec

IPsec and NAT

- Problems:
  - NAT cannot multiplex IPsec: impossible to modify SPI or port number because they are authenticated

  → Host behind a NAT could not use IPsec
- NAT traversal (NAT-T):
  - UDP-encapsulated ESP (port 4500)
  - NAT detection: extension of IKEv1 and IKEv2 for sending the original source address in initial packets

  → Host behind a NAT can use IPsec

27

IPsec and mobility

- Problem:
  IPsec policies and SAs are bound to IP addresses. Mobile node's address changes
- Mobile IPv6 helps: home address (HoA) is stable. But mobile IPv6 depends on IPsec for the tunnel between HA and MN.
  → Chicken-and-egg problem
- Solution:
  IPsec changed to indexed SAs by SPI only
- IPsec-based VPNs from mobile hosts do not use the IP address as selector. Instead, proprietary solutions
- MOBIKE mobility protocol

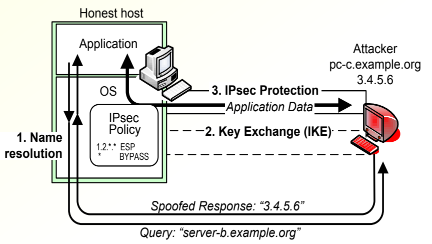
28

IPsec and Identifiers

- Application opens a connection to an IP address. IPsec uses the IP addresses as policy selector
- IKE usually authenticates the remote node by its DNS
- Problem: No secure mapping between the two identifier spaces: DNS names and IP addresses

29

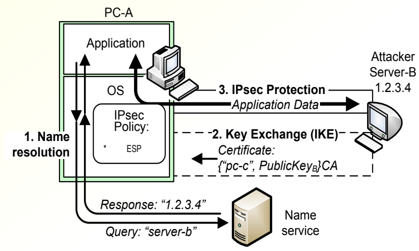
Classic IPsec/DNS Vulnerability



- IPsec policy selection depends on secure DNS

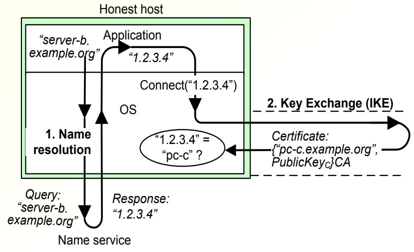
30

IPsec and Certificates



31

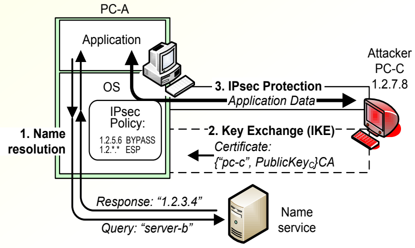
IPsec and Certificates - Details



- IKE knows the peer IP address, not its name, but the certificate only contains the name

32

IPsec and Certificates - Attack



- IPsec cannot detect the attack
- Result: group authentication only → maybe of for VPN where the goal is to keep outsiders out

33

Peer authorization database (PAD)

- IPsec spec [RFC4301] defines a database that maps authenticated names to the IP addresses which they are allowed to represent
  - How implemented? Secure reverses DNS would be the best solution — but it does not exist.
- Other solutions:
  - Accept that group authentication is ok — short-term solution
  - Secure DNS — both secure forward and reverse lookup needed, which is unrealistic
  - Give up transparency — extend the socket API so that applications can query for the authenticated name and other security state
  - Connect by name — change the socket API so that the OK knows the name to which the application wants to connect

34

Exercises

- For the IPsec policy examples of this lecture, define the IPsec policy for the peer nodes
- Try to configure the IPsec policy between two computers. What difficulties did you meet? Use ping to test connectivity. Use a network sniffer to observe the key exchange and to check that packets on the wire are encrypted
- Each SAD entry stores (caches) policy selector values from the policy that was used when creating it. Inbound packets are compared against these selectors to check that the packet arrives on the correct SA.
  - What security problem would arise without this check?
  - What security weakness does the caching have?
  - Some IPsec implementations stored a pointer to the policy entry, instead of caching the selector. What weakness did this have?
  - RFC 4301 solves these problems by requiring the SPD to be decorrelated, i.e. for the selectors of policy entries not to overlap. Yet, the policies created by system administrators almost always have overlapping selectors. Device an algorithm for transforming any IPsec policy to an equivalent decorrelated one.

35