SSL, SSH and IPSec
Overview of things to come

• Security can be implemented at many levels
  – Kerberos, SSL and SSH are implemented at the application level
  – No need to change the OS
  – Applications must be specially designed to work with Kerberos, SSL or SSH

  – IPSec is implemented at the transport level
  – Inside the OS
  – More transparent to user

<table>
<thead>
<tr>
<th>SSL</th>
<th>TCP</th>
<th>IP</th>
<th>Lower layers</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TCP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lower layers</td>
</tr>
</tbody>
</table>
Security facilities in the TCP/IP protocol stack

(a) Network Level

(b) Transport Level

(c) Application Level
SSL and TLS

• SSL provides a secure transport connection between applications
  – Usually client and server

• SSL (Secure Socket Layer) was originally proposed by Netscape

• SSL v3.0 was specified in an Internet Draft (1996)
  – has been widely implemented in web browsers and web servers
    • e.g., Netscape Navigator and MS Internet Explorer

• TLS (Transport Layer Security) --1999
  – TLS can be viewed as SSL v3.1
    • TLS is not interoperable with SSL v3 (slightly different crypto)
  – TLSv1.2 (2008): crypto update
Components

- **SSL Handshake Protocol**
  - negotiation of security algorithms and parameters
  - key exchange
  - server authentication -- optionally client authentication

- **SSL Change Cipher Spec Protocol**
  - a single byte message-- indicates end of the SSL handshake

- **SSL Record Protocol**
  - fragmentation
  - compression
  - message authentication and integrity protection
  - encryption

- **SSL Alert Protocol**
  - error messages (fatal alerts and warnings)
Important Concepts

• SSL works in terms of connection and sessions between client and server:
  – SSL Session:
    • An association between a server and a client
    • Stateful
      – cryptographic security parameters
    • Can be multiple sessions between parties (but not common)
    • Sessions are created by the handshake protocol
  – SSL Connection:
    • Peer-to-peer relationship, transient
    • Every connection is associated with a session
    • A session can have multiple connection
Lower Layer: SSL Record Protocol

- Receive messages from upper layer
- Breaks messages into blocks
- Compresses blocks
- Computes MAC for each block
  - Each block has implicit sequence number to prevent reordering
- Encrypts blocks
  - Note: if encryption and MAC key not selected then no encryption or MAC used
- Adds header
SSL Record Protocol Operation

Application Data

Fragment

Compress

Add MAC

Encrypt

Append SSL Record Header
SSL Record Format

<table>
<thead>
<tr>
<th>Content Type</th>
<th>Major Version</th>
<th>Minor Version</th>
<th>Compressed Length</th>
</tr>
</thead>
</table>

- Plaintext (optionally compressed)
- MAC (0, 16, or 20 bytes)
Handshake Protocol

• The most complex part of SSL.

• Authenticate both server and client

• Negotiate encryption, MAC algorithm and cryptographic keys.

• Used before any application data are transmitted.
SSL Handshake Protocol

**Phase 1:** Negotiation of the session ID, key exchange algorithm, MAC algorithm, encryption algorithm, and exchange of initial random numbers.

**Phase 2:** Server may send its certificate and key exchange message, and it may request the client to send a certificate. Server signals end of hello phase.

**Phase 3:** Client sends certificate if requested and may send an explicit certificate verification message. Client always sends its key exchange message.

**Phase 4:** Change cipher spec and finish handshake.
Certificate

- **Serial Number**: Used to uniquely identify the certificate.
- **Subject**: The person, or entity identified.
- **Signature Algorithm**: The algorithm used to create the signature.
- **Signature**: The actual signature to verify that it came from the issuer.
- **Issuer**: The entity that verified the information and issued the certificate.
- **Valid-From**: The date the certificate is first valid from.
- **Valid-To**: The expiration date.
- **Key-Usage**: Purpose of the public key (e.g. encipherment, signature, certificate signing...).
- **Public Key**: The public key.
- **Thumbprint Algorithm**: The algorithm used to hash the public key certificate.
- **Thumbprint** (also known as fingerprint): The hash itself, used as an abbreviated form of the public key certificate.
Change Cipher Spec Protocol

- Sent by both the client and server to notify the other party that the following records will be protected using the just-negotiated CipherSpec and keys.

- Consists of single message -- a single byte with the value 1.

- The purpose of the message is to updates the cipher suite to be used on the connection.
Alert Protocol

• Used to convey SSL-related alerts to the peer entity.

• Alert messages are compressed and encrypted.

• The message is two bytes:
  – 1 byte: warning (1) or fatal (2)
  – 1 byte: status of the certificate & other specific alerts
Second byte

- fatal
  - unexpected_message
  - bad_record_MAC
  - decompression_failure
  - handshake_failure
  - illegal_parameter
- warning
  - no_certificate
  - bad_certificate
  - unsupported_certificate
  - certificate_revoked
  - certificate_expired
  - certificate_unknown

- in case of a fatal alert
  - connection is terminated
  - session ID is invalidated
    - no new connection can be established within this session
Attacks on SSL

• SSL is claimed to protects against MITM attack using
  – End point authentication
  – Encryption
  – Attacks are even more dangerous because of perceived security.

• MITM attacks rely on spoofing ARP and DNS.

• Causes of reported attack
  – Improper use of cryptography
  – Mis-configured clients
  – Bad implementation

• Crypto 2009: use weakness in MD5 to forge a ‘rogue’ certificate
• Lack of user awareness and education.
  – Users click-through on certificate warnings.
Flaws in SSL v2

- Same key used for encryption and MAC
- Weak keys due to old export restrictions in US
- Weak MAC based on MD5 only
- Weakness in handshake can allow for MITM attack
- Uses TCP close to indicate end of data (can lead to truncation attack)
- SSL v2 now disabled by default in IE v7, Firefox v2
SSL-TLS

- Version number
  - TLS 1.1 is the SSL version 3.2

- Cipher suites
  - TLS doesn’t support Fortezza key exchange and encryption

- Padding
  - variable length padding is allowed (max 255 padding bytes)

- MAC
  - TLS uses the latest version of HMAC
  - the MAC covers the version field of the record header too

- certificate_verify message
  - the hash is computed only over the handshake messages
  - in SSL, the hash contained the master_secret and pads

- More alert codes
SSL-TLS

• TLS v1.2 (2008)
  – MD5-SHA1 based MAC replaced by cipher-based MAC
  – Added AES-based ciphers to list
  – Slight modifications about how hashes are used
OpenSSL Project

- A collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit.

- It implements:
  - Secure Sockets Layer (SSL v2/v3)
  - Transport Layer Security (TLS v1) protocols
  - A full-strength general purpose cryptography library.

http://www.openssl.org/
Secure Shell (SSH)

(teaser)
SSH

• Started as secure replacement for *telnet*.
  – SSH-1 Tatu Ylönen (1995)
  – SSH-3 (1999) OSSH → OpenSSH

• Provides confidentiality
  – Credential used for login
  – Content of the remote login session

• SSH provides security at **Application Layer**.
  – Secure copying of files between client and server
  – Also can be used for tunnelling other protocols
  – **Transport layer** security for those protocols
SSH

• SSH authenticates both the client and the server.

• Authentication:
  – Server: Public/Private key pair
    • Client uses a locally stored PK of the server to verify the server’s signature
  – Client:
    • Username/password
    • Asymmetric key pair – the server need to know the PK
Applications

• Anonymous ftp for software updates, patches...
  – client authentication not needed
  – clients want to be sure of origin and integrity of software

• Secure ftp.
  – e.g. upload of webpages to webserver using sftp
  – Server needs to authenticate clients
  – Username and password sufficient
  – transmission over secure SSH transport layer protocol

• Secure remote administration
  – SysAdmin (client) sets up terminal on remote machine
  – SysAdmin password protected by SSH transport layer protocol

• Secure remote login
SSH-2 Architecture

• Three layer architecture: [http://www.ietf.org/rfc/rfc4251.txt](http://www.ietf.org/rfc/rfc4251.txt)

• Transport Layer Protocol: provides
  – Initial connection
  – server authentication,
  – confidentiality, and integrity with perfect forward secrecy
  – Key re-exchange after 1Gb of data transmitted or after 1 hour

• User Authentication Protocol
  – Authenticates client to the serve

• Connection Protocol
  – Supports multiple connections (channels) over a single transport layer protocol secure channel.
SSH-2 Architecture

- Applications
- SSH Connection Protocol
- SSH Authentication Protocol
- SSH Transport Layer Protocol
- TCP
Key Exchange and Authentication

- Server listens to port 22
Port forwarding

- Also tunneling: a way to forward TCP traffic through SSH.
  - e.g securing POP3, SMTP and HTTP connections
    - insecure connections
  - The client-server applications will run their normal authentication over the encrypted tunnel.

- There are two types of port forwarding:
  - local (outgoing tunnel)
  - remote forwarding. (incoming tunnel)

- Local port forwarding:
  - forwards traffic coming to a local port to a specified remote port.
  - E.g. the command,

    ```
    ssh2 -L 1234:localhost:23 username@host
    ```
  - traffic to port 1234 on the client will be forwarded to port 23 on the server (host).
Remote port forwarding

• Does the opposite:
  – forwards traffic coming to a remote port to a specified local port.

• For example, the command

  ssh2 -R 1234:localhost:23 username@host

  – traffic that comes from port 1234 on the server (host) will be forwarded to port 23 on the client (localhost).
Causes of Insecurity

• Implementation weaknesses (e.g. plaintext recovery attack 2008)
• Does not use very standard crypto (attack in 2004)
• Weak server platform security
  – Worms, malicious code, rootkits,…
• Weak user platform security
  – Keystroke loggers, malware,…
• Absence of use of certificates for public key (rely on user)
• Lack of user awareness and education.
  – Users click-through on warnings.
Causes of Insecurity

• Implementation weaknesses (e.g. plaintext recovery attack 2008)
• Does not use very standard crypto (attack in 2004)
• Weak server platform security
  – Worms, malicious code, rootkits,…
• Weak user platform security
  – Keystroke loggers, malware,…

• Absence of use of certificates for public key (rely on user)
• Lack of user awareness and education.
  – Users click-through on warnings.
IPSec
History

• In 1994, the Internet Architecture Board (IAB) issued a report entitled *Security in the Internet Architecture* (RFC 1636).
  – Many incidents that affected many sites, IP spoofing attacks, packet sniffing, etc.
  – General consensus: the Internet needed better security.

• Internet Protocol Security (IPsec): a suite of protocols for securing IP communications. (approx Layer 3)

• IPsec is specified by the Internet Engineering Task Force (IETF)

http://www.networksorcery.com/enp/topic/ipsecsuite.htm
Internet Protocol Security (IPsec)

• Protection of communication between,
  – hosts
  – gateways
  – a host and a gateway
Packet-oriented security

• Comparison with SSL, TLS, SSH:
  – These are at higher level of OSI stack
  – Applications must be altered to incorporate these

• IPsec provides application-transparent Security
  – Network services that use IP (e.g. telnet, FTP) or user application that uses IP (TCP BSD Socket) can use IPSec without modification.
Components

• ISAKMP (Internet Security Association and Key Management Protocol)
  – Security association
  – IKE (Internet key exchange) for establishing security association
  – a collection of algorithms and parameters, keys, etc to encrypt and authenticate a data flow in one direction.

• Security protocols:
  1. Authentication Header (AH)
  2. Encapsulating Security Payload (ESP)

• Databases
  – Security Association Database (SADb)
  – Policy Database
IPSec Architecture

• Security Policy Database (SPD)
  – Given source and destination IP addresses, determines which if packets are kept or discarded, and whether IPSec is applied or bypassed

• Security Association (SA)
  – Association between peers for security services
  – Unidirectional
  – Defined uniquely by destination address, security protocol (AH or ESP) and security parameter index (SPI)
  – Contains only one security protocol (if both AH and ESP are used, there will be two SAs)
IPSec Architecture

- Security Association bundle
  - Sequence of SAs that IPSec applies to packets

- Security Association Database (SADb)
  - Indexed by destination (or incoming) address and SPI
  - Its key fields are
    - Crypto algorithm identifier and keys
    - Lifetime of the SA
    - Whether in tunnel mode or transport mode
Modes

- Two modes of applying IPsec protection
- Both modes are applicable to both security protocols (AH and ESP)

1. Transport mode
   - Default mode for end-to-end security
     - Client-server communication
   - IPsec information is added between IP header and the rest of the packet

2. Tunnel modes
   - Used for protecting traffic between two networks when packets have to pass through an untrusted network
   - Whole IP packet becomes payload to a new IP packet protected by IPSec
Modes

- **Transport mode**
  - IP header | IPsec | Rest of packet

- **Tunnel mode**
  - New IP hdr | IPsec | IP head | Rest of the packet
Modes

- In transport mode, IP datagram contains only *one* IP header,
  - specifies the source address and the ultimate destination

- In tunnel mode, an IP datagram contains two IP headers:
  - an outer IP header: specifies the IPSec processing destination
  - an inner IP header: contains the source and the ultimate destination of the packet
  - the inner packet is usually encrypted
Encapsulating Security Payload (ESP)

• The Encapsulating Security Payload protocol provides
  – confidentiality service
  – limited traffic-flow confidentiality
  – authentication service
    • Applied to payload only

• In transport mode, ESP secures upper-layer protocols.
• In tunnel mode, ESP extends protection to the inner IP header.
ESP Protocol in Transport Mode
ESP Protocol in Tunnel Mode
Authentication Header (AH) Protocol

- Authentication Header protocol provides an authentication service for
  - data origin authentication
  - connectionless integrity
  - an optional anti-replay service
  - Applied to both payload and IP header
AH Modes

Transport Mode

Tunnel Mode
Mutable & Immutable

• Some fields of IP header can be modified in transit,
  – the value of the field is set to zero for purposes of the MAC computation.
    • TTL, must be decremented by every router
    • Hop counts.

• If a field is mutable, but its value at the (IPsec) receiver is predictable, then that value is inserted into the field for purposes of the MAC calculation.
  – Destination address in “source routing”
    • Source specifies the route - so the address changes but the destination address is known
Internet Security Association and Key Management Protocol (ISAKMP)

- ISAKMP defines procedures and packet formats to establish, negotiate, modify and delete Security Associations (SA).

- ISAKMP supports the negotiation of SAs for security protocols at all layers of the network stack (e.g., IPSEC, TLS etc.).

- ISAKMP is distinct from key exchange protocols
  - separating the details of security association management (and key management) from the details of key exchange.

- Better security requires authentication and key exchanges to be combined
Negotiation phases

- Phase 1: two entities such as ISAKMP servers agree on how to protect further negotiation traffic.
  - establishing ISAKMP SA.

- Phase 2: ISAKMP SA is used for security associations for other protocols such as IPSEC
Internet Key Exchange (IKE)

• A protocol for mutual authentication and establishing shared key for IPsec SA

• Two phases:
  – Phase 1: Mutual authentication and establishment of session keys between two identities
    • It’s known as the ISAKMP SA, or IKE SA.

  – Phase 2: using the keys from phase 1, multiple phase 2 SA between the two identities
IKE Phase 1

• Phase-1 can be two types, called *modes*:
  – Main mode:
    • six messages
    • mutual authentication and session key establishment
    • additional functionalities
      – e.g. hide endpoint identifiers
  – Aggressive mode:
    • three messages
    • mutual authentication and session key establishment
IKE Phase 1 – Main Mode

1. Alice → Bob: C1, *Crypto suites I support*
2. Bob → Alice: C1,C2, *Crypto suite I choose*
3. Alice → Bob: C1,C2, $g^a \mod p$
4. Bob → Alice: C1,C2, $g^b \mod p \Rightarrow Alice$ and Bab have $K = g^{ab}$
5. Alice → Bob: C1,C2, $Enc( Alice, \text{ “Proof” } Alice; K )$
6. Bob → Alice: C1,C2, $Enc( Bob, \text{ “Proof” } Bob; K )$

All cryptographic algorithms can be negotiated.
IKE Phase 1 – Aggressive Mode

1. Alice → Bob: $Alice, g^a \mod p$, crypto proposal
2. Bob → Alice: $g^b \mod p$, crypto choice, "proof" I’m Bob
3. Alice → Bob: proof I’m Alice

- Alice proposes the security parameter, and use them in message 1.
  - Unclear what if Bob refuse

- Message 2 and 3 is to show DH value is known, the secret identifying Alice (or Bob) is known
Key Types

• Three types of keys:
  – Pre-shared
  – Public key pairs - encryption
  – Public key pairs- signature

• Proof of identity is different for each key type
  – Proving the sender knows the key associated with an identity
  – A ‘proof’ is some hash of the key, DH values, nonces, crypto choices offered and cookies
Which to Choose

We described three protocols that add security to network communication. Which is ‘the best’?

• Depends on situation

• Use application-level security (SSL or SSH) when
  – Few applications require security
  – Environment doesn’t provide security
  – Application needs to be authenticated
Which to Choose

• Use IPSec when
  – Many applications require security
  – Application itself doesn’t provide it
  – Protect packet itself
  – Hide destination (tunnel mode)

➢ Often used for VPN