

CMPUT 299 (B1) Winter 2008

Security in a Networked World

Higher Layer Protocols

yannis@cs.ualberta.ca

e-mail

- ▶ Origins:
 - ▶ The SMTP protocol was designed without security in mind. There are reasons why SMTP is still a “weak” protocol. Somehow users with no prior interaction have to be able to communicate via e-mail,
 - ▶ An SMTP session can be carried out in printable text with a server (`telnet smtpserver 25`)
 - ▶ Commands: HELO, MAIL FROM, RCPT TO
 - ▶ Revealing commands: VRFY / EXPN / (DEBUG)
 - ▶ SMTP server implementations (e.g., *sendmail*) have a long history of being complex, vulnerable and difficult to configure properly.
 - ▶ Note: mail delivery requires file access privileges

e-mail (cont'd)

- ▶ Part of the problem is SMTP's legacy of being a store-and-forward protocol.
 - ▶ Intermediate servers can store a message (not from their user, neither intended for their users) to forward it further.
 - ▶ An echo of times past when mail forwarding was performed on occasional basis, but the principle is still useful today.
 - ▶ Servers that act as non-discriminating “conveyors” of other user's e-mails are known as *open relays*.
 - ▶ Most organization limit the role of their SMTP servers to move outbound (inbound) e-mail that only comes from (is destined to) their users.
 - ▶ Open relays are usually blacklisted.

e-mail contents

- ▶ MIME (Multipurpose Internet Mail Extensions) is a standard to encode and convey primarily non-ASCII content over SMTP.
 - ▶ MIME is powerful:
 - ▶ Content-Type: Message/External-Body;
 - ▶ Retrieves via a separate (non-SMTP) mechanism the contents from possibly a remote location.
 - ▶ A message can be split into fragments
 - ▶ When the attached content is huge by SMTP single-message expectations. The same mechanism can be used to evade virus checking schemes that decode MIME attachments and scan them.
 - ▶ The real culprit can be the attachment
 - ▶ Executables, or some scripting language, even PostScript. (Note that PDF files are also a matter of concern, as they can contain JavaScript).

Remote Procedure Call (RPC)

- ▶ RPCs are the network-wide generalization of function calls. Arguments fly out, results fly in.
 - ▶ Identifying the function to be called is accomplished by compiling in user programs “stub” code generated by an interface description language.
 - ▶ The stub in the client code will perform the remote call(s) to the server(s) by (un)marshaling (output)input arguments.
 - ▶ How does the client know what port is the server is listening? (Services not necessarily tied to “low” ports.)
 - ▶ An RPC services directory provides the mapping in the form of a “portmap” service at a well known port.
 - ▶ The request first looks up the service in the portmap to determine whether the service runs and on which port (try rpcinfo to see) and then issues the request to the given port.

RPC Liabilities

- ▶ RPC was initially built on top of weak authentication schemes, but has gradually adopted stronger ones.
 - ▶ Services are “registered” and “unregistered” by communicating with the portmapper (DOS prone).
 - ▶ Usually the portmapper provides a list of the registered services just for the asking to hosts within one subnet or organization.
 - ▶ Allows indirect calls (originally meant for performance improvement) with trust implications.
- ▶ Network Information Services on top of RPC
 - ▶ NIS runs over RPC and provides support for file/database lookup services (incl. password hashes!)

Remote & Network File Systems

- ▶ NFS: stateless server, state at the client
 - ▶ Individual authentication of each request.
 - ▶ File handles are opaque pointers provided to the client that identify a file (for the server's purposes).
 - ▶ A file handle to the root directory of the file system is acquired at mount time.
 - ▶ NFS inherits the logic of Unix's user and group IDs and assumes they are consistently used across the machines that mount the server(s).
 - ▶ NFS is exceptionally listening on port 2049 but additional services around NFS register with portmap
 - ▶ Client machines are essentially at the mercy of an attacker taking over control of the server.

File Transfer Protocols

- ▶ Trivial File Transfer Protocol (TFTP)
 - ▶ No authentication whatsoever. Still used for acquiring boot images for diskless devices.
- ▶ File Transfer Protocol (FTP)
 - ▶ Separation of data and control connections.
 - ▶ Data connection port indicated either via PORT command sent from client to server or requested by the PASV command to the server (the server complies and responds).
 - ▶ PORT command requires the firewall's attention and the addition of an inbound rule exception (for the duration of the transfer). Thus, a compromised applet can indirectly trigger the firewall to open a hole for attack to another port/host!
 - ▶ Anonymous FTP (privileges issues, incoming files)

Location of Security in the Protocol Stack

- ▶ Application layer (S/MIME, PGP, shttp)
- ▶ Transport layer (SSL/TLS, ssh?)
- ▶ Network layer (IPSec, IKE)
- ▶ Link layer (802.10)
- ▶ Physical layer (spread spectrum)

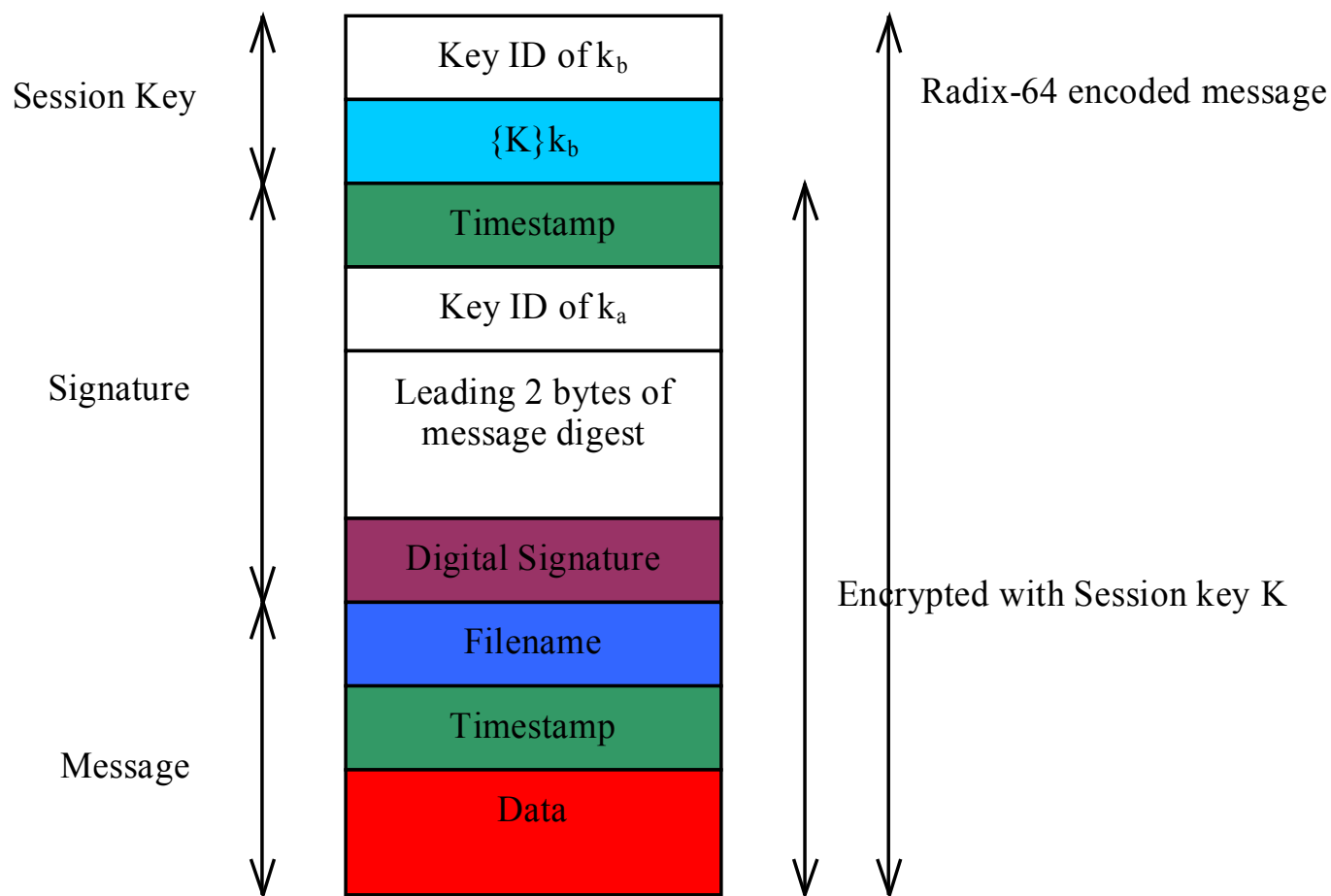
What is “protected” at each layer?

(Note that a process can have many connections, a user can run many processes, a host can have many users, a LAN may have many hosts, a physical channel can support many LANs.)

Application Layer Example: PGP (E-Mail)

- ▶ Assume A knows B's public key, k_b
- ▶ A generates a pseudorandom “session”/message key K which is used as encryption key to a symmetric encryption scheme (3DES, IDEA etc.)
- ▶ A encrypts the session key using, e.g. RSA.
- ▶ A can sign the message using RSA and its private key.
- ▶ Note that this is not an “interactive” or “real-time” application of public key cryptography.
- ▶ The messages, once produced could be delivered after a substantial delay.

Example: PGP E-Mail Message Format (Approximately)



S/MIME is not very different

Network Layer or Transport Layer?

- ▶ Secure Socket Layer (SSL)
 - ▶ Requires some changes to applications but does not need modification of the protocol stack. Usually implemented in a library that can be used by applications.
 - ▶ Mostly used for server authentication.
- ▶ IPsec
 - ▶ Applications need not be modified but the protocol stack is modified, i.e., kernel-level IPsec support.
 - ▶ Mostly used for authenticating and encrypting node-to-node communication. (NOT geared to authentication of users – something else needs to handle this part)

SSL

- ▶ Distinction of connection (transient) and session (shared among connections).
- ▶ Session setup is expensive, and usually many connections are used between the same endpoints (think of https transfers).
- ▶ A session state is identified by (a) an ID, (b) the X509.c3 cert of the peer, (c) compression mode, (d) cipher spec (encryption and hash), (e) master secret, (f) resumability indicator.
- ▶ A connections parameters are (a) keys for encryption (client & server), (b) keys to MAC functions (client & server), (c) seq. #, ...

SSL Modus Operandi (“Record Protocol”)

- ▶ Chop application data to fragments.
 - ▶ Compress (optionally) each fragment.
 - ▶ Add MAC to each (compressed) fragment.
 - ▶ Encrypt the result using symmetric encryption.
 - ▶ Prepend SSL record header.
-
- ▶ The header identifies the disposition of the data, as well as the version number and the length.

SSL Types of Messages

- ▶ Change Cipher Spec Protocol
- ▶ Alert Protocol
- ▶ Handshake Protocol
- ▶ Application Data Protocol (the “real” data)
- ▶ Many of the important tasks are carried out by the handshake protocol:
 - ▶ Version negotiation
 - ▶ Cipher Suite supported
 - ▶ Nonces (random values used during key exchange)
 - ▶ Session ID
 - ▶ Compression Methods

SSL Cipher Suites

- ▶ Description of how the key exchange is to take place. The key will be subsequently used for a symmetric cipher.
 - ▶ RSA or varieties of DH
- ▶ Cipher Specifications:
 - ▶ Available Cipher Algorithms and
 - ▶ Available MAC Algorithms
- ▶ Note that a server responds with a similarly structured message (but choosing from the options offered) plus *its certificate* (or chain of certificates). A client certificate may or may not exist.

SSL Connections

- ▶ The end result of the SSL handshake is the creation of a master secret.
- ▶ Connection parameters are derived from the master secret by hashing it into a sufficient number of bytes for the required parameter.

IPsec

- ▶ IPsec is defined as an assortment of protocols.
- ▶ The two main ones are:
 - ▶ Authentication Header (Protocol) = AH
 - ▶ Encapsulating Security Payload (Protocol) = ESP
- ▶ As the name suggests AH is meant only for authentication and not confidentiality of IP payloads.
- ▶ A particular feature of IPsec: any required key management is defined outside the context of the AH and ESP protocols. AH and ESP are in essence packet content encapsulation schemes. One needs IKE or something similar for keys.

IPsec Security Associations (SAs)

- ▶ An SA is a one-way relationship between a sender and receiver for security purposes.
 - ▶ Not that illuminating, right?
- ▶ Think of SAs as “rules” linking IPsec parameters to a particular destination (e.g., IP address, port, etc. so called “SA selectors”).
- ▶ One little caveat: the sender must inform the receiver of what it had applied to the transformation of the data as per the Security Association so the receiver can handle it. This is conveyed (in a locally significant manner) by the Security Parameter Index (SPI).

IPsec AH vs ESP

- ▶ AH supports data integrity and authentication of IP packets. Useful, e.g., in router advertisements and updates, ICMP messages, etc. and of course for application traffic (but security conscious users frequently also want encryption.)
- ▶ ESP supports confidentiality by encrypting the payload of the higher layer protocols. ESP also includes authentication.
- ▶ A problem: if we encrypt (or just authenticate) the higher layer protocol content, we still reveal the IP addresses of the endpoints (they must be in the clear to get routed!). Solution: use a “tunnel” mode to encrypt application payload and IP header inside another IP packet (with different IP source and destination address). This is done, e.g., to link corporate firewalls over the Internet.