Email and Web Security

Computer Security Lecture 14

Mike Just[1]

School of Informatics
University of Edinburgh

4th March 2010

[1] Based on original lecture notes by David Aspinall

Outline

Secure Email: PGP and S/MIME

Issues of trust

Web security: transport

X.509

Email infrastructure security

- Email, the first widely available Internet service, has a very simple backbone. **SMTP** (*Simple Mail Transport Protocol*) uses plain-text commands in a telnet session, with little or no authentication. You can type them directly. with no authentication Moreover, some delivery agents like POP3 use plaintext and clear passwords. Secure IMAP is better.
- This is the reason that **email is trivially forgeable**. Moreover, most email is sent as plaintext which provides **no confidentiality**.
- Email today is typically stored on private organizational mail servers, or at Internet or email service providers (e.g., Google).
- Email clients are either specialized for email (e.g., MS Outlook), or browser-based.

Email infrastructure security . . .

- *Server attacks*. Sending **spam** through **open relays** (accept mail from anywhere), also **gaining access** to systems. Most infamous is `sendmail`, which allowed command execution: exploited by Morris Internet Worm in 1988. Solutions: run server in restricted environment; use application gateway.
- *Client attacks*. Plain text poses no direct threat. Many virus warnings (e.g., "Good Times") are hoaxes or themselves the virus. But **macro languages** and **active content** have led to serious email-transmitted viruses and Trojans. Also phishing.

Secure Email

- Two good PK models supported by email applications: **S/MIME** (*Secure Multipurpose Internet Mail Extension*) and **PGP** (*Pretty Good Privacy*). Many less-good mechanisms.
- Same general technique:

$$A \to B : \underbrace{\{M\}_K,}_{\text{encrypted message}} \quad \underbrace{\{K\}_{K_b},}_{\text{encrypted session key}} \quad \underbrace{\mathbf{S}_A(h(M),T)}_{\text{digital signature}}$$

  Alice makes one-time session key K and encrypts email M with it. She encrypts the session key with Bob's public key K_b (sometimes called a *digital envelope*). Optionally, she includes a signature, by signing a hash of the message $h(M)$ and a timestamp T. Sometimes $\{M,T\}_K$ is sent (replay).
- For multiple recipients include multiple envelopes.

S/MIME

- Newer than PGP, but standardization began sooner, in RSA labs. Built into email clients of Netscape and Microsoft, amongst others.
- S/MIME uses **X.509 personal certificates**, signed by a certification authority, using a **trust hierarchy** model. Usually certs cost money.
- S/MIME uses the same PKI (*Public Key Infrastructure*) as SSL. Integrated email clients of web browsers implement it hand-in-hand.
- Organisations can implement their own internal private, or better, *closed* PKIs.

S/MIME . . .

- Pros: same infrastructure as SSL, use of hierarchical trust model more appropriate in some circumstances.
- Cons: Requires separate process to acquire (and pay for) certificate.
- Has open source implementation in the **OpenSSL Project**. See Mozilla's PKI page for details of other open-source PKI software.

PGP

- PGP has a venerable history. Invented by Phil Zimmerman in 1991, strong believer in privacy rights. Made available as freeware. Source code published in a book in 1995 to circumvent US export restrictions: a team used OCR to reconstruct an international version of the program, PGPI.
- PGP is supported by mail client plugins and local proxies. Crypto specs and file formats are now standardised in **OpenPGP**, a developer consortium and IETF Working Group [RFC 2440].
- **OpenPGP public keys** are hosted on a network of PGP keyservers, and can be countersigned, using PGP's **web-of-trust** model, without TTPs.
- Messages in OpenPGP (ASCII) and **PGP/MIME**.
- Pros: open source code scrutinized for years; not limited to email. Some PGP products have NIST security certification. Cons: trust model not suited to commercial application.

Trust Structures

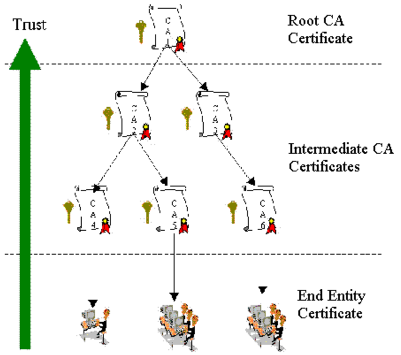
Hieararchy of trust

- Original PEM (*Privacy Enhanced Mail*) project planned a single hierarchy with the (**Internet Policy Registration Authority**) at its root.
  - This didn't happen. Instead, a forest of CAs evolved.
  - Browsers have certificates of many root CAs built-in.
  - Revocation is by CA-supplied **Certificate Revocation Lists** (CRLs), cf. credit card hot-lists. Emerging alternative is **Online Certificate Status Protocol** (OCSP) for real-time checks.

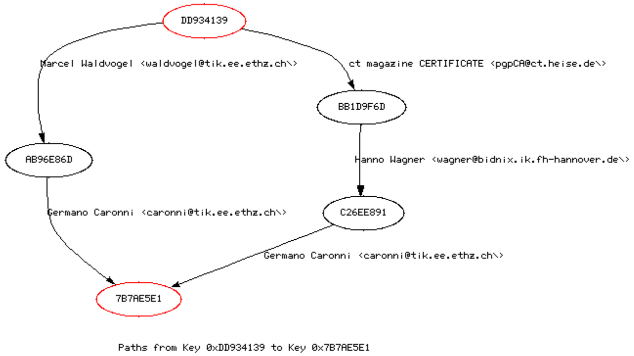
Web of Trust

- Individuals assign trust values to public keys by signing them.
- Should users allow transitive trust? Maybe yes, maybe no: global advantage in trust spread widely, but risk as it gets further away.
- Signing: social contract and PGP "signing parties".
- Revocation is by owner issuing a revocation certificate, to revoke a signature on a key.

Hierarchy of Trust



Web of Trust


DD934139
Marcel Waldvogel <waldvogel@tik.ee.ethz.ch\>
ct magazine CERTIFICATE <pgpCA@ct.heise.de\>
BB1D9F6D
AB96E86D
Hanno Wagner <wagner@bidnix.ik.fh-hannover.de\>
Germano Caronni <caronni@tik.ee.ethz.ch\>
C26EE891
Germano Caronni <caronni@tik.ee.ethz.ch\>
7B7AE5E1
Paths from Key 0xDD934139 to Key 0x7B7AE5E1

See `http://www.rubin.ch/pgp/weboftrust.en.html`

Other points of trust

- Many **interface issues**.
  - Is your sent (or received) encrypted mail stored locally unencrypted or encrypted? (If the latter, you must be on the recipient list to decrypt it!)
  - How are your private keys stored? Are they protected? Can they be extracted and shared with other applications?
- The browser and mail client implementations are critical: they are trusted to invoke the crypto operations securely, as claimed.
- **Revocation check**: ideally, email client ought to allow CRL or key-server check to see if a public key has been revoked.

SSL/TLS

- The **SSL** (*Secure Sockets Layer*) protocol provides security on top of the TCP/IP transport layer, below application-specific protocols. Each connection needs a dedicated TCP/IP socket. But each SSL session can allow multiple underlying connections (reduces keying overhead).
- Provides authentication, confidentiality, integrity. Also has built-in data compression. Layers: *handshake protocol* and *record protocol*.
- Commonly used on web for secure communications with a web server, in the http-over-SSL **`https`** protocol. Usually TCP port 443 on the server.

SSL/TLS . . .

- In `https`, *everything* is encrypted (URL requested, HTTP header, form contents, cookies, as well as web page itself). Only thing undisguised is connection to particular server.
- **But** not everything relies upon SSL for protection. Some view the additional overhead as too costly.
- Numerous other SSL-enhanced protocols also take advantage of SSL (SSLtelnet, SSLftp, stunnel).

SSL History

- Introduced 1994 by Netscape Navigator (major selling point).
- Competing S-HTTP launched at same time by CommerceNet group. S-HTTP designed to work with Web only. Initially had to be purchased in modified NCSA Mosaic. S-HTTP sunk.
- Microsoft's competing attempt **PCT** was introduced in first version of Internet Explorer. MS backed down after release of SSLv3.
- Versions of SSL:
  - **SSL v1** used internally in Netscape (flawed)
  - **SSL v2** in Navigator 1.0–2.x. Had MIM attacks, buggy random number generators. 40-bit encryption was broken.
  - **SSL v3**, 1996. Navigator 3.0, IE 3.0. Added fixes and more features, including Diffie-Hellman anonymous key-exchange.
  - **TLS v1**, 1999. Close to SSLv3.

SSL cipher suites

Client and server negotiate a **cipher suite**. Details of possibilities depend on version of SSL, as well as which suites supported by client or server.

Authentication	v2	RSA public keys & X.509 certificates
	v3	anonymous Diffie-Hellman key-exchange
Encryption	v2	40-bit RC2, RC4 ("export grade")
	v3	56-bit DES-CBC, 128-bit RC2, RC4, 3DES CBC 168-bit
MAC	v2	MD5
	v3	SHA

Some browsers let you specify which of their supported cipher suites you're willing to use.
The IETF working group are looking at future extensions for TLS, including supporting the use of OpenPGP keys as well as X.509 certificates.

SSL handshake protocol outline I

Three phases: **hello**, **key agreement**, **authentication**.

1. Client hello	$A \rightarrow S$:	$A,\ A\#,\ N_a,\ CipherPreferences$
2. Server hello	$S \rightarrow A$:	$S,\ S\#,\ N_s,\ CipherChoice$
3. Server certificate	$S \rightarrow A$:	S, CS
4. Pre-master secre.	$A \rightarrow S$:	$\{ K_0 \}_{K_s}$
5. Client finished	$A \rightarrow S$:	$\{ Finished, MAC(K_1, DataSoFar) \}_{K_{as}}$
6. Server finished.	$S \rightarrow A$:	$\{ Finished, MAC(K_1, DataSoFar) \}_{K_{sa}}$

Commentary:

1. Client hello: Alice sends name, session ID, nonce, cipher prefs.
2. Server hello: server choose best cipher suite, sends own data.
3. Server sends certificate *CS* with public key, which Alice can check. (Server may ask Alice for certificate here, or may choose anonymity)

SSL handshake protocol outline II

Three phases: **hello**, **key agreement**, **authentication**.

1. Client hello	$A \rightarrow S$:	$A,\ A\#,\ N_a,\ CipherPreferences$
2. Server hello	$S \rightarrow A$:	$S,\ S\#,\ N_s,\ CipherChoice$
3. Server certificate	$S \rightarrow A$:	S, CS
4. Pre-master secret	$A \rightarrow S$:	$\{ K_0 \}_{K_s}$
5. Client finished	$A \rightarrow S$:	$\{ Finished, MAC(K_1, DataSoFar) \}_{K_{as}}$
6. Server finished.	$S \rightarrow A$:	$\{ Finished, MAC(K_1, DataSoFar) \}_{K_{sa}}$

Commentary:

4 Alice computes a 48-byte *pre-master secret key* K_0, which sends to the server encrypted under the server's public key.

5,6 Alice and server each compute six shared secrets, 3 for each direction. First, $K_1 = h(K_0, N_a, N_s)$ is the *master secret*. Then K_{as} and K_{sa} are the symmetric cipher secret keys (e.g., DES keys) used for encryption thereafter. The second key is used for the MAC. The third key is an IV used to initialize the symmetric cipher.

SSL handshake protocol outline III

- The handshake protocol establishes an **SSL session**;
- Alternatively, it allows **resumption** of an on-going session in the first step, if Alice's session id *A#* is non-zero and the server agrees to resume the session by responding with the same value.
- This allows both resuming an SSL communication and opening another connection without undergoing key-exchange and authentication again.

X.509 Standard

- X.509 is an ITU standard (*International Telecoms Union*, formerly the CCITT), part of the X.500 series which specifies a directory service.

  Recommendation X.509
  OSI — The Directory: Authentication Framework.

  X.509 specifies a PKI. Version 3 published 1997.
- **X.509 certificates** have a specification in ASN.1, but it's open to some interpretation. This has lead to various "profiles" which pin down some of the choices.
- Examples of profiles: US **Federal PKI**, various other governments, and IETF **PKIX**.
- PKIX is used for S/MIME and SSL. See

  `http://www.ietf.org/html.charters/pkix-charter.html`

PKIX Distinguished names

- An X.500 *distinguished name* (DN) is a list of specific names each with an attribute, which specifies a path through an X.500 directory.
- X.500 presumed every subject in the world would have a globally unique DN. Not practical in reality: no single entity is trusted by everybody (one of the reasons PEM failed). In PKIX, names have **local scope** (like DNS names and IP numbers).
- Standard attribute types are defined in X.520, PKIX requires that implementations handle some, e.g.:

  common name
  organizational unit
  organization
  state or province name
  country

  I am represented as CN=*David Aspinall*, OU=*School of Informatics*, O=*University of Edinburgh*, C=*Scotland UK*.

X.509 Certificates

X.509 certificates have 10 fields:

version	*v1, v2, or v3*
serial number	*unique amongst certificates issued by a CA*
signature alg ID	*identifies signature algorithm*
issuer	*X.500 DN*
validity	*[start,end] times in UTC (2 digit yr) / generalised time.*
subject	*X.500 DN*
PK info	*algorithm, parameters and key material*
issuer ID	*bitstream added in v2 to uniquify names (in case of DN reuse)*
subject ID	*ditto*
extensions	*added in v3, various extra information*
signature value	*the signature proper:* *signed hash of fields 1 to 10*

Official field name of "signature value" is "Encrypted"

Future: Identifier-based PKs?

- PKIs of either trust model involve vast and unwieldy chains of trust and certifications. Big expense to set up and maintain, effort to retrieve certificates before communications, and verify trust chains.
- Dream future: **Identifier-based PKC**. A public key is derived from a user's identifying information: Bob's public key is derived from his e-mail address, for example. Certification implicit.
- To read messages, Bob must verify his identity (once) to an authority to obtain the private key corresponding to his public key.
- Big advantage: no need to fetch public key securely or check a trust chain. The relationship between public-private key pairs is based on a secret held by the authority that issues private keys.
  - For more, see CESG's ideas on ID-PKC at:

    `http://www.cesg.gov.uk/site/ast`

    along with their pilot for HMG PKI...

References

Jalal Feghhi, Jalil Feghhi, and Peter Williams. *Digital Certificates — Applied Internet Security*.
Addison-Wesley, 1999.

Aviel Rubin, Daniel Geer, and Marcus J Ranum. *Web Security Sourcebook*.
John Wiley & Sons, 1997.

Lincoln D Stein. *Web Security — A Step-by-Step Reference Guide*.
Addison-Wesley, 1998.