Security building blocks: protocols

TkL Markus Peuhkuri

2007-03-27

Lecture topics

- Basic security protocols
- Security protocols
- Security problems in protocols
- After this lecture, you should
  - understand where to locate encryption
  - have basic idea of Kerberos
  - know benefits and problems of PKI

What are security protocols needed for

- Exchanging session keys
- Authenticating
- Securely transfering data

Where to locate encryption

- Link layer
  - all communication protected on protected links
  - intermediate nodes must be trusted
  - popular on wireless links: great need, lower speeds
  - problems on high-speed links: many algorithms are not suitable for a parallel processing
  - GSM, WEP, PPP Encryption[7]
- Network layer
  - end-to-end encryption (if not a tunnel mode)
  - all communication between hosts protected
  - OS modifications needed, key management
  - applications may work as is
  - IPSec
- Transport layer
  - underlying protocol provides retransmissions
  - applications may need to be adapted
  - faster to deploy
  - transmission layer DoS
  - TLS

- Application layer
  - application-specific optimisation
  - must be implemented for each application
  - possibly protects data also when not in transit
  - faster to deploy
  - PGP, S/MIME

Generating keys

- (Session) keys must not be predictable
  - Netscape 1.1 used only the time of day, the process ID, and the parent process ID to generate SSL keys
- True randomness difficult in computers
- Pseudo-random numbers $x_{n+1} = (1103515245x_n + 12345) \bmod 2^{31}$, $x_1 = seed$[1]
- Real-world sources
  - quantum physics `http://www.randomnumbers.info/`
  - time measurements from real-world events
    - keyboard codes and intervals
    - mouse movements
    - interrupts (disk, network)
- Pool of randomness (in Linux)
  - source for cryptographic hash function[2]

`/dev/random` does not return more than estimated amount of randomness

`/dev/urandom` returns as much as wanted
⇒ maybe vulnerable to hash analysis

Key exchange with symmetric ciphers

- Use of trusted third party $\mathcal{C}$
  1. $\mathcal{A} \rightarrow \mathcal{C} : \{\text{request for session key to Bob}\}\mathcal{K}_{\mathcal{A}}$
  2. $\mathcal{C} \rightarrow \mathcal{A} : \{\mathcal{K}_{\mathcal{A},\mathcal{B}}\}\mathcal{K}_{\mathcal{A}} || \{\mathcal{K}_{\mathcal{A},\mathcal{B}}\}\mathcal{K}_{\mathcal{B}}$
  3. $\mathcal{A} \rightarrow \mathcal{B} : \{\mathcal{K}_{\mathcal{A},\mathcal{B}}\}\mathcal{K}_{\mathcal{B}}$
- Problems
  - $\mathcal{B}$ does not know with whom to talk
  - replay attacks possible

Needham-Schroeder [8]

1. $\mathcal{A} \rightarrow \mathcal{C} : \{\mathcal{A}||\mathcal{B}||rand_1\}$
2. $\mathcal{C} \rightarrow \mathcal{A} : \{\mathcal{A}||\mathcal{B}||rand_1||\mathcal{K}_{\mathcal{A},\mathcal{B}}||\{\mathcal{A}||\mathcal{K}_{\mathcal{A},\mathcal{B}}\}\mathcal{K}_{\mathcal{B}}\}\mathcal{K}_{\mathcal{A}}$
3. $\mathcal{A} \rightarrow \mathcal{B} : \{\mathcal{A}||\mathcal{K}_{\mathcal{A},\mathcal{B}}\}\mathcal{K}_{\mathcal{B}}$
4. $\mathcal{B} \rightarrow \mathcal{A} : \{rand_2\}\mathcal{K}_{\mathcal{A},\mathcal{B}}$
5. $\mathcal{A} \rightarrow \mathcal{B} : \{rand_2 - 1\}\mathcal{K}_{\mathcal{A},\mathcal{B}}$

[1] Not a very good generator: period of 2^{31} and lowest bits are not very random.
[2] strong mixing function

- *rand*s used to defeat replay
  - must not be repeated
  - nonce[3]
- If a session key is compromised, it is possible to masquerade using old exchange
  - use timestamps $\mathcal{T}$ [2]
    $\Rightarrow$ need for accurate and *secure* clocks
  - use message numbers [9]
    $\Rightarrow$ keep track with numbers with each party

Kerberos 5 [6]

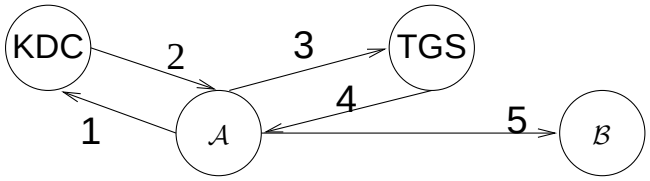
- Uses Needham-Schroeder with timestamps
- Provides

  secure passwords are not stored on a disk or transmitted over a network
  single-sign-on as the user logs in only once for all resources
  trusted third-party a central authentication server
  mutual authentication as the services are authenticated too
- Assumptions
  - denial of service attacks are not handled by Kerberos
  - principals keep their keys secret
  - passwords are of sufficient strength
  - loosely synchronised clocks
  - principal identifiers are not recycled

Messages in Kerberos 5

Components: KDC = Key Distribution Center, TGS = Ticket Granting Service

- Uses tickets $T_{\mathcal{A},\mathcal{B}} = \mathcal{B}||\{\mathcal{A}||\text{Address}_{\mathcal{A}}||\mathcal{T}_{\text{valid}}||\mathcal{K}_{\mathcal{A},\mathcal{B}}\}\mathcal{K}_{\mathcal{B}}$
  1. $\mathcal{A} \rightarrow \mathsf{KDC} : \{\mathcal{A}||\mathsf{TGS}\}$
  2. $\mathsf{KDC} \rightarrow \mathcal{A} : \{\mathcal{K}_{\mathcal{A},\mathsf{TGS}}\}\mathcal{K}_{\mathcal{A}}||T_{\mathcal{A},\mathsf{TGS}}$
  3. $\mathcal{A} \rightarrow \mathsf{TGS} : \{\mathcal{B}||Auth_{\mathcal{A},\mathsf{TGS}}||T_{\mathcal{A},\mathsf{TGS}}\}$
  4. $\mathsf{TGS} \rightarrow \mathcal{A} : \mathcal{A}||\{\mathcal{K}_{\mathcal{A},\mathcal{B}}\}\mathcal{K}_{\mathcal{A},\mathsf{TGS}}||T_{\mathcal{A},\mathcal{B}}$
  5. $\mathcal{A} \rightarrow \mathcal{B} : \{Auth_{\mathcal{A},\mathcal{B}}||T_{\mathcal{A},\mathcal{B}}\}$[4]
  6. $\mathcal{B} \rightarrow \mathcal{A} : \{\mathcal{T}+1\}\mathcal{K}_{\mathcal{A},\mathcal{B}}$
     The last step is optional.



[3] Designating a lexical item formed for use on a specific occasion.
[4] *Auth* includes recent timestamp encrypted with session key.

Kerberos weakness

- A need for synchronised clocks
  - authenticator valid for 5 minutes
- Known plaintext attacks
  - fixed fields in messages
  - no "salt" used
- Consequences of compromises
  - root on KDC
    - *all credentials compromised*
  - Kerberos administrator credentials
    - all credentials compromised
  - root on a server machine
    - all services on the machine. If there is some distributed service, such as Andrew File System (AFS) that shares service principals across multiple machines, all of those are compromised. Note that no client credentials are exposed expect for brute-force attack like if captured from a network.
    - can impersonate as the server
    - all traffic between clients and the server can be decrypted over the period that same key has been used
  - root on a client machine
    - cached tickets on that machine
    - passwords entered into the machine when compromised. The passwords entered earlier are not vulnerable as they are not stored after $\mathcal{K}_{\mathcal{A},\mathsf{TGS}}$ is received.
  - user credentials
    - the tickets can be used for their lifetime
    - if the password is known, then the account is compromised

Public key exchange

- Simple in principle: $\mathcal{A} \rightarrow \mathcal{B} : \{\mathcal{K}_{\mathcal{A},\mathcal{B}}\}\mathcal{K}_{\mathcal{B}p}$
- However, $\mathcal{A}$ not authenticated
  $\Rightarrow \mathcal{A} \rightarrow \mathcal{B} : \{\{\mathcal{K}_{\mathcal{A},\mathcal{B}}\}\mathcal{K}_{\mathcal{A}s}\}\mathcal{K}_{\mathcal{B}p}$
- Where to get $\mathcal{K}_{\mathcal{A}p}$, $\mathcal{K}_{\mathcal{B}p}$?
  - a directory would be mother of all targets
    $\Rightarrow$ something not depending integrity of a large, public access database
- Certificate: a token that binds an *identity* to a key
  - tree structure: *root* is known out-of-band, like information about root certificates are installed with browser or operating system
  - free-form chain: a different trust is placed depending on chain members

X.509

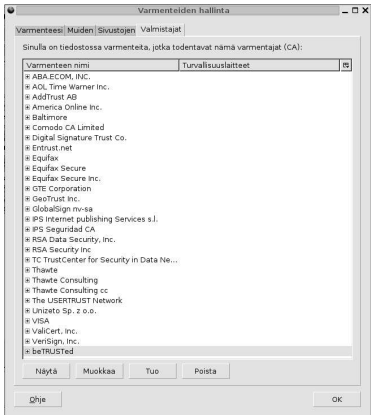
- Directory authentication Framework
- ITU X.509v3 issued 1993
- Certificate has following components
  1. version (=3)

2. serial number that is unique for issuer
3. signature algorithm
4. issuer's distinguished name
5. validity interval
6. subject's distinguished name
7. public key information
8. issuer's unique identifier
9. subject's unique identifier
10. extensions
11. signature (calculated over other components)

Ideal certificate/PKI world

- A certificate for each
  - web site
  - email sender
  - citizen
  - corporation, society, community
- Each transaction would become verifiable
- All certificates could be verified using certificate tree

Who do you trust



Obstacles in Certificates

- Bootstrap information needed
  - which CAs to select?
  - one with USD 1000 / year / server or USD 25 / year: the first with 100 % browser compatibility, the other with 95 – 99 % compatibility.
- Transferring *trust*
- Certificate revocation list (CRL)
  - similar to credit card blacklists
  - often not used
- Naming identities
  - how about namesakes

  - identities may not be unique across certificate registries
  - implement or rely on a existing registry, such as domain name system but this creates yet further trust issues.
    ⇒ complicates implementation
- Certificate information does not give enough information to build trust
- Keeping root secret key really secret

OpenPGP [1]

- Based on original PGP developed 1991 by Philip Zimmerman
- Web of trust
  - different trust levels

    undefined nothing is known
    marginal maybe valid
    complete key is valid
  - keys self-signed
  - multiple signatures, with different levels of trust on signatures

    unknown nothing is known
    none improperly signing one
    marginal understands implications and validates keys
    full as good as you
- Multiple signatures

Transport Layer Security [3]

- Provides data privacy and confidentiality
  - protocol-independent
  - supports a large selection of cryptographic algorithms
- TLS Record protocol
  - symmetric cryptology for privacy
  - reliable connection
  - runs top on a reliable byte stream (TCP)
  - encapsulates other protocols
- TLS Handshake protocol
  - peer identity can be authenticated: optional, but in normal case at least one of the peers is authenticated using asymmetric cryptography.
  - a shared secret negotiated in a secure way, protecting also from man-in-middle attack
  - reliable negotiation

TLS handshake

1. $\mathcal{C} \rightarrow \mathcal{S}$: ServerHello,
2. $\mathcal{S} \rightarrow \mathcal{C}$: *Sertificate, ServerKeyExchange, CertifacateRequest* ServerHelloDone
3. $\mathcal{C} \rightarrow \mathcal{S}$: *Sertificate*, ClientKeyExchange*CertificateVerity*, ChangeCiperSpec,Finished
4. $\mathcal{S} \rightarrow \mathcal{C}$: ChangeCiperSpec,Finished
5. $\mathcal{C} \leftrightarrow \mathcal{S}$: Application Data

Internet Key Exchange

- Used for IPSec key exchange (IKE)[4, 5]
- Security association established
  - encryption algorithm
  - hash algorithm
  - authentication method
  - Diffie-Hellman modulus
- Main and aggressive mode (IKEv1)
- Maintains security association descriptions

Key storage

- Computer memory
  - superuser can read
  - other processes by the same user
  - may end to a hard disk as a result from paging
- Special hardware on a computer
  - may provide key functions
  - no access to key
- External token
  - can be removed from the system when not in use
  - becomes a protected asset
- Smart card
  - can keep secret keys internal
  - a small system easy to evaluate for security

Summary

- Use encryption and integrity checks on right level
  - know which ones you need to trust
- Avoid protocols that need central point
- Securing one location is easier than securing multiple locations, however

References

[1] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. *OpenPGP Message Format*, November 1998. RFC 2440. URL:`http://www.ietf.org/rfc/rfc2440.txt`.

[2] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Commun. ACM*, 24(8):533–536, 1981.

[3] T. Dierks and C. Allen. *The TLS Protocol Version 1.0*, January 1999. RFC 2246. URL:`http://www.ietf.org/rfc/rfc2246.txt`.

[4] D. Harkins and D. Carrel. *The Internet Key Exchange (IKE)*, November 1998. RFC 2409. URL:`http://www.ietf.org/rfc/rfc2409.txt`.

[5] C. Kaufman and Ed. *Internet Key Exchange (IKEv2) Protocol*, December 2005. RFC 4306. URL:`http://www.ietf.org/rfc/rfc4306.txt`.

[6] J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (V5)*, September 1993. RFC 1510. URL:`http://www.ietf.org/rfc/rfc1510.txt`.

[7] G. Meyer. *The PPP Encryption Control Protocol (ECP)*, June 1996. RFC 1968. URL:`http://www.ietf.org/rfc/rfc1968.txt`.

[8] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[9] Dave Otway and Owen Rees. Efficient and timely mutual authentication. *SIGOPS Oper. Syst. Rev.*, 21(1):8–10, 1987.