Security I – retired slides

Markus Kuhn

Computer Laboratory, University of Cambridge

http://www.cl.cam.ac.uk/teaching/1213/SecurityI/

Lent 2013 – Part IB

Secure hash functions

A *hash function* $h : \{0,1\}^* \to \{0,1\}^n$ efficiently maps arbitrary-length input bit strings onto (usually short) fixed-length bitstrings such that the output is uniformly distributed (for non-repeating input values).

Hash functions are commonly used for fast table lookup or as checksums.

A *secure n-bit hash function* is in addition expected to offer the following properties:

- Preimage resistance (one-way): For a given value y, it is computationally infeasible to find x with $h(x) = y$.
- Second preimage resistance (weak collision resistance): For a given value x, it is computationally infeasible to find x' with $h(x') = h(x)$.
- Collision resistance: It is computationally infeasible to find a pair $x \neq y$ with $h(x) = h(y)$.

Secure hash functions: standards

- MD5: $n = 128$
  still widely used today, but collisions were found in 2004
  `http://www.ietf.org/rfc/rfc1321.txt`
- SHA-1: $n = 160$
  widely used today in many applications, but 2^{69}-step algorithm to find collisions found in 2005, being phased out
- SHA-2: $n = 224$, 256, 384, or 512
  close relative of SHA-1, therefore long-term collision-resistance questionable, best existing standard
  FIPS 180-3 US government secure hash standard,
  `http://csrc.nist.gov/publications/fips/`
- SHA-3: KECCAK wins 5-year NIST contest in October 2012
  no length-extension attack, arbitrary-length output,
  can also operate as PRNG, very different from SHA-1/2.
  (other finalists: BLAKE, Grøstl, JH, Skein)
  `http://csrc.nist.gov/groups/ST/hash/sha-3/`
  `http://keccak.noekeon.org/`

Secure hash functions: Merkle–Damgård construction

Fast secure hash functions such as MD5 or SHA-1 are based on a PRF $C : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ called *compression function*.

First, the input bitstring X is padded in an unambiguous way to a multiple of the compression function's input block size k. If we would just add zero bits for padding, for instance, then the padded versions of two strings which differ just in the number of trailing "zero" bits would be indistinguishable ($10101 + 000 = 10101000 = 1010100 + 0$). By padding with a "one" bit (even if the length was already a multiple of k bits!), followed by between 0 and $k - 1$ "zero" bits, the padding could always unambiguously be removed and therefore this careful padding destroys no information.

Then the padded bitstring X' is split into m k-bit blocks $X_1, \ldots, X_m$, and the n-bit hash value $H(X) = H_m$ is calculated via the recursion

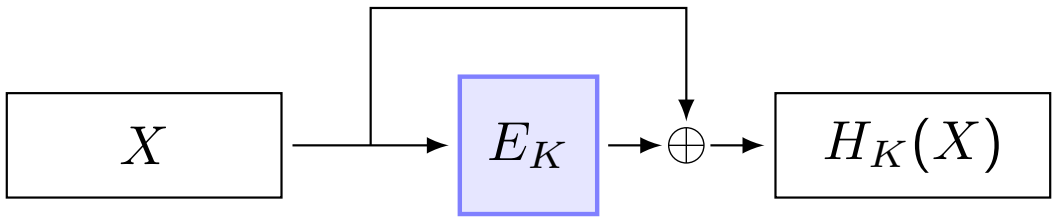
$$H_i = C(H_{i-1}, X_i)$$

where H_0 is a constant n-bit start value.

MD5 and SHA-1 for instance use block sizes of $k = 512$ bits.

One-way function from block cipher (Davies–Meyer)

A block cipher can be turned into a one-way function by XORing the input onto the output. This prevents decryption, as the output of the blockcipher cannot be reconstructed from the output of the one-way function.



Another way of getting a one-way function is to use the input as a key in a block cipher to encrypt a fixed value.

Both approaches can be combined to use a block cipher E as the compression function in a secure hash function:

$$H_i = E_{X_i}(H_{i-1}) \oplus H_{i-1}$$

Hash-based message authentication code

Hash a message M concatenated with a key K:

$$\mathrm{MAC}_K(M) = h(K, M)$$

This construct is secure if h is a pseudo-random function or is a modern secure hash function such as SHA-3.

Danger: If h uses the Merkle–Damgård construction, an attacker can call the compression function again on the MAC to add more blocks to M, and obtain the MAC of a longer M' without knowing the key!

To prevent such a message-extension attack, variants like

$$\mathrm{MAC}_K(M) = h(h(K, M))$$

can be used to terminate the iteration of the compression function in a way that the attacker cannot continue.

HMAC

HMAC is a standardized technique that is widely used to calculate a message-authentication code using a Merkle–Damgård-style secure hash function h, such as MD5 or SHA-1:

$$\mathrm{HMAC}_K = h(K \oplus X_1, h(K \oplus X_2, M))$$

The fixed padding values X_1, X_2 used in HMAC extend the length of the key to the input size of the compression function, thereby permitting precomputation of its first iteration.

`http://www.ietf.org/rfc/rfc2104.txt`

More applications of secure hash functions I

Password hash chain

$$\begin{aligned} R_0 &= \text{random} \\ R_{i+1} &= h(R_i) \quad (0 \leq i < n) \end{aligned}$$

Store R_n in a host and give list $R_{n-1}, R_{n-2}, \ldots, R_0$ as one-time passwords to user. When user enters password R_{i-1}, its hash $h(R_{i-1})$ is compared with the password R_i stored on the server. If they match, the user is granted access and R_{i-1} replaces R_i.

Leslie Lamport: *Password authentication with insecure communication*. CACM 24(11)770–772, 1981. `http://doi.acm.org/10.1145/358790.358797`

Proof of prior knowledge / secure commitment

You have today an idea that you write down in message M. You do not want to publish M yet, but you want to be able to prove later that you knew M already today. So you publish $h(M)$ today.

If the entropy of M is small (e.g., M is a simple password), there is a risk that h can be inverted successfully via brute-force search. Solution: publish $h(N, M)$ where N is a random bit string (like a key). When the time comes to reveal M, also reveal N. Publishing $h(N, M)$ can also be used to commit yourself to M, without revealing it yet.

More applications of secure hash functions II

Hash tree

Leaves contain hash values of messages, each inner node contains the hash of the concatenated values in the child nodes directly below it.

Advantages of tree over hashing concatenation of all messages:

- Update of a single message requires only recalculation of hash values along path to root.
- Verification of a message requires only knowledge of values in all direct children of nodes in path to root.

One-time signatures

Secret key: $2n$ random bit strings $R_{i,j}$ $(i \in \{0,1\}, 1 \leq j \leq n)$

Public key: $2n$ bit strings $h(R_{i,j})$

Signature: $(R_{b_1,1}, R_{b_2,2}, \ldots, R_{b_n,n})$, where $h(M) = b_1 b_2 \ldots b_n$

More applications of secure hash functions III

Stream authentication

Alice sends to Bob a long stream of messages $M_1, M_2, \ldots, M_n$. Bob wants to verify Alice's signature on each packet immediately upon arrival, but it is too expensive to sign each message individually.

Alice calculates

$$\begin{aligned} C_1 &= h(C_2, M_1) \\ C_2 &= h(C_3, M_2) \\ C_3 &= h(C_4, M_3) \\ &\cdots \\ C_n &= h(0, M_n) \end{aligned}$$

and then sends to Bob the stream

$$C_1, \text{Signature}(C_1), (C_2, M_1), (C_3, M_2), \ldots, (0, M_n).$$

Only the first check value is signed, all other packets are bound together in a hash chain that is linked to that single signature.

Secret sharing

A (t, n) secret sharing scheme is a mechanism to distribute shares $S_1, \ldots, S_n$ of a secret key S $(0 \leq S < m)$ among parties $P_1, \ldots, P_n$ such that any t of them can together reconstruct the key, but any group of $t - 1$ cannot.

Unanimous consent control – (n, n) secret sharing

- For all $1 \leq i < n$ generate random number $0 \leq S_i < m$ and give it to P_i.
- Give $S_n = S - \sum_{i=1}^{n-1} S_i \bmod m$ to P_n.
- Recover secret as $S = \sum_{i=1}^{n} S_i \bmod m$.

Can also be implemented with bitstrings and XOR instead of modular arithmetic.

Secret sharing – Shamir's threshold scheme

- Choose a prime $p > \max(S, n)$.
- Choose a polynomial

$$f(x) = \sum_{j=0}^{t-1} a_j x^j$$

  with $a_0 = S$ and random numbers $0 \leq a_j < p$ $(1 \leq j < t)$.
- For all $1 \leq i \leq n$ compute $S_i = f(i) \bmod p$ and give it to P_i.
- Recover secret $S = f(0)$ by Lagrange interpolation of f through any t points $(x_i, y_i) = (i, S_i)$. Note that $\deg(f) = t - 1$.

Lagrange interpolation:

If (x_i, y_i) for $1 \leq i \leq t$ are points of a polynomial f with $\deg(f) < t$:

$$f(x) = \sum_{i=1}^{t} y_i \prod_{\substack{1 \leq j \leq t \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

Diffie-Hellman key exchange

How can two parties achieve message confidentiality who have no prior shared secret and no secure channel to exchange one?

Select a suitably large prime number p and a generator $g \in \mathbb{Z}_p^*$ ($2 \leq g \leq p-2$), which can be made public. A generates x and B generates y, both random numbers out of $\{1, \ldots, p-2\}$.

$$A \to B: \quad g^x \bmod p$$
$$B \to A: \quad g^y \bmod p$$

Now both can form $(g^x)^y = (g^y)^x$ and use a hash of it as a shared key.

The eavesdropper faces the *Diffie-Hellman Problem* of determining g^{xy} from g^x, g^y and g, which is believed to be equally difficult to the *Discrete Logarithm Problem* of finding x from g^x and g in $\mathbb{Z}_p^*$. This is infeasible if $p > 2^{1000}$ and $p-1$ has a large prime factor.

The DH key exchange is secure against a passive eavesdropper, but not against middleperson attacks, where g^x and g^y are replaced by the attacker with other values.

W. Diffie, M.E. Hellman: New Directions in Cryptography. IEEE IT-22(6), 1976-11, pp 644–654.

ElGamal encryption

The DH key exchange requires two messages. This can be eliminated if everyone publishes his g^x as a *public key* in a sort of phonebook.

If A has published (p, g, g^x) as her *public key* and kept x as her *private key*, then B can also generate for each message a new y and send

$$B \to A: \qquad g^y \bmod p,\ (g^x)^y \cdot M \bmod p$$

where $M \in \mathbb{Z}_p$ is the message that B sends to A in this asymmetric encryption scheme. Then A calculates

$$[(g^x)^y \cdot M] \cdot [(g^y)^{p-1-x}] \bmod p = M$$

to decrypt M.

In practice, M is again not the real message, but only the key for an efficient block cipher that protects confidentiality and integrity of the bulk of the message (hybrid cryptography).

With the also widely used RSA asymmetric cryptography scheme, encryption and decryption commute. This allows the owner of a secret key to sign a message by "decrypting" it with her secret key, and then everyone can recover the message and verify this way the signature by "encrypting" it with the public key.

ElGamal signature

Asymmetric cryptography also provides digital signature algorithms, where only the owner of a secret key can generate a signatures for a message M that can be verified by anyone with the public key.

If A has published (p, g, g^x) as her *public key* and kept x as her *private key*, then in order to sign a message $M \in \mathbb{Z}_p$ (usually hash of real message), she generates a random number y (with $0 < y < p-1$ and $\gcd(y, p-1) = 1$) and solves the linear equation

$$x \cdot g^y + y \cdot s \equiv M \pmod{p-1} \tag{1}$$

for s and sends to the verifier B the signed message

$$A \to B: \qquad M,\ g^y \bmod p,\ s = (M - x \cdot g^y)/y \bmod (p-1)$$

who will raise g to the power of both sides of (1) and test the resulting equation:

$$(g^x)^{g^y} \cdot (g^y)^s \equiv g^M \pmod{p}$$

Warning: Unless p and g are carefully chosen, ElGamal signatures can be vulnerable to forgery:
D. Bleichenbacher: Generating ElGamal signatures without knowing the secret key.
EUROCRYPT '96. `http://www.springerlink.com/link.asp?id=xbwmv0b564gwlq7a`

Public-key infrastructure I

Public key encryption and signature algorithms allow the establishment of confidential and authenticated communication links with the owners of public/private key pairs.

Public keys still need to be reliably associated with identities of owners. In the absence of a personal exchange of public keys, this can be mediated via a trusted third party. Such a *certification authority* C issues a digitally signed *public key certificate*

$$\mathrm{Cert}_C(A) = \{A, K_A, T, L\}_{K_C^{-1}}$$

in which C confirms that the public key K_A belongs to A starting at time T and that this confirmation is valid for the time interval L, and all this is digitally signed with C's private signing key K_C^{-1}.

Anyone who knows C's public key K_C from a trustworthy source can use it to verify the certificate $\mathrm{Cert}_C(A)$ and obtain a trustworthy copy of A's key K_A this way.

Public-key infrastructure II

We can use the operator $\bullet$ to describe the extraction of A's public key K_A from a certificate $\mathsf{Cert}_C(A)$ with the certification authority public key K_C:

$$K_C \bullet \mathsf{Cert}_C(A) = \begin{cases} K_A & \text{if certificate valid} \\ \text{failure} & \text{otherwise} \end{cases}$$

The $\bullet$ operation involves not only the verification of the certificate signature, but also the validity time and other restrictions specified in the signature. For instance, a certificate issued by C might contain a reference to an online *certificate revocation list* published by C, which lists all public keys that might have become compromised (e.g., the smartcard containing K_A^{-1} was stolen or the server storing K_A^{-1} was broken into) and whose certificates have not yet expired.

Public-key infrastructure III

Public keys can also be verified via several trusted intermediaries in a *certificate chain*:

$$K_{C_1} \bullet \mathsf{Cert}_{C_1}(C_2) \bullet \mathsf{Cert}_{C_2}(C_3) \bullet \cdots \bullet \mathsf{Cert}_{C_{n-1}}(C_n) \bullet \mathsf{Cert}_{C_n}(B) = K_B$$

A has received directly a trustworthy copy of K_{C_1} (which many implementations store locally as a certificate $\mathsf{Cert}_A(C_1)$ to minimise the number of keys that must be kept in tamper-resistant storage).

Certification authorities can be made part of a hierarchical tree, in which members of layer n verify the identity of members in layer $n-1$ and $n+1$. For example layer 1 can be a national CA, layer 2 the computing services of universities and layer 3 the system administrators of individual departments.

Practical example: A personally receives K_{C_1} from her local system administrator C_1, who confirmed the identity of the university's computing service C_2 in $\mathsf{Cert}_{C_1}(C_2)$, who confirmed the national network operator C_3, who confirmed the IT department of B's employer C_3 who finally confirms the identity of B. An online directory service allows A to retrieve all these certificates (plus related certificate revocation lists) efficiently.

Some popular Unix cryptography tools

- `ssh [user@]hostname [command]` — Log in via encrypted link to remote machine (and if provided execute "`command`"). RSA or DSA signature is used to protect Diffie-Hellman session-key exchange and to identify machine or user. Various authentication mechanisms, e.g. remote machine will not ask for password, if user's private key (`~/.ssh/id_rsa`) fits one of the public keys listed in the home directory on the remote machine (`~/.ssh/authorized_keys`). Generate key pairs with `ssh-keygen`.

  `http://www.openssh.org/`

- pgp, gpg — Offer both symmetric and asymmetric encryption, digital signing and generation, verification, storage and management of public-key certificates in a form suitable for transmission via email.

  `http://www.gnupg.org/`, `http://www.pgpi.org/`

- `openssl` — Tool and library that implements numerous standard cryptographic primitives, including AES, X.509 certificates, and SSL-encrypted TCP connections.

  `http://www.openssl.org/`

TEA, a Tiny Encryption Algorithm

TEA is a 64-bit block cipher with 128-bit key and 64-round Feistel structure, designed at the Computer Lab by David Wheeler and Roger Needham for 32-bit processors. The aim was to find a cipher so simple that the implementation can be memorised, not maximum performance:

```
void code(long *v, long *k)
{
  unsigned long y = v[0], z = v[1], sum = 0;
  unsigned long delta=0x9e3779b9, n = 32;
  while (n-- > 0) {
    sum += delta;
    y += ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
    z += ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
  }
  v[0]=y ; v[1]=z;
}
```

ftp://ftp.cl.cam.ac.uk/users/djw3/tea.ps

Classification of operating-system security I

In 1983, the US DoD published the "Trusted computer system evaluation criteria (TCSEC)", also known as "Orange Book".

It defines several classes of security functionality required in the TCB of an operating system:

- Class D: Minimal protection – no authentication, access control, or object reuse (example: MS-DOS, Windows98)
- Class C1: Discretionary security protection – support for discretionary access control, user identification/authentication, tamper-resistant kernel, security tested and documented (e.g., classic Unix versions)
- Class C2: Controlled access protection – adds object reuse, audit trail of object access, access control lists with single user granularity (e.g., Unix with some auditing extensions, Windows NT in a special configuration)

Classification of operating-system security II

- Class B1: Labeled security protection – adds confidentiality labels for objects, mandatory access control policy, thorough security testing
- Class B2: Structured protection – adds trusted path from user to TCB, formal security policy model, minimum/maximum security levels for devices, well-structured TCB and user interface, accurate high-level description, identify covert storage channels and estimate bandwidth, system administration functions, penetration testing, TCB source code revision control and auditing
- Class B3: Security domains – adds security alarm mechanisms, minimal TCB, covert channel analysis, separation of system administrator and security administrator
- Class A1: Verified design – adds formal model for security policy, formal description of TCB must be proved to match the implementation, strict protection of source code against unauthorised modification

Common Criteria

In 1999, TCSEC and its European equivalent ITSEC were merged into the *Common Criteria for Information Technology Security Evaluation*.

- Covers not only operating systems but a broad spectrum of security products and associated security requirements
- Provides a framework for defining new product and application specific sets of security requirements (*protection profiles*)

  E.g., NSA's Controlled Access Protection Profile (CAPP) replaces Orange Book C2.
- Separates functional and security requirements from the intensity of required testing (*evaluation assurance level*, EAL)

EAL1: tester reads documentation, performs some functionality tests
EAL2: developer provides test documentation and vulnerability analysis for review
EAL3: developer uses RCS, provides more test and design documentation
EAL4: low-level design docs, some TCB source code, secure delivery, independent vul. analysis (highest level considered economically feasible for existing product)
EAL5: Formal security policy, semiformal high-level design, full TCB source code, indep. testing
EAL6: Well-structured source code, reference monitor for access control, intensive pen. testing
EAL7: Formal high-level design and correctness proof of implementation

E.g., Windows Vista Enterprise was evaluated for CAPP at EAL4 + ALC_FLR.3 (flaw remediation).
`http://www.commoncriteriaportal.org/`

Network security

> "It is easy to run a secure computer system. You merely have to disconnect all connections and permit only direct-wired terminals, put the machine in a shielded room, and post a guard at the door." — Grampp/Morris

Problems:

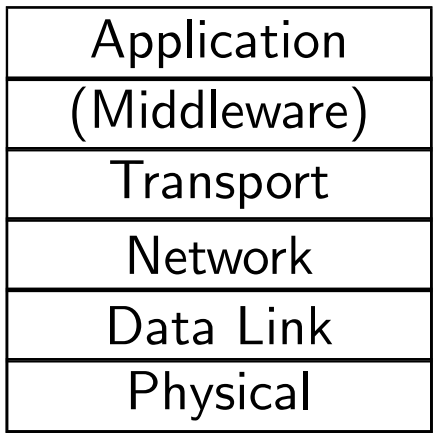
- Wide area networks allow attacks from anywhere, often via several compromised intermediary machines, international law enforcement difficult
- Commonly used protocols not designed for hostile environment
  - authentication missing or based on source address, cleartext password, or integrity of remote host
  - missing protection against denial-of-service attacks
- Use of bus and broadcast technologies, promiscuous-mode network interfaces
- Vulnerable protocol implementations
- Distributed denial-of-service attacks

TCP/IP security

TCP/IP transport connections are characterised by:

- Source IP address
- Destination IP address
- Source Port
- Destination Port

Network protocol stack:



IP addresses identify hosts and port numbers distinguish between different processes within a host. Port numbers < 1024 are "privileged"; under Unix only root can open them. This is used by some Unix network services (e.g., rsh) to authenticate peer system processes.

Example destination ports:
20–21=FTP, 22=SSH, 23=telnet, 25=SMTP (email), 79=finger, 80=HTTP, 111=Sun RPC, 137–139=NETBIOS (Windows file/printer sharing), 143=IMAP, 161=SNMP, 60xx=X11, etc.
See `/etc/services` or `http://www.iana.org/assignments/port-numbers` for more.

Address spoofing

IP addresses are 32-bit words (IPv6: 128-bit) split into a network and a host identifier. Destination IP address is used for routing. The IP source address is provided by the originating host, which can provide wrong information ("address spoofing"). It is verified during the TCP 3-way handshake:

$$
\begin{aligned}
S \to D: &\quad \texttt{SYN}_x \\
D \to S: &\quad \texttt{SYN}_y, \texttt{ACK}_{x+1} \\
S \to D: &\quad \texttt{ACK}_{y+1}
\end{aligned}
$$

Only the third message starts data delivery, therefore data communication will only proceed after the claimed originator has confirmed the reception of a TCP sequence number in an ACK message. From then on, TCP will ignore messages with sequence numbers outside the confirmation window. In the absence of an eavesdropper, the start sequence number can act like an authentication nonce.

Examples of TCP/IP vulnerabilities I

- The IP *loose source route* option allows S to dictate an explicit path to D and old specifications (RFC 1122) require destination machines to use the inverse path for the reply, eliminating the authentication value of the 3-way TCP handshake.
- The connectionless *User Datagram Protocol (UDP)* has no sequence numbers and is therefore more vulnerable to address spoofing.
- Implementations still have predictable start sequence numbers, therefore even without having access to reply packets sent from D to S, an attacker can
  - impersonate S by performing the entire handshake without receiving the second message ("sequence number attack")
  - disrupt an ongoing communication by inserting data packets with the right sequence numbers ("session hijacking")

Examples of TCP/IP vulnerabilities II

- In many older TCP implementations, D allocates a temporary data record for every half-open connection between the second and third message of the handshake in a very small buffer. A very small number of `SYN` packets with spoofed IP address can exhaust this buffer and prevent any further TCP communication with D for considerable time ("SYN flooding").
- For convenience, network services are usually configured with alphanumeric names mapped by the *Domain Name System (DNS)*, which features its own set of vulnerabilities:
  - DNS implementations cache query results, and many older versions even cache unsolicited ones, allowing an attacker to fill the cache with desired name/address mappings before launching an impersonation attack.
  - Many DNS resolvers are configured to complete name prefixes automatically, e.g. the hostname n could result in queries n`.cl.cam.ac.uk`, n`.cam.ac.uk`, n`.ac.uk`, n. So attacker registers `hotmail.com.ac.uk`.

Firewalls I

Firewalls are dedicated gateways between intranets/LANs and wide area networks. All traffic between the "inside" and "outside" world must pass through the firewall and is checked there for compliance with a local security policy. Firewalls themselves are supposed to be highly penetration resistant. They can filter network traffic at various levels of sophistication:

- A basic firewall function drops or passes TCP/IP packets based on matches with configured sets of IP addresses and port numbers. This allows system administrators to control at a single configuration point which network services are reachable at which host.
- A basic packet filter can distinguish incoming and outgoing TCP traffic because the opening packet lacks the `ACK` bit. More sophisticated filtering requires the implementation of a TCP state machine, which is beyond the capabilities of most normal routing hardware.

Firewalls II

- Firewalls should perform plausibility checks on source IP addresses, e.g. not accept from the outside a packet with an inside source address and vice versa.
- Good firewalls check for protocol violations to protect vulnerable implementations on the intranet. Some implement entire application protocol stacks in order to sanitise the syntax of protocol data units and suppress unwanted content (e.g., executable email attachments $\rightarrow$ viruses).
- Logging and auditing functions record suspicious activity and generate alarms. An example are port scans, where a single outside host sends packets to all hosts of a subnet, a characteristic sign of someone mapping the network topology or searching systematically for vulnerable machines.

Firewalls are also used to create encrypted tunnels to the firewalls of other trusted intranets, in order to set up a *virtual private network (VPN)*, which provides cryptographic protection for the confidentiality and authenticity of messages between the intranets in the VPN.

Limits of firewalls

- Once a host on an intranet behind a firewall has been compromised, the attacker can communicate with this machine by tunnelling traffic over an open protocol (e.g., HTTPS) and launch further intrusions unhindered from there.
- Little protection is provided against insider attacks.
- Centrally administered rigid firewall policies severely disrupt the deployment of new services. The ability to "tunnel" new services through existing firewalls with fixed policies has become a major protocol design criterion. Many new protocols (e.g., SOAP) are for this reason designed to resemble HTTP, which typical firewall configurations will allow to pass.

Firewalls can be seen as a compromise solution for environments, where the central administration of the network configuration of each host on an intranet is not feasible. Much of firewall protection can be obtained by simply deactivating the relevant network services on end machines directly.

"Is this product/technique/service secure?"

Simple Yes/No answers are often wanted, but typically inappropriate. Security of an item depends much on the context in which it is used.

Complex systems can provide a very large number of elements and interactions that are open to abuse. An effective protection can therefore only be obtained as the result of a systematic planning approach.

Security Management and Engineering II

"No worries, our product is 100% secure. All data is encrypted with 128-bit keys. It takes billions of years to break these."

Such statements are abundant in marketing literature. A security manager should ask:

- What does the mechanism achieve?
- Do we need confidentiality, integrity or availability of exactly this data?
- Who will generate the keys and how?
- Who will store / have access to the keys?
- Can we lose keys and with them data?
- Will it interfere with other security measures (backup, auditing, scanning, ...)?
- Will it introduce new vulnerabilities or can it somehow be used against us?
- What if it breaks or is broken?
- ...

Security policy development

Step 1: Security requirements analysis

- Identify assets and their value
- Identify vulnerabilities, threats and risk priorities
- Identify legal and contractual requirements

Step 2: Work out a suitable security policy

The security requirements identified can be complex and may have to be abstracted first into a high-level **security policy**, a set of rules that clarifies which are or are not authorised, required, and prohibited activities, states and information flows.

Security policy models are techniques for the precise and even formal definition of such protection goals. They can describe both automatically enforced policies (e.g., a mandatory access control configuration in an operating system, a policy description language for a database management system, etc.) and procedures for employees (e.g., segregation of duties).

Step 3: Security policy document

Once a good understanding exists of what exactly security means for an organisation and what needs to be protected or enforced, the high-level security policy should be documented as a reference for anyone involved in implementing controls. It should clearly lay out the overall objectives, principles and the underlying threat model that are to guide the choice of mechanisms in the next step.

Step 4: Selection and implementation of controls

Issues addressed in a typical low-level organisational security policy:

- General (affecting everyone) and specific responsibilities for security
- Names manager who "owns" the overall policy and is in charge of its continued enforcement, maintenance, review, and evaluation of effectiveness
- Names individual managers who "own" individual information assets and are responsible for their day-to-day security
- Reporting responsibilities for security incidents, vulnerabilities, software malfunctions
- Mechanisms for learning from incidents

- Incentives, disciplinary process, consequences of policy violations
- User training, documentation and revision of procedures
- Personnel security (depending on sensitivity of job)

  Background checks, supervision, confidentiality agreement
- Regulation of third-party access
- Physical security

  Definition of security perimeters, locating facilities to minimise traffic across perimeters, alarmed fire doors, physical barriers that penetrate false floors/ceilings, entrance controls, handling of visitors and public access, visible identification, responsibility to challenge unescorted strangers, location of backup equipment at safe distance, prohibition of recording equipment, redundant power supplies, access to cabling, authorisation procedure for removal of property, clear desk/screen policy, etc.
- Segregation of duties

  Avoid that a single person can abuse authority without detection (e.g., different people must raise purchase order and confirm delivery of goods, croupier vs. cashier in casino)
- Audit trails

  What activities are logged, how are log files protected from manipulation
- Separation of development and operational facilities
- Protection against unauthorised and malicious software

- Organising backup and rehearsing restoration
- File/document access control, sensitivity labeling of documents and media
- Disposal of media

  Zeroise, degauss, reformat, or shred and destroy storage media, paper, carbon paper, printer ribbons, etc. before discarding it.
- Network and software configuration management
- Line and file encryption, authentication, key and password management
- Duress alarms, terminal timeouts, clock synchronisation, ...

For more detailed check lists and guidelines for writing informal security policy documents along these lines, see for example

- British Standard 7799 "Code of practice for information security management"
- German Information Security Agency's "IT Baseline Protection Manual" `http://www.bsi.bund.de/english/gshb/manual/`
- US DoD National Computer Security Center Rainbow Series, for military policy guidelines `http://en.wikipedia.org/wiki/Rainbow_Series`

UK Computer Misuse Act 1990

- Knowingly causing a computer to perform a function with the intent to access without authorisation any program or data held on it $\Rightarrow$ up to 6 months in prison and/or a fine
- Doing so to further a more serious crime
  $\Rightarrow$ up to 5 years in prison and/or a fine
- Knowingly causing an unauthorised modification of the contents of any computer to impair its operation or hinder access to its programs or data $\Rightarrow$ up to 5 years in prison and/or a fine

The intent does not have to be directed against any particular computer, program or data. In other words, starting automated and self-replicating tools (viruses, worms, etc.) that randomly pick where they attack is covered by the Act as well. Denial-of-service attacks in the form of overloading public services are not yet covered explicitly.

`http://www.hmso.gov.uk/acts/acts1990/Ukpga_19900018_en_1.htm`

UK Data Protection Act 1998

Anyone processing personal data must comply with the eight principles of data protection, which require that data must be

1. fairly and lawfully processed

   Person's consent or organisation's legitimate interest needed, no deception about purpose, sensitive data (ethnic origin, political opinions, religion, trade union membership, health, sex life, offences) may only be processed with consent or for medical research or equal opportunity monitoring, etc.

2. processed for limited purposes

   In general, personal data can't be used without consent for purposes other than those for which it was originally collected.

3. adequate, relevant and not excessive
4. accurate
5. not kept longer than necessary
6. processed in accordance with the data subject's rights

   Persons have the right to access data about them, unless this would breach another person's privacy, and can request that inaccurate data is corrected.

7. secure
8. not transferred to countries without adequate protection

   This means, no transfer outside the European Free Trade Area. Special "safe harbour" contract arrangements with data controllers in the US are possible.

Some terminology:

"Personal data" is any data that relates to a living identifiable individual ("data subject"), both digitally stored and on paper.

A "data controller" is the person or organisation that controls the purpose and way in which personal data is processed.

`http://www.hmso.gov.uk/acts/acts1998/19980029.htm`
`http://www.ico.gov.uk/`
`http://www.admin.cam.ac.uk/univ/dpa/`