COMP 3704 Computer Security

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

Key Size

- Consider how much the information is worth
- Even advancements in computing are not going to change energy requirements by more than a few orders of magnitude
- Assume some weakness in your system (cipher, protocol or key generation) allows adversary to reduce search space from 2^n to $2^{n/2}$

Guidelines

- For symmetric keys, use between 128 and 256 bits
- For asymmetric keys, use about 8x the size of your symmetric key
- More important than key size are key generation and key management

Key Management

- If the keys are compromised, even the most secure protocols can be broken
- Key Management is about generating, protecting and distributing keys

Key Generation and Protection

- Make sure you have enough entropy to actually generate all possible values in the keyspace
- Example: there are only 2^{56} 8-character 7-bit passwords

$\Rightarrow$ Guarding a 256-bit key with an 8-character password makes no sense

Use hashing to convert long, low-entropy user input into short, high-entropy keys.

Software Key Management

- Cryptographic methods need keys in plaintext in memory for encyrption and decryption
- Operating system may write memory with key to disk (swap)
- Operating system may give pages still containing keys to other applications

$\Rightarrow$ Protect memory with keys (and intermediate results) from swapping, overwrite with random data before releasing!

Key Storage

- Ideal: dedicated hardware (for example, smart cards) storing the key and providing encryption/decryption functionality
- Alternative: store keys encrypted with (strong) passphrase on disk
- Reality: keys stored in plaintext in files protected by OS permissions

Key Distribution

X.509 PKI A certificate authority (CA) signs a certificate vouching for the relation between a public key and some physical entity

Web of trust Users meet and certify (sign) each other's keys together with a confidence rating; trust in validity of a binding is based on confidence on all discovered paths

X.509

+ accountability

+ easy to use, widely supported

 - need to establish identity with CA (costly)

 - need to trust CAs (check your browser's list!)

Web of trust

+ distributed, cheap

+ geeky & social

Web of trust

+ distributed, cheap
+ geeky & social
 - geeky & social!?
 - accountability, validation standards hard to enforce
 - path finding is costly (need to download certificate data)
 - certificate revocation is difficult to communicate

CAcert

- Web of trust used to associate key and entity
- CAcert then issues X.509 certificate

\+ free

\- security disadvantages from both approaches

Self-signed certificates

- Sign your own X.509 certificate
- No actual authenticiation
- Allows use of SSL (for encryption) without CA

Key revocation

- Key revocation certificates are used to communicate that a key was compromised
- Key lifetime limitations (as part of certification) are used to expire keys automatically
- The more frequently used a key is, the shorter its lifetime should be

Cryptography tools

- OpenSSL
- gpg

OpenSSL

- Standard library (libcrypto)
- Provides common cryptographic primitives
- Provides high-level protocol implementations (X.509, SSL)
- Binary (`openssl`) allows command-line access
- Used by Apache for SSL support

OpenSSL usage

`$ openssl help` list commands

`$ openssl md5 FILENAME` $\Rightarrow$ MD5 hash of FILENAME

```
$ openssl des -in FILENAME -out FILENAME.enc -e
  -k 'foo bar'
```

A Self-Certified CA

$ wget http://grothoff.org/.../3704/openssl.cnf

$ echo 01 > ca.db.serial; mkdir ca.db.certs ca.db.index

$ openssl genrsa 2048 > ca.key

$ openssl req -new -x509 -key ca.key -out ca.cert -config openssl.cnf

$ cat ca.cert ca.key > ca.pem

Creating a Certificate

$ openssl genrsa 1024 > server.key

$ openssl req -new -key server.key -out server.csr -config openssl.cnf

$ openssl x509 -in server.csr -out server.cert -req -signkey server.key

$ openssl ca -out server.cert -infiles server.csr

$ openssl verify -CAfile ca.cert server.cert

Configuring Apache to use Certificate

- SSLEngine on
- SSLCertificateFile server.cert
- SSLCertificateKeyFile server.key
- You need to listen on a port $>$ 1024

$ apache2 -f apache2.conf

GnuPG

- Free version of PGP, with library (libgcrypt)
- Provides common cryptographic primitives
- Provides implementation of OpenPGP (RFC 2440)
- Commonly used for secure E-mail
- Provides web of trust

Using GnuPG

$ gpg –gen-key

$ gpg –export

$ gpg –import FILENAME

$ gpg –edit-key EMAIL; > fpr > sign > trust

$ gpg –clearsign FILENAME

Questions

?

Problem

You are running the secure webportal for the Denver Credit Union. You are already using SSL and have your certificate signed by major CAs. What else could be done to improve the security of your site?

Problem

You are running the secure webportal for the Denver Credit Union. Customers are scared to use the Internet for banking. How can you improve their **impression** of the security of your site to gain more customers?