CS 580 Client-Server Programming
Fall Semester, 2002
Doc 25 Security
Contents

References

NIH Security Web Site
http://www.alw.nih.gov/Security/security.html

Applied Cryptography Second Edition, Bruce Schneier, John Wiley & Sons, 1996

Red Team versus the Agents, Scientific American, December 2000, pp. 20, 24.

Security

Some common issues

- Authentication

  Client (server) should be able to ascertain the origin of the message

  Intruder should not be able to masquerade as someone else

- Integrity

  Messages should not be modified in transit

- Nonrepudiation

  Sender should not be able to falsely deny sending a message later

- Confidentiality

  Intruders should not be able to learn sensitive data from a message

- Restricted Access

  Users should be able to perform requests that they are authorized to perform

Some Bad Ideas

- Security by Obscurity
- Security in the wrong place
- Authentication without checking
- Back doors

Security through Obscurity

Security relies on encryption/authentication methods are not obvious.

Some examples:

- Reverse the byte order of a message
- Swap bytes in some "secret" way
- Add garbage to data
- Use some "secret" algorithm

Security in the Wrong Place

Always think about what you're trying to accomplish with a security system.

Examples:

- Client performs authentication locally

Authentication without Checking

A server that has an authentication and authorization should precede actions that require authentication.

Example:

POP server without the different states

Back doors

Programmers have the tendency to add debug code to their servers to make testing easier.

This debug code may circumvent any security features of the server.

Example

- sendmail "WIZARD"

  Wizard command gave full root privileges to the user
  The default distribution had this command enabled
  The "Internet worm" used this to attack machines throughout the Internet.

- Sandia National Labs Security Agents Software

  Agent software based on Lisp
  Agents could perform any Lisp string
  Agents could request other agents to perform tasks
  Intruders could masquerade as an agent

General Security Issues

Points to keep in mind when dealing with security:

- The security system is only as strong as its weakest link
- A flawed security system is worse than no security at all
- US government considers encryption algorithms munitions

- Several security systems together are better than only one
- Security by obscurity really doesn't work

Some existing tools to look at:

- PGP (Pretty Good Privacy)
- RCA
- Kerberos
- SSL (Secure Socket Layer)

Network Authentication

Network packets can travel through many routers and computers.

The added risks

- Network sniffing
- Traffic logging
- Etc.

Some issues:

- Passwords can be "sniffed" from the network
- Traffic patterns can be analyzed

Authentication methods: Basic

Username and password

Protocols we have seen that use this:

- POP
- HTTP

Problems

POP clients check for new mail periodically

Each check requires sending name/password

HTTP requests are made every time a page is requested.

Username and password get embedding in html pages

The more frequent the authentication information is sent over a network, the higher the chance that it will be sniffed.

One-Way Hash Functions

Let M be a message (sequence of bytes)

A one-way hash function f() such that:

- f maps arrays of bytes to arrays of bytes
- f(M) is always the same length
- Given an M it is easy to compute f(M)
- Given f(M) it is hard to compute M
- Given M it is hard to find N such that f(M) = f(N)

Common One-way Hash Functions

MD5 - Message Digest 5
SHA - Secure Hash Algorithm

One-Way Hash and Logins

Let f() be a one-way hash function

Client has password and userName

Client with IP XXX connects to server

Client sends userName and f(password + XXX)

Server knows the client's IP address XXX from the connection

Server computes f(password + XXX) to validate password

Sniffer can only see f(password + XXX)

f(password + XXX) only works from the machine with IP XXX

Using MD5 & SHA in Java

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class SampleCode
  {
  public static void main(String args[])
    throws NoSuchAlgorithmException
    {
    MessageDigest sha = MessageDigest.getInstance("SHA");
    sha.update("Hi mom".getBytes());
    byte[] shaHash = sha.digest();
    System.out.println(new String(shaHash));

    MessageDigest md5 = MessageDigest.getInstance("MD5");
    md5.update("Hi mom".getBytes());
    byte[] md5Hash = md5.digest();
    System.out.println(new String(md5Hash));
    }
  }
```

Using MD5 & SHA in Smalltalk

Load the MD5 & SHA parcels

'Hi mom' asByteArray md5Value

151963197781583193299119603806589129963

'Hi mom' asByteArray shaValue

559954799469170145248016473141621118947215272720

Using security tokens, tickets, or cookies

Applications using stateless protocols (http) need to authenticate every request

Server gives a security token to a client

The token identifies the client to the server

The client only authenticates once and thereafter uses the cookie

The authentication may involve "expensive" encryption

Some requirements for security tokens:

- Cannot be faked
- Need to expire after some time

Practical security token issues

How can a server identify a client with a token?

Rely on a piece of information that is hard to change: IP address of the client

The server must correlate the token with the address when the client uses it.

How can this be done?

A random string which identifies the client
A one-way scrambled string with client information

Encryption

After a client has been authenticated, the traffic on a network can still be sniffed.

A solution is encryption of all traffic.

This can be done at any layer of the protocol stack

Two basic types of encryption:

- Shared key encryption

  One key both encrypts and decrypts

- Public/Private key encryption

  One key encrypts, another decrypts

Public/Private Key Encryption

A public key is something that is well known, i.e. published.

A client can send authentication information by encrypting the info with the server's public key.

The server will then use its own private key to decrypt the information.

Advantages:

- The information the client sends to the server cannot be decrypted by anyone except the server

Common Algorithms

RSA (Rivist, Shamir, Adleman)
DSA (Digital Signature Algorithm)

RSA

Public Key

Key contains n & e where

n = p*q, p & q are primes
e relatively prime to (p-1)*(q-1)

p & q must be kept secret

Private Key

d is e^{-1} mod ((p-1)*(q-1))

that is (d*e) mod ((p-1)*(q-1)) = 1

Encrypting

Let m be a message such that m < n

Let c be the encrypted message

$c = m^e \bmod n$

If m >= n then break into block smaller than n and encrypt each block

Decrypting

$m = c^d \bmod n$

Example

Example is from page 467-8 of Schneier

Alice's Keys

Let

p = 47.
q = 71.

Then n = p*q = 3337

e = 79.

Then d = 79^{-1} mod 3220 = 1019

So Alice's public key is

n = 3337
e = 79

Alice's private key is

d = 1019

Sending a Message to Alice

Let m = 41

To send the message to Alice we compute

$c = m^e \bmod n = 41^{79} \bmod 3337 = 875$

We send 875 to Alice

Alice computes

$c^d \text{ mode } n = 857^{1019} \bmod 3337 = 41$

Digital Signatures

The same encryption method can be used to authenticate a message:

A client encrypts information with its own private key.

The server will lookup the client's public key and decrypt the information.

Advantages:

The information can only be decrypted with the client's public key.

If the public key distribution center can be trusted, the information is guaranteed to come from the client.

A combination of the previous two methods allows for mutual authentication.

Alice Signing a Document

Let the document be 67

Alice uses private key to compute

67^d mode n = 3081.

To validate the signature anyone can compute

3081^e mod n = 3081^{79} mod 3337 = 67