HTTPS
HTTP Secure
The World-Wide Web

Authentication Methods

- Several authentication methods are used:
  - **Declarative Security**
    - HTTP-level mechanisms
      - Basic authentication scheme
      - Digest access authentication scheme
    - Server-level mechanisms
  - Programmatic Security
HTTP Basic Mechanism

- In the basic authentication scheme of HTTP, the user’s name and password need to be sent with each request for a protected resource.
- When the server receives a request for a protected resource, it checks whether that request has the HTTP header `Authorization: Basic username:password`.
- If the name and password are accepted by the server (i.e., are those of a user that has the privilege to get the page), then the requested page is returned.

HTTP Basic Mechanism

- If the request does not have the authorization header or the name and password are not accepted, then the server replies with `401` (unauthorized).
- An 401 response can have the header `WWW-Authenticate: Basic realm="realm-name"`.
- That is, "in order to get this resource, you will have to authenticate using the basic method”.
- Tell the user to supply authentication for pages in `realm-name`.
Declarative Security: BASIC

Realm A
/a/A.html
/a/B.jsp

OK + Content
GET E.xsl

Realm B
/b/C.css
/b/D.xml

E.xsl
F.xml

GET /a/B.jsp
401 + Basic realm="A"

Realm A
/a/A.html
/a/B.jsp

GET /a/B.jsp
Declarative Security: BASIC

Realm A
/a/A.html
/a/B.jsp

OK + Content
GET /a/B.jsp + user:pass

Realm B
/b/C.css
/b/D.xml

E.xsl
F.xml

Declarative Security: BASIC

Realm A
/a/A.html
/a/B.jsp

OK + Content
GET /a/A.html + user:pass

Realm B
/b/C.css
/b/D.xml

E.xsl
F.xml
Browser Cooperation

- Throughout the session, the browser stores the user-name and password and automatically sends the authorization header in either one of the following cases:
  - The requested resource is under the directory of the originally authenticated resource
  - The browser received 401 from the Web server and the `WWW-Authenticate` header has the same realm as the previous protected resource

Basic Authentication in City Nexus System
Digest Access Scheme

- The most serious security flaw in the basic scheme is that the name and password are sent unencrypted, and hence everyone on the network path can read it.
- If an attacker snoops a request with basic authentication, she can access to the whole protection space of the resource.
- The *digest access authentication scheme* solves many of the flaws of the basic schemes, such as the one above.

Digest Operation

- Like the *basic*, the *digest scheme* requires that authentication data is sent with each request for a protected resource.
- However, passwords are not sent as simple text.
- The idea is to use a *one-way hash*, such as MD5.
- A one-way hash $H$ is a mapping of strings that has the following properties:
  - It is "easy" to compute $H(x)$, given the input $x$.
  - It is "hard" to compute $x$, given the mapping $H(x)$.
Digest Operation (cont)

- In the digest scheme, instead of sending the password $x$ as simple text, the client sends $H(y)$
- $y$ is the concatenation of the user name, the password, an opaque generated by the server, the request URI, and more (why is the opaque needed?)
- A server that gets digested authentication data repeats the same encryption process and compares its output with the given $H(y)$
- More details can be found in RFC 2617

Server-Level Authentication

- A Web server can use its own authentication mechanisms rather than those of HTTP
- Typically, server-level mechanisms act as follows
  - The server requires authentication by redirecting the client to a special HTML form
  - If authentication succeeds, then the server stores the username in the corresponding session object
- Note that the browser and the HTTP headers are oblivious to server-level authentication
Programmatic Security

- In declarative security, a page is either accessible to a user or is not.
- But what if we wanted a page to include some data that will only be shown to privileged users?
  - E.g., the grades of the user
- In programmatic security, we enhance security checks in dynamic pages (e.g., JSP)
- Using this approach, an application can generate different contents for different users

Declarative-Security: Pros & Cons

- **Advantage:** Application programs (i.e., JSP and Servlets) do not have to do anything special
- **Advantage:** Security holes due to bugs are less probable
- **Disadvantage:** Server-specific process
- **Disadvantage:** All or nothing security
  - users can or cannot see the page
  - sometimes, what we really want is for the page content to be dependent on the user
SSL Connections

Security on the Internet

- The Internet is used to transmit sensitive data from clients to servers and vice-versa
  - User passwords
  - Credit card numbers
  - Private client data on remote servers (e.g., Banks)
- However, data packets are read by several computers on the way from the client to the server (and vice-versa)
  - Routers, proxies, etc.
Security on the Internet (cont)

- For **secure communication**, the following should be provided:
  - Only the server can read the client requests
  - Only the client can read the server's responses
  - Only the client can send requests on behalf of itself
  - Only the server can send responses on behalf of itself
- In short, no one should be able to interfere in the interaction, either by reading the transferred data or by impersonating one of the sides
Symmetric and Asymmetric Keys

- Data can be encrypted and decrypted using *keys*, which are simply large numbers
- **Symmetric keys**: the same key is used for both encoding and decoding of the message
- **Asymmetric keys**: one key is used to encode the message, and another is used to decode it
- It is considered practically impossible to decode a message without knowing the decoding key

The RSA Cryptography System

- RSA was developed in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman
- It is based on the asymmetric key mechanism:
  - Each participant has a *private key* and a *public key*
  - The *public key* is known to all and the *private key* is kept in secret within its owner
  - Asymmetric keys: the *public key* is the encoding key and the *private key* is the decoding key
Secure Connection: A Naive Approach

- Consider the following protocol:
  - Server and Client send their public keys to each other
  - Data is encrypted using the public key of the receiver
- What is wrong with this protocol?
  - Encryption methods (public keys) are known to everyone - everyone can impersonate the participants
  - A participant cannot tell whether its received key was indeed sent by the other participant

SSL Connections

- The SSL (Secure Socket Layer) protocol is used to manage security of message transmission on the Internet
- Data encryption and decryption is based on symmetric and asymmetric keys
- The HTTPS (HTTP over SSL) protocol is actually the HTTP protocol above SSL transportation
SSL in the Network Layers

HTTP
Email Protocols
SSL
TCP/IP

The SSL Handshake

1. The client gets the Server's certificate

Is this a good certificate?

Client

Server

hello + SSL settings

SSL Settings + Certificate
The SSL Handshake

2. The client creates a master secret and shares it with the server

Client \( \overset{\text{E}}{\Rightarrow} \)

\( \overset{\text{E}}{\Rightarrow} \)

Server

3. The client and the server create symmetric session keys from the master secret

Client \( \overset{\text{E}}{\Rightarrow} \)

Server \( \overset{\text{E}}{\Rightarrow} \)
The SSL Handshake

Data is transferred using the session keys

SSL Certificates

- To assure that the replier of the first request is the server, the server sends a certificate.
- The certificate contains both the server's name and its public key.
- The certificate is issued by a Certificate Authority (CA), which is known to the client in advance.
  - For example: VeriSign, Thawte, RSA Secure Server, etc.
- CA signs the certificate using a digital signature, which the client can verify using a method similar to the private-public key method.
The Server's Certificate

<table>
<thead>
<tr>
<th>Public Key</th>
<th>Serial Number</th>
<th>Validity Period</th>
<th>Server's Name</th>
<th>Issuer's Name</th>
<th>Issuer's Digital Signature</th>
</tr>
</thead>
</table>

An Example: The Certificate of bankhapoalim.co.il
Authentication via SSL

- If the server needs to assure the client’s identity, the first interaction after the SSL handshake will typically be a client authentication.
- Client authentication is done using the regular (e.g., HTTP) authentication mechanisms.
- *What is the difference, though?*

**HTTPS approach**

```
Client hello

Server hello

Server Certificate

serverHelloDone

ClientKeyExchange E(Kserv, PK)

ChangeCipherSpec

FIN Handshake (MAC)

ChangeCipherSpec

FIN Handshake (MAC)

Application_data http request

Application_data http response

Alert : close_notify

Alert : close_notify
```

Handshake

Data

Close
Web Servers

Handles Security

Http

Authentication

SSL - Secure connection

Logs

statistics

Content negotiation

Compression

Dealing with large files – chunked transfer

Virtual hosting

Bandwidth throttling

Resources

• HTTP Basic Access Authentication:
  http://www.ietf.org/rfc/rfc2617.txt

• Digest Access Authentication:

• Secure Sockets Layer (SSL):
  http://tools.ietf.org/html/rfc6101,

• Transport Layer Security (TLS):