Overview

- History of the Web
- Architecture of the WWW
  - Packets and protocols
  - Hosts, gateways and firewalls
  - Networks and Domains
- Goals of e-business security
- Definition of the problem
  - Securing the server
  - Securing the pathway
  - Securing the user environment

The Evolution of the WWW

- The WWW grew from a number of basic ideas:
  - Bush’s notion of hypertext
  - The promise of SGML and Structured Markup
  - The development of a simplified protocol for transfer of documents
  - A method for identifying resources in a distributed fashion
- The web was significantly complicated by:
  - The browser wars
  - The inclusion of ever more sophisticated distribution of processing capability
  - The development of Web 2.0 capabilities (AJAX)
  - The emergence of mobile devices and wireless networks
- The problems may be insurmountable given:
  - The centrality of the end user
  - The emergence of the cloud
  - The complexity of the distributed system
The Emergence of E-Commerce

- The web has taken on a variety of ecommerce roles – both B2B and B2C
- Along with this important new role have come a number of concerns and problems
  - DDOS attacks have made operating sites inaccessible and resulted in revenue losses
  - DBMS back end data has been compromised resulting in customer information loss
  - Shoddy programming has made private information accessible inadvertently
  - Compromises of websites have resulted in incorrect or malicious information being exposed.

The Architecture of the Internet/Web

- The internet as a network and the WWW as a killer app are on the one hand accidents of history – there were many other alternatives
- On the other hand, compared to the alternatives, they were:
  - Competitively superior solutions – for their time
  - Very open to developers
  - Easy to use
  - Free

How the WWW Works

- A web client first resolves the address of a server using DNS
- An http protocol request is then dispatched to port 80 of that address to request a resource
  - Resources were initially simply files
  - Resources quickly came to include any file or any program that could produce HTML
- The resource returned to the client consists of an HTML document
Web Components

- The standards for the WWW define the capabilities of the software
- There are a half dozen pieces of software that are important to the WWW
  - Browsers are the clients
  - Proxies provide filtering and intermediate caches
  - Web servers maintain the libraries
    - SSL/TLS modules provide encrypted messages
    - Certificate servers provide authentication
  - Streaming media servers provide continuous connections

URLs

- A URL is made up of multiple parts:
  http://spring.michael@www.sis.pitt.edu:80/~cascade/index.html?a=b&c
  - Scheme
  - Hierarchical URL indicator
  - Authority
    - Access credentials
    - Server
    - Port
  - Resource path
  - Query String
  - Fragment Identifier

Some Notes on URL Processing

- Every browser processes URL's differently
- There are a number of issues that should be of concern:
  - How the credentials of the authority section are determined can create problems
  - The server address can be specified many different ways
  - The port specification creates some vulnerabilities
  - Character encoding schemes present many complications.
### Http Protocol

- A simple protocol
  - GET, POST
  - PUT, HEAD, OPTIONS, TRACE, DELETE
- A simple message
  - Here is some data
  - Here is a "document"
- An increasingly complex server (state, authentication, encryption, application serving)
- An increasing complex client (parse a variety of documents, trace links, spawn applications)

### Web Technology

![Diagram of web technology](image)

- Certificate Servers
- SSL Pipe
- Multi-function browser
- http request
- Multi-function server
- http response
- Pages for delivery
- Programs to produce pages

### Browser Technology

![Diagram of browser technology](image)

- Document Cache
- Cookies
- Plugins
- Input Formalization
  - DNS Lookup
  - Cookie Management
  - SSL Management
- Document Parsing
  - Java/Vbscript
  - URLs
  - CSS/XSLT
- Document Display
  - HTML/XML
  - Applet
- Plugin Management

http request

http response
Server Technology

- Request Parsing
- Authentication
- Security
- Error Handling
- Document Prep
- Finding
- Generation
- SSI
- JSP/CTM

Pages for delivery
- Passwords and controls
- Programs that produce pages
- SSI and J/ASP
- CTM

The http protocol

- The web protocol is very robust and very simple
- For each request, the client:
  - Does a DNS lookup if needed
  - Opens a connection to the server
  - Sends a request for a resource
- The server:
  - Checks the availability of the resource
  - Returns the resource or an error message
  - Closes the connection

The structure of requests and responses

- Requests have a header and a body
  - The header has many lines but:
    - Begins with one of seven standard request types
  - The body is null for five of the request types and contains data for the POST and PUT types

- Responses have a header and a body
  - The header has many lines but:
    - Begins with a status
    - Ends with a content type
    - The body contains either the resource or an explanatory message
A Sample Request

• The user types the following in their client:
  http://www.sis.pitt.edu/~cascade/index.html

• The client sends only a header:
  GET /~cascade/index.html HTTP/1.0
  If-Modified-Since: Fri, 10 Oct 1997 17:35:54 GMT;
  User-Agent: Mozilla/4.7 [en] (X11; I; SunOS 5.6 sun4u)
  Pragma: no-cache
  Host: www.sis.pitt.edu
  Accept: image/gif, image/jpeg, image/pjpeg, image/png, */*
  Accept-Encoding: gzip
  Accept-Language: en-US, en
  Accept-Charset: iso-8859-1,*,utf-8

A Sample Response

HTTP/1.1 200 OK
Date: Wed, 01 Dec 1999 16:11:19 GMT
Server: Apache/1.3.1 (Unix)
Last-Modified: Wed, 12 May 1999 20:31:56 GMT
ETag: "7a108-16c2-370c3e53c"
Content-Length: 5826
Connection: close
Content-Type: text/html

<html>
<head>
<title> CASCADE </title>
</head>
<body>...
</body>...

Request/Response Headers

Authorization: encoding, name and password
Content-Encoding: how the body is encoded
Content-Length: length of the body
Content-Type: type(mime) of the body
Date: the date and time the request was generated
From: email address of the requestor
Last-Modified: date/time of last modification
Pragma: directives to the client – e.g. no-cache
Server/User Agent: server/browser type
Referer: the address of the resource of the link
Request Types

• Get – to get a document or run a CGI script. When a form is used, the form info is placed in the request
• Post – to get a document or run a CGI script. When a form is used, the form info is placed in the body
• Head – used to ask for header info only
• Put – used to place a document on the server
• Delete – deletes a document on a server
• Options – tells what options the server has
• Trace – for debugging, tells what the server is doing

Status Codes

• Five categories of status code
  • 1xx: informational – used for development
  • 2xx: Successful response
  • 3xx: Redirection
  • 4xx: Client Error
  • 5xx: Server Error

• Frequently used codes:
  • 200 – success
  • 301 and 302 – moved permanently or temporarily
  • 400 – bad request
  • 401 – unauthorized
  • 403 – forbidden
  • 404 – not found

Content Types

• The content types for WWW documents are all allowable mime types
• Whether the browser can handle them is not the concern of the server
• The common content-types are:
  • text/ascii  text/html
  • image/jpg  image/gif
  • application/pdf  application/msword
Development of web capability

• With time, it became clear that web was too static
• The Common Gateway Interface (CGI)
  • CGI created a capability to develop dynamic pages based on server program execution. Perl became the language of choice.
• Scripting Languages
  • As the CGI load on networks and servers grew, scripting languages were developed to offload some of the demand to the client
  • Full client side applications – applets emerged as well
  • Stylesheets were also added for clients
• Active Server Pages (ASP)
  • ASPs are pages that call functions that yield specific pieces of text.
  • These provided an alternative to CGI -- programs that wrote pages.
  • Java Server Pages (JSP) parallel Microsoft's ASP

Authentication

• Various mechanisms can be used to control or limit access to a web site
  • Apache on Unix uses a file called .htaccess
  • A restricted resource causes a 401 challenge to be issued
    • If the user doesn’t authenticate, a message is displayed
    • If the user authenticates correctly, the resource is provided
  • Basic authentication involves encoding, not encryption

Cache Control

• Caches exist in many places on the web – servers, clients, and intermediate proxies.
• Cache policy can be set by the client or the server. They include:
  • No-store: keep no copies
  • No-Cache: check the server before providing the cache
  • Public: the response may be cached by end user and proxy
  • Private: the response may not be cached by a proxy
  • Max-age: number of seconds for which the message is valid
Sessions and Cookies

- One of the biggest problems with HTTP is the lack of state information.
- There is no perfect solution to this problem, but approaches include:
  - URL rewriting
  - Hidden Fields
  - Cookies
- Server side languages abstract these methods and provide consistent mechanism for holding state information.

Persistent Connection and Ranges

- Under HTTP 1.1, connections are by default persistent and must be explicitly closed.
- Under HTTP 1.0, this was accomplished using the headers:
  - `Keep-alive: #of seconds`
  - `Connection: keep-alive`
- Under HTTP 1.1 ranges of resources can also be requested and a 206 status code is returned if the range is returned.

The First HTML

- In 1991, Berners-Lee provided the first simple HTML document elements.
- It was based on an SGML DTD used at CERN:
  - It included the heading elements
  - Some paragraph and list elements
  - The anchor element
- As time went on new elements were appearing based on the browsers.
**HTML 2.0**

- The IETF formed a working group in 1994 that completed a specification for HTML in 1995.
- It combined the original HTML tags and several things from Dave Reggett's HTML+
  - Tables
  - Forms
- HTML 3.0 followed but was abandoned

**HTML 4.01**

- HTML 4.0 and 4.01 were developed to bring the DTD into conformance with SGML
- It introduced new elements:
  - Span and div
  - Frames, framesets and iframes
  - More extensive table elements
- New attributes were defined
- Xhtml was based on html 4.01 but tried to move away from frames

**XML and xhtml**

- There was a belief that xhtml would eventually lead to a diverse set of schema's based on xml.
- Early on, several new presentation schemas were developed
  - Mathml
  - SVG
  - MusicXML
  - RSS
- In addition, hundreds of XML schemas for special purposes were developed
A growing problem

- Initial attacks in cyberspace were mostly a matter of defacing websites – to prove that it could be done.
- Despite attacks and theft of user data (e.g. credit card numbers), there was little actual abuse of the data.
- As the web becomes a more normal place to do business – given the economies of scale – it will become increasingly important to secure all aspects of the transactions.

Solutions to the problem

- The earliest response to the problem was to offer “secure servers” as a solution – meaning that data was encrypted in transport
- It is increasingly important that the data be protected at the terminus – once it is collected.
- It is equally important that the data only be released to legitimate owners of the data
- It is important that data sent to customer be clean – free of viruses and that it be correct

The Web Security Landscape

- Web security has three facets
  - Securing the web server and the data that is on it.
    - Data cannot be modified
    - It is only sent to authorized individuals
  - Securing data in transit between the web server and the user
    - Keep the data confidential
    - Keep the “connection open”
    - Keep the end users computer secure
Security Pragmatics

• Develop a good authentication system
• Provide for off-site backup of data
• Provide for physical security of systems
• Provide for replication and load balancing
  • Servers
  • Internet connections
• Make sure infrastructure (e.g. IP and DNS) can’t be compromised
• Make sure network operations are monitored for intrusions

Securing Web Servers

• The computer itself must be secured
  • Minimize accounts
  • Restrict account access to services
• The operating system must be secured
  • Minimize services
• The web server and scripts that make up your site must be secured
  • Check server configuration
  • Check all scripts

Securing Information in Transit

• Practically speaking, the only way to secure information in transit is to encrypt it.
  • SSL/TLS provide this function, but only if the parties agree
  • There are issues of spoofing, man-in-the-middle attacks, and certificate authorities
  • The best solution is the use of both server and client certificates
• Security of information in transit also includes insuring the availability of the service – protected from denial of service attacks
Securing the user's machine

- The weakest link in any effort to secure e-commerce is the end users system
- Many end user systems have been compromised without the end user knowing they are compromised.
- Spyware and worms have invaded many machines and user education seems to be of limited values
- Even automatic updates are sometimes suspicious

Risk Analysis vs Best Practices

- Traditional security practices have often involved doing a risk analysis – given a possible event, what is the potential it will occur and what will be the damage that results.
- Web security does not lend itself to this systematic way of thinking – because the probability of some events can be difficult to estimate
- Best practices provides an alternative, but it does not provide a complete answer either in that best practices vary from site to site and application to application.

The Goals of E-Business Security

- Establish reasons for confidence and trust of the partners in the transaction
  - Provide security and privacy policies
  - Provide mechanisms for insuring identity
- Establish appropriate security for data in storage and in transit
  - Insure that data is secured at the server end
  - Insure that data is secured in transit
  - Insure that data is secured at the client end