CSE 543 - Computer Security (Fall 2006)

Lecture 19 - Web Security
November 9, 2006
URL: http://www.cse.psu.edu/~tjaeger/cse543-f06/
Network vs. Web Security
What is the web?

• A collection of application-layer services used to distribute content
  – Web content (HTML)
  – Multimedia
  – Email
  – Instant messaging

• Many applications
  – News outlets, entertainment, education, research and technology, …
  – Commercial, consumer and B2B
Web security: the high bits

- The largest distributed system in existence
  - threats are as diverse as applications and users
  - But need to be thought out carefully …

- The stakeholders are …
  - Consumers (users, businesses, agents, …)
  - Providers (web-servers, IM services, …)

- Another way of seeing web security is
  - Securing the web infrastructure such that the integrity, confidentiality, and availability of content and user information is maintained
Secure socket Layer (SSL/TLS)

- Used to authenticate servers
  - Uses certificates, “root” CAs
- Can authenticate clients
- Inclusive security protocol
- Security at the socket layer
  - Transport Layer Security (TLS)
  - Provides
    - authentication
    - confidentiality
    - integrity
SSL Handshake

1. Client Hello (algorithms, …)
2. Server Hello (alg. selection, …)
3. Server Certificate
4. ClientKeyRequest
5. ChangeCipherSuite
6. ChangeCipherSuite
7. Finished
8. Finished
**Simplified Protocol Detail**

*Participants:* Alice/A (client) and Bob/B (server)

*Crypto Elements:* Random $R$, Certificate $C$, $k_i^+$ Public Key (of $i$)

*Crypto Functions:* Hash function $H(x)$, Encryption $E(k, d)$, Decryption $D(k, d)$, Keyed MAC $HMAC(k, d)$

1. Alice $\rightarrow$ Bob $\quad R_A$
2. Bob $\rightarrow$ Alice $\quad R_B, C_B$
   - Alice pick pre-master secret $S$
   - Alice calculate master secret $K = H(S, R_A, R_B)$
3. Alice $\rightarrow$ Bob $\quad E(k_B^+, S), HMAC(K', CLNT' + [#1, #2])$
   - Bob recover pre-master secret $S = D(k_B^-, E(k_B^+, S))$
   - Bob calculate master secret $K = H(S, R_A, R_B)$
4. Bob $\rightarrow$ Alice $\quad HMAC(K', SRV R' + [#1, #2])$

**Note:** Alice and Bob : IV Keys, Encryption Keys, and Integrity Keys 6 keys, where each key $k_i = g_i(K, R_A, R_B)$, and $g_i$ is key generator function.
Advantages of SSL

• Confidential session
• Server authentication*
• GUI clues for users
• Built into every browser
• Easy to configure on the server
• Protocol has been analyzed like crazy
• Seems like you are getting security “for free”
Disadvantages of SSL

- Users don’t check certificates
  - most don’t know what they mean
- Too easy to obtain certificates
- Too many roots in the browsers
- Some settings are terrible
  - ssl v2 is on
  - totally insecure cipher suites are included
- very little use of client-side certificates
- performance!
  - early days had sites turning off
  - getting better (crypto coprocessors, etc.)
Reality of SSL

• SSL is here to stay no matter what
• credit card over SSL connection is probably safer than credit card to waiter
• biggest hurdles:
  – performance
  – user education (check those certificates)
  – too many trusted sites (edit your browser prefs)
  – misconfiguration (turn off bad ciphersuites)
  – can be used for many non-web applications
Cookies

• Cookies were designed to offload server state to browsers
  – Not initially part of web tools (Netscape)
  – Allows users to have cohesive experience
  – E.g., flow from page to page,

• Someone made a design choice
  – Use cookies to authenticate and authorize users
  – E.g. Amazon.com shopping cart, WSJ.com
Cookie Issues …

• New design choice means
  – Cookies must be protected
    • Against forgery (integrity)
    • Against disclosure (confidentiality)

• Cookies not robust against web designer mistakes
  – Were never intended to be
  – Need the same scrutiny as any other tech.

Many security problems arise out of a technology built for one thing incorrectly applied to something else.
Cookie Design 1: mygorilla.com

- Requirement: authenticate users on site mygorilla.com

- Design:
  1. use digest authentication to login user
  2. set cookie containing hashed username
  3. check cookie for hashed username

- Q: Is there anything wrong with this design?
Cookie Design 2: mygorilla.com

• Requirement: authenticate users on site

mygorilla.com

• Design:
  1. use digest authentication to login user
  2. set cookie containing encrypted username
  3. check cookie for encrypted username

• Q: Is there anything wrong with this design?
Exercise: Cookie Design

• Design a secure cookie for mygorilla.com that meets the following requirements

• Requirements
  – Users must be authenticated (assume digest completed)
  – Time limited (to 24 hours)
  – Unforgeable (only server can create)
  – Privacy-protected (username not exposed)
  – Location safe (cannot be replayed by another host)
Library Attack ....

• I am sitting in the local library using the computer ...
• ... to buy some stuff ...
• ... and walk away ...
Dynamic Content

• Server generates content at run time
  – For time-sensitive information (stock ticker)
  – For user customization (Amazon.com)
  – Provide HTML interface to complex system (e.g., course management system)
Dynamic Content: CGI

- Common Gateway Interface (CGI)
  - Generic way to call out to external applications on the server
  - Passes URL to external program (e.g., form)
  - Result is captured and return to requestor

- Historically
  - "shell" scripts used to generate content
    - Very, very dangerous

- NOTE: server extensions are no better (e.g., servlets)
DC: Embedded Scripting

• Program placed directly in content, run at during request time and output returned in content
  – MS active server pages (ASP)
  – PHP
  – mod_perl
  – server-side JavaScript
  – python, ....

• Nice at generating output
  – Dangerous if tied to user input
Web Server Security

• Microsoft IIS 5.0 had many flaws
  – Buffer overflows (Code Red)
  – All services (ftp, smtp, etc) ON by default
  – Permissions on server are loose (modify system files)
  – ISAPI enables access to many libraries (ON by default)
  – Default website, so everyone knows where you are
  – Unnecessary code is installed
  – Poor authentication

• IIS 6.0 is better
  – OFF by default
  – Perms still a challenge
  – No default website
Warning: Cross-Site Scripting

• Note Assume the following is posted to a message board on your favorite website:

   Hello message board.
   <SCRIPT>malicious code</SCRIPT>
   This is the end of my message.

• Now a reasonable ASP (or some other dynamic content generator) uses the input to create a webpage (e.g., blogger nonsense).

• Now a malicious script is now running
  – Applet, ActiveX control, …
Dynamic Content Security

• Largely just applications
  – Inasmuch as application are secure
  – Command shells, interpreters, are dangerous

• Three things to prevent DC vulnerabilities
  – Validate input
    • Input often received as part of user supplied data
    • E.g., cookie
  – Limit program functionality
    • Don’t leave open ended-functionality
  – Execute with limited privileges