SSL/TLS Overview

- SSL offers security for HTTP protocol
- Authentication of server to client
- Optional authentication of client to server
  - Incompatibly implemented in different browsers
  - CA infrastructure not in widespread use
- Confidentiality of communications
- Integrity protection of communications
Purpose in more detail

- **Authentication based on certification authorities (CAs)**
  - Trusted third party with well-known public key
  - Certifies who belongs to a public key (domain name and real name of company)
  - Example: Verisign

- **What SSL Does Not Address**
  - Privacy
  - Traffic analysis
  - Trust management
Ciphersuites: Negotiating ciphers

- Server authentication algorithm (RSA, DSS)
- Key exchange algorithm (RSA, DHE)
- Symmetric cipher for confidentiality (RC4, DES)
- MAC (HMAC-MD5, HMAC-SHA)
Overview of SSL Handshake

Client

Supported ciphers, client random

Chosen cipher, server random, certificate

Compute keys

Encrypted pre-master secret

MAC of handshake messages

MAC of handshake messages

Server

From “SSL and TLS” by Eric Rescorla
Simplified SSL Handshake

- Client and server negotiate on cipher selection.
- Cooperatively establish session keys.
- Use session keys for secure communication.
Client Authentication Handshake

- Server requests that client send its certificate.
- Client signs a signed digest of the handshake messages.
SSL Client Certificate

From “SSL and TLS” by Eric Rescorla
Establishing a Session Key

- Server and client both contribute randomness.
- Client sends server a “pre-master secret” encrypted with server’s public key.
- Use randomness and pre-master secret to create session keys:
  - Client MAC
  - Server MAC
  - Client Write
  - Server Write
  - Client IV
  - Server IV
Establishing a Session Key

From “SSL and TLS” by Eric Rescorla
Session Resumption

- Problem: Public key crypto expensive
- New TCP connection, reuse master secret.
  - Avoids unnecessary public key cryptography.
- Combines cached master secret with new randomness to generate new session keys.
- Works even when the client IP changes (servers cache on session ID, clients cache on server hostname).
What does a CA-issued Certificate Mean?

• No one knows exactly.

• That a public key belongs to someone authorized to represent a hostname?

• That a public key belongs to someone who is associated in some way with a hostname?

• That a public key belongs to someone who has lots of paper trails associated to a company related to a hostname?

• That the CA has no liability?
How to get a Verisign certificate

- Pay Verisign ($300)
- Get DBA license from city call ($20)
  - No on-line check for name conflicts...can I do business as Microsoft?
- Letterhead from company ($0)
- Notarized document (need driver’s license) ($0)

Conclusions:
- Easy to get a fraudulent certificate
- Maybe not so easy to avoid prosecution afterwards

But that’s only Verisign’s policy
- Many CAs can issue certificates
So many CAs...
Client Authentication on the Web

E*TRADE Customer & Member Log On

New! Earn $50 for each new customer you refer to E*TRADE. Get started now (customer logon required)

E*TRADE User Name: [Field]
Password: [Field]
Start In: [Field]

Log on to OptionsLink®
(For Business Solutions clients only)

For our Chinese language investors, we now offer E*TRADE Chinese

Interrogative adversaries

- Adaptively query a Web server a reasonable number of times
- Treat server as an oracle for an adaptive chosen message attack
- Don’t need any eavesdropping or other network tampering
- Anyone can do it, but surprisingly powerful attack
  - C.f., adaptive chosen-ciphertext attacks—sounded improbable
Cookies

- A Web server can store key/value pairs on a client
- The browser resends cookies in subsequent requests to the server
- Cookies can implement login sessions
Netscape cookie example

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>.wsj.com</td>
</tr>
<tr>
<td>Path</td>
<td>/cgi</td>
</tr>
<tr>
<td>SSL?</td>
<td>FALSE</td>
</tr>
<tr>
<td>Expiration</td>
<td>941452067</td>
</tr>
<tr>
<td>Variable name</td>
<td>fastlogin</td>
</tr>
<tr>
<td>Value</td>
<td>bitdiddleMaRdw2J1h6Lfc</td>
</tr>
</tbody>
</table>
Cookies for login sessions

Why? Enter a password once per session
SSL can’t protect data sent without SSL

- Problem: Secure content can leak through plaintext channels
- Cookie file has flag to require SSL
  - Not set by BankOnline.com
- Trick user into visiting HTTP port
  - Just need a link from an unrelated web page
  - Cookie automatically sent in the clear
  - Network eavesdropper can record it
  - Might as well not have used SSL
Letting clients name the price: Instant Shop

- Problem: Servers trust clients not to modify HTML variables.
- Price determined by hidden variable in Web page.
- Make a personal copy of the web page. Modify it.
Instant Shop example: What a browser displays

To confirm your purchase, submit below.

Batteries $10  
Biology textbook $99  
Britney Spears CD $25

Submit Query  Confirm purchase
Instant Shop example: What’s inside

<html><body>
<form action=commit_sale.cgi>

<input type=hidden name=item1 value=10>Batteries $10<br>
<input type=hidden name=item2 value=99>Biology textbook $99<br>
<input type=hidden name=item3 value=25>Britney Spears CD $25<br>
<input type=submit>Confirm purchase

</form>
</body></html>
Instant Shop example: Malicious client

<html><body>
<form action=commit_sale.cgi>

<input type=hidden name=item1 value=0>Batteries $10<br>
<input type=hidden name=item2 value=0>Biology textbook $99<br>
<input type=hidden name=item3 value=0>Britney Spears CD $25<br>
<input type=submit>Confirm purchase

</form>
</body></html>
Security through obscurity: NeBride.com

- Problem: No cryptographic authentication at all
- Cookie (authenticator) is the username
- Create a cookie with someone’s username
  - Instant access to her name, address, phone number, e-mail address, wedding date and place, and password.
Predictable sequence numbers: fatbrain.com

- Problem: Customer can determine the authenticator for any other user.
- Authenticators are sequence numbers in the URL.
  https://www.fatbrain.com/HelpAccount.asp?t=0&p1=fubob@mit.edu&p2=540555758
  https://www.fatbrain.com/HelpAccount.asp?t=0&p1=nobob@mit.edu&p2=540555759
- Guess a victim’s sequence number by decrementing.
- Access to personal information
- Change address, receive password by email!
Welcome to Your Account.
Manage your account information, check on orders you have placed and more.

Use the menu bar on the left to:

- Change Sign-in E-mail -- change your sign-in e-mail. More...
- Change Password -- change your sign-in password. More...
- Edit Profiles -- edit your shipping, billing and payment information or create a new profile. More...
- Order Status -- view your order history or check the status of orders en route. More...
- Keep Me Posted -- view your email notifications. More...
- Password Reminder -- send yourself an email containing your password. More...

For detailed information on what you can do with Your Account, click the "More..." link next to your topic of interest or simply scroll down this page.

Thanks and we hope you enjoy the flexibility available with Your Account.
wsj.com

- Authenticate subscribers with stateless servers
- Half million paid-subscriber accounts
- Purchase articles, track stock portfolios
The server interactive.wsj.com wishes to set a cookie that will be sent to any server in the domain .wsj.com. The name and value of the cookie are: fastlogin=\*.*.*.*.*.*.*.*.*.*.*.*.*. This cookie will persist until Sun Feb 25 07:26:53 2001. Do you wish to allow the cookie to be set?
Background: The crypt() hash function

- **Hash function “salted” with 12 extra bits**
  - Prevent attacker from building dictionary of hashes of common passwords
  - Permute the hash function based on 12-bit seed
  - Prepend seed to hashed password for use in verification

- **Produces one-way function of password**
  - Only hashes first 8 characters
  - Encrypt 0s 25 times with password as key

- **Used by Unix login**
  - So put hashed password in world-readable /etc/passwd
  - To validate password, hash it and compare to stored hash
wsj.com analysis

- **Design:** fastlogin = \{user, \text{MAC}_k (user)\}
- **Reality:** fastlogin = 
  \text{user} + \text{UNIX-crypt (user + server secret)}

- Easily produce fastlogin cookies

  \begin{tabular}{lll}
  username & crypt() Output & fastlogin cookie \\
  \hline
  bitdiddl & \text{MaRdw2J1h6Lfc} & \text{bitdiddlMaRdw2J1h6Lfc} \\
  bitdiddle & \text{MaRdw2J1h6Lfc} & \text{bitdiddleMaRdw2J1h6Lfc} \\
  \end{tabular}

- Usernames matching first 8 characters have same authenticator

- No revocation or expiration.

- This is already bad, but it gets worse...
Obtaining the server secret?

- Adaptive chosen message attack
- Perl script queried WSJ with invalid cookies
- Runs in max $128 \times 8$ queries rather than intended $128^8$ ($1024$ vs. $72057594037927936$)
- 1 sec/query yields 17 minutes vs. $10^9$ years
- The key is “March20”
How the attack works

<table>
<thead>
<tr>
<th>Secret guess</th>
<th>username</th>
<th>crypt input</th>
<th>worked?</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitdiddl</td>
<td>bitdiddl</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>bitdidd</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>bitdidd</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MA</td>
<td>bitdid</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Ma</td>
<td>bitdid</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>March20</td>
<td>b</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Lack of cryptography: highschoolalumni.com

- Problem: No cryptographic authentication at all
- Cookie authenticator is the public username and public user ID
The server www.highschoolalumni.com wishes to set a cookie that will be sent to any server in the domain .highschoolalumni.com. The name and value of the cookie are: Beacon=hsareg; expires=Fri Apr 27 06:07:05 2004. This cookie will persist until Tue Apr 27 06:07:05 2004. Do you wish to allow the cookie to be set?
Leaking secrets: sprintpcs.com

- Problem: Secure content can leak through plaintext channels.
- Site didn’t set SSL flag on cookies (like BankOnline.com)
- User logs in with HTTPS, then clicks back to main HTTP page.
- Vulnerable to passive eavesdropper.
The server m27.sprintpcs.com wishes to set a cookie that will be sent to any server in the domain sprintpcs.com. The name and value of the cookie are:
SPCS%5FRM=RM%5FON=Y&CN1=cono&915=cono

This cookie will persist until Tue Mar 27 19:01:45 2001

Do you wish to allow the cookie to be set?
Google

- Google indexed many cookie files inadvertently places on the Web.

- Search for:
  - cookies.txt
  - avenuea.com FALSE FALSE (cookie set by advertising co.)
  - CERT7.DB or text:CERT7.DB (in many cookies.txt files)
A simple scheme that works

\[
\text{auth} = \text{expire} + \text{data} + \text{MAC}_k(\text{expire} + \text{data})
\]

where \(\text{MAC}\) could be HMAC-SHA1,
\(\text{data}\) could be a username or capability, and
\(\'+\'\) denotes concatenation with a delimiter
Secure against interrogative adversary
But of course, MAC what you mean!

- Sign *marshalled* data, not data with multiple interpretations

- badauth = MAC (key, username + expiration)
  - (Alice, 21-Apr-2001) → MAC (key, Alice21-Apr-2001)
  - (Alice2, 1-Apr-2001) → MAC (key, Alice21-Apr-2001)

- Same authenticator!

- Use unambiguous representation or delimiters