Web Security
Web Security

- Crypto (SSL)
- Client security
- Server security
SSL

Web Security

SSL

SSL

Trusting SSL

The Server’s

Knowledge of the

Client

SET

The Failure of SET

Aside: The SET

Root Certificate

The Client’s

Knowledge of the

Server

Who Issues Web

Certificates?

Mountain America

Credit Union

A Fake Certificate

A Technical Attack

Conclusions on SSL

Recent Changes in TLS

Protecting the Client

Active Content

Web Authentication

SSL
- Mostly covered last time
- Crypto is insufficient for Web security
- One issue: linkage between crypto layer and applications
T rusting SSL

- What does the server *really* know about the client?
- What does the client *really* know about the server?
What has SSL told the server?
- Unless client-side certificates are used, absolutely nothing
- SSL provides a secure pipe. Someone is at the other end; you don’t know whom
- No linkage to transactions
In theory, we could have had digitally-signed purchase orders linked to credit card accounts.
Visa and Mastercard (and eventually Amex) tried, after the Web became popular.
They developed a protocol called SET (Secure Electronic Transactions).
It provided client-side certificates linked to credit cards.
In theory, merchants wouldn’t need to know (and store) credit card numbers.
Virtually no one used it.
The reasons were both technical and financial.
The Failure of SET

- It required client-side software
  - ⇒ Very few people install extra software
- Client-side certificates are hard to use — what if you use several computers?
- There was too little financial incentive for merchants, so they couldn’t give customers a discount for using SET
- It *still* permitted merchants to store credit card numbers; in fact, they were present, albeit encrypted, in the certificate
  - ⇒ Merchants use credit card numbers as customer tracking keys for databases
- Good crypto alone isn’t sufficient!
Who should control the SET root certificate, used to sign the Visa, Mastercard, etc., top-level certificates?

(SET certified Visa et al.; they certified banks, who in turn issued customer certificates)

It would be catastrophic if the root’s private key were compromised

Visa didn’t trust Mastercard, or vice-versa

Solution: a sacrificial PC signed all of the second-level certificates, at which point it was physically smashed. Different organizations took home different pieces...
The Client’s Knowledge of the Server

- The client receives the server’s certificate. Does that help?
- A certificate means that someone has attested to the binding of some name to a public key.
- Who has done the certification? Is it the right name?
Who Issues Web Certificates?

- Every browser has a list of built-in certificate authorities.
- The latest version of Firefox has about 180 certificate authorities!
- Do you trust them all to be honest and competent?
- Do you even know them all?
- (One CA has a 512-bit RSA key.)
- (Baltimore Cybertrust is listed. It sold its PKI business in 2003. Are the new owners trustworthy?)
In 2006, someone persuaded a reputable CA to issue them a certificate for Mountain America, a credit union.

The DNS name was www.mountain-america.net.

It looks legitimate, but the real credit union site is at www.mtnamerica.org.

(There’s also www.mountainamerica.com, a Las Vegas travel site)

Which site was intended by the user?
A Fake Certificate

Web Security
SSL
Trusting SSL
The Server’s Knowledge of the Client
SET
The Failure of SET
Aside: The SET Root Certificate
The Client’s Knowledge of the Server
Who Issues Web Certificates?
Mountain America Credit Union

A Fake Certificate
A Technical Attack
Conclusions on SSL
Recent Changes in TLS

Protecting the Client
Active Content
Web Authentication
A Technical Attack

- Usually, you shop via unencrypted pages
- You click “Checkout” (or “Login” on a bank web site)
- The next page — downloaded without SSL protection — has the login link, which will use SSL
- What if an attacker tampers with that page, and changes the link to something different? Will you notice?
- Note that some small sites outsource payment processing...
Conclusions on SSL

- The cryptography itself seems correct
- The human factors are dubious
- Most users don’t know what a certificate is, or how to verify one
- Even when they do know, it’s hard to know what it should say in any given situation
- There is no rational basis for deciding whether or not to trust a given CA
Recent Changes in TLS

Protecting the Client
Active Content
Web Authentication
Recent Changes in TLS

- Client host name
- Client CA list
- More standard PRFs; those are specified in the cipher suites
- Changes to cipher suites
In hosting centers, many web sites (with different DNS names) sometimes share the same IP address.

Distinguished in HTTP by a Host: header.

But — with TLS (or SSL), the server sends its certificate before the Host: header is sent. Which certificate should be offered by the server?

New extension: include the host name in the ClientHello message.
TLS uses hash functions for several things: certificates, MACs, PRFs

What hash functions are supported?

For the entire life of SSL and TLS, we’ve had MD5 and SHA-1 — but MD5 has been cracked and SHA-1 is falling

Which functions are supported by the client?

MACs are easy; that’s part of the cipher suite

New extension: ClientHello announces hash function support

Should have been done originally — but no protocol designer anticipated the hash function problem
Protecting the Client

Web Browser Security
The Attackers’ Goals
Buggy Code
Why Are Browsers So Insecure?
Active Content
Web Authentication
Web Browser Security

- User interface
- Buggy code
- Active content
The Attackers’ Goals

- Steal personal information, especially financial site passwords
- Turn computers into “bots”
- Bots can be used for denial of service attacks, sending spam, hosting phishing web sites, etc.
## Buggy Code

- **All browsers are vulnerable, and getting worse**
- **Browser bugs (Symantec):**
<table>
<thead>
<tr>
<th>Browser</th>
<th>1H2005</th>
<th>2H2005</th>
<th>1H2006</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>25</td>
<td>25</td>
<td>38</td>
</tr>
<tr>
<td>Firefox</td>
<td>32</td>
<td>17</td>
<td>47</td>
</tr>
<tr>
<td>Opera</td>
<td>7</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Safari</td>
<td>4</td>
<td>6</td>
<td>12</td>
</tr>
</tbody>
</table>

- **Exposure period (Symantec):**
<table>
<thead>
<tr>
<th>Browser</th>
<th>2H2005</th>
<th>1H2006</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>25</td>
<td>9</td>
</tr>
<tr>
<td>Firefox</td>
<td>-2</td>
<td>1</td>
</tr>
<tr>
<td>Safari</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Opera</td>
<td>18</td>
<td>2</td>
</tr>
</tbody>
</table>
Why Are Browsers So Insecure?

- Their task is complex
- They are dealing with many untrusted sites
- By definition, browser inputs cross *protection domains*
- It is likely that no browser is significantly better than any other in this regard — they’re *all bad*
Active Content
Active Content

- There’s worse yet for web users: active content
- Typical active content: JavaScript, Java, Flash, ActiveX
- Web pages can contain more-or-less arbitrary programs or references to programs
- To view certain web pages, users are told “please install this plug-in”, i.e., a program
- “Given a choice between dancing pigs and security, users will pick dancing pigs every time.” (Ed Felten)
JavaScript

- No relationship to Java — originally called LiveScript (EvilScript?)
- Source of most recent security holes, in Firefox and IE
- No clear security model
- Crucial link in *cross-site scripting* attacks
AJAX — Asynchronous JavaScript and XHTML

- Permits highly interactive web pages, i.e., Google Maps
- Security implications for client and server are still quite unclear (but are likely to be bad...)
ActiveX

- The biggest active content design error
- Over 1,000 ActiveX controls on a typical new, out-of-the box, machine
- Translation: over 1,000 different pieces of code that can be run by almost any web page
- But wait, there’s more!
Any web page can download other controls
Translation: any web page can download an arbitrary piece of code to run on a user’s machine
The only protection is a digital signature on the downloaded code
But at best that identifies the author — see the previous discussion of certificates!
There is no restriction on what the code can do
Why ActiveX?

- It can be used for some very beneficial things, such as Windows Update.
- It can be used to “enhance” the user’s web experience, i.e., provide dancing pigs.
- Business reasons? Tie web sites to Windows and IE?
- Only IE has ActiveX. This is the single biggest security difference between IE and Firefox.
Web Authentication

Web Security
SSL
Recent Changes in TLS
Protecting the Client
Active Content
Web Authentication
Web Authentication
HTTP Authentication
How They Work
Basic Authentication
User Prompt
Digest Authentication
Password Storage
Limitations of HTTP Authentication
Three options: client-side certificates, HTTP authentication, site-specific

Client-side certificate uses SSL

Storing and protecting the private key is hard

Where does the key live? How is it moved from machine to machine?

Site-specific — a login screen — is by far the most common
HTTP Authentication

- Transaction between the web browser and the web server
- Two types, Basic and Digest
- Generally used together with SSL
- Often seen as unaesthetic
How They Work

- Client sends an HTTP request
- Server replies with a WWW-Authenticate: challenge
- Client prompts user for credentials
- Client retries request with Authorization: header included
- Can be used to authenticate to proxies, but that’s rare
Basic Authentication

- Server send a challenge with a realm
- Realm is displayed to the user (but not tied to a certificate)
- Client replies with base-64 encoded (but not encrypted) password
- For userid **Aladdin** and password **open sesame**, client sends

  Authorization: Basic QWxhZGRpbjpvcGVuc2l0b3IgQ29uc3V0

  which is **Aladdin:open sesame in base 64**
User Prompt

Enter username and password for "File Access" at https://www.cs.columbia.edu

User Name:

Password:
Digest Authentication

- Uses challenge/response authentication
- Server sends a nonce in the WWW-Authenticate: message
- Client reply includes MD5 hash of username, password, nonce, HTTP method, and requested URL
- Can’t replay, because the nonce will be different each time
- Password not sent in the clear
- (Actually somewhat more complex than this)
Password Storage

- With Basic authentication, Unix-style hashed passwords can be stored
- Digest (and most forms of challenge/response) require plaintext passwords
- That file can be stolen — and people often reuse their passwords for other web sites
- Note that this applies to web page-based authentication, too; it’s not a limitation of HTTP authentication
Limitations of HTTP Authentication

- No fancy login screen
- No “Forgot your password?” link
- No easy recovery from authentication failure; just a 401 error
- Generally used only by low-end web sites
- Not very friendly for token-based authentication (though Digest is better)