CSE 484/M584: Computer Security (and Privacy)

Spring 2025

David Kohlbrenner
dkohlbre@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner, Nirvan Tyagi. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials

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

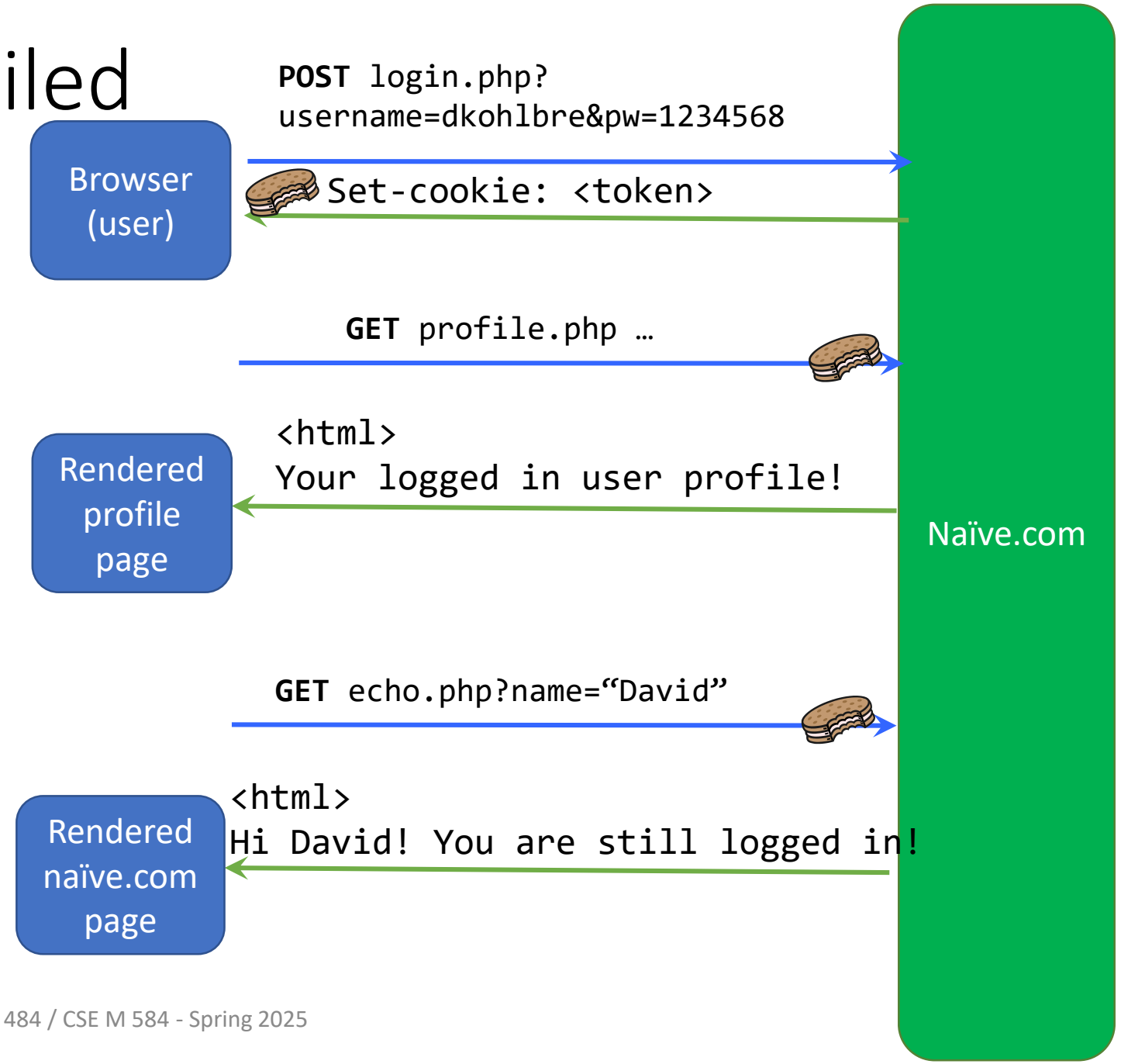
In all XSS there are 3 actors

- Adversary
- Server victim
- User victim

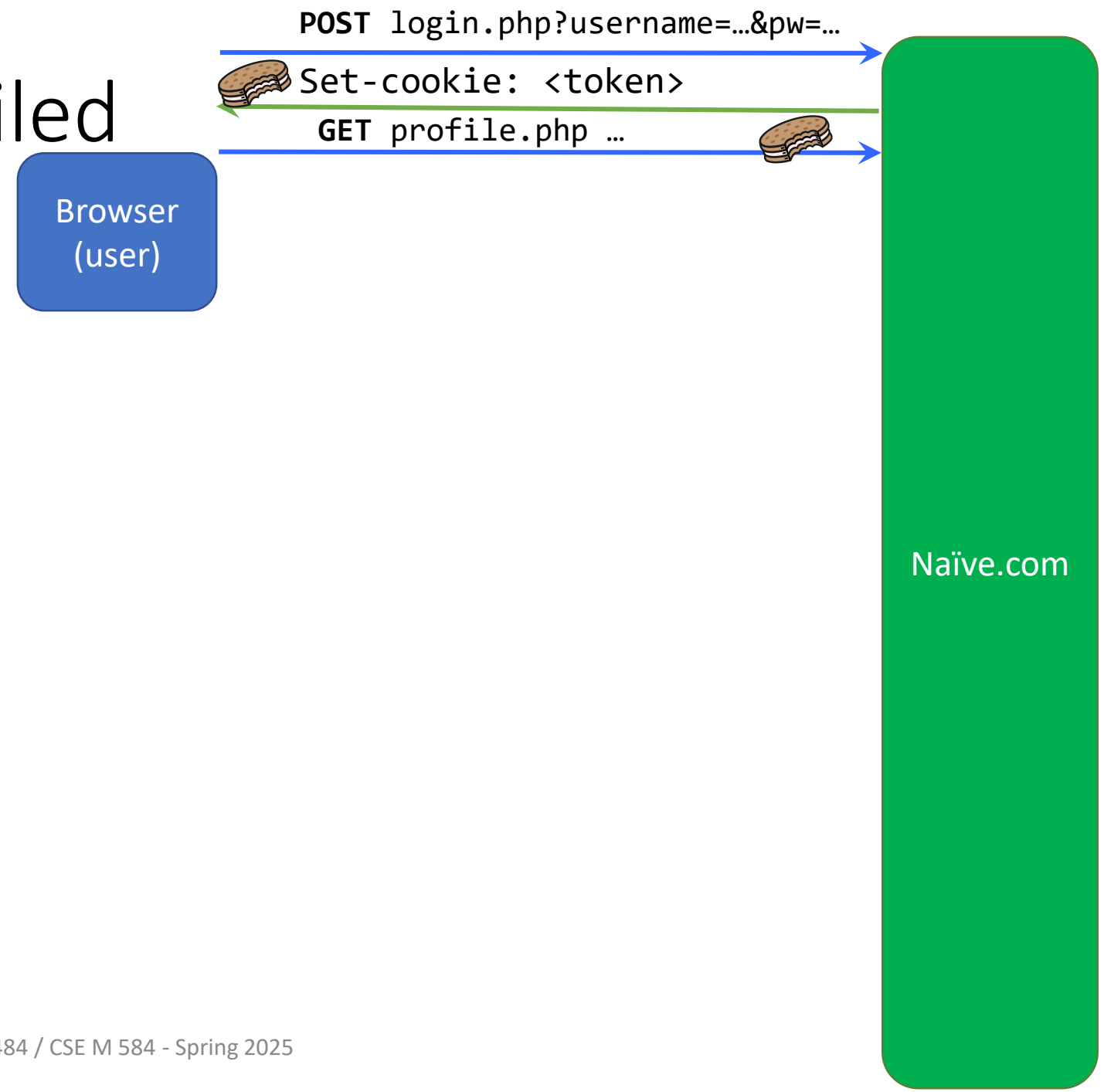
Reflected XSS

- 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

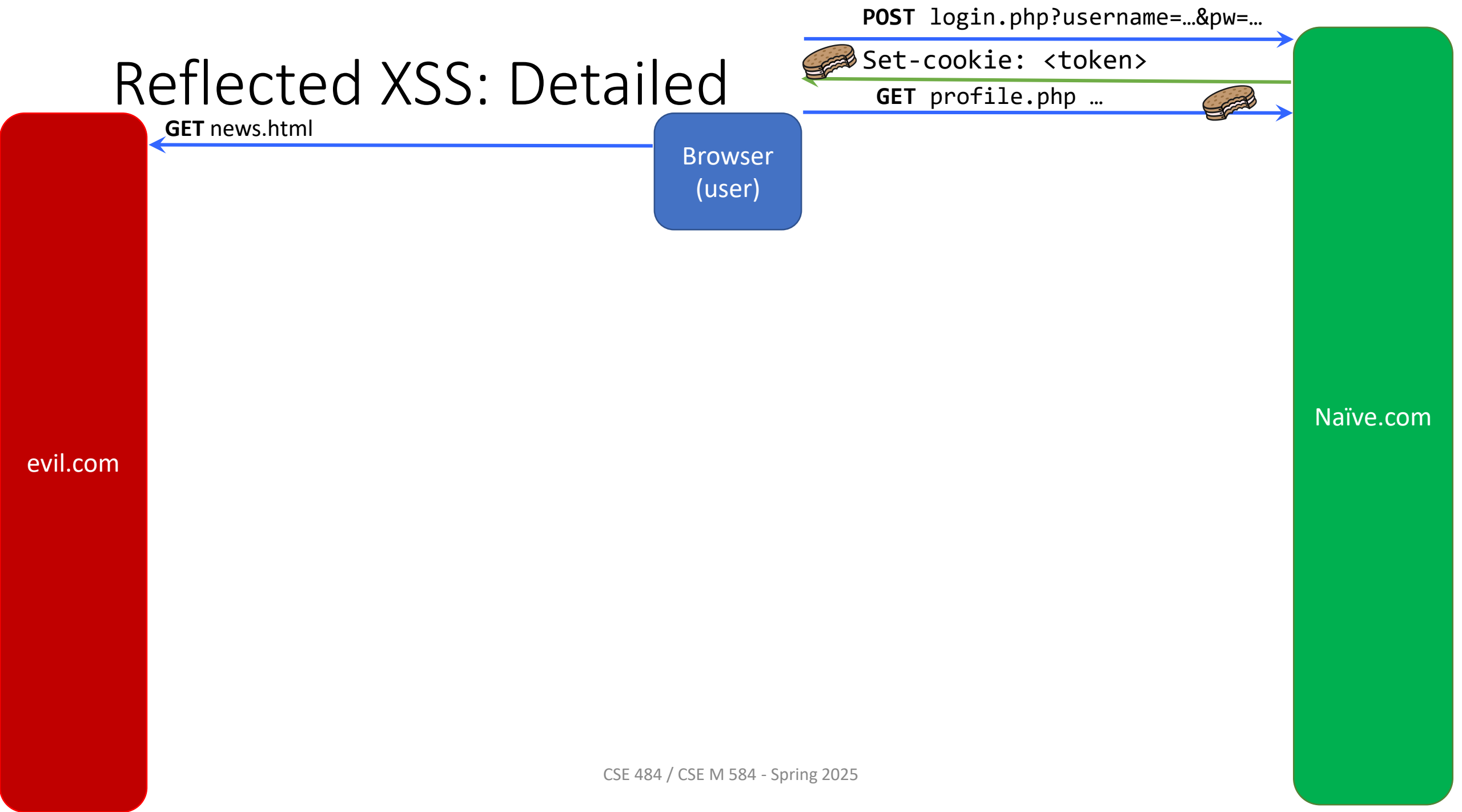


Reflected XSS: Detailed

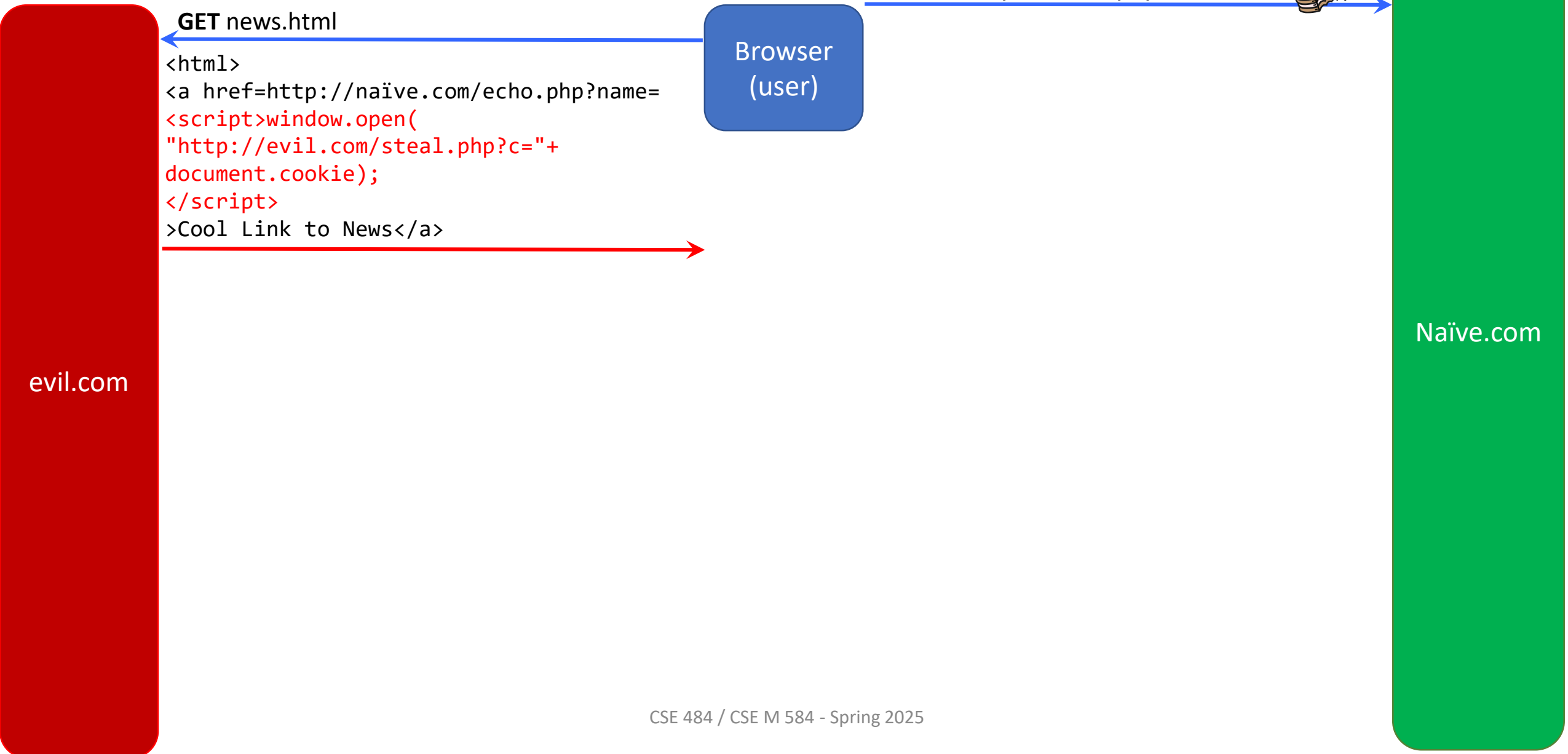


Reflected XSS: Detailed

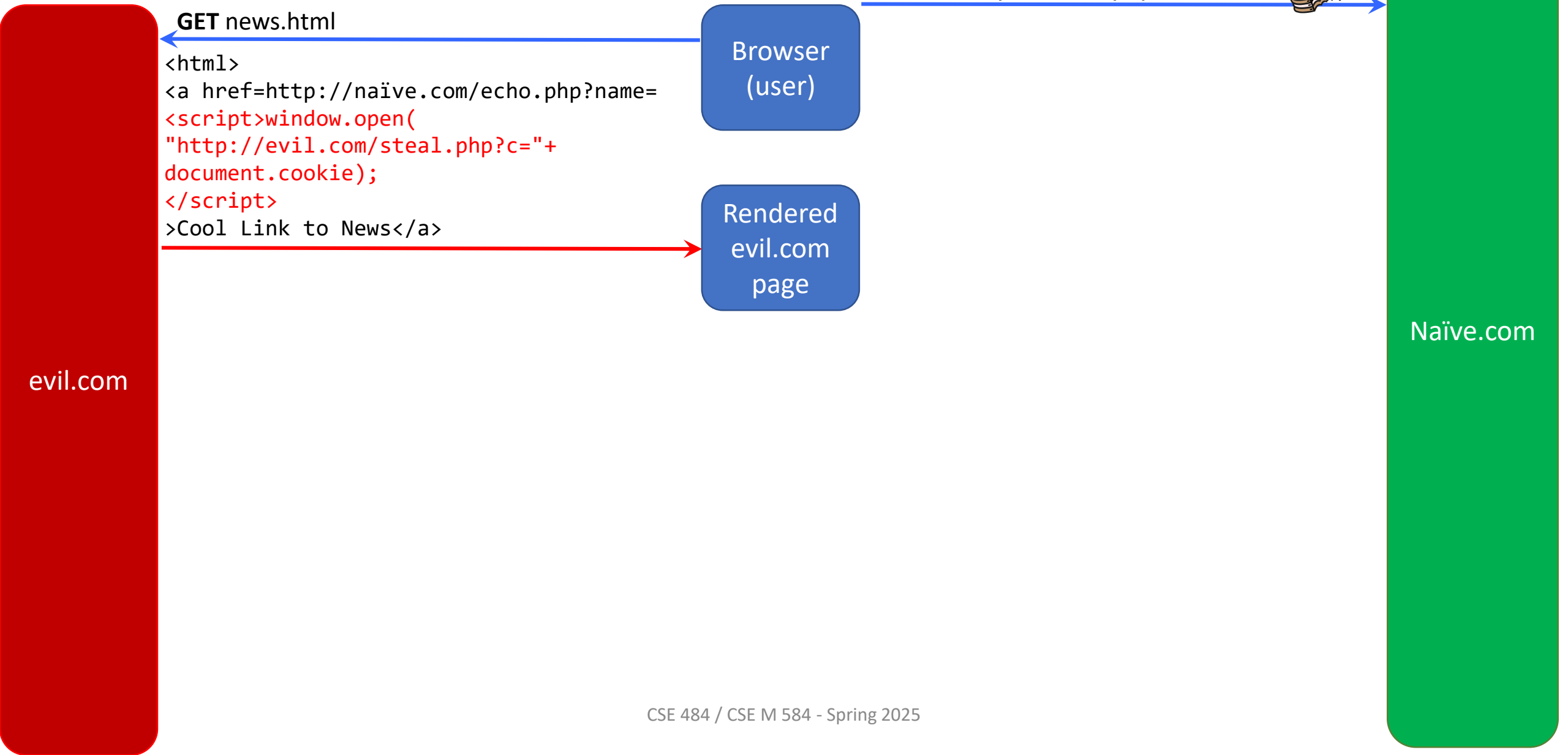
Reflected XSS: Detailed



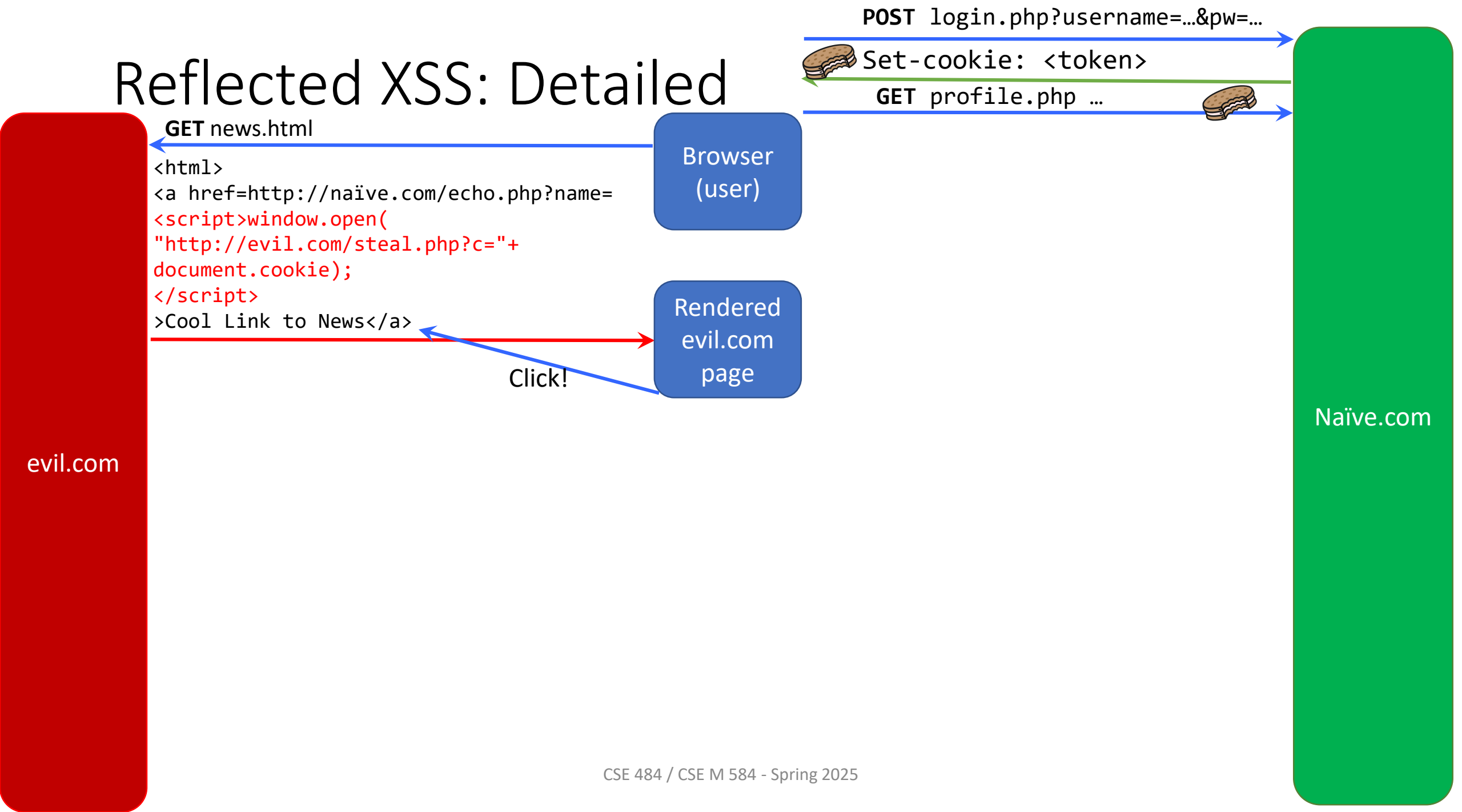
Reflected XSS: Detailed



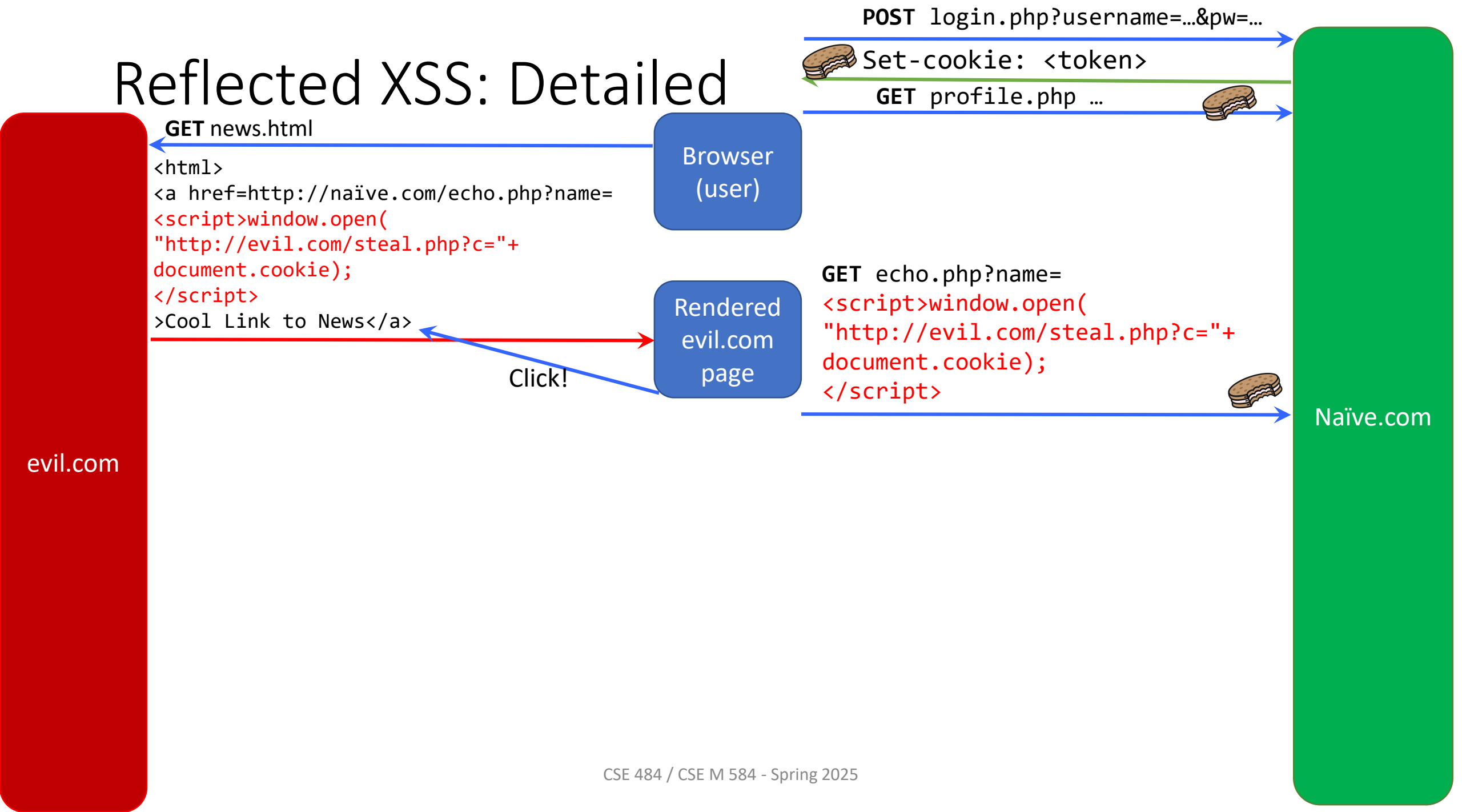
Reflected XSS: Detailed



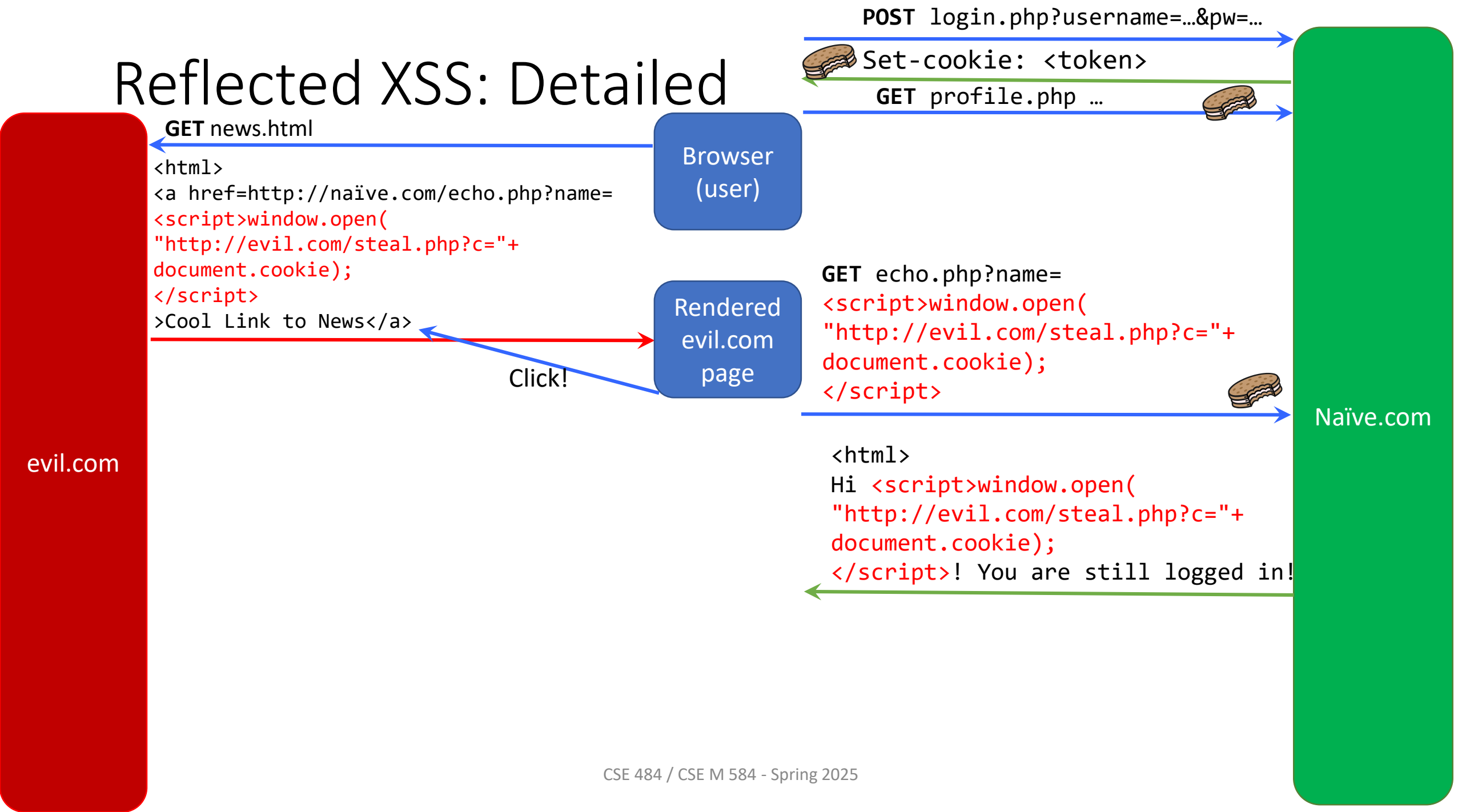
Reflected XSS: Detailed



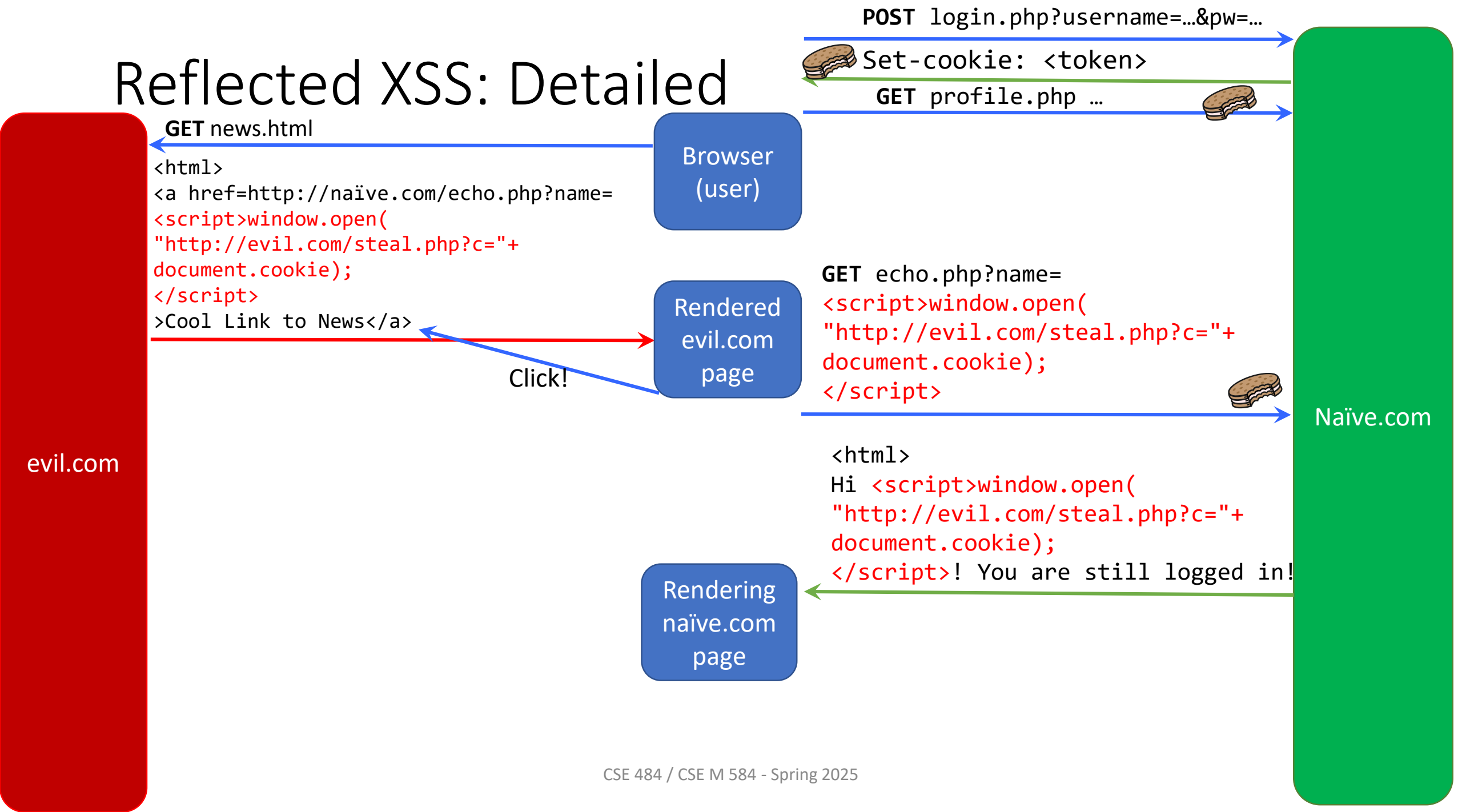
Reflected XSS: Detailed



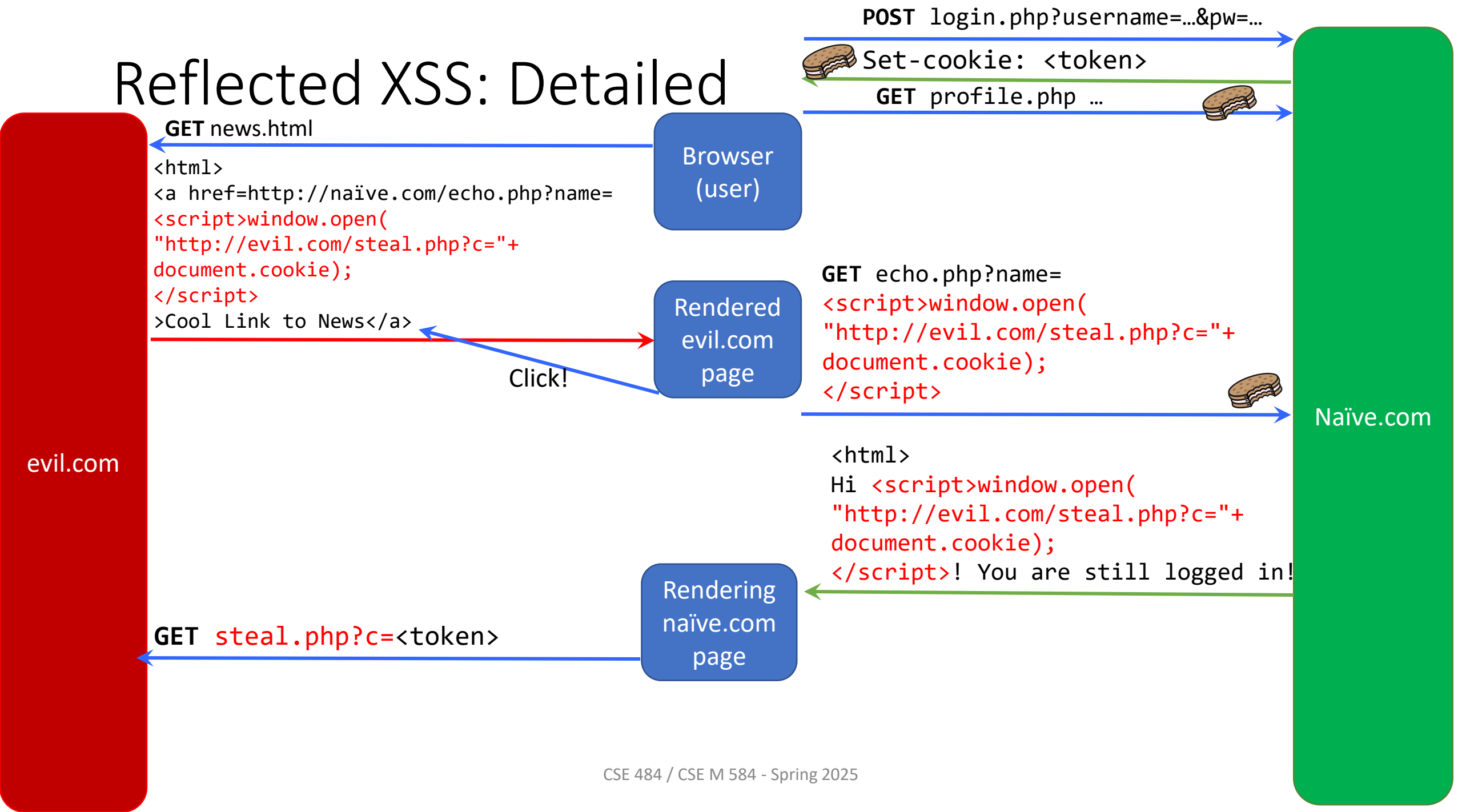
Reflected XSS: Detailed



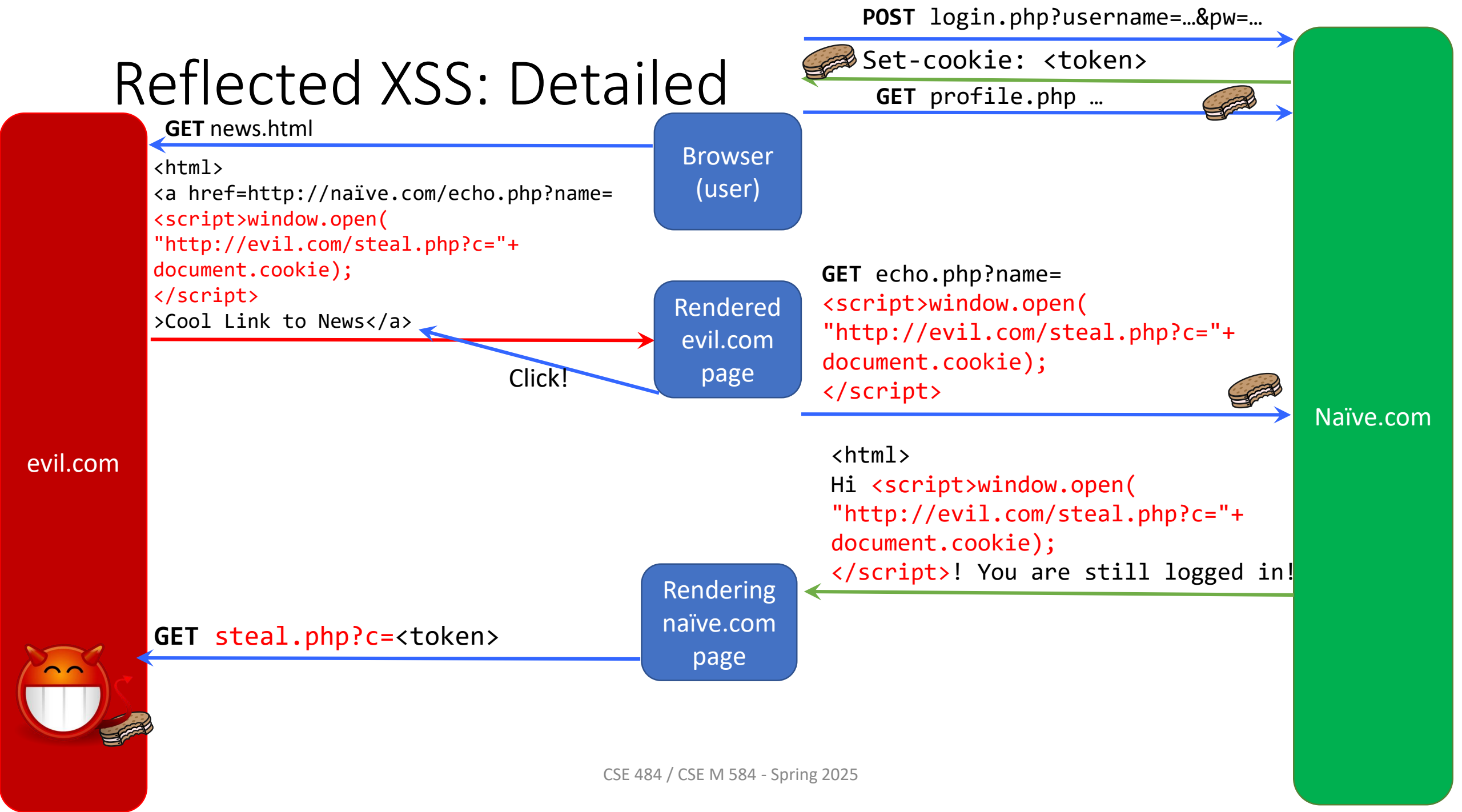
Reflected XSS: Detailed



Reflected XSS: Detailed

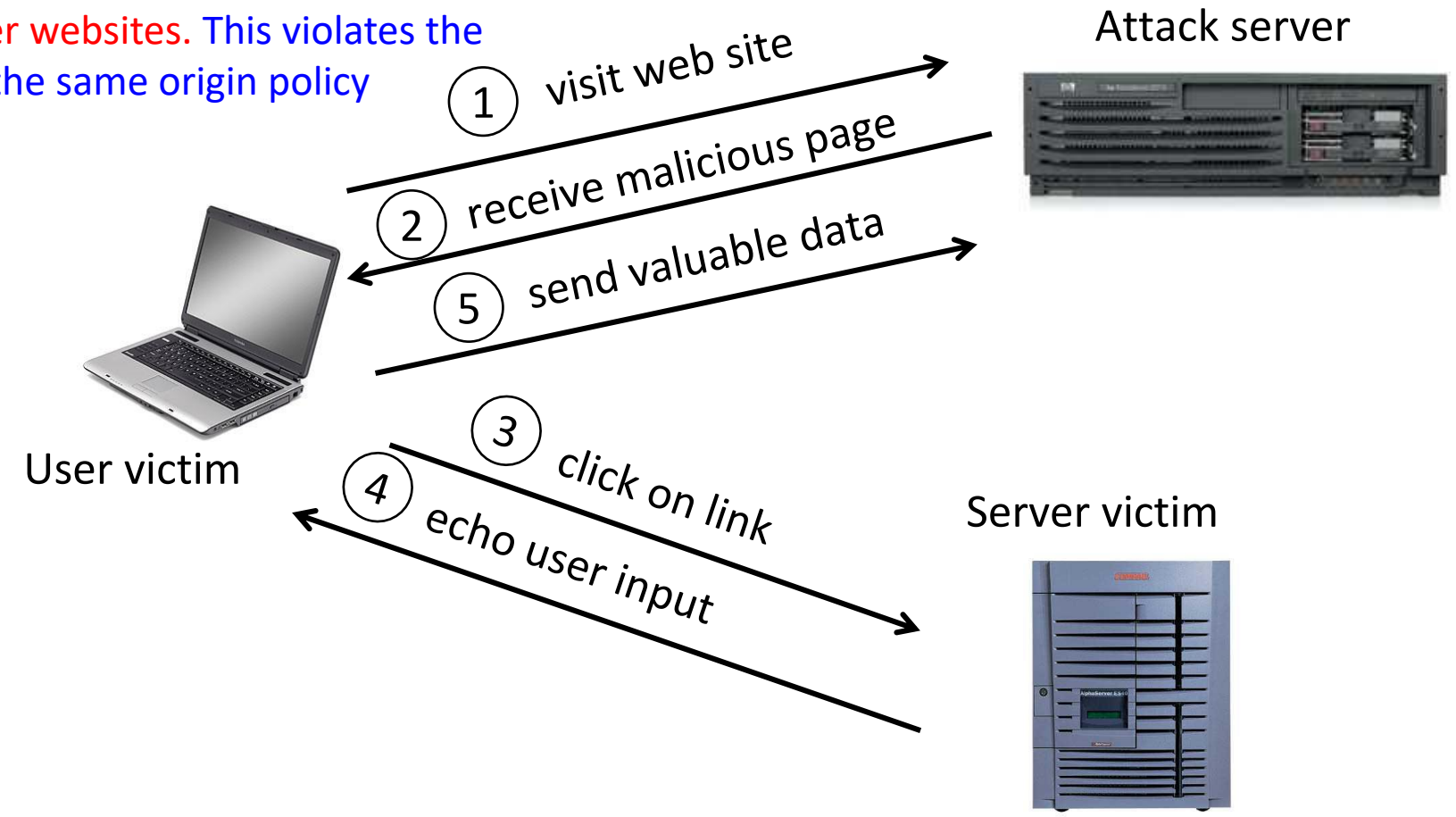


Reflected XSS: Detailed

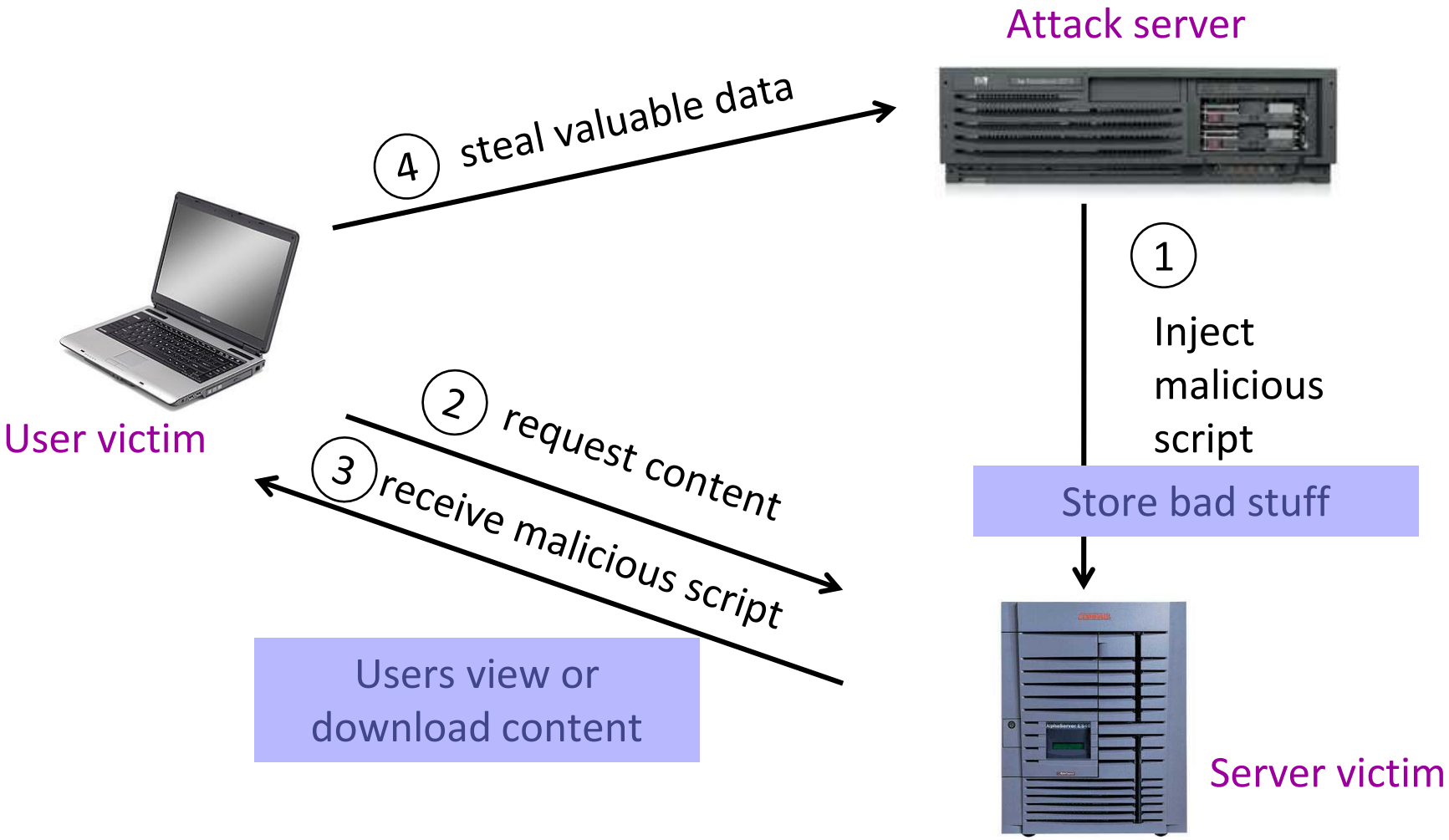


Basic Pattern for Reflected XSS

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

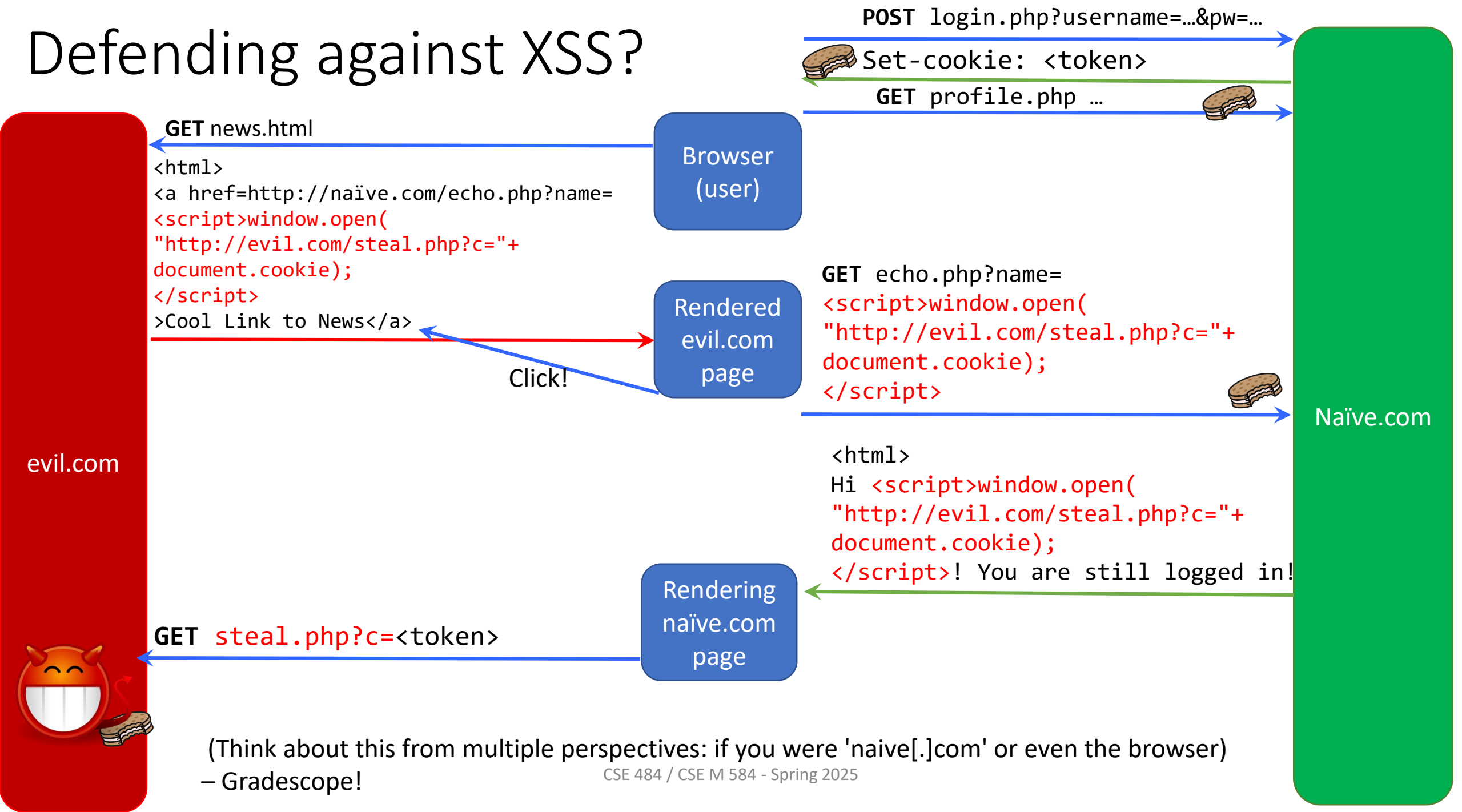

Attack server
4 steal valuable data
1
Inject malicious script
2 request content
3 receive malicious script
User victim
Store bad stuff
Users view or download content
Server victim

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?



(Think about this from multiple perspectives: if you were 'naive[.]com' or even the browser) – Gradescope!

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

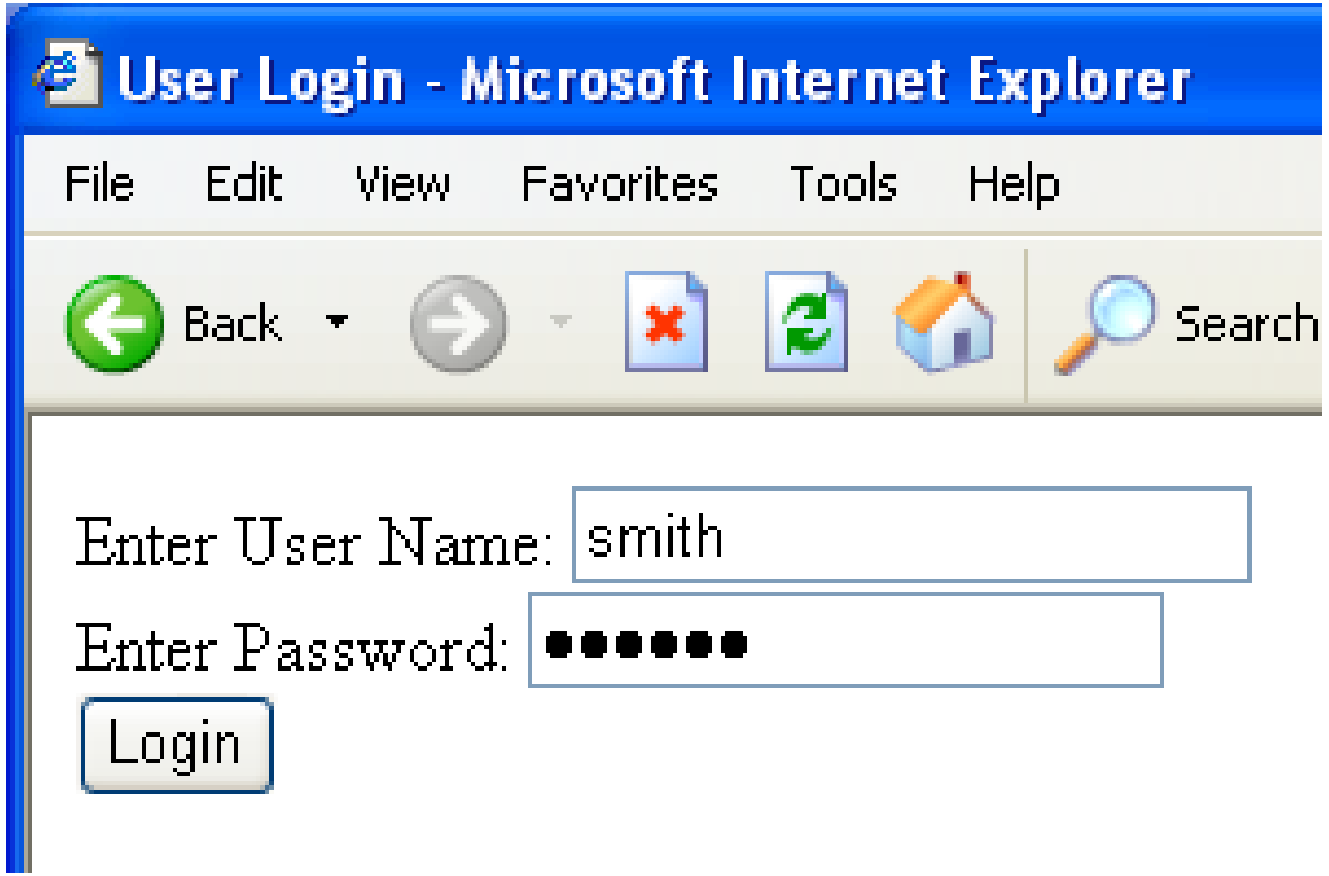
- Preventing injection of scripts into HTML is hard! → Use standard APIs
  - Blocking “<” and “>” is not enough
  - Event handlers, stylesheets, encoded inputs (%3C), etc.
  - phpBB allowed simple HTML tags like <b>

    <b c=“>” onmouseover=“script” x=“<b ”>Hello<b>
- Beware of filter evasion tricks (XSS Cheat Sheet)
  - If filter allows quoting (of <script>, etc.), beware of malformed quoting:
    <IMG """><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



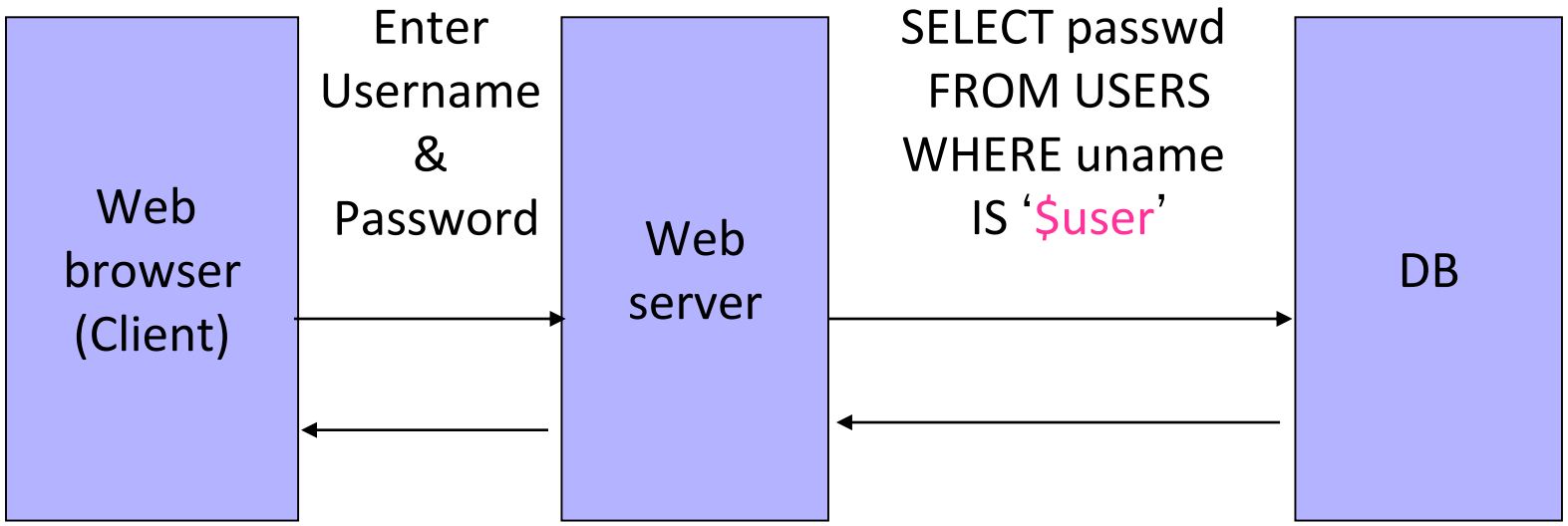
Typical (bad) Query Generation Code

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

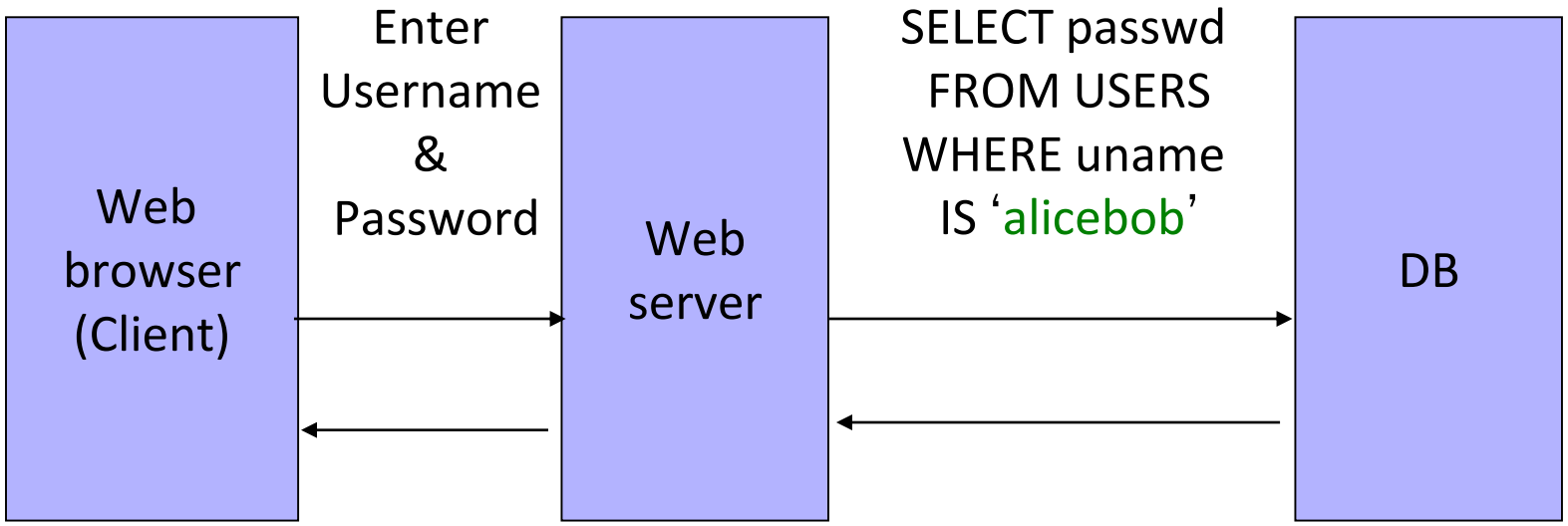
What if `'user'` is a malicious string that changes the meaning of the query?

Demo!

User Input Becomes Part of Query

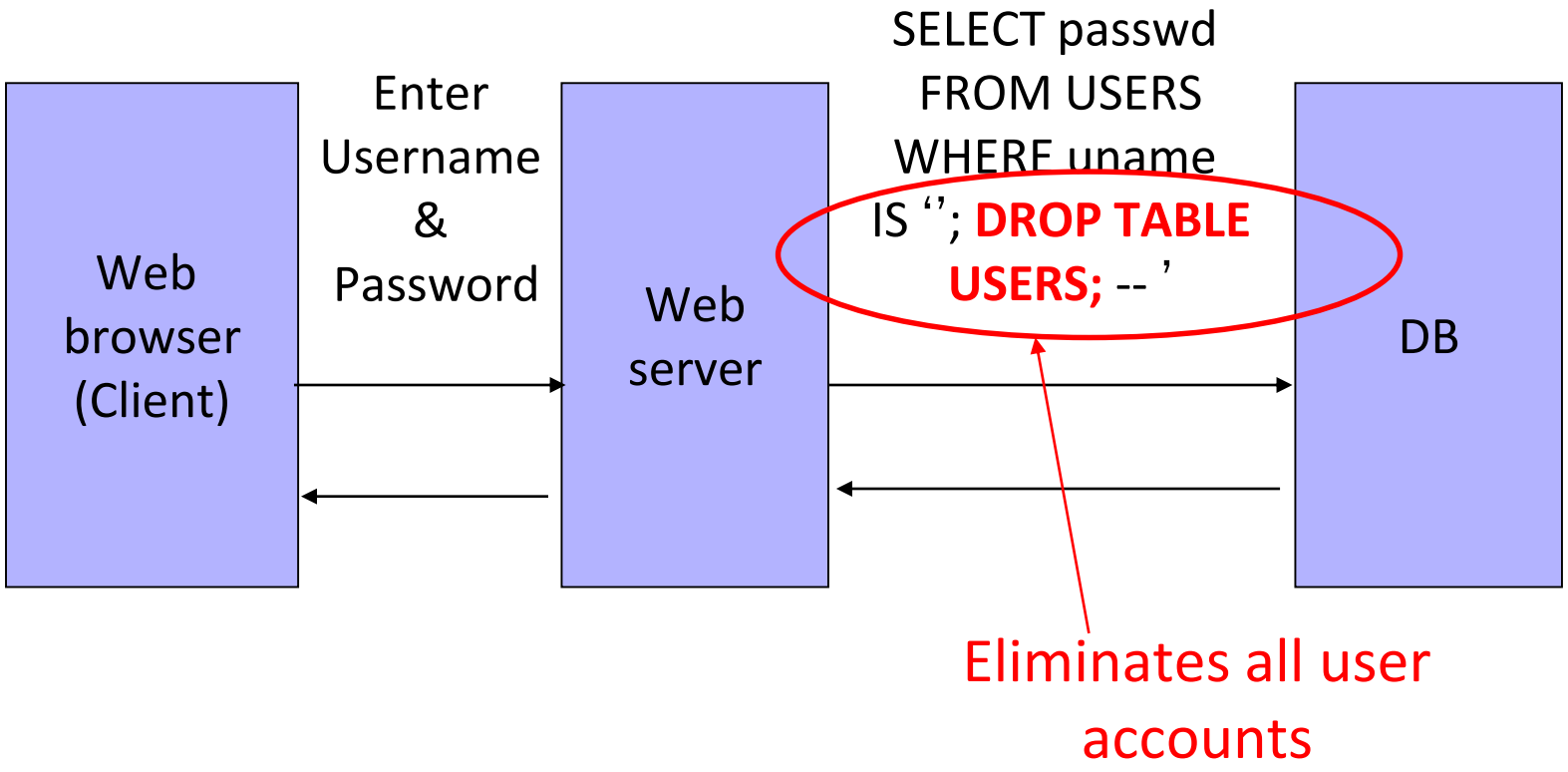


Normal Login



Malicious User Input

SQL Injection Attack



Cross-Site Scripting (XSS) (alt diagram)