CSc 466/566

Computer Security

21 : Web Security — Introduction

Version: 2019/11/13 09:59:58

Department of Computer Science
University of Arizona

collberg@gmail.com

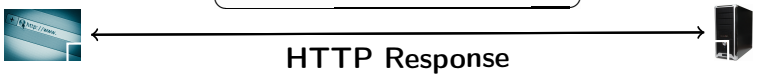
Christian Collberg

Outline

Static Web Content

HTTP Request

```
GET /index.html HTTP/1.1
Host: www.site.com
```



HTTP Response

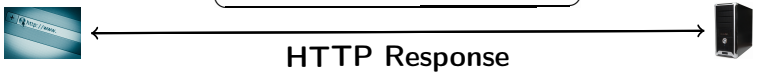
```
HTTP/1.1 200 OK
Server: Appache
Date: Mon, 16 Apr 2012 21:44:29 GMT
Expires: -1
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: ...
Content-Length: 314

<!doctype html>
<html><body>
...
</body></html>
```


HTTP/HTML

HTTP Request

```
GET /index.html HTTP/1.1
Host: www.site.com
```



HTTP Response

```
<b>bold text</b>
<ul>
   <li> list item 1
   <li> list item 2
</ul>
<a href="site.com/boat.jpg">Link!</a>
<script>
   document.location=...
</script>
<img src="boat.jpg">
```

Exercise: HTTP Requests and Responses

- Install curl on your computer: https://curl.haxx.se.
- Try this command that shows the outgoing message:

```
> curl -lvs http://example.com | & grep '> '
```

- What is a "User-Agent"?

Exercise: HTTP Requests and Responses...

- Try this command which shows the returned page:

```
> curl -i http://www.example.com
```

Can you identify the sections of the returned page?

Forms

HTTP Request

```
www.site.com/register.php?
name="Alice"&
email="alice@gmail.com"
```

www.site.com/register.php

```
<html><title>Registration</title>
<HTML>
   <TITLE>Registration</TITLE>
<BODY>
   <FORM ACTION="register.php" METHOD="GET">
      <INPUT TYPE="text" NAME="name">
      <INPUT TYPE="text" NAME="email">
      <INPUT TYPE="submit" VALUE="Submit">
   </FORM>
</BODY>
</HTML>
```


Confidentiality

- HTTP requests and responses are delivered via TCP on port 80.
- All traffic is in the clear!
- MITM attacks.

Exercise: Sending Form Data

- Try this command that is sent to a site that just echos what you sent it:

```
> curl --request POST --url https://postman-
    echo.com/post -d'name=alice' -d'password=
    bob4evah'
```

Exercise: Sending Form Data...

- Add –trace-ascii /dev/stdout to the previous command:

```
> curl --trace-ascii /dev/stdout ...
```

- Can you see how the form data is sent?

Outline

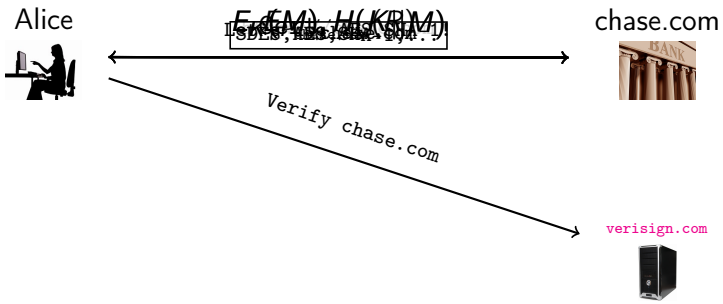
HTTP over Secure Socket Layer (HTTPS)

1. Alice browses to `https://chase.com`
2. The browser sends `chase.com` a list of cryptographic ciphers/hash functions it supports.
3. The server selects the strongest ciphers/hash functions they both support.
4. `chase.com` tells the browser of its cryptographic choices.
5. `chase.com` sends the browser its certificate $\texttt{Cert}_{\texttt{chase.com}}$, containing its public key $P_{\texttt{chase.com}}$.

HTTP over Secure Socket Layer...

6. The browser verifies the authenticity of $\texttt{Cert}_{\texttt{chase.com}}$.
7. Browser generates a random number R.
8. The browser encrypts R with $P_{\texttt{chase.com}}$ and sends it to chase.com.
9. Starting from R, the browser and chase.com generate a shared secret key K.
10. Subsequent messages M: send $E_K(M), H(K||M)$.

HTTP over Secure Socket Layer (HTTPS)



Exercise: https

- Try to connect to https://example.com:

```
> curl -v https://example.com
```

- Can you find the exchange of cryptographic primitives?
- Who issued the server certificate?

Outline

Dynamic Content

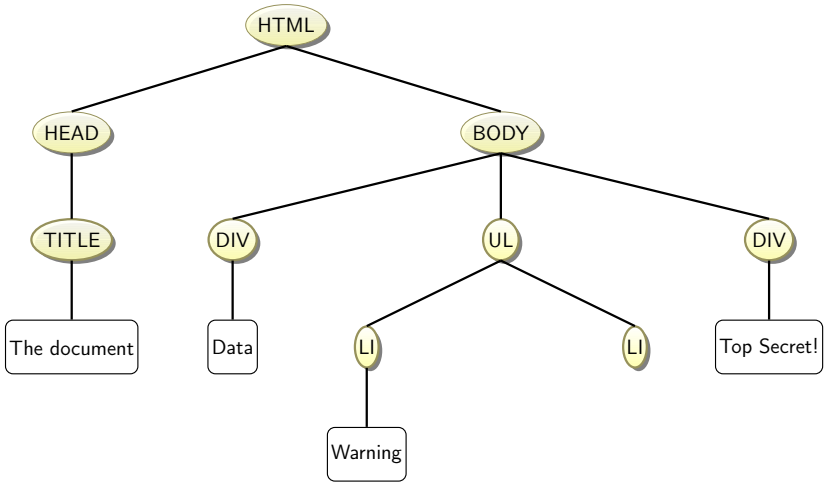
- Plain `html` pages are static.
- Dynamic content can change, even without reloading the page.
- Client-side scripts are included in web pages to provide dynamic content.
- Web pages are represented internally in the browser as DOM trees (Document Object Model).
- Scripts can manipulate the DOM tree.
- Most scripts are written in JavaScript.

DOM Tree Example

```
<html>
    <head>
        <title>The document</title>
    </head>
    <body>
        <div>Data</div>
        <ul>
            <li>Warning</li>
            <li></li>
        </ul>
        <div>Top Secret!</div>
    </body>
</html>
```

Source: http://javascript.info/tutorial/dom-nodes

DOM Tree Example...



Exercise: Visualizing the DOM Tree

- Go to https://codepen.io/pavlovsk/pen/QKGpQA and enter this text:

```
<div id="tree" class="tree">
<ul>
   <li> Warning
</ul>
<div> Top Secret! </div>
<ol>
   <li> Error
</ol>
</div>
```

JavaScript

- JavaScript code can be included within HTML documents:

```
<script type="text/javascript">
function hello() {
   alert("Hello world!");
}
</script>
```

- JavaScript functions can be invoked as a result of clicks, etc.:

```
<img src="..."
     onMouseOver="javascript:hello()">
```


DOM Tree Traversal

- DOM tree node properties:

name	description
firstChild, lastChild	start/end of this node's list of children
childNodes	array of all this node's children
nextSibling, previousSibling	neighboring nodes with the same parent
parentNode	the element that contains this node

- Thus, you can traverse the DOM tree from within JavaScript:

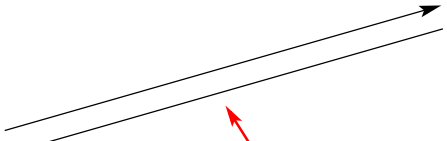
```
window.document.childNodes[0].childNodes[1].
   childNodes[4]
```


Sessions

- HTTP is a state-less protocol:
  - every time a browser asks for a page is a new event to the server;
  - the server keeps no information (automatically) between page loads.
- A session is extra information stored about a visitor between interactions.
- Three methods to keep track of sessions:
  1. Hidden form fields,
  2. Client-side cookies,
  3. Server-side session.
- We must protect against session hijacking — an attacker getting hold of a user's session information and impersonating him to the server.

HTTP is Stateless

www.site.com



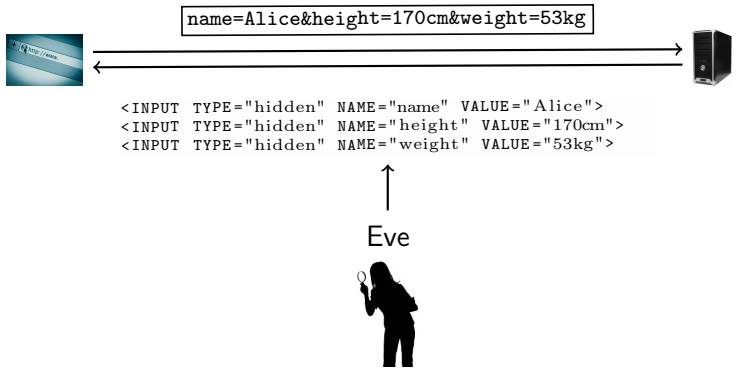
Sessions Using Hidden Form Fields

- Any information that needs to survive between interactions is stored in the browser in hidden fields in the HTML.
- The information is sent back to the server in POST or GET requests.

```
<HTML><BODY><FORM
   ACTION="http://www.victoriassecret.com/buy.jsp"
   METHOD="get">
<INPUT TYPE="hidden" NAME="name" VALUE="Alice">
<INPUT TYPE="hidden" NAME="weight" VALUE="53kg">
<INPUT TYPE="submit">
</FORM></BODY></HTML>
```

- HTTP is sent in cleartext — susceptible to MITM attack.

Sessions Using Hidden Form Fields...



- Use HTTPS instead.

Sessions Using Cookies

- A cookie is a piece of data sent to the client by the web server.
- The cookie is stored on the client.
- When the user returns to the site, the cookie is sent to the web server.

cookie

```
"name"="Alice"
"height"="170cm"
"weight"="53kg"
expire=10 Dec, 2012
domain=.victoriassecret.com
path: /
send for: any type
```


Exercise: Sending/Receiving Cookies

- Try this command which sends cookie data to a server:

```
> curl -c /tmp/cookies 'http://httpbin.org/
    cookies/set?name=alice&pw=bob'
```

Exercise: Sending/Receiving Cookies

- Look at the file which stores the cookies on your machine:

```
> cat /tmp/cookies
```

What information does it have?

Exercise: Sending/Storing/Receiving Cookies...

- Now try this, and check the cookie file again:

```
> curl -v -c /tmp/cookies 'http://httpbin.org/
    cookies/set?name=alice&pw=eric'
```

What changed?

Exercise: Sending/Storing/Receiving Cookies...

- Try this, which sends the cookie file to the server:

```
> curl -v -b /tmp/cookies 'http://httpbin.org/
    cookies'
```

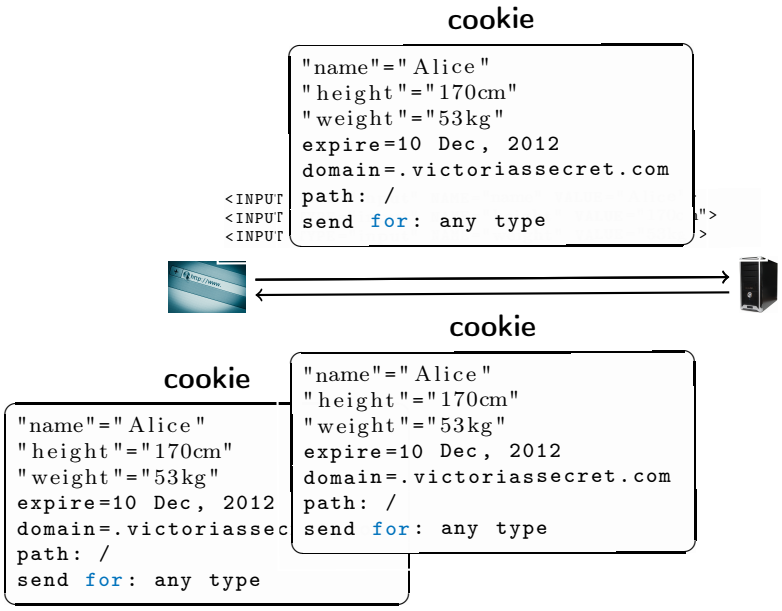
What cookie data was sent?

Sessions Using Cookies

- Let's assume Alice is browsing to http://www.victoriassecret.com.
- She fills out a form with her personal data:

```
<HTML><BODY>
<FORM ACTION="http://www.victoriassecret.com/buy.jsp"
      METHOD="get">
<INPUT TYPE="input" NAME="name"   VALUE="Alice">
<INPUT TYPE="input" NAME="height" VALUE="170cm">
<INPUT TYPE="input" NAME="weight" VALUE="53kg">
<INPUT TYPE="submit">
</FORM>
</BODY></HTML>
```

Sessions Using Cookies. . .



Sessions Using Cookies — Cookie Properties

- Expiration date: none specified, the cookie is deleted when the user exits the browser.
- Domain name — the site for which this cookie is valid.

Cookie Transport

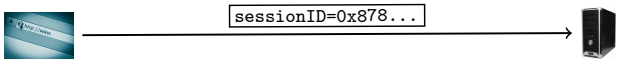
- Cookies, by default, are sent using HTTP.
- MITM attacks!
- Countermeasures:
  1. Set the secure flag: HTTPS is used instead.
  2. Encrypt the cookie value.
  3. Obfuscate the cookie name.

Server-Side Sessions

- User information is kept in a database on the server.
- A session ID (session token) identifies the user's session.
- `GET`/`POST` variables or cookies are used to store the token on the client.
- When the user browses to a page, the token is sent to the server, and the user's data is looked up from the database.

```
<HTML><BODY>
<FORM ACTION="http://www.victoriassecret.com/buy.jsp"
      METHOD="get">
<INPUT TYPE="hidden" NAME="sessionID" VALUE="0x324A...">
</FORM>
</BODY></HTML>
```

Server-Side Sessions



sessionID	data
0x878...	name="Alice",height="170cm",...
0x9A5...	name="Bob",height="180cm",...

- The session ID should be hard to guess.

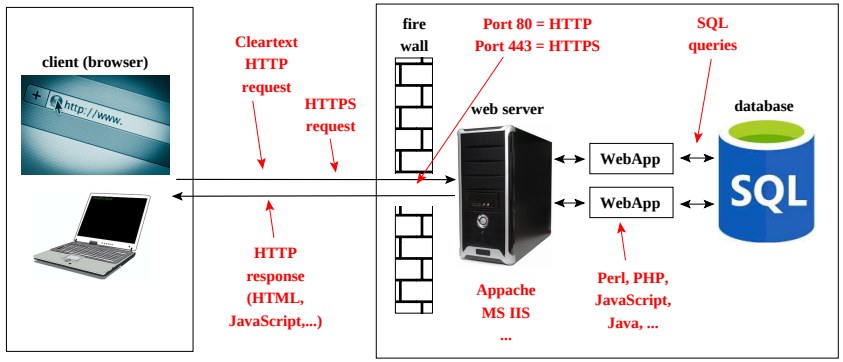
Exercise: Goodrich & Tamassia C-7.8

- Suppose a web client and web server for a popular shopping web site have performed a key exchange so that they are now sharing a secret session key.
- Describe a secure method for the web client to then navigate around various pages of the shopping site, optionally placing things into a shopping cart.
- Your solution is allowed to use one-way hash functions and pseudo-random number generators, but it cannot use HTTPS, so it does not need to achieve confidentiality.
- Your solution should be resistant to HTTP session hijacking even from someone who can sniff all the packets.

Exercise: Goodrich & Tamassia C-7.8...

Outline

Web server Attack Surface



https://www.blackhat.com/presentations/bh-asia-02/bh-asia-02-shah.pdf

What is a URL?

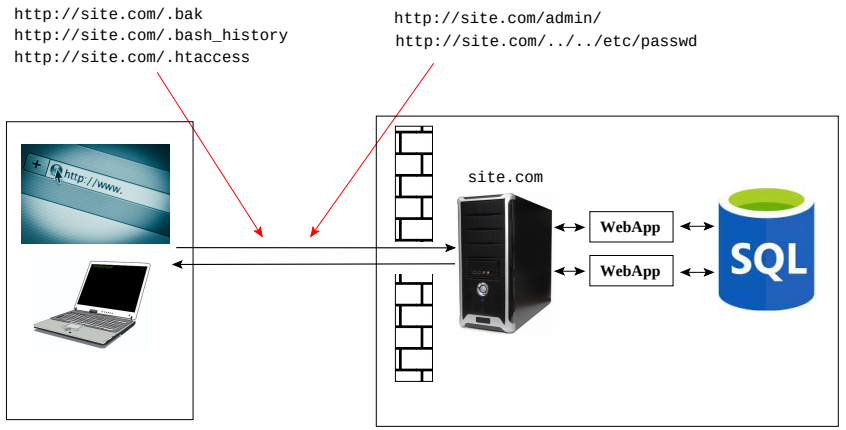
```
http://user:password@www.site.com:80/forum/code.php3
```

1. Protocol: http, https, ftp, ...
2. Username+Password: optional
3. Server: IP address or name
4. Port: 80=HTTP (default), 443=HTTPS
5. Path: Directory path to file on server

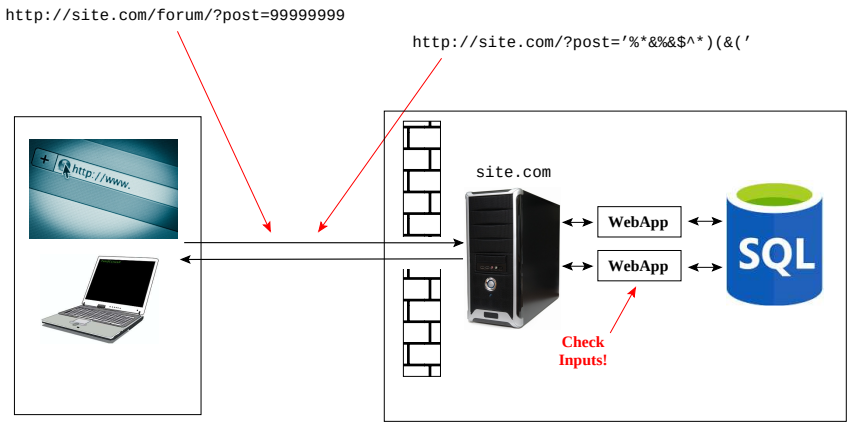
URL Manipulation Attacks

- Try different paths by trial and error
- http://ccm.net/contents/31-url-manipulation-attacks
- https://en.wikipedia.org/wiki/Directory_traversal_attack

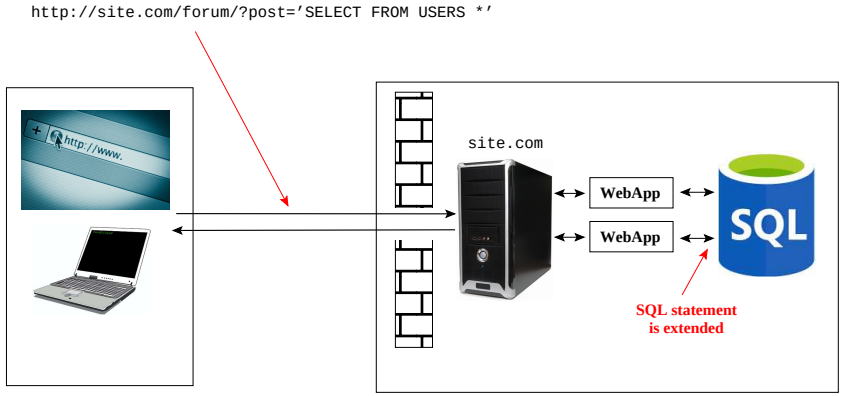
URL Manipulation Attacks...



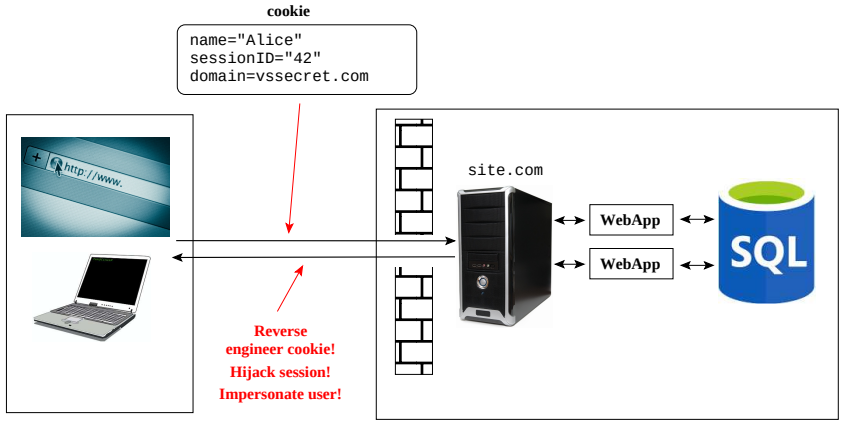
Input Validation Attacks



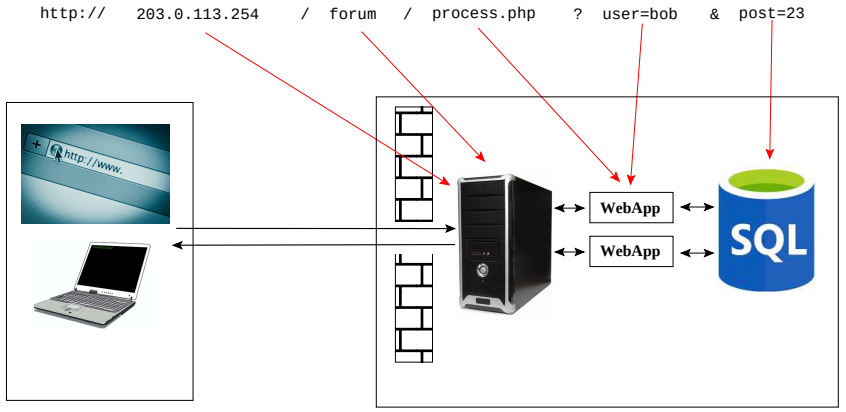
SQL Query Poisoning



Reverse Engineer Cookies

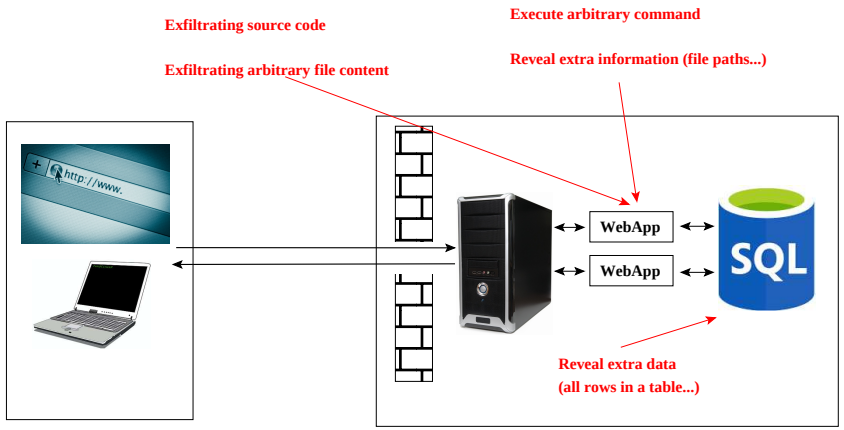


URLs Pointing Into Servers

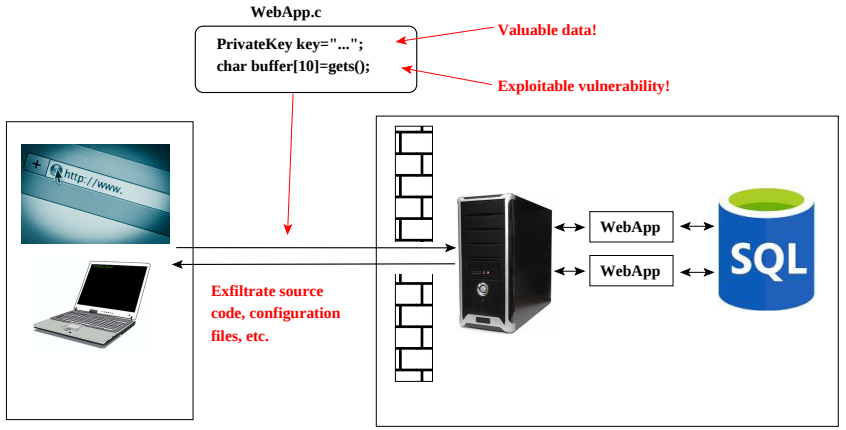


- URLs wind their way into the server.

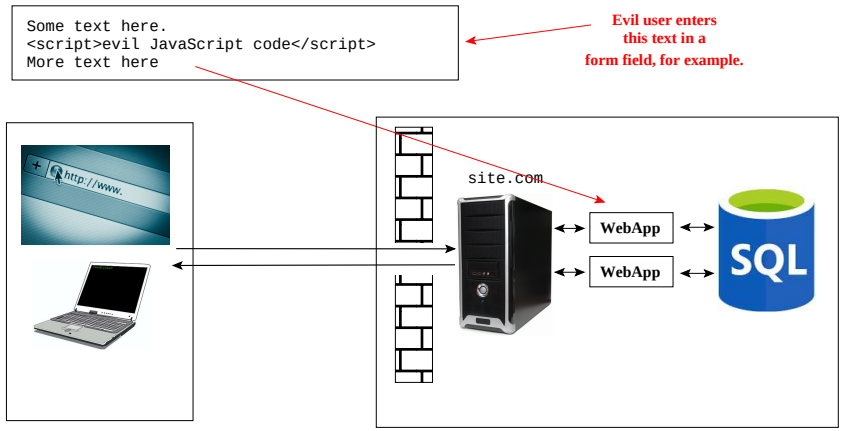
What are the effects?



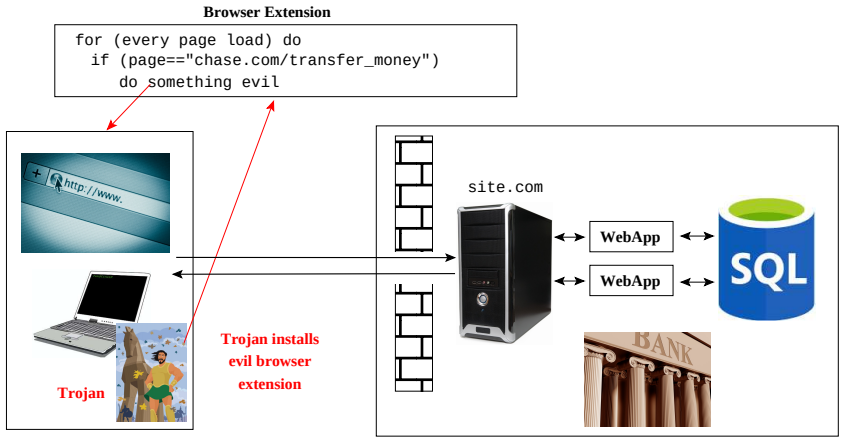
Source Code Disclosure



HTML Script Injection

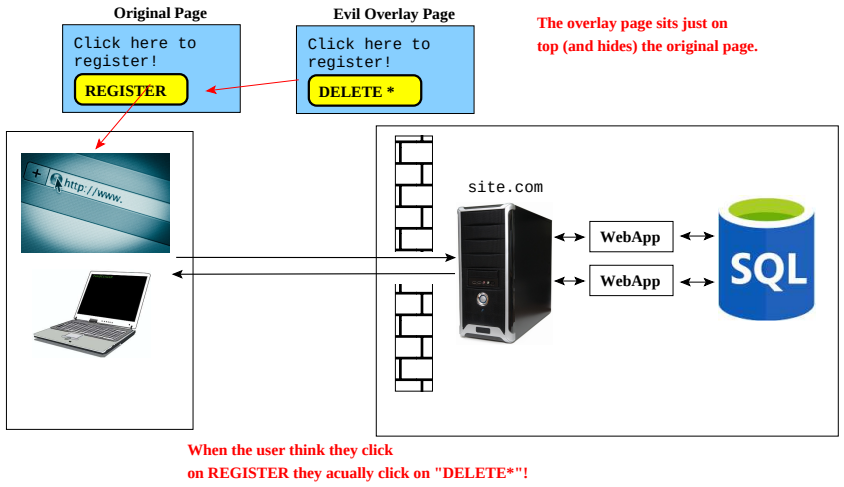


Man-In-The-Browser



https://www.owasp.org/index.php/Man-in-the-browser_attack

Clickjacking



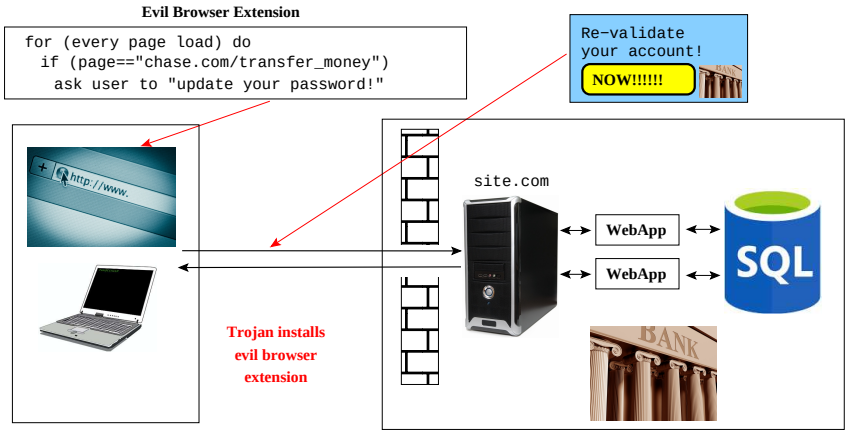
https://www.owasp.org/index.php/Clickjacking

Phishing

- Criminals use social engineering to appear trusted.
- Goal is to gain information that can help in identity theft.
- Steal account details, mother's maiden name, SSN, ...
- Use information to open accounts, get loans, buy stuff, ...
- 5% of users fall for these scams.

https://www.owasp.org/index.php/Phishing

Phishing...



Exercise

- Get in groups with a few friends.
- Go to https://cve.mitre.org/cve/search_cve_list.html. This is a list of known vulnerabilities.
- Search for one of the web attacks above. Record the CVE.
- Tell the class about the vulnerability you found!

Outline

Readings and References

- Chapter 7 in *Introduction to Computer Security*, by Goodrich and Tamassia.