THE WEB AS A DISTRIBUTED SYSTEM
WEB HACKING
SESSION
3-TIER

Client

Web-Server

DB

Presentation
- HTML
- JavaScript
- AJAX

Application Logic
- HTTP
- PHP
- EJB
- WebObjects

Data
- SQL
- Relations

persistent state
SCENARIO

- user visits a service
- attacker tries to disturb
- various complex layers
- independently developed technologies are being combined
- what you see may not be what you get…
• goal: manipulate state stored in the database
• not directly accessible (hopefully)
• improper input checking in frontend server required
• nice: inconsistency is persistent
$password = $_POST['password'];
$id = $_POST['id'];
$sql = "UPDATE Accounts SET PASSWORD = '$password' WHERE account_id = $id";

Now imagine:  password=';--

SQL injection
Comic by Randall Munroe, xkcd.com
FRONTEND

- **goal:** manipulate content delivered to the browser
- **infrastructure attacks** like DNS cache poisoning
- **solution for this:** make sure you use SSL
- **improper input checking** can still bite you
http://example.com/?query=query string
generates website containing:
<p>You are looking for: query string</p>

so how about that:
http://example.com/?query=HTML code

remember that?
http://www.wolfgang-schaeuble.de/?search=&gt;&lt;/strong&gt;&lt;/div&gt;…
Dr. Wolfgang Schäuble MdB
Bundesminister des Innern

CDU/CSU-Bundestagsfraktion
CDU-Deutschlands

24.05.2008

Bundesinnenminister tritt zurück

wäre eine Meldung, die sicher viele gerne lesen würden. Allerdings handelt es sich nur um eine Cross-Site-Scripting-Schwachstelle im der Webseite des Politikers, der gerne die Online-Durchsuchung einführen möchte. Scherzbole können dadurch beliebige Meldungen unter der Domaine wolfgang-schauble.de erstellen.

Der Fehler liegt in der Suchfunktion des Internetauftritts, die HTML- und Skriptcode in Anfragen nicht ausfiltert. Grüße an dmk.
- can inject `<script>` code
- this code will run with the privileges of the embedding site (think IE zones)
- **cross-site scripting**
- Can you steal site credentials with this?
- imagine a bank website allowing injection
- How do you exfiltrate the password?
- JavaScript can access password fields
- you cannot use AJAX to send the password
- **same origin policy**
  - JavaScript may only connect back to the originating server (with some tolerance)
- can be defeated with `<img>` tags
  - encode password in URL to ping your server
- JavaScript can also read cookies…
- disallow cross-site image loading?
  - lots of sites use this
- no JavaScript access to password field?
  - AJAX logins need this
- fix web application
  - well…
- never click on suspicious links
- always use SSL
So you think SSL works?

- You explicitly type https://?
- Your site loads all JavaScript securely?
- Your platform checks for certificate revocation?
- … and for X.509 Basic Constraints?
- You trust all CAs on this planet to never issue broken certs?
REMOTE CONTROL

- goal: trick the browser to not show what’s actually happening
- or: how to pull strings behind the user’s back
- or: can one website control another one?
- no mischief with the server communication
SCENARIO

- user visits a regular website you control

Can you use credentials of a different site?

some preconditions

- user is logged in to the target site in another browser tab

- the target site identifies the user session with a cookie

- no cross-site cookie leakage in browser
- same origin policy denies AJAX to target again, `<img>` is your friend
- one website can send arbitrary requests to another, unrelated site
- **cross site request forgery**
- a special case of the **confused deputy problem**
- requests are blindly operating the target
- send requests and GET parameters
  - click buttons in the UI of the target site
  - operate search fields and other text input
- basic or digest authentication? cookies?
  - browser automatically sends credential
  - session riding
- POST requests?
  - manufacture a `<form>` instead of `<img>`
study in late 2008: high-profile bank websites vulnerable

browser-based port scanning

this is behind the corporate firewall

WiFi routers with web interface

disable firewall
reset wifi protection
enable UPnP
- disable cross-site POST requests
  - GET requests should by definition never change persistent state
- never authenticate a change of persistent state by cookie only
- pass an additional credential
  - session ID in URL, edit tokens
Log in

Don't have an account? Create an account.
You must have cookies enabled to log in to OSWiki.

Username: 
Password: 

Remember my login on this computer
Log in  E-mail new password
BLINDNESS

- goal: mislead the user to not seeing what’s actually happening
- nothing going on behind your back
- the internal state of the browser is properly displayed
- but you don’t notice…
www.paypal.com

homograph attack
The image shows a page from the PayPal website, focusing on the login and security aspects. The page includes fields for entering a username, email address, and password, with options for login and account creation. The main heading reads "Das Prinzip PayPal. Online zahlen – einfach und sicher." which translates to "The principle of PayPal. Online payment – simple and secure."
- this only works when logged in
  - always log out explicitly
  - do not use persistent logins
- you may want to check whether your password manager autofills inside frames
CONCLUSION

- web standards have gotten complex
- even bug-free behavior is vulnerable
- browsers are a bad application platform
- we did not even talk about WebSockets, WebGL, …
Is everything lost?

Yes