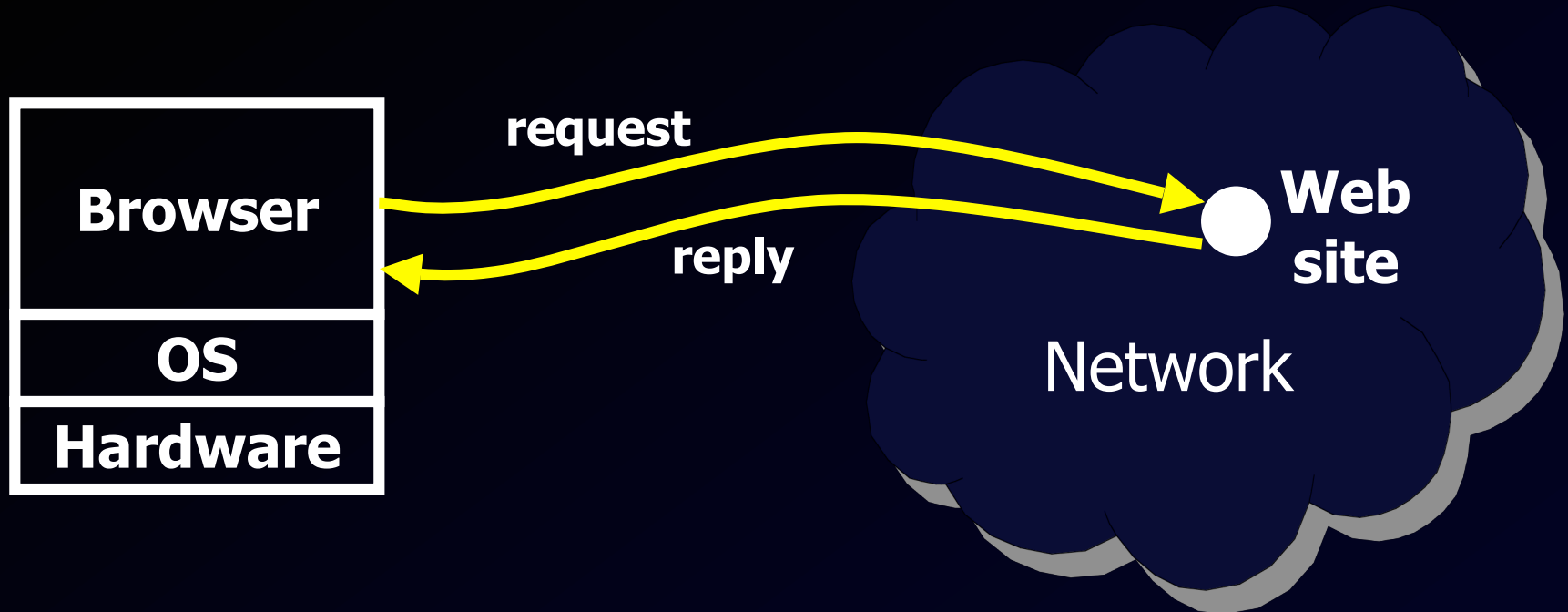


UMSSIA

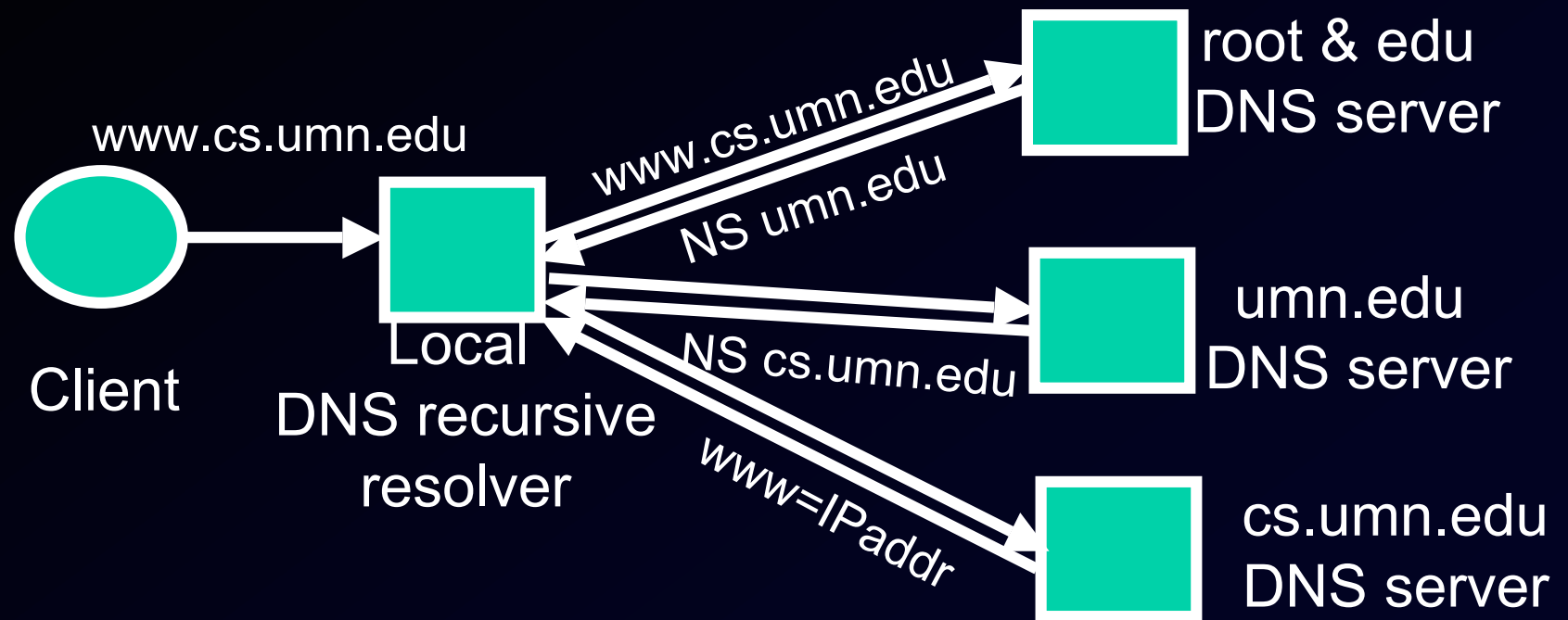
DAY VIII: WEB FUN

WEB SECURITY

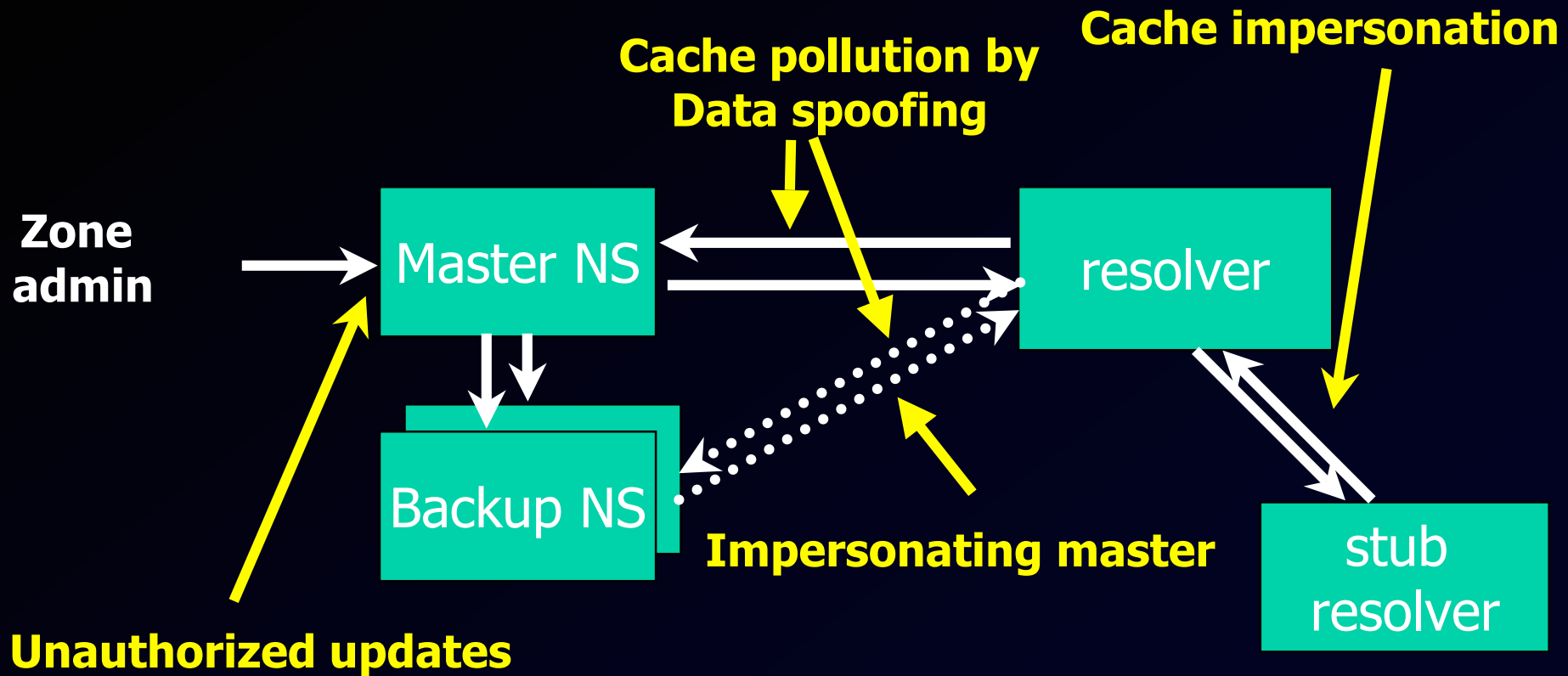


- **Browser sends requests to server**
- **Server processes requests**
- **Browser receives information and code**
- **Repeat as necessary.**

DNS RESOLUTION



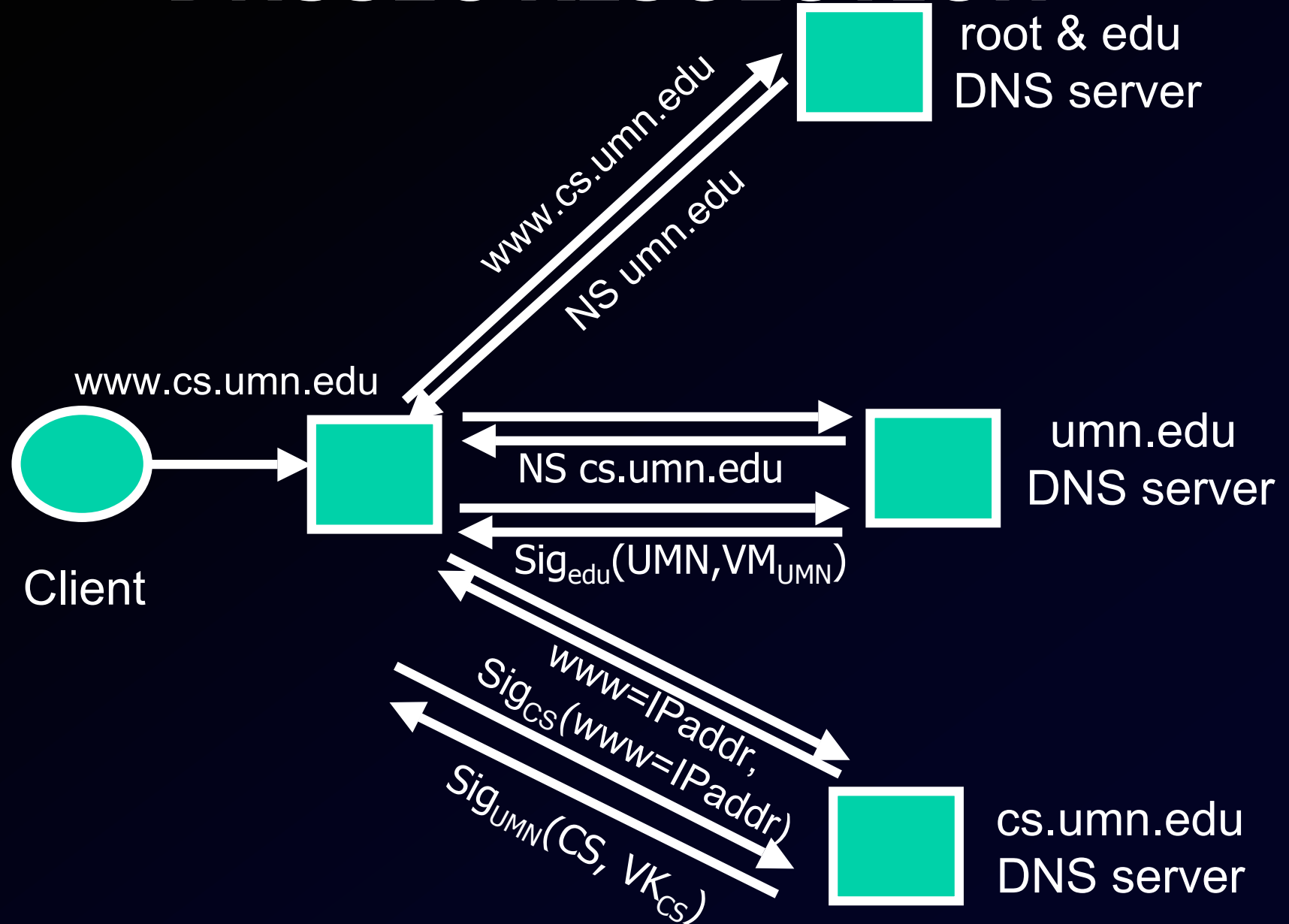
DNS DATA FLOW



DNSSEC

- **DNSSEC Attempts to address these issues cryptographically.**
- **In the "Ideal DNSSEC Deployment":**
 - **There is a public key for the "root" domain.**
 - **This is used to sign certificates for the administrators of top-level domains**
 - **These admins sign certificates for subdomains**
- **DNSSEC has three new kinds of records**
 - **Signatures on results, uploaded by zone admins**
 - **Certificates, to allow verification**
 - **NSEC records, to prove negative results**

DNSSEC RESOLUTION



HTTP

Method



File



HTTP version



Headers



```
GET /default.asp HTTP/1.0
Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
Connection: Keep-Alive
If-Modified-Since: Sunday, 17-Apr-96 04:32:58 GMT
```



Blank line

HTTP version



Status code

Data – none for GET

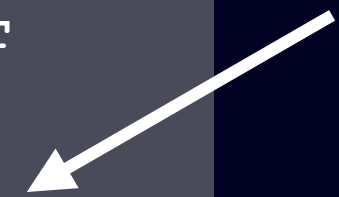
Reason phrase

Headers



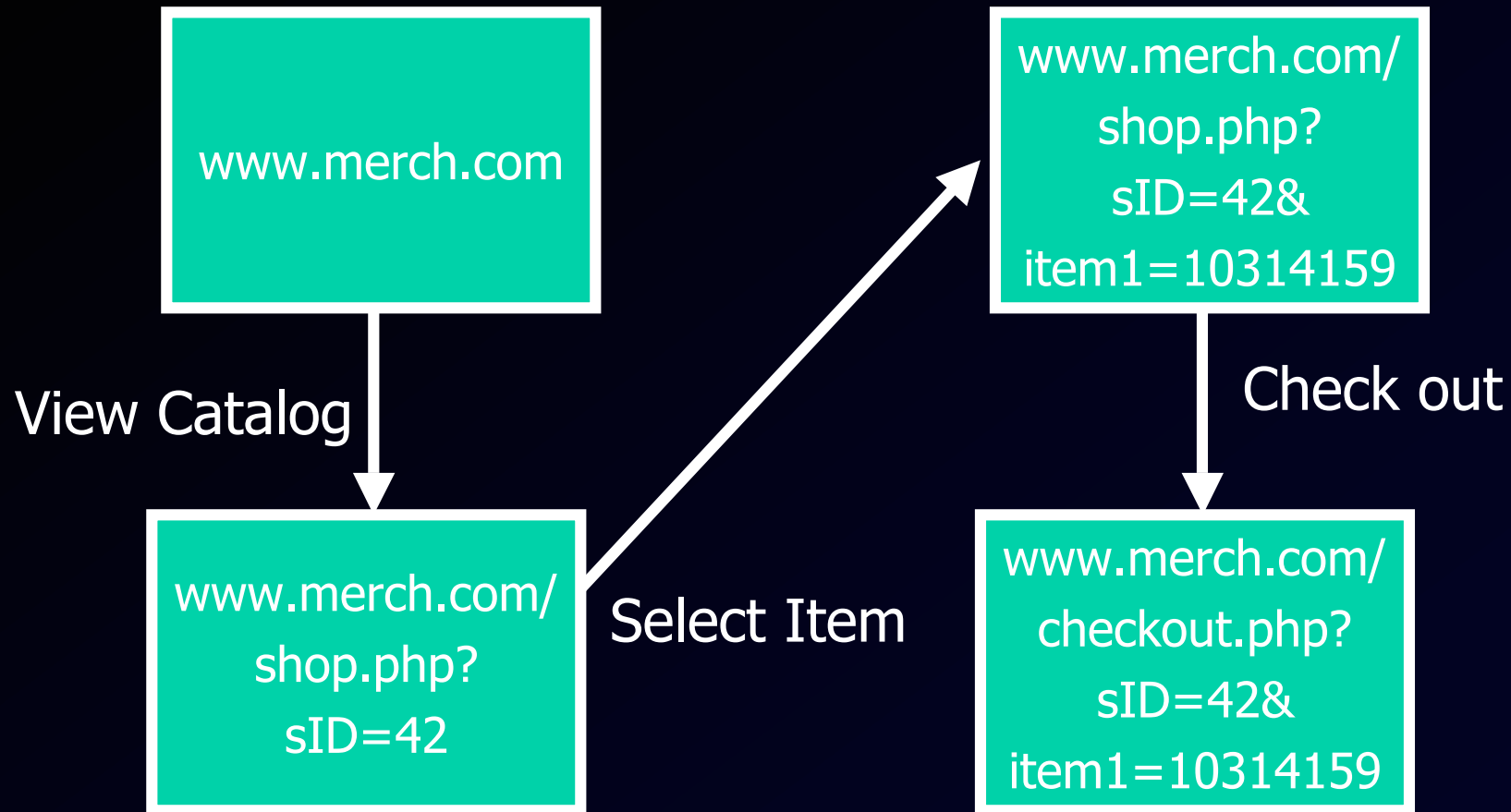
```
HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Content-Length: 2543
```

Data



```
<HTML> Some data... blah, blah, blah </HTML>
```

BROWSER SESSIONS



Store session information in URL

COOKIES

A **cookie** is a persistent file created by a server to store information at the client

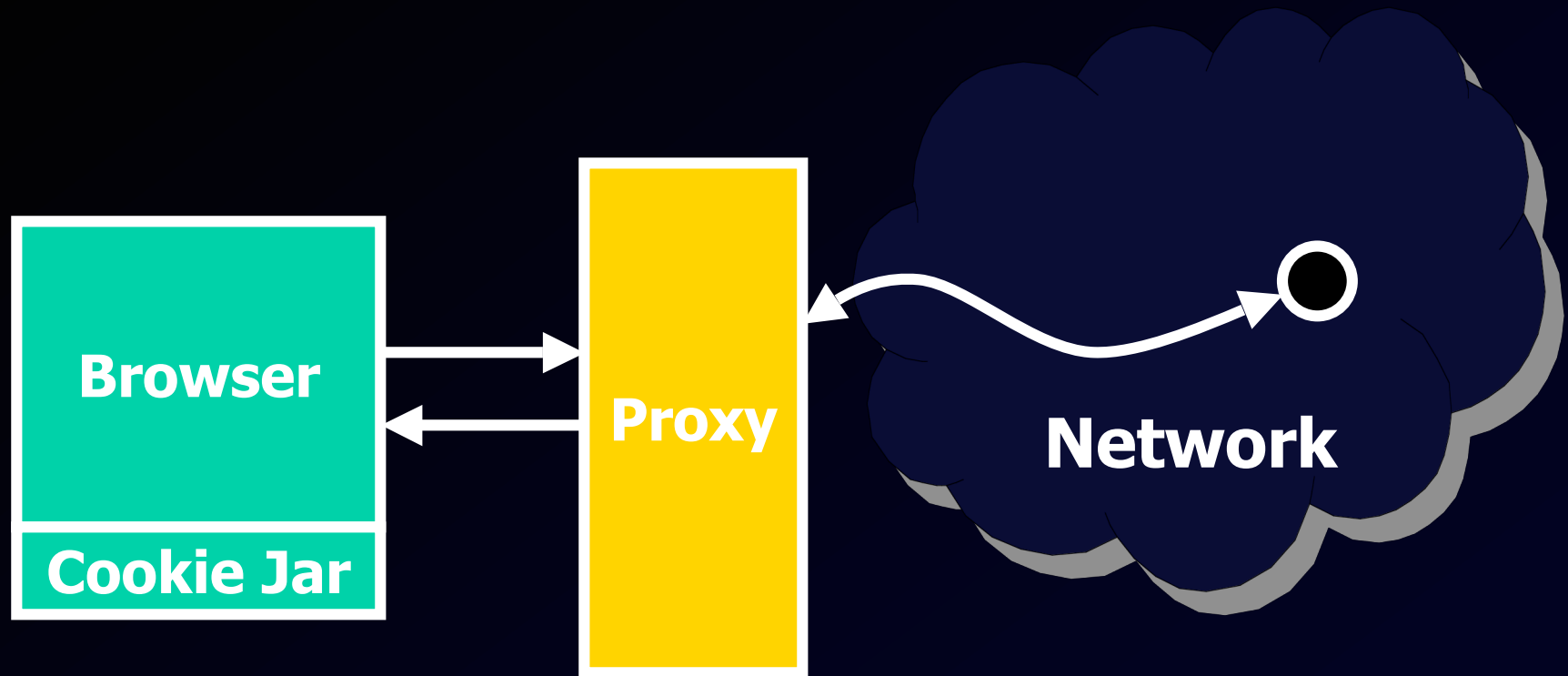


HTTP is a stateless protocol; cookies add state

COOKIES

- ... specify which web sites can access them. Typically only the creating site can access the cookie.
- ... Come in persistent and temporary flavors.
- ... Are a privacy risk: can be used to store browsing habits, personal info, etc.
- ... Are often not secured against malicious attackers.

PRIVACY PROXIES



... intercept HTTP requests and responses to help enforce privacy policy.

... e.g., modify cookies before sending to browser or server; filter out ads; block sites...

PRIVACY POLICY

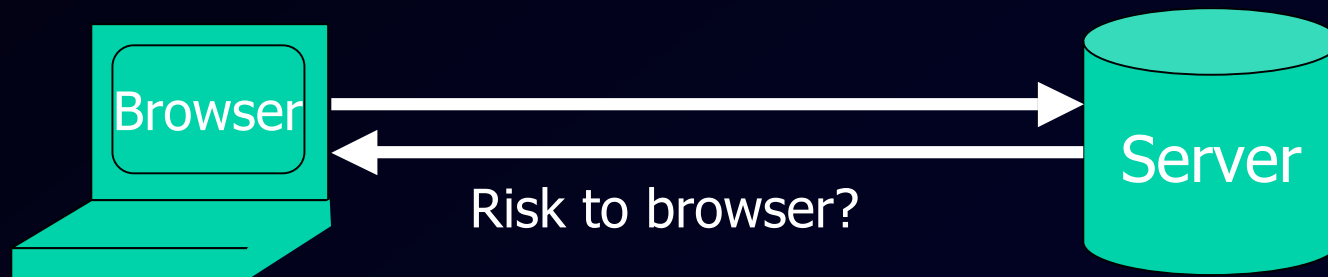
- Web sites collect a lot of personal information. What do they do with it?
- The P3P framework allows agreement on the use of this info, a formalized version of a "Privacy Policy."



- **Enforcement at server side is another matter...**
beware the market for lemons!

CLIENT-SIDE THREATS

- Given correct implementation, **data** is essentially “harmless...”
- Risks come from **code** received from web
 - Scripts in web pages
 - ActiveX controls and Browser extensions
 - Applets



JAVASCRIPT

- **... is an interpreted language executed by browser, and used in many attacks**
- **... can run:**
 - **before the HTML is loaded**
 - **before the document is viewed**
 - **while the document is viewed**
 - **as the browser is leaving**
 - **when user changes focus**
- **... code has access to:**
 - **User inputs: Keyboard monitoring and logging**
 - **Cookies: hijack authentication, browsing data**
 - **Screen IO: Spoof parts of web browser UI**
 - **Network: Communicate across network**

ACTIVEX

ActiveX controls are programs that reside on clients and are activated by HTML pages. They are:

- not interpreted by the browser**
- typically downloaded and installed on demand**

The ActiveX “security model” relies on:

- Digital signatures to verify the source of a binary**
- policy to reject controls from network zones**
- A type bit: *safe for initialization*, or *safe for scripting* which affects the way the control is used**

Once an ActiveX control is installed, it runs with the privileges and in the memory context of the browser and there is no way to control or limit its execution.

JAVA

... is a general purpose, strongly-typed programming language.

Browsers execute java programs in the Java Virtual Machine (JVM), which attempts to isolate remote code from sensitive system resources.

The JVM can also verify code security properties such as exception safety, modularity, etc.

This design allows for portable implementation, flexible sandboxing, etc...

JAVA SECURITY RISKS

... include several types:

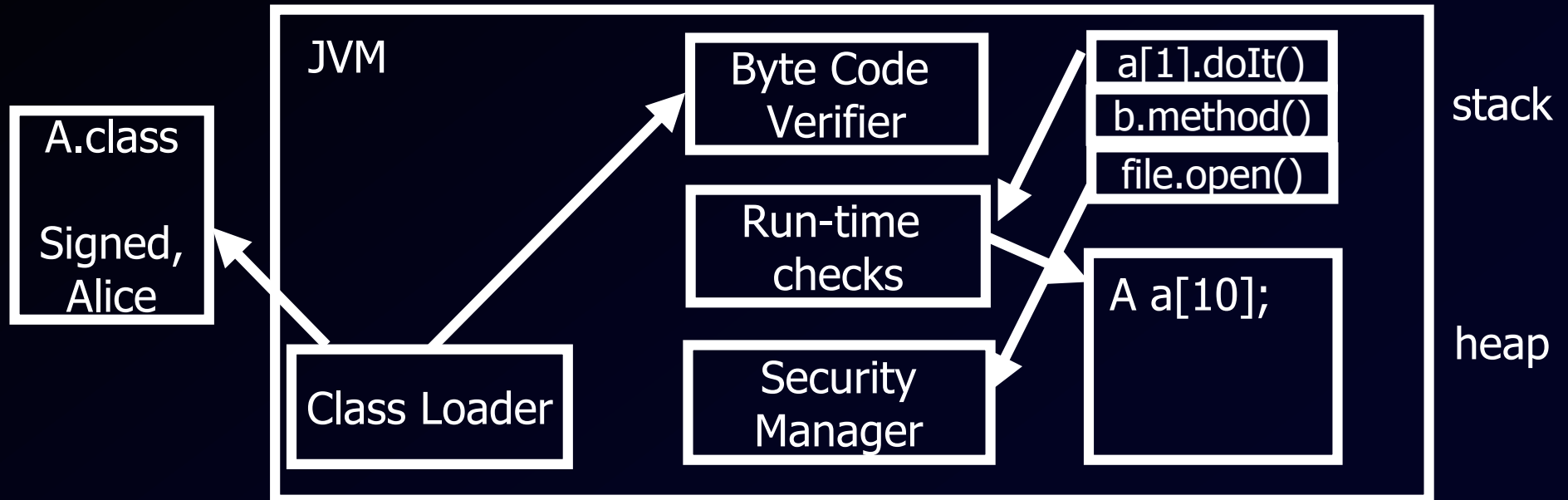
- **Annoyance or inconvenience:**
 - Display large window that ignores mouse input
 - Play irritating sound and do not stop
 - Consume CPU cycles, memory, network bandwidth ...
- **Information theft:**
 - Communication to remote machine
 - Access to password file, credit card number, ...
 - Subtle attack: trick dialog boxes, or trick UI
- **Modify or compromise system:**
 - Delete files, call system functions

JAVA SANDBOX

Class loaders create new types, and enforce isolation through namespaces and **protection domains**.

The **Byte Code Verifier** and JVM **run-time tests** ensure that type safety, array bounds, and visibility levels are preserved.

The **Security Manager** checks access requests based on protection domains and **stack inspection**.



WEB SITE SECURITY

Securing web sites can be challenging for many reasons:

- Sessions store state at the "client"**
- Inputs come from unknown, untrusted sources**
- The statelessness of HTTP allows replays, modification, injection, etc.**
- Mutually untrusted applications may use the same server**

TRANSACTIONS OVER HTTP

Potential for bugs come up in many places:

- **Session creation and identification:
Creating and assigning unique IDs**
- **Concurrency issues: contention and duplication of sessions**
- **Session termination and timeout:**
 - Clean up stale sessions
 - Handle stale requests
 - Concurrency in session termination
- **Session state storage**
 - Distributed or local? Performance issues
 - Fail-over, load balancing issues

EXAMPLE: COOKIE AUTHENTICATION

- **Fu et al., 2002 study of cookies for 27 major web vendors:**
 - **Obtained access to other accounts on 8**
 - **Obtained access to arbitrary accounts on 1**
- **Common mistakes:**
 - **Weak or misused crypto**
 - **Username/UID = authentication**
 - **Session ID = authentication**

EXAMPLES

Bad Crypto:

- www.ichat.com : cookie =
username || password © "secret string"
- www.wsj.com : cookie =
UNIX crypt(username || "secret string")

No Crypto:

- www.highschoolalumni.com cookie =
user="name"&id="uid"
- www.nebride.com : cookie=
userid="id"&email="blah"
asks for password or **send via email.**

EXAMPLES

Use of nonsecrets:

- www.fatbrain.com
 - User logs in, is assigned a session ID in URL.
 - Knowing the sID allows access to a session.
 - IDs are assigned as follows:

```
global_current_id = random_integer();
while(more sessions)
    next_id = global_current_id++;
```
- Same problem: www.verizonwireless.com

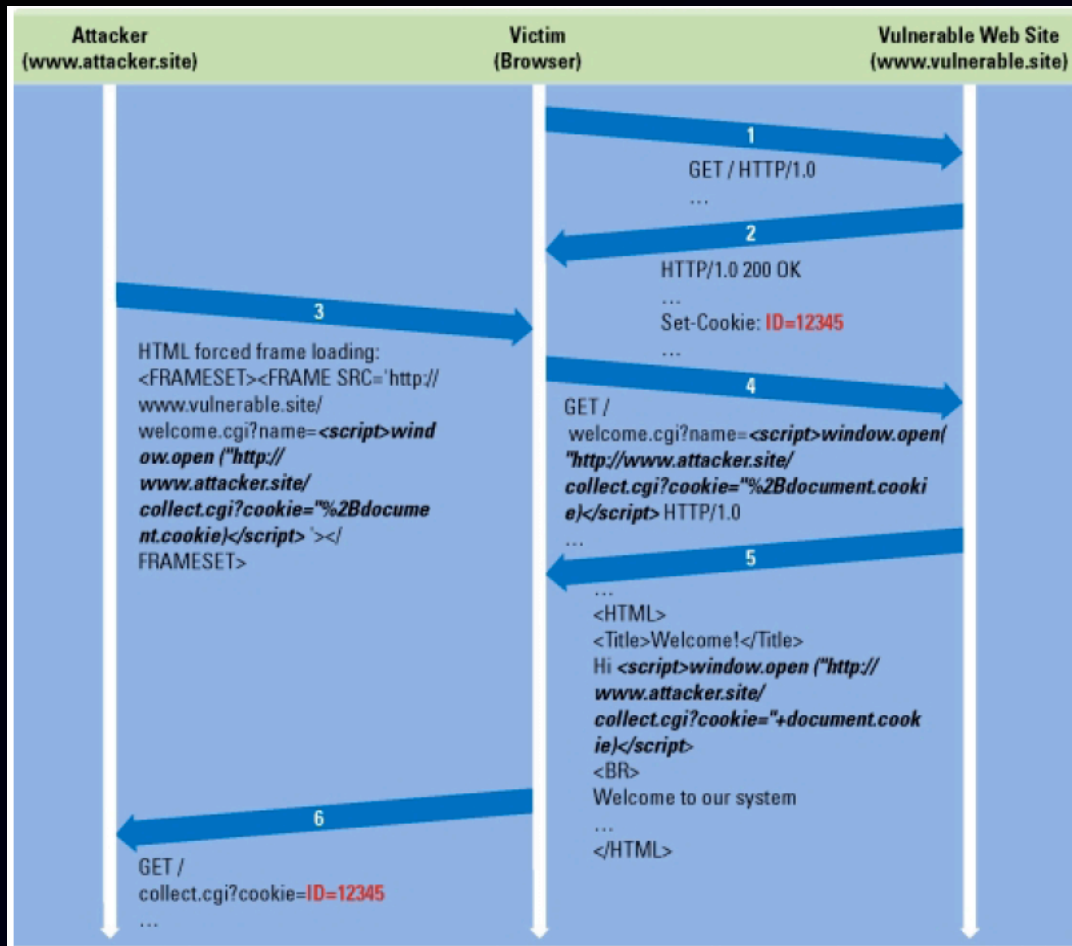
DELICIOUS COOKIE RECIPE

- For safe cookies, use crypto integrity, e.g.
- Cookie: `exp=t`
 - `&data1=<blah1>`
 - `&data2=<blah2> ...`
 - `&auth=MACK(exp=t&data1=...)`
- Where K is a secret key held by the server (shared between servers if multiple)
- Session numbers should be chosen randomly
- If data[i] should be secret, apply encryption to `<blah[i]>`. (MAC the encrypted `<blah>`)

SERVER-SIDE ATTACKS

- ... include the typical buffer overflows, format strings, integer overflows, etc. But also:
- **Command-line injection:**
 - “cat reply | mail \$user” =>
“cat reply | mail user@example.com | rm -rf /”
- **“SQL/XPath Injection”**
 - Alter database queries to reveal or modify application database
 - Change user passwords, login as bogus user, learn entire tables, run commands as DBMS user, etc....
- **URL injection:**
 - PHP fopen(“/local/file”, “r”) opens /local/file.
 - PHP fopen(“http://www.example.com/remote/file” , “r”)

CLIENT-SIDE ATTACKS



- Cross-Site Scripting
 - Run my script on your site's page
- Cache poisoning
 - My site/script loads as your URL
- Cookie poisoning
 - Set my cookies for your site

DEFENSE

INPUT VALIDATION

- **Check to make sure inputs are valid**
 - Eg. Email addr: only alphanumeric, `_`, `.`, `@`
- **Whitelist, don't blacklist**
 - **DON'T** look for `|` and `;`
- **Apply checking before and after all decoding and conversions**
 - IIS unicode bug, etc...
- **Use defenses available, e.g. Perl tainting (`perl -w`), etc...**