CSC 405
Web Intro

Adam Gaweda
agaweda@ncsu.edu

Alexandros Kapravelos
akaprav@ncsu.edu

Birth of the Internet

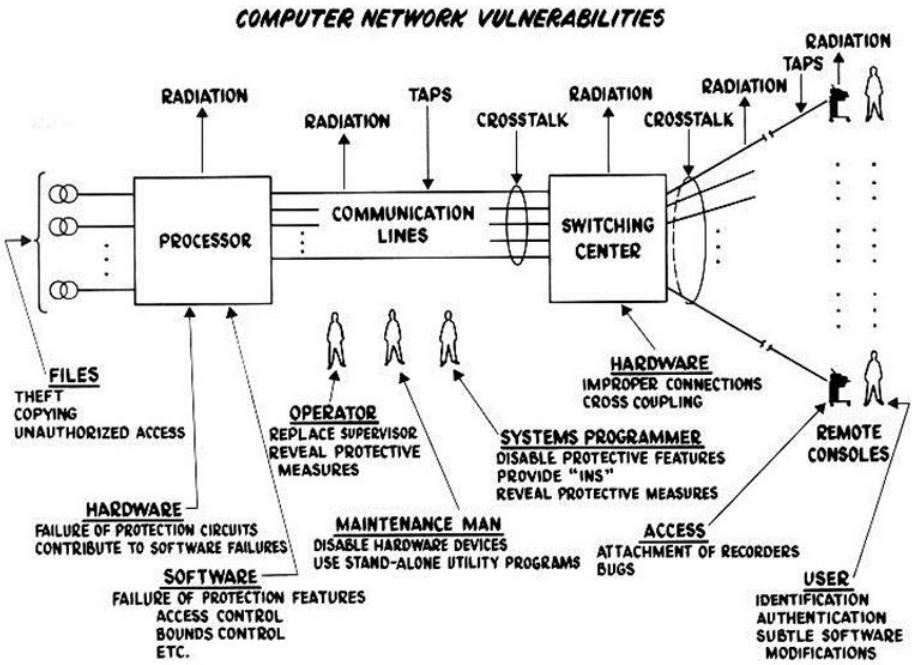
The US Dept of Defense wanted a redundant, networked communication system for the military

Dr. Larry Roberts designed **ARPAnet** in Dec 1969 for $3.4m

1969

Birth of the Internet

Willis H. Ware chairs RAND R-609, identifying all of ARPAnet's vulnerabilities



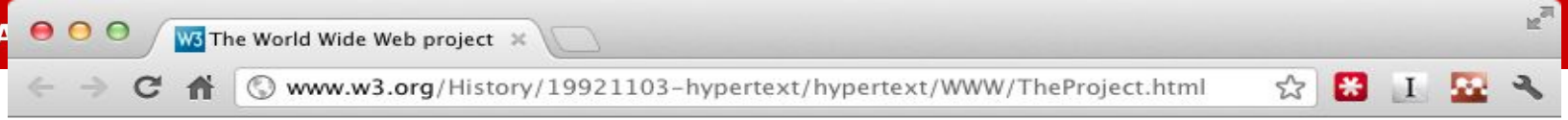
Birth of the Internet

In 1983, ARPAnet adopted TCP/IP and the Internet was born

Sir Tim Burners-Lee developed HTML and the World Wide Web

[World Wide Web Project (the first webpage)]

ACM Turing Award 2016



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.

Help
on the browser you are using

Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)

Technical
Details of protocols, formats, program internals etc

Bibliography
Paper documentation on W3 and references.

People
A list of some people involved in the project.

History
A summary of the history of the project.

How can I help ?
If you would like to support the web..

Getting code
Getting the code by anonymous FTP , etc.

Birth of the Web

- Created by Tim Berners-Lee while he was working at CERN
  - First CERN proposal in 1989
  - Finished first website end of 1990

- Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web, Tim Berners-Lee


PROPRIETE CERN
PROPRIETE CERN

Design

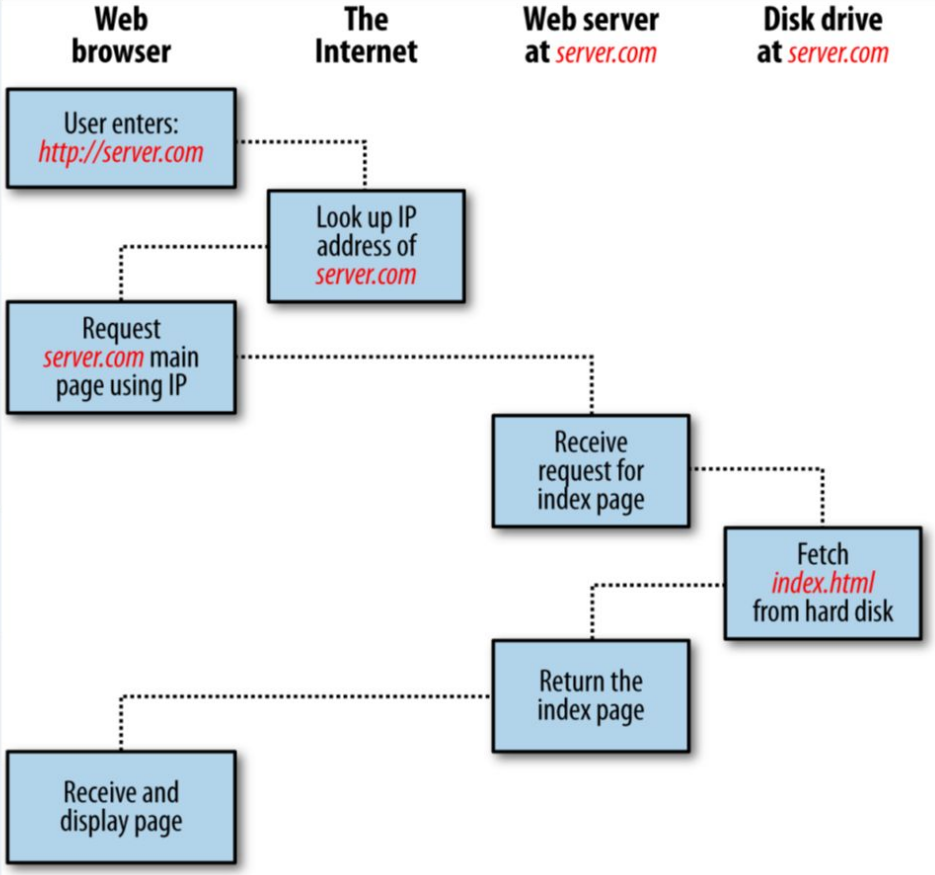
- Originally envisioned as a way to share research results and information at CERN

- Combined multiple emerging technologies
  - Hypertext
  - Internet (TCP/IP)

- Idea grew into "universal access to a large universe of documents"

Workflow

1	User enters web address in address bar
2	Browser looks up the IP address of the URL via DNS
3	Browser submits a request for content from IP
4	Server receives the request
5	Server looks for the content on its hard disk
6	Binary Digits are Transferred via the Internet
7	Browser processes the binary into HTML, which it then interprets



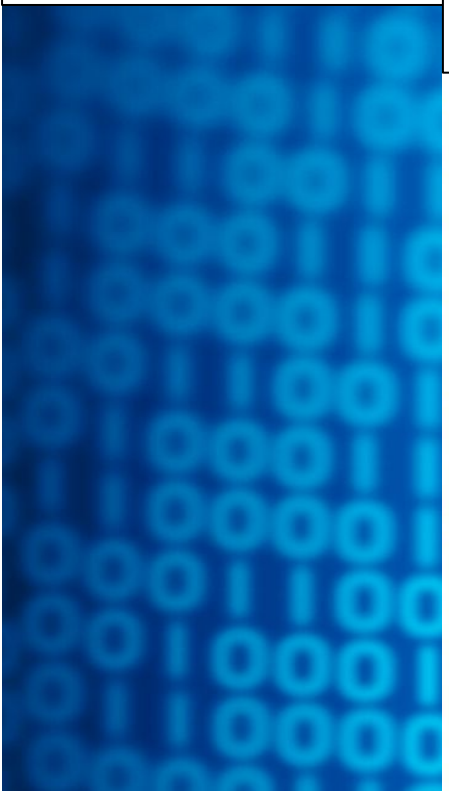
Workflow

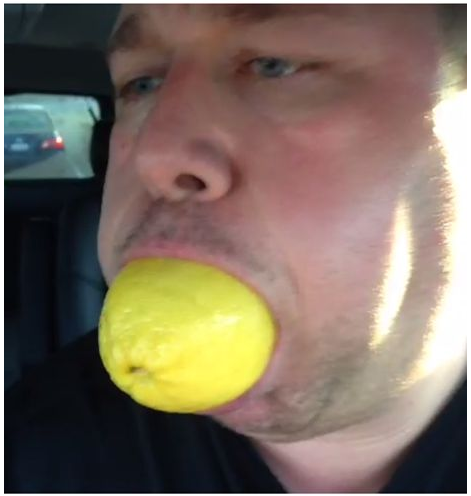
Binary

Converts to Unicode

Which Builds HTML

Which Renders





Three Central Questions

How to name a resource?

How to request and serve a resource?

How to create hypertext?

Three Central Questions

How to name a resource?

- Uniform Resource Identifier (URI/URL)

How to request and serve a resource?

- Hypertext Transfer Protocol (HTTP)

How to create hypertext?

- Hypertext Markup Language (HTML)

Uniform Resource Identifier

- Essential metadata to reach/find a resource
- Answers the following questions:
  - Which server has it?
  - How do I ask?
  - How can the server locate the resource?
- Latest definition in RFC 3986 (January 2005)

URI – Syntax

<scheme>:<authority>/<path>?<query>#<fragment>

URI – Syntax

<scheme>:<authority>/<path>?<query>#<fragment>

- **scheme**
  - The protocol to use to request the resource

URI – Syntax

`<scheme>:<authority>/<path>?<query>#<fragment>`

- scheme
  - The protocol to use to request the resource
- **authority**
  - The entity that controls the interpretation of the rest of the URI
  - Usually a server name
    - `<username>@<host>:<port>`

URI – Syntax

<scheme>:<authority>/<path>?<query>#<fragment>

- scheme
  - The protocol to use to request the resource
- authority
  - The entity that controls the interpretation of the rest of the URI
  - Usually a server name
    - <username>@<host>:<port>
- **path**
  - Usually a hierarchical pathname composed of "/" separated strings

URI – Syntax

<scheme>:<authority>/<path>?<query>#<fragment>

- scheme
  - The protocol to use to request the resource
- authority
  - The entity that controls the interpretation of the rest of the URI
  - Usually a server name
    - <username>@<host>:<port>
- path
  - Usually a hierarchical pathname composed of "/" separated strings
- **query**
  - Used to pass non-hierarchical data

URI – Syntax

`<scheme>:<authority>/<path>?<query>#<fragment>`

- scheme
  - The protocol to use to request the resource
- authority
  - The entity that controls the interpretation of the rest of the URI
  - Usually a server name
    - `<username>@<host>:<port>`
- path
  - Usually a hierarchical pathname composed of "/" separated strings
- query
  - Used to pass non-hierarchical data
- **fragment**
  - Used to identify a subsection or subresource of the resource

URI – Syntax

```
<scheme>:<authority>/<path>?<query>#<fragment>
```

Examples:

```
foo://example.com:8042/over/there?test=bar#nose
ftp://ftp.ietf.org/rfc/rfc1808.txt
mailto:classtech@ncsu.edu
https://example.com/test/example:1.html?/hello
```

URI – Reserved Characters

:	&
/	‘
?	(
#	)
[	*
]	+
@	,
!	;
$	=

URI – Percent Encoding

Must be used to encode anything that is **not** of the following:

Alpha	`[a-zA-Z]`
Numeric	`[0-9]`
Dash	`-`
Period	`.`
Underscore	`_`
Tilde	`~`

URI – Percent Encoding

Encode a byte outside of the previous list with percent sign (%) followed by hexadecimal representation of byte

- **`& -> %26`**
- **`% -> %25`**
- **`<space> -> %20`**
- **`...`**

Let's fix our previous example:

`https://example.com/test/example:1.html?/hello`

↓

`https://example.com/test/example%3A1.html?%2Fhello`

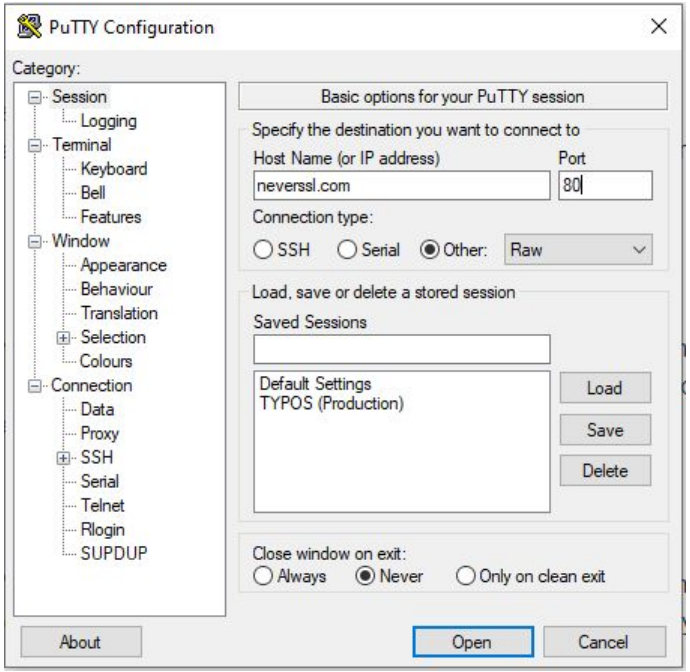
HTTP – Overview

- **Client**
  - Opens TCP connection to the server
  - Sends request to the server

- **Server**
  - Listens for incoming TCP connections
  - Reads request
  - Sends response

Demo

Request by Client

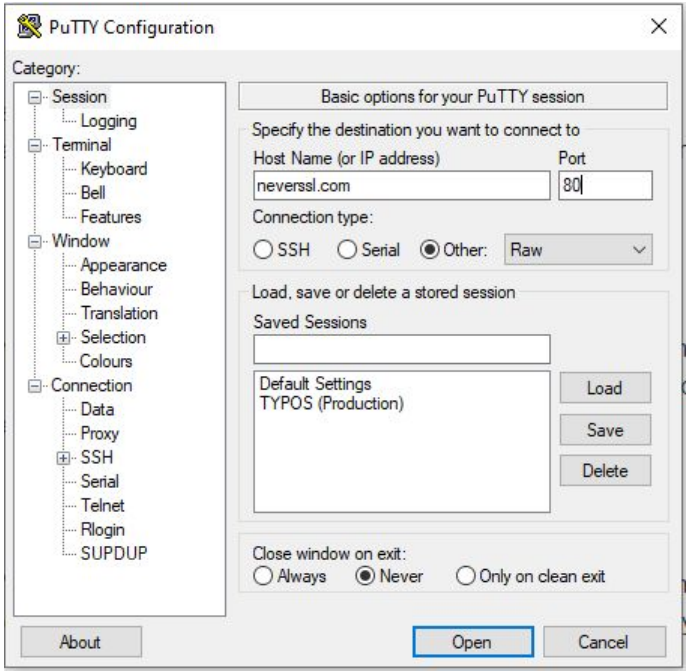



```
GET / HTTP/1.1
User-Agent: curl/7.37.1
Host: neverssl.com
Accept: */*
```

Demo

Request by Client

Response by Server




```
GET / HTTP/1.1
User-Agent: curl/7.37.1
Host: neverssl.com
Accept: */*

HTTP/1.1 200 OK
Date: Thu, 09 Mar 2024 03:22:05 GMT
Server: Apache/2.4.54 ()
Upgrade: h2,h2c
Connection: Upgrade
Last-Modified: Wed, 29 Jun 2022 00:23:33 GMT
ETag: "f79-5e28b29d38e93"
Accept-Ranges: bytes
Content-Length: 3961
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8

<html>
<head>
<title>NeverSSL - Connecting ... </title>
<style>
body {
font-family: Montserrat, helvetica,
…
```

Requests

- An HTTP request consists of:
  - method
  - resource (derived from the URI)
  - protocol version
  - header fields
  - body (optional)

```
GET / HTTP/1.1
User-Agent: curl/7.37.1
Host: neverssl.com
Accept: */*
```

Requests

- An HTTP request consists of:
  - **method**
  - resource (derived from the URI)
  - protocol version
  - header fields
  - body (optional)

```
GET / HTTP/1.1
User-Agent: curl/7.37.1
Host: neverssl.com
Accept: */*
```

Requests – Methods

The method that that client wants applied to the resource

- **`GET`** – request transfer of the entity referred to by the URI
- **`POST`** – ask the server to process the included body as "data" associated with the resource identified by the URI
- **`PUT`** – request that the enclosed entity be stored under the supplied URI
- **`HEAD`** – identical to **`GET`** except server **must not** return a body

Requests – Methods

- **`OPTIONS`** – request information about the communication options available on the request/response chain identified by the URL
- **`DELETE`** – request the server deletes the resource identified by the URI
- **`TRACE`** – invoke a remote, application-layer loop-back of the request message and the server should reflect the message received back to the client in its body
- **`CONNECT`** – used with proxies

Web servers can also define **arbitrary methods**

Requests

- An HTTP request consists of:
  - method
  - **resource** (derived from the URI)
  - protocol version
  - header fields
  - body (optional)

```
GET / HTTP/1.1
User-Agent: curl/7.37.1
Host: neverssl.com
Accept: */*
```

Requests – Resources

- URI can specify the absolute location of the resource
  - `https://example.com/test/help.html`
- Or the URI can specify a location relative to the current resource
  - `//example.com/example/demo.html`
    - Relative to the current network-path (scheme)
  - `/test/help.html`
    - Relative to the current authority
  - `../../people.html`
    - Relative to the current authority and path
- Context important in all cases
  - `http://localhost:8080/test`

Endpoint Attacks

- **Distributed Denial of Service** (DDOS) attacks are one of the most common web attacks

Endpoint attacks spike early in 2023

The proliferation of cloud and endpoint attacks is making 2023 a more challenging year than many CISOs bargained — and budgeted — for. CISOs in the banking, financial services and insurance industries told VentureBeat, on condition of anonymity, that attacks on every type of endpoint have quadrupled in just four months. Data they can capture shows cloud infrastructure, Active Directory, ransomware, web application, vulnerability exploitation, and distributed denial of service (DDOS) attacks spiking sharply in the last 120 days.

https://venturebeat.com/security/defining-endpoint-security-in-a-zero-trust-world/

```
GET /.env
GET //administrator/.env
GET //laravel/.env
GET /1674310391
GET /_profiler/phpinfo
GET /actuator/health
GET /CSS/Miniweb.css
GET /go/add-on/business-continuity/api/plugin?folderName=&pluginName=../../../etc/passwd
GET /laravel/.env
GET /menu.jsp
GET /nmaplowercheck1672624376
GET /nmaplowercheck1675000731
GET /Portal0000.htm
GET /Public/home/js/check.js
GET /server-status
HEAD /
HEAD /login
POST /
POST //admin/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //api/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //backup/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //blog/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //cms/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //demo/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //dev/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //laravel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //lib/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //lib/phpunit/phpunit/Util/PHP/eval-stdin.php
POST //lib/phpunit/src/Util/PHP/eval-stdin.php
POST //lib/phpunit/Util/PHP/eval-stdin.php
POST //new/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //old/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //panel/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //phpunit/phpunit/src/Util/PHP/eval-stdin.php
POST //phpunit/phpunit/Util/PHP/eval-stdin.php
POST //phpunit/src/Util/PHP/eval-stdin.php
POST //phpunit/Util/PHP/eval-stdin.php
POST //protected/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
```

URL Endpoint Scans from https://typos.csc.ncsu.edu

Common DDOS Web Attacks

- Computationally Expensive Operations
  - Database Lookups
  - PDF Generation
  - Large file uploads
  - ZIP Bombing

- Operations that take time to process result in taking up
  - memory
  - server connections
  - etc.

Requests

- An HTTP request consists of:
  - method
  - resource (derived from the URI)
  - **protocol version**
  - header fields
  - body (optional)

```
GET / HTTP/1.1
User-Agent: curl/7.37.1
Host: neverssl.com
Accept: */*
```

Requests - Protocol

Based on TCP, uses port **`80`** by default

- **`HTTP/1.0`**
  - Defined in RFC 1945 (May 1996)
- **`HTTP/1.1`**
  - Defined in RFC 2616 (June 1999)
- **`HTTP/2.0`**
  - Based on SPDY, still under discussion
- **`HTTPS/2`** and **`HTTPS/3`** (Port **`443`**)
  - Creates private encryption to strengthen communication

Requests

- An HTTP request consists of:
  - method
  - resource (derived from the URI)
  - protocol version
  - **header fields**
  - body (optional)

```
GET / HTTP/1.1
User-Agent: curl/7.37.1
Host: neverssl.com
Accept: */*
```

Requests - Header Fields

- Defines information about the client
  - Key-Value Pairs transmitted in clear-text
  - Separated by **`CR-LF`**

- **`Accept: text/html`**
  - Define the media type client is expecting
- **`User-Agent: Googlebot/2.1 (+http://www.google.com/bot.html)`**
  - Identifies the software the client is using to access the server
- Full List

Modern Requests

```
GET / HTTP/1.1
Host: www.google.com
Accept-Encoding: deflate, gzip
Accept:
text/html,application/xhtml+xml,applicati
on/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_10_1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/39.0.2171.95
Safari/537.36
```

Modern Requests

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36
```

- **Mozilla/5.0**
  - Indicates compatibility with Mozilla rendering engine
- **(Macintosh; Intel Mac OS X 10_10_1)**
  - System browser is running
- **AppleWebKit/537.36**
  - Platform browser uses
- **(KHTML, like Gecko)**
  - Additional details
- **Chrome/39.0.2171.95 Safari/537.36**
  - Additional details

Header Attacks

- **Slow Header Attacks (SlowLoris)**
  - Establish multiple **HTTP** connections in parallel
  - However, never complete the requests, only sending partial headers
    - Server will assume the requests are genuine and wait for them to complete
  - To continue the attack, new **HTTP** headers get added to the attack
    - "Oh, the user is on an unreliable network, we can wait"

Header Attacks

- **Slow Header Attacks (SlowLoris)**
  - Establish multiple `HTTP` connections in parallel
  - However, never complete the requests, only sending partial headers
    - Server will assume the requests are genuine and wait for them to complete
  - To continue the attack, new `HTTP` headers get added to the attack
    - "Oh, the user is on an unreliable network, we can wait"

- **Slow POST Attacks**
  - Similar to Header Attacks, but partial `POST` data is sent
  - The `Content-Length` header tells the server how much to expect, but the attacker delays sending the entire payload

SlowLoris Demo

Managing Header Attacks

- Increase maximum concurrent connections
  - **Load Balancers** to evenly distribute connections between servers
- Limit concurrent connections by a single IP address
  - **fail2ban** malicious IP addresses
- Limit time span a client request can stay alive
  - **Apache**: `mod_reqtimeout`, `mod_qos`
  - **Nginx**: `client_header_timeout`, `client_body_timeout`

Responses

- An HTTP response consists of:
  - protocol version
  - status code
  - short reason
  - headers
  - body

```
HTTP/1.1 200 OK
Date: Thu, 09 Mar 2024 03:22:05 GMT
Server: Apache/2.4.54 ()
Upgrade: h2,h2c
Connection: Upgrade
Last-Modified: Wed, 29 Jun 2022 00:23:33 GMT
ETag: "f79-5e28b29d38e93"
Accept-Ranges: bytes
Content-Length: 3961
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8

<html>
<head>
<title>NeverSSL - Connecting ... </title>
<style>
body {
font-family: Montserrat, helvetica,
…
```

Responses

- An HTTP response consists of:
  - **protocol version**
  - status code
  - short reason
  - headers
  - body

```
HTTP/1.1 200 OK
Date: Thu, 09 Mar 2024 03:22:05 GMT
Server: Apache/2.4.54 ()
Upgrade: h2,h2c
Connection: Upgrade
Last-Modified: Wed, 29 Jun 2022 00:23:33 GMT
ETag: "f79-5e28b29d38e93"
Accept-Ranges: bytes
Content-Length: 3961
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8

<html>
<head>
<title>NeverSSL - Connecting ... </title>
<style>
body {
font-family: Montserrat, helvetica,
…
```

Responses

- An HTTP response consists of:
  - protocol version
  - **status code**
  - **short reason**
  - headers
  - body

```
HTTP/1.1 200 OK
Date: Thu, 09 Mar 2024 03:22:05 GMT
Server: Apache/2.4.54 ()
Upgrade: h2,h2c
Connection: Upgrade
Last-Modified: Wed, 29 Jun 2022 00:23:33 GMT
ETag: "f79-5e28b29d38e93"
Accept-Ranges: bytes
Content-Length: 3961
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8

<html>
<head>
<title>NeverSSL - Connecting ... </title>
<style>
body {
font-family: Montserrat, helvetica,
…
```

Responses – Status Codes

- **1XX** – Informational: request received, continuing to process
- **2XX** – Successful: request received, understood, and accepted
- **3XX** – Redirection: user agent needs to take further action to fulfill the request
- **4XX** – Client error: request cannot be fulfilled or error in request
- **5XX** – Server error: the server is aware that it has erred or is incapable of performing the request

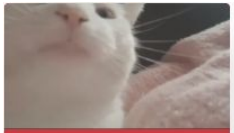
Responses – Short Reason

- "200" -> OK
- "201" -> Created
- "202" -> Accepted
- "204" -> No Content
- "301" -> Moved Permanently
- "307" -> Temporary Redirect

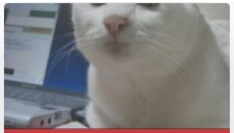
Responses – Status Codes

- "400" -> Bad Request
- "401" -> Unauthorized
- "403" -> Forbidden
- "404" -> Not Found
- "500" -> Internal Server Error
- "501" -> Not Implemented
- "502" -> Bad Gateway
- "503" -> Service Unavailable

http.cat



103
Early Hints

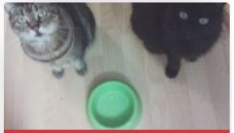


200
OK

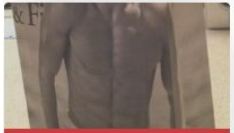
201
Created

202
Accepted

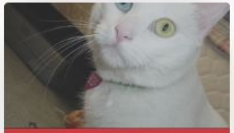
203
Non-Authoritative Information



204
No Content



206
Partial Content

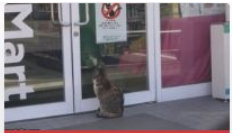


207
Multi-Status

300
Multiple Choices

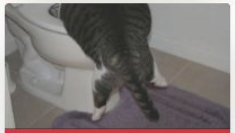
308
Permanent Redirect

400
Bad Request



401
Unauthorized

402
Payment Required

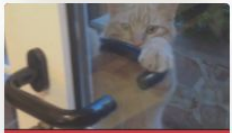


403
Forbidden

404
Not Found

405
Method Not Allowed

406
Not Acceptable



407
Proxy Authentication Required

HTTP Authentication

- Based on a simple **challenge-response** scheme
- The **challenge** is returned by the server as part of a **`401`** (unauthorized) reply message and specifies the authentication schema to be used
- An authentication request refers to a **realm**, that is, a set of resources on the server
- The client must include an Authorization header field with the required (valid) credentials

HTTP Basic Authentication

- The server replies to an unauthorized request with a 401 message containing the header field

```
WWW-Authenticate: Basic realm="ReservedDocs"
```

- The client retries the access including in the header a field containing a cookie composed of a **base64** encoded **username** and **password** (RFC 2045)

```
Authorization: Basic UmFscGllOkRyaW5rTW9yZU92YWx0aW5l==
```

- Can you crack the username/password?

HTTP 1.1 Authentication

- Defines an additional authentication scheme based on cryptographic digests (RFC 2617)
  - Server sends a **nonce** as challenge
  - Client sends request with digest of the username, the password, the given nonce value, the HTTP method, and the requested URL

- To authenticate the users, the server needs access to **clear-text** user passwords

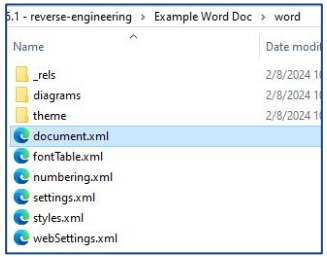
Monitoring and Modifying HTTP Traffic

- HTTP traffic can be analyzed in different ways
  - **Sniffers** can be used to collect traffic
  - Servers can be configured to create extensive **logs**
  - **Browsers** can be used to analyze the content received from a server
  - Client-side/server-side **proxies** can be used to analyze the traffic without having to modify the target environment

- Client-side proxies are especially effective in performing vulnerability analysis because they allow one to examine and modify each request and reply
  - Firefox extensions: LiveHTTPHeaders, Tamper Data
  - Burp Proxy

Hypertext Markup Language

- A **markup language** designed to 'present' data in a certain format with the ability to 'link' to other resources
- Based on Standard Generalized Markup Language (SGML) (ISO 8879:1986)
- HTML is a specialized version of a **Document Object Model** (DOM) for the web
  - Microsoft Office formats all of its files w/ a DOM



Tags

HTML contains **tags** that explain what the content is suppose to be

> **Most** tags have an opening and closing tag (with a handful of exceptions)

Every HTML file starts with an **<html>** tag and ends with an **</html>** tag

HTML – Tags

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div>
      <p>I am the example text</p>
    </div>
  </body>
</html>
```

HTML – Tags

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div>
      <p>I am the example text</p>
    </div>
  </body>
</html>
```

Informs the application of the specific DOM format used

HTML – Tags

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div>
      <p>I am the example text</p>
    </div>
  </body>
</html>
```

Contains metadata about the document

HTML – Tags

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div>
      <p>I am the example text</p>
    </div>
  </body>
</html>
```

HTML – Tags

```
<html>
  <head>
    <title>Hello World</title>
    <link rel="stylesheet" href="stylesheets/main.css">
  </head>
  <body>
    <div>
      <p>I am the example text</p>
    </div>
  </body>
</html>
```

Or resources the page needs to import

HTML – Tags

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div>
      <p>I am the example text</p>
    </div>
  </body>
</html>
```

The body contains the actual contents of the resource

HTML – Tags

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div>
      <p>I am the example text</p>
    </div>
  </body>
</html>
```

And tags like ** ** and **<p>** inform the application about how the content should be organized (and formatted)

HTML – Tags

- html // Document Language
  - head // Metadata
    - title
      - "Hello World"
  - body // Contents
    - div // Section
      - p // Paragraph
        - "I am the example text"

HTML – Tags

- Tags can have "attributes" that provide metadata about the tag

- Attributes live inside the start tag after the tag name

- `<input type="text" name="email" disabled>`
  - `input` is the tag name with…
  - `type` as an attribute with the value `"text"`
  - `name` as an attribute with the value `"email"`
  - `disabled` as an attribute with no value
  - `input` also does not need a closing `</input>`

HTML – Hyperlink

- anchor tag is used to create a hyperlink
- href attribute is used provide the URI
- Text inside the anchor tag is the text of the hyperlink

- `<a href="http://google.com">Example</a>`

Example

HTML – Browsers

- User agent is responsible for parsing and interpreting the HTML and displaying it to the user

HTML – Character References

- How to include HTML special characters as text/data?

< > ' " & =

- Encode the character reference
- Also referred to in HTML < 5.0 as "entity reference" or "entity encoding"

HTML – Character References

- Three variations, each start with **&** and end with **;**
  - Named character reference
    - **&<predefined_name>;**
  - Decimal numeric character reference
    - **&#<decimal_unicode>;**
  - Hexadecimal numeric character reference
    - **&#x<hexadecimal_unicode>;**

- **Note:** This will be the **root** of a significant number of vulnerabilities and is **critical** to understand

HTML – Character References Example

- The ampersand (**&**) is used to start a character reference, so it needs to be encoded as a character reference

HTML – Character References Example

- The ampersand (**&**) is used to start a character reference, so it needs to be encoded as a character reference
  - `&` ⇒ &
  - `&` ⇒ & in ASCII
  - `&` ⇒ & in HEX
  - `&` ⇒ & in longer form HEX

HTML – Character References Example

é

- `é`
- `é`
- `é`

Modern browsers can handle files with special characters, but these standards come from a time when they did not

HTML – Character References Example

- `🐛`
- `🐛`

HTML – Character References Example

- Why must ‘<’ be encoded as a character reference?
  - <
  - <
  - 0
  - 0

Web Scrapers / Crawlers / Spiders

- Because HTML is highly structured, it leaves it very susceptible to programs that extract information from the webpage
  - Can then render phishing pages to appear authentic
  - Or host other people's content with injected affiliate links
  - Or be genuine web indexing companies

Web Scrapers / Crawlers / Spiders

- Because HTML is highly structured, it leaves it very susceptible to programs that extract information from the webpage
  - Can then render phishing pages to appear authentic
  - Or host other people's content with injected affiliate links
  - Or be genuine web indexing companies

- Good rule of thumb is to include a **robots.txt** file in the root directory to inform "good" bots what to **not** index
  - https://www.robotstxt.org/
  - ```
    User-agent: *
    Disallow: /cgi-bin/
    Disallow: /~csc405/
    Disallow: *.gif
    Disallow: /flag/challengeXX.txt
    ```
    ```

Web Scrapers / Crawlers / Spiders

- Because HTML is highly structured, it leaves it very susceptible to programs that extract information from the webpage
  - Can then render phishing pages to appear authentic
  - Or host other people's content with injected affiliate links
  - Or be genuine web indexing companies

- Good rule of thumb is to include a **robots.txt** file in the root directory to inform "good" bots what to **not** index
  - https://www.robotstxt.org/
  -
```
User-agent: *
Disallow: /cgi-bin/
Disallow: /~csc405/
Disallow: *.gif
Disallow: /flag/challengeXX.txt
```

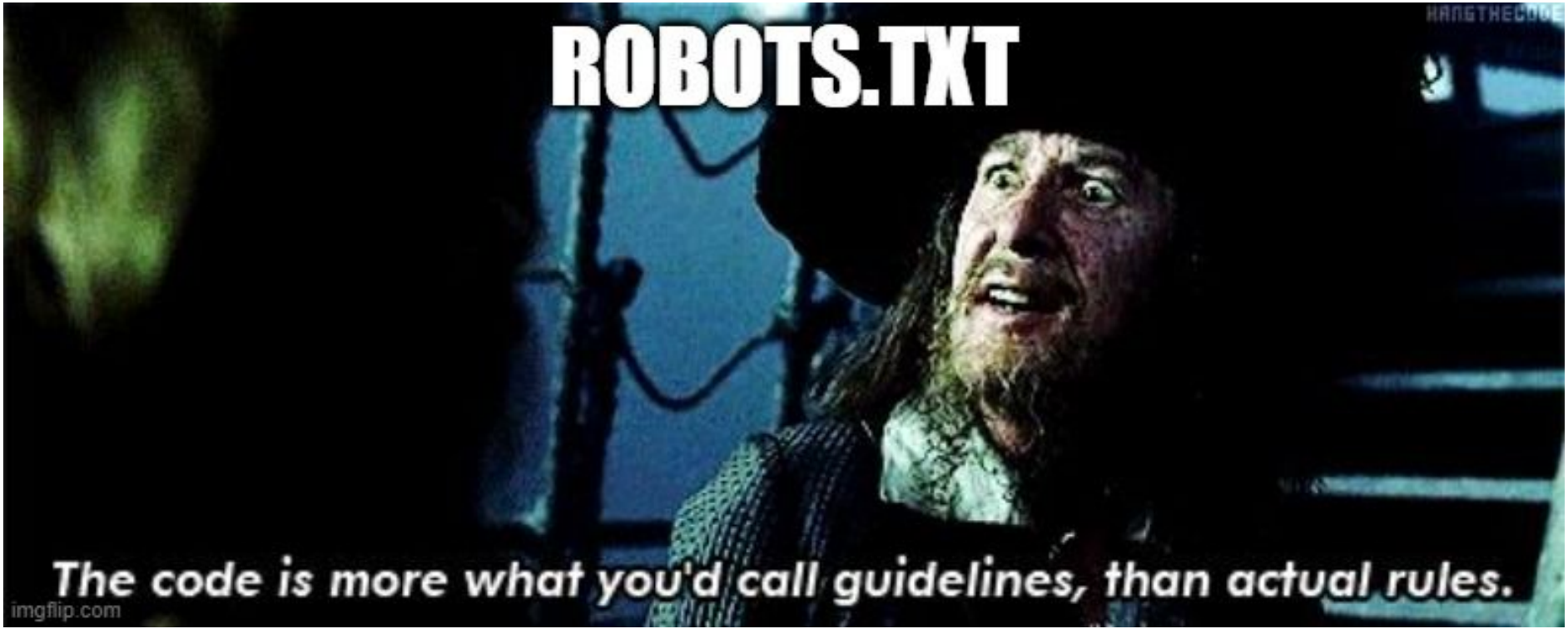
Web Scrapers / Crawlers / Spiders

- Because HTML is highly structured, it leaves it very susceptible to programs that extract information from the webpage
  - Can then render phishing pages to appear authentic
  - Or host other people's content with injected affiliate links
  - Or be genuine web indexing companies

- Good rule of thumb is to include a **robots.txt** file in the root directory to inform "good" bots what to **not** index
  - https://www.robotstxt.org/
  -
```
User-agent: *
Disallow: /cgi-bin/
Disallow: /~csc405/
Disallow: *.gif
Disallow: /flag/challengeXX.txt
```

Do not index /cgi-bin/, /~csc405/, any gifs, or explicitly do not index specific files

What about Bad Bots?



What about Bad Bots?

- Honeypots
  - Include **`Disallow: <location>`** in **`robots.txt`** but include a link to it in your webpage
  - Grab the IP address of the malicious bot for processing later

robots.txt

```
User-agent: *
Disallow: /honeypot/trap/
```

What about Bad Bots?

- Honeypots
  - Include `Disallow: <location>` in **robots.txt** but include a link to it in your webpage
  - Grab the IP address of the malicious bot for processing later

robots.txt

```
User-agent: *
Disallow: /honeypot/trap/
```

Website

```
<a href="/honeypot/trap/"></a>
```

What about Bad Bots?

- Honeypots
  - Include `Disallow: <location>` in **robots.txt** but include a link to it in your webpage
  - Grab the IP address of the malicious bot for processing later

robots.txt

```
User-agent: *
Disallow: /honeypot/trap/
```

Website

```
<a href="/honeypot/trap/"></a>
```

/honeypot/trap/index.php

```
file_put_contents('bad-bots.txt', GetIp() . "rn", FILE_APPEND);
```

What about Bad Bots?

- Frequently change the HTML structure
  - Auto generate random attribute values for tags with **`id`** and **`class`** attributes
  - Instead of

```
<div class="article-content" id="main">
```

use

```
<div class="U2ARCQs4oH" id="91JpNLuG51">
```

What about Bad Bots?

- Frequently change the HTML structure
  - Regularly change the nesting structure of the page
  - Instead of

```
<div class="article-content" id="main">
  ...content...
</div>
```

use

```
<div class="U2ARCQs4oH" id="91JpNLuG51">
  <div class="rhG7k8p7q091JpNLuG51">
    ...content...
  </div>
</div>
```

Amazon Prime

✓prime

Eligible for Free Shipping

Free Shipping by Amazon

Get FREE Shipping on eligible orders shipped by Amazon

From Our Brands

Amazon Brands

Top Brands in Electronics

Top Brands

span.a-declarative 308 × 590

Elements Recorder Console Sources Network Performance Memory Application Security Lighthouse Cookie-Editor

```
<div data-normaliseheight="false" class="a-section a-spacing-medium _octopus-search-result-card_style_apbSearchResultsContai
  <div class="_octopus-search-result-card_style_apbSearchResultItem__2-mx4">
    <span class="a-declarative" data-version-id="v13poyu19nzrc72qk2wbqfze5xu" data-render-id="r17g4jroadhhe426zh4ty8e8xf0" d
    da-puis-card-container-declarative" data-csa-c-id="5gikwq-85mybo-vs5urr-oz6oh9"> == $0
      <div class="puis-card-container s-card-container s-overflow-hidden aok-relative puis-expand-height puis-include-conten
        ::before
        <div class="a-section a-spacing-base">
          <div class="s-product-image-container aok-relative s-text-center s-image-overlay-grey puis-image-overlay-grey s-pad
          zrc72qk2wbqfze5xu" style="padding-top: 0px;">
            <span data-component-type="s-product-image" class="rush-component" data-version-id="v13poyu19nzrc72qk2wbqfze5xu"
```

Your Security Zen

SO YOU WANT TO STOP HOTLINKING AND BANDWIDTH THEFT

Would you like to stop hotlinkers? Is your web hosting bandwidth bill growing each month? Looking for web hosting? Want to test image hotlink protection for your web site? Here is information on using an **htaccess** file to stop hotlinking and bandwidth theft.

HOW DO I STOP HOTLINKING AND BANDWIDTH THEFT?

You can stop others from hotlinking your site's files by placing a file called **.htaccess** in your Apache site root (main) directory. The period before the name means the file is hidden, so you may want to edit your file as *htaccess.txt*, upload it to your server, then rename the txt file to *.htaccess* in your directory. Contact your web host on how to access your directories and configure your **.htaccess** file.

Example: Your site url is *www.mysite.com*. To stop hotlinking of your images from other sites and display a replacement image called *qX4w7.gif* from an image host, place this code in your **.htaccess** file:

```
RewriteEngine On
RewriteCond %{HTTP_REFERER} !^http://(.+\.)?mysite\.com/ [NC]
RewriteCond %{HTTP_REFERER} !^$
RewriteRule .*\.(jpe?g|gif|bmp|png)$ https://i.imgur.com/ZtXiCBw.gif [L]
```

If a request for an image on my server **is not** coming from my server, redirect them to this image

Source: https://altlab.com/htaccess_tutorial.html

Your Security Zen - "Permissions" Bypass

Missouri's Department of Elementary and Secondary Education listed the Social Security numbers of teachers and administrators, hidden in the HTML of their webpage

Missouri governor threatens legal action against journalist who found flaw that exposed Social Security numbers

By Sean Lyngaas, CNN

Published 7:11 PM EDT, Thu October 14, 2021

Source: https://www.cnn.com/2021/10/14/media/mike-parson-st-louis-post-dispatch/index.html