

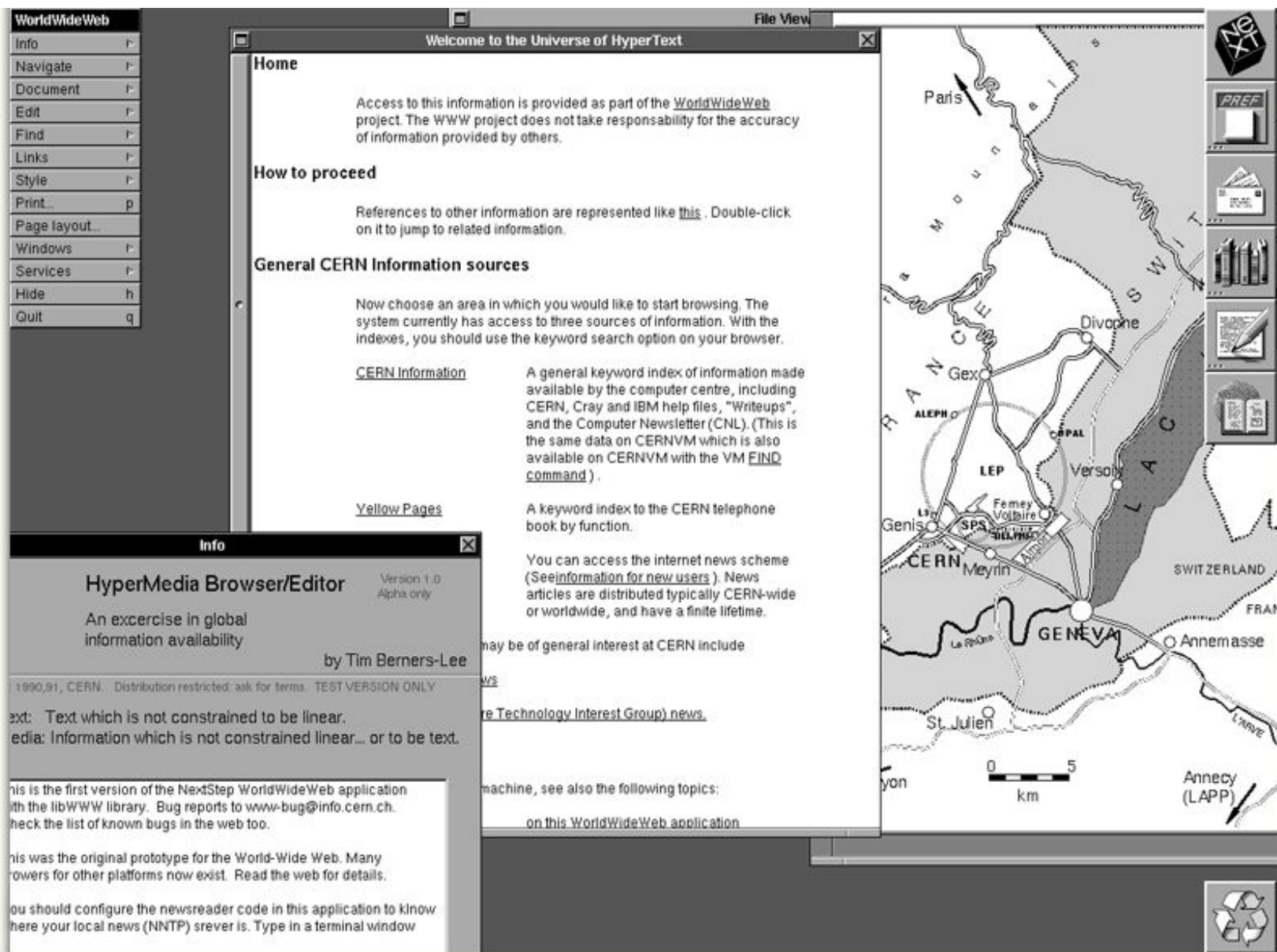
CSC 405

Computer Security

Web Security

Alexandros Kapravelos
akaprav@ncsu.edu

(Derived from slides by Giovanni Vigna and Adam Doupe)



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.

Sir Tim Berners-Lee



**ACM Turing
Award 2016**

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

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”

Three Central Questions

- How to name a resource?
- How to request and serve a resource?
- How to create hypertext?

Three Central Technologies

- 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)

HTML

URI

HTTP



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
- 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:akapprav@ncsu.edu`

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

URI – Reserved Characters

:
/
?

[
]
@
!
\$

&
,
(
)
*
+
,
;
=

URI – Percent Encoding

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

Alpha [a-zA-Z]

Digit [0-9]

-

.

|

~

URI – Percent Encoding

- Encode a byte outside the range 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?/alex`
 - `https://example.com/test/example%3A1.html?%2Falex`

URI – Absolute vs. Relative

- 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`

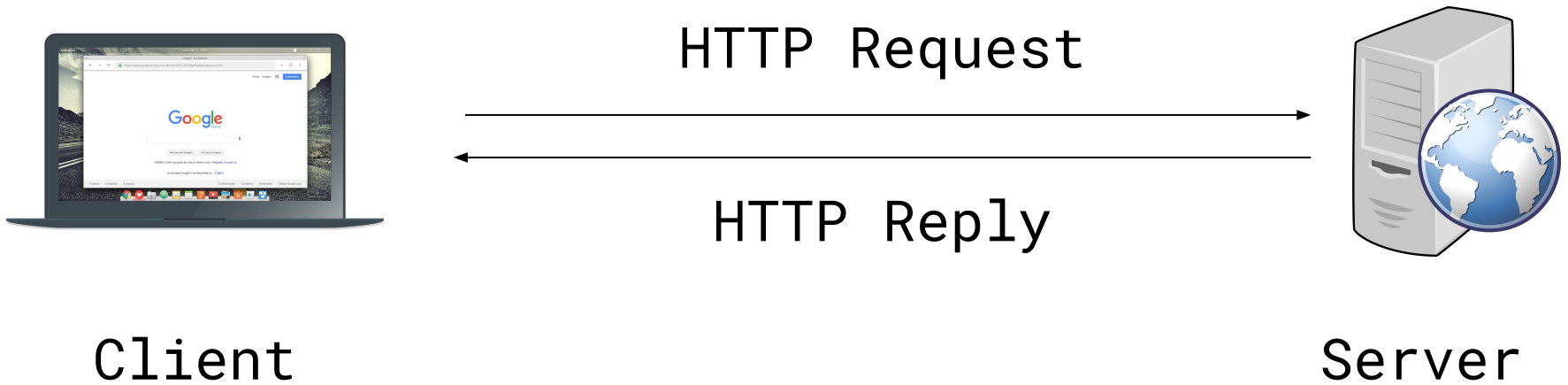
Hypertext Transport Protocol

- Protocol for how a web client can request a resource from a web server
- Based on TCP, uses port 80 by default
- Version 1.0
 - Defined in RFC 1945 (May 1996)
- Version 1.1
 - Defined in RFC 2616 (June 1999)
- Version 2.0
 - Based on SPDY, still under discussion

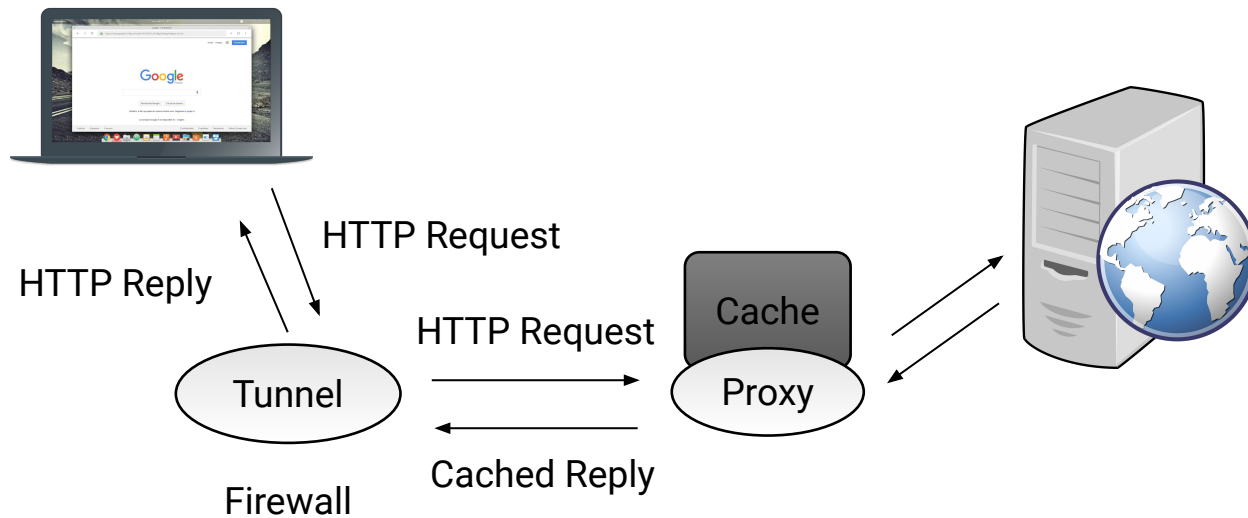
HTTP – Overview

- Client
 - Opens TCP connection to the server
 - Sends request to the server
- Server
 - Listens for incoming TCP connections
 - Reads request
 - Sends response

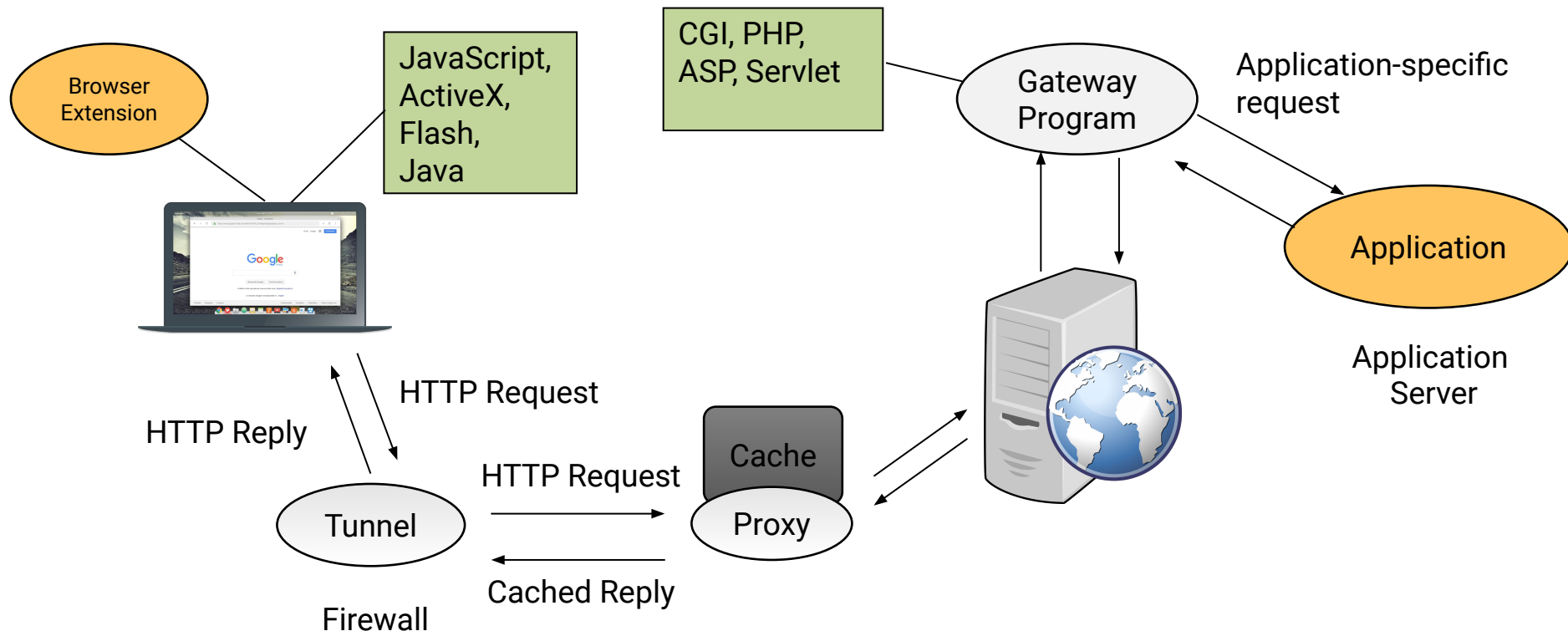
Architecture



Architecture



Architecture



Requests

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

Requests – Syntax

- Start line, followed by headers, followed by body
 - Each line separated by CRLF
- Headers separated by body via empty line (just CRLF)

Requests – Methods

- The method that that client wants applied to the resource
- Common methods
 - 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 that the server delete the resource identified by the URI
- TRACE – used to invoke a remote, application-layer loop-back of the request message and the server should reflect the message received back to the client as the body of the response
- CONNECT – used with proxies
- ...
 - A webserver can define arbitrary extension methods

Requests – Example

GET / HTTP/1.1

User-Agent: curl/7.37.1

Host: www.google.com

Accept: */*

Modern Requests

GET / HTTP/1.1

Host: www.google.com

Accept-Encoding: deflate, gzip

Accept:

text/html,application/xhtml+xml,application/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

Responses

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

Responses – Syntax

- Status line, followed by headers, followed by body
 - Each line separated by CRLF
- Headers separated by body via empty line (just CRLF)
- Almost the same overall structure as request

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 – Status Codes

- "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

Requests – Example

GET / HTTP/1.1

User-Agent: curl/7.37.1

Host: www.google.com

Accept: */*

Responses – Example

HTTP/1.1 200 OK

Date: Tue, 13 Jan 2015 03:57:26 GMT

Expires: -1

Cache-Control: private, max-age=0

Content-type: text/html; charset=ISO-8859-1

Set-Cookie: ...

Server: gws

X-XSS-Protection: 1; mode=block

X-Frame-Options: SAMEORIGIN

Alternate-Protocol: 80:quic,p=0.02

Accept-Ranges: none

Vary: Accept-Encoding

Transfer-Encoding: chunked

```
<!doctype html><html itemscope=""  
itemtype="http://schema.org/WebPage" lang="en"><head><meta  
content="Search the world's information, including webpages,  
images, videos and more. Go ...
```

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 base64 encoded (RFC 2045) username and password

`Authorization: Basic QWxhZGRpbjpvYVUyIHVlc2FtZQ==`

- 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 web server has to have 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
 - This is a professional-grade tool that I use

Hypertext Markup Language

- A simple data format used to create hypertext documents that are portable from one platform to another
- Based on Standard Generalized Markup Language (SGML) (ISO 8879:1986)
- HTML 2.0
 - Proposed in RFC 1866 (November 1995)
- HTML 3.2
 - Proposed as World Wide Web Consortium (W3C) recommendation (January 1997)
- HTML 4.01
 - Proposed as W3C recommendation (December 1999)
- XHTML 1.0
 - Attempt by W3C to reformulate HTML into Extensible Markup Language (XML) (January 2000)
- HTML 5.0
 - Proposed as W3C recommendation (October 2014)
- HTML 5.1
 - Under development

HTML – Overview

- Basic idea is to “markup” document with tags, which add meaning to raw text
- Start tag:
 - `<foo>`
- Followed by text
- End tag:
 - `</foo>`
- Self-closing tag:
 - `<bar />`
- Void tags (have no end tag):
 - ``

HTML – Tags

- Tag are hierarchical

HTML – Tags

```
<html>  
  <head>  
    <title>Example</title>  
  </head>  
  <body>  
    <p>I am the example text</p>  
  </body>  
</html>
```

HTML – Tags

- `<html>`
 - `<head>`
 - `<title>`
 - Example
 - `<body>`
 - `<p>`
 - 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
- Four different syntax
 - `<foo bar>`
 - foo is the tag name and bar is an attribute
 - `<foo bar=baz>`
 - The attribute bar has the value baz
 - `<foo bar='baz'>`
 - `<foo bar="baz">`
- Multiple attributes are separated by spaces
 - `<foo bar='baz' disabled required="true">`

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
- `Example`

[Example](#)

HTML – Basic HTML 5 Page

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>CSC 591</title>
  </head>

  <body>
    <a href="http://example.com/">Text</a>
  </body>
</html>
```


HTML – Browsers

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

HTML – Parsed HTML 5 Page

DEMO

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”
- Three types, each starts with & and ends with ;
 - Named character reference
 - &<predefined_name>;
 - Decimal numeric character reference
 - &#<decimal_unicode_code_point>;
 - Hexadecimal numeric character reference
 - &#x<hexadecimal_unicode_code_point>;
- 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 must be encoded as a character reference
- `&`
- `&`
- `&`
- `&`

HTML – Character References

Example

- é
- é
- é
- é

HTML – Character References

Example

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

Your Security Zen

Adversarial Patch

An image-independent patch that is extremely salient to a neural network. This patch can then be placed anywhere within the field of view of the classifier, and causes the classifier to output a targeted class.

