HTTP/2

A Pentester's Perspective

What is it?

- A new version of Hyper Text Transfer Protocol (HTTP)
- Big Changes
- Lots to talk about

A Brief History

HTTP/0.9

- March 12, 1989: Tim Berners-Lee submits proposal for what will become HTTP
- Plain-Text protocol
- FTP with links
- Read Only (GET)
- HTML only (no images, stylesheets, or any other files)
- One request is one connection

HTTP/1.0

- First official version (RFC 1945) May 1996
- Non-HTML formats supported
- More methods introduced:
  - GET / HEAD / POST / PUT / DELETE
- No virtual hosting
- One request is still one connection

HTTP/1.1

- Introduced June 1999 with RFC 2616
  - Replaced with later RFCs in 2014
- All the contents
  - Support for compression
- Virtual Hosting
  - Host header added (Required)
- Persistent connection
  - One connection can be established for multiple requests (Pipelined)

Example Request

GET /docs/index.html HTTP/1.1

Host: www.nowhere123.com

Accept: image/gif, image/jpeg, */*

Accept-Language: en-us

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

(Blank Line \r\n)

Example Response

HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 08:56:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sun, 20 May 2018 07:16:26 GMT
Accept-Ranges: bytes
Content-Length: 44
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug

<html><body><h1>It works!</h1></body></html>

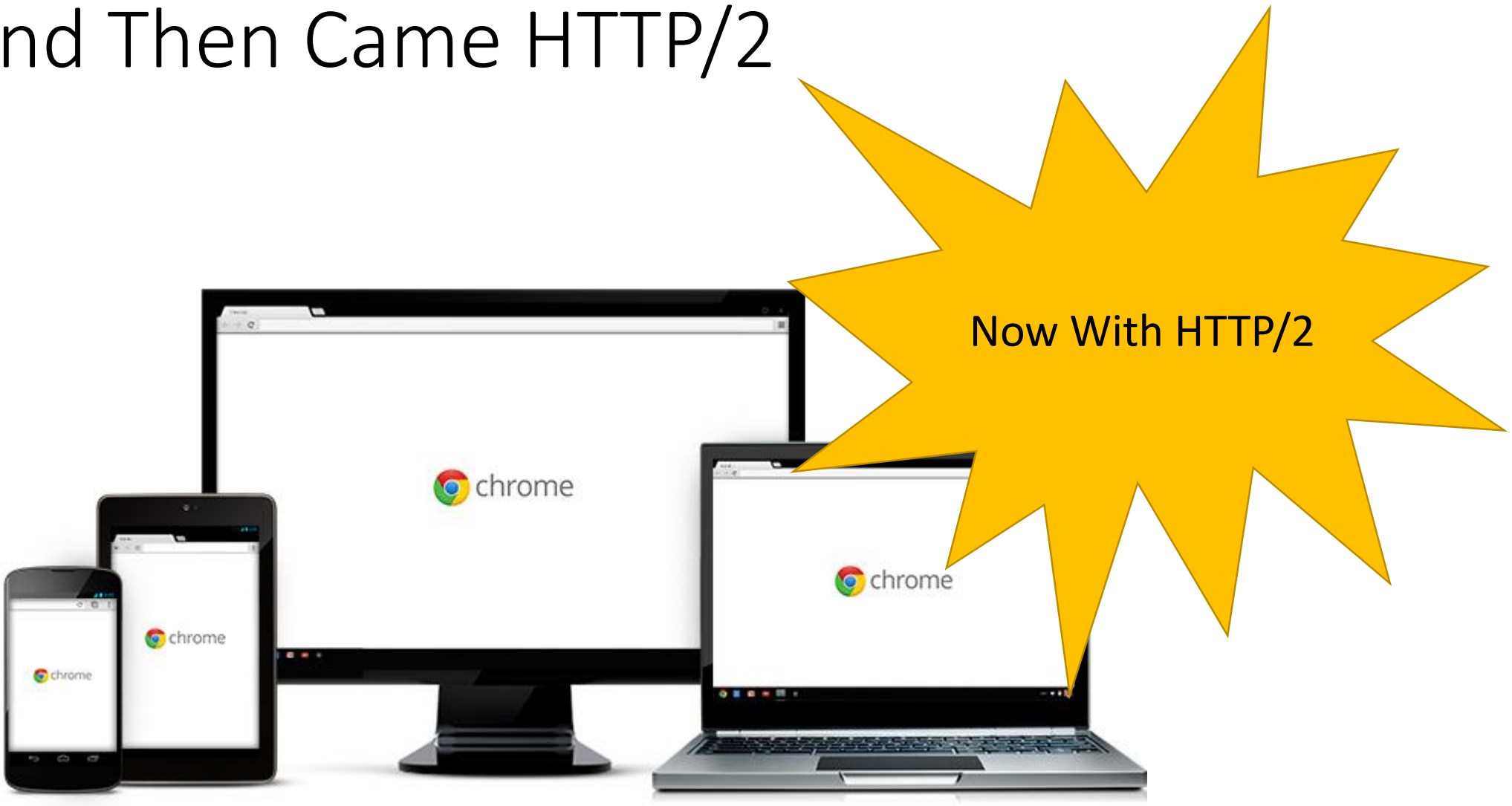
HTTP/2

The new hotness

First there was SPDY

- Published proposal in 2009
- Requires HTTPS
- Published by Google and incorporated in Chrome
- Encapsulates most of what will become HTTP/2

And Then Came HTTP/2

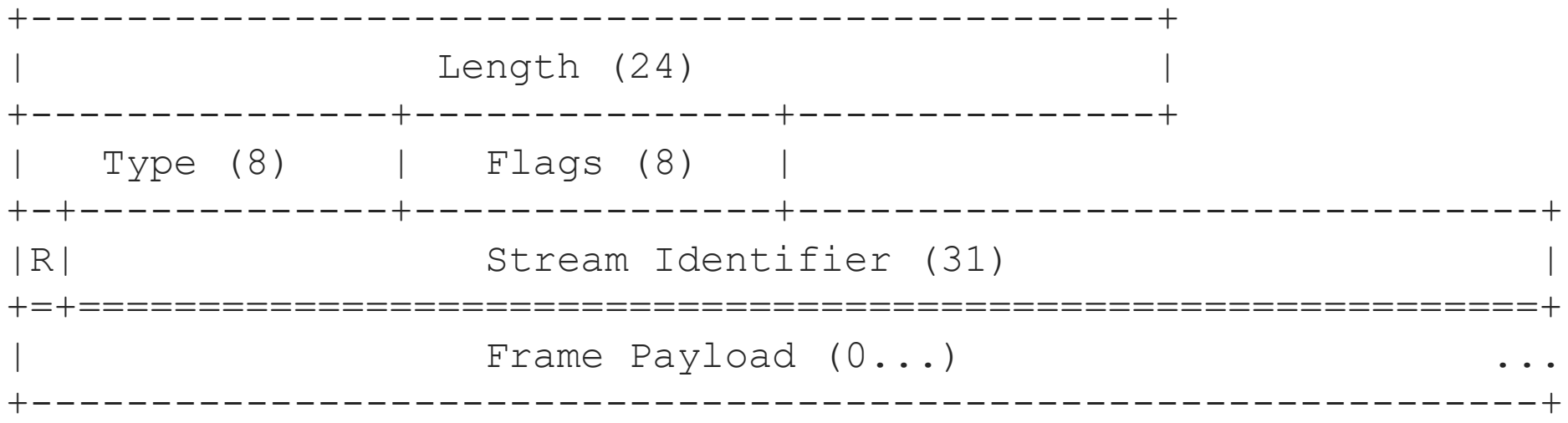


What did we get?

- Binary Protocol
- Header Compression (HPACK)
- Multiplexing
- Server Push

Binary Protocol

- Binary is better / simpler
- Lighter weight
- Reduced Network Latency
- Better Compression
- Framing



Header Compression

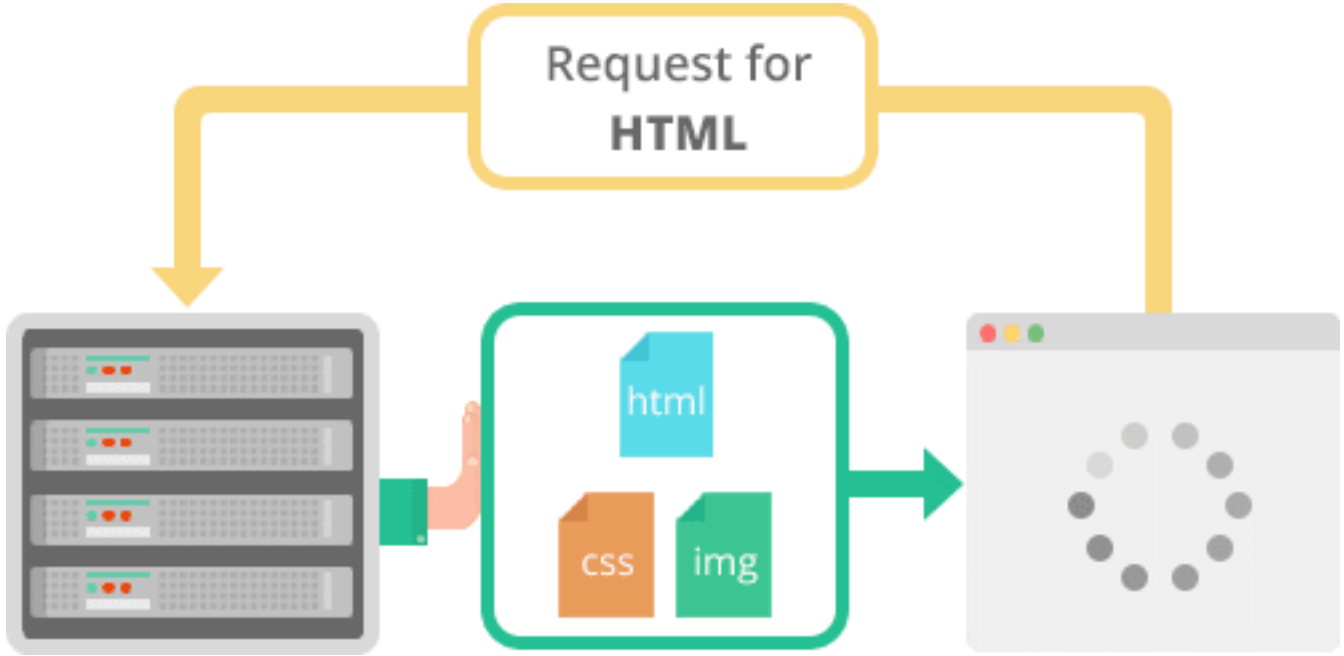
- Uses HPACK
- Reduced Overhead
- Smaller == Quicker
- Not vulnerable to attacks like CRIME or BREACH

Multiplexing

- Not pipelined
- Streaming
- Bi-directional sequence of frames
- Stream Prioritization

Server Push

- Server can push data to client before requested
- Cached by client
- Reused across pages
- Client can decline
- Great for resources



The Handshake

HTTP/1.1 Version

- Request sent with Upgrade header
  - “h2c” For HTTP
  - “h2” for HTTPS
- Includes “HTTP2-Settings” header (Base64 Encoded)
- Receives a “101 Changing Protocol” response
- Client connecting to HTTPS URI uses TLS-ALPN extension

Attacking

Put on your hacker hats

Server Attacks

- Parsing is Hard!
- Same Old Mistakes All Over Again
  - H20 – Directory Traversal (CVE-2016-1133)
  - Lots Of Denial Of Service
    - HPACK Bombs
    - Slow Read Attack (similar to SlowLoris)
    - Misconfiguration (Dependency and Priority)

Security Defenses

- Parsing is Hard!
- Web Application Firewalls
  - Most don’t support HTTP/2
- Intrusion Detection / Prevention Systems
  - Most don’t fully support
  - Some listed as experimental

Application Attacks

- All the Data
- Misconfigured Server Push
  - Pushing more than intended
  - Information Leakage
- Look in the Cache

Tools

Tools

- Proxies
  - MITM Proxy
  - Charles Proxy
- Testing Tools
  - cUrl (requires nghttp)
  - Wireshark
  - http2fuzz
  - h2load
  - h2spec
  - And more https://github.com/http2/http2-spec/wiki/tools

Conclusion

Things to consider

- Are you seeing everything?
  - Look for 101 Changing Protocol
  - Look for server push cache (information leakage)
- Is the server configured correctly
  - Look for vulnerable implementations
  - Look for misconfigurations

The Future

- Much more research is needed
  - Servers need to be tested and fuzzed
- Tools need to be updated
  - Proxies need to work with HTTP/2
  - Security Defenses need to understand it

Thanks - Questions

- Joshua Barone
- @tygarsai