OWASP 2021
>irtual
APPSEC

PRESENTED BY:
Erik Elbieh, Security Consultant & Researcher
Palindrome Technologies

We’re not in HTTP anymore: Investigating WebSocket Server Security

Talk Summary

1. How WebSockets Work
2. Summary of WebSockets Research
3. New STEWS tool(s)

Erik Elbieh's Brief Bio

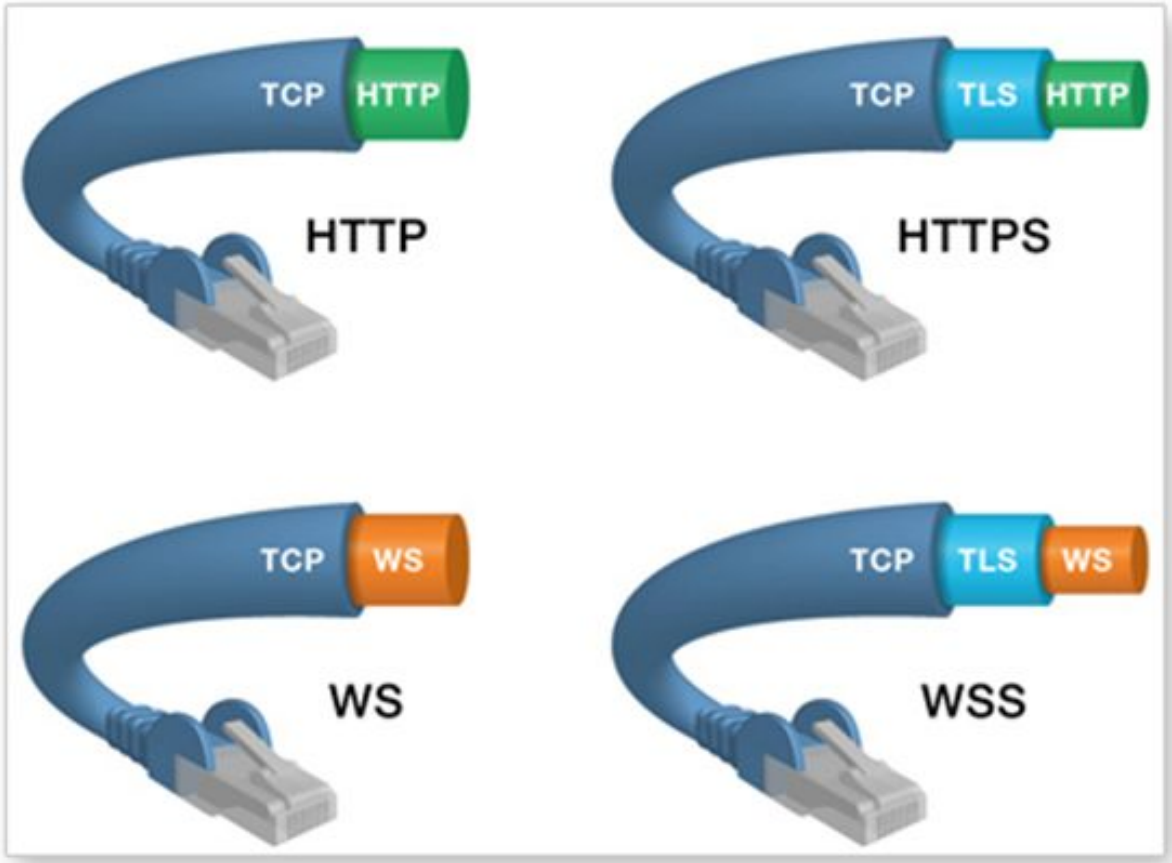
- Security Researcher and Consultant at Palindrome Technologies
  - Pen testing telecom systems, web apps, Kubernetes, and more
- Previously a Security Engineer at General Motors
  - Secured vehicle modules, Bluetooth specialist
- OSCP since 2019
- Graduated from Columbia University and Bard College at Simon's Rock
- More at erikelbieh.com

Part 1: How WebSockets Work

WebSocket Protocol History

- Created in 2010-2011 (RFC6455)
- Provides a low-overhead web protocol for real-time communications
- WebSocket servers are often distinct from HTTP servers

WebSocket vs. HTTP



Source: https://developerinsider.co/difference-between-http-and-http-2-0-websocket/

WebSocket vs. HTTP

- WebSockets don't use the request/response approach that HTTP does. WebSockets remain open until closed. This allows webpage updates to happen without refreshing the webpage (alternative to XHR, etc.)
  - Note: Proxies are usually built for the request/response approach HTTP uses and can have WebSockets vulnerabilities
- HTTP has a lengthy header with every request/response, but after a WebSocket is started, there is no similar header. Lower overhead is good for frequent back-and-forth real time communication.

WebSockets Stack

Any Protocol

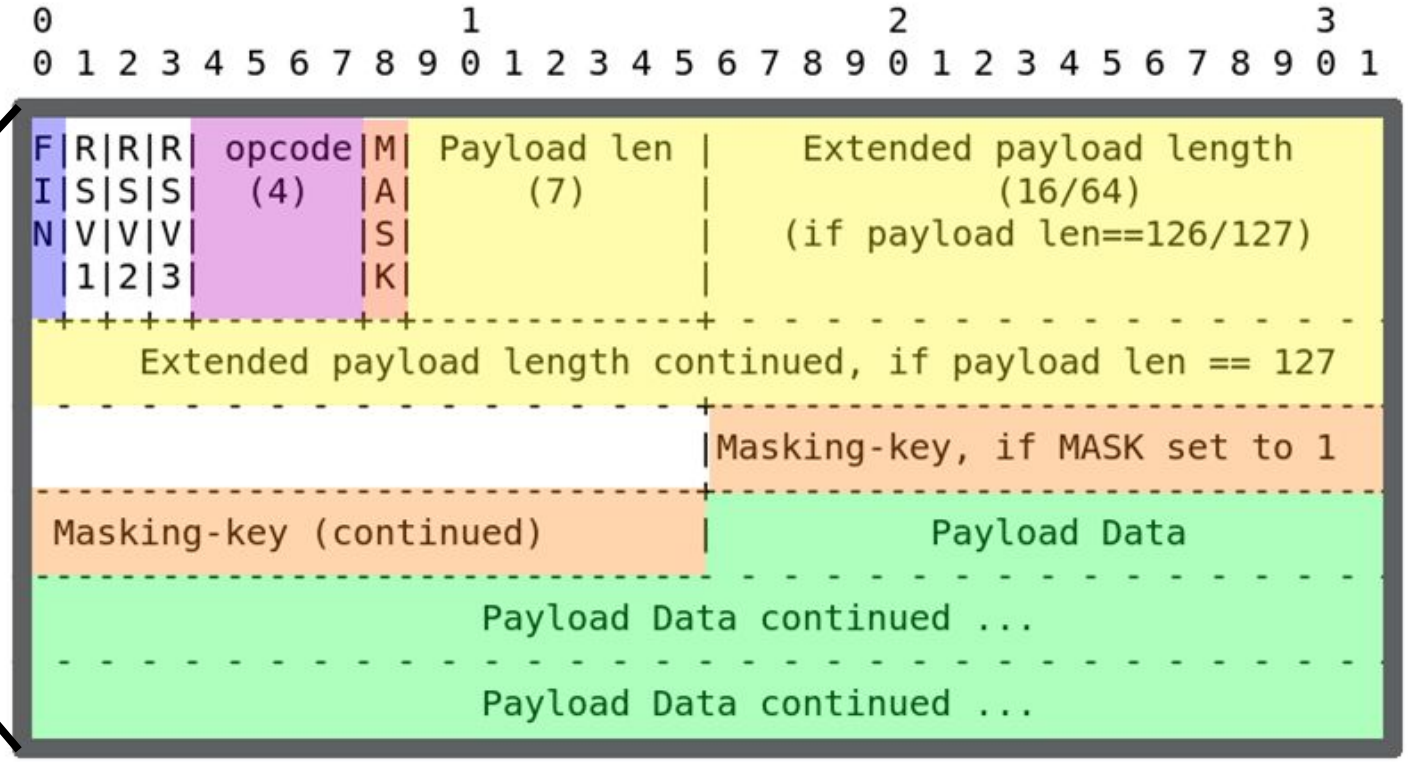
(socket-io, engine-io, STOMP, WAMP, MQTT, etc.)

WebSockets

(!= HTTP)

TCP/IP

WebSockets Frame



WebSockets Higher-Level Protocols

- Some protocols are (or can be) implemented on top of WebSockets:
  - Socket.io
  - Engine.io
  - STOMP
  - WAMP
  - MQTT

WebSocket Example: Phase 1

Key Point: WebSockets use HTTP to "kickstart" the WebSocket protocol

Step 1: HTTP request from browser

(Note the many uses of the word "WebSocket")

```
> GET / HTTP/1.1
> Host: 127.0.0.1:8085
> User-Agent: curl/7.74.0
> Accept: */*
> Upgrade: websocket
> Sec-WebSocket-Key: dXP3jD9Ipw0B2EmWrMDTEw==
> Sec-WebSocket-Version: 13
> Connection: upgrade
>
```

Step 2: HTTP response from server

"101 Switching Protocols" is a 'rare' HTTP status code that often indicates a WebSocket was started

```
< HTTP/1.1 101 Switching Protocols
< Upgrade: websocket
< Connection: Upgrade
< Sec-WebSocket-Accept: GLWt4W8Ogwo6lmX9ZGa314RMRr0=
< X-Powered-By: Ratchet/0.4.3
```

WebSocket Example: Phase 2

Not much to see because the WebSocket Protocol focuses on minimizing overhead.
Chat application demo shown below

```
> Look, matey, I know a dead parrot when I see one, and I'm looking at one right now.
< No no he's not dead, he's, he's restin'! Remarkable bird, the Norwegian Blue, idn'it, ay? Beautiful plumage!
> The plumage don't enter into it. It's stone dead.
< Nononono, no, no! 'E's resting!
```

WebSockets in the Wild

Use cases include:

- Chat bots, especially customer service
- Slack, Discord, and other chat platforms
- Maps tracking real-time movement
- Live finance data websites
- Cryptocurrency websites
- Smart TV remote control!?
- Kubernetes/Docker API!?

Try This at Home Kids!

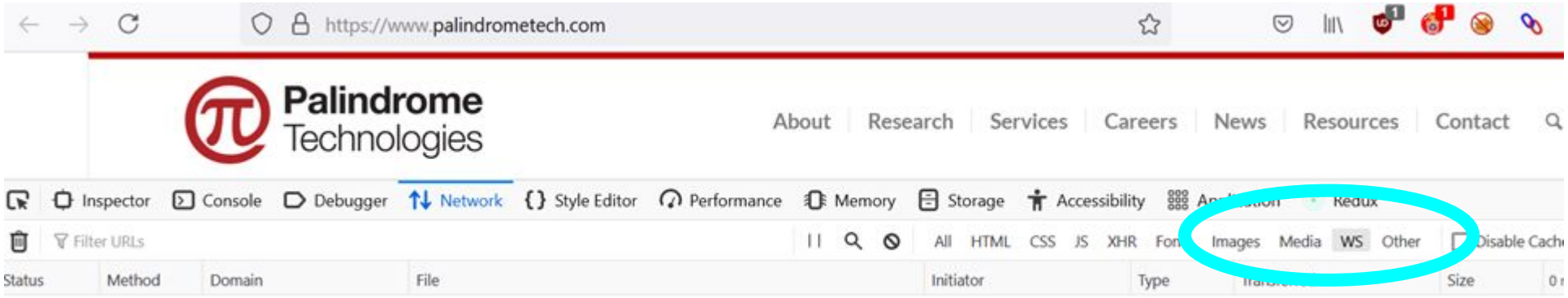
Try This at Home Kids!

1. Open web browser developer tools (Control+Shift+I in Firefox or Chrome) and visit the Network tab
2. Click "WS" to filter for only WebSockets traffic
3. Visit a webpage with WebSockets, such as:
   a. Finance: https://finance.yahoo.com/
   b. Sports: https://www.livescore.in/
   c. Chat: https://support.zoom.us
   d. Live maps: https://www.marinetraffic.com
4. Observe initial WebSocket request and response

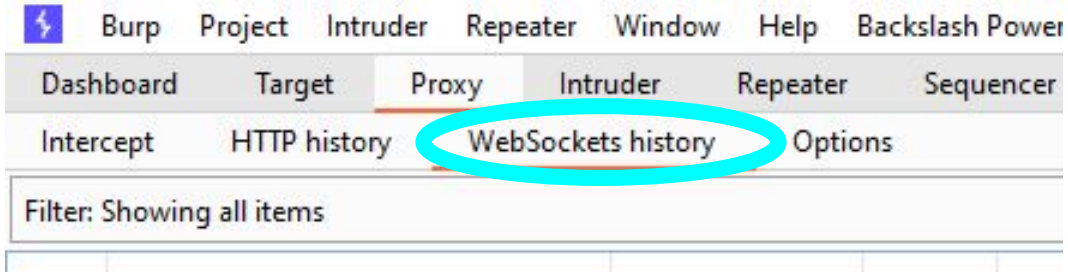
Note: Web proxy tools like Burp Suite and OWASP ZAP store WebSocket traffic in a separate tab from HTTP traffic

Finding WebSockets

Firefox



Burp Suite

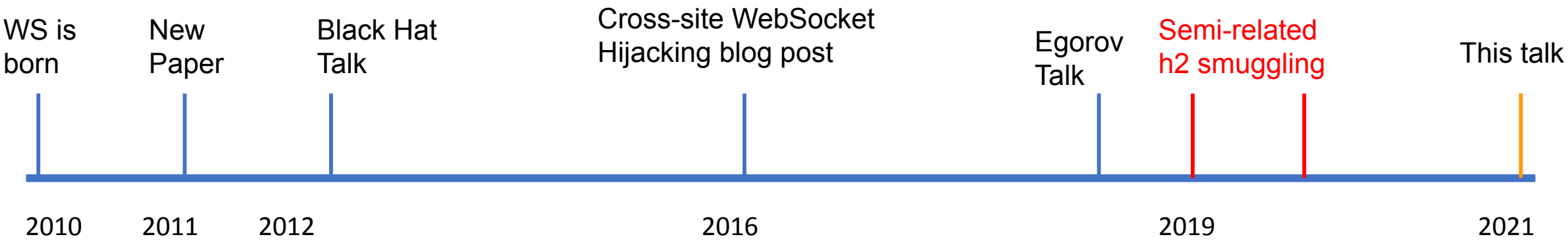


Part 2: Summary of WebSockets Research

Highlights of Prior WebSockets Research

- 2011: Firefox 4 temporarily removes WebSocket support due to protocol issue
- 2016: CORS, a HTTP CSRF mitigation, doesn't apply to WebSockets -> Cross Site WebSocket Hijacking (CSWSH)
- 2019: Proxies that don't properly handle WebSockets can leave to WebSocket Smuggling

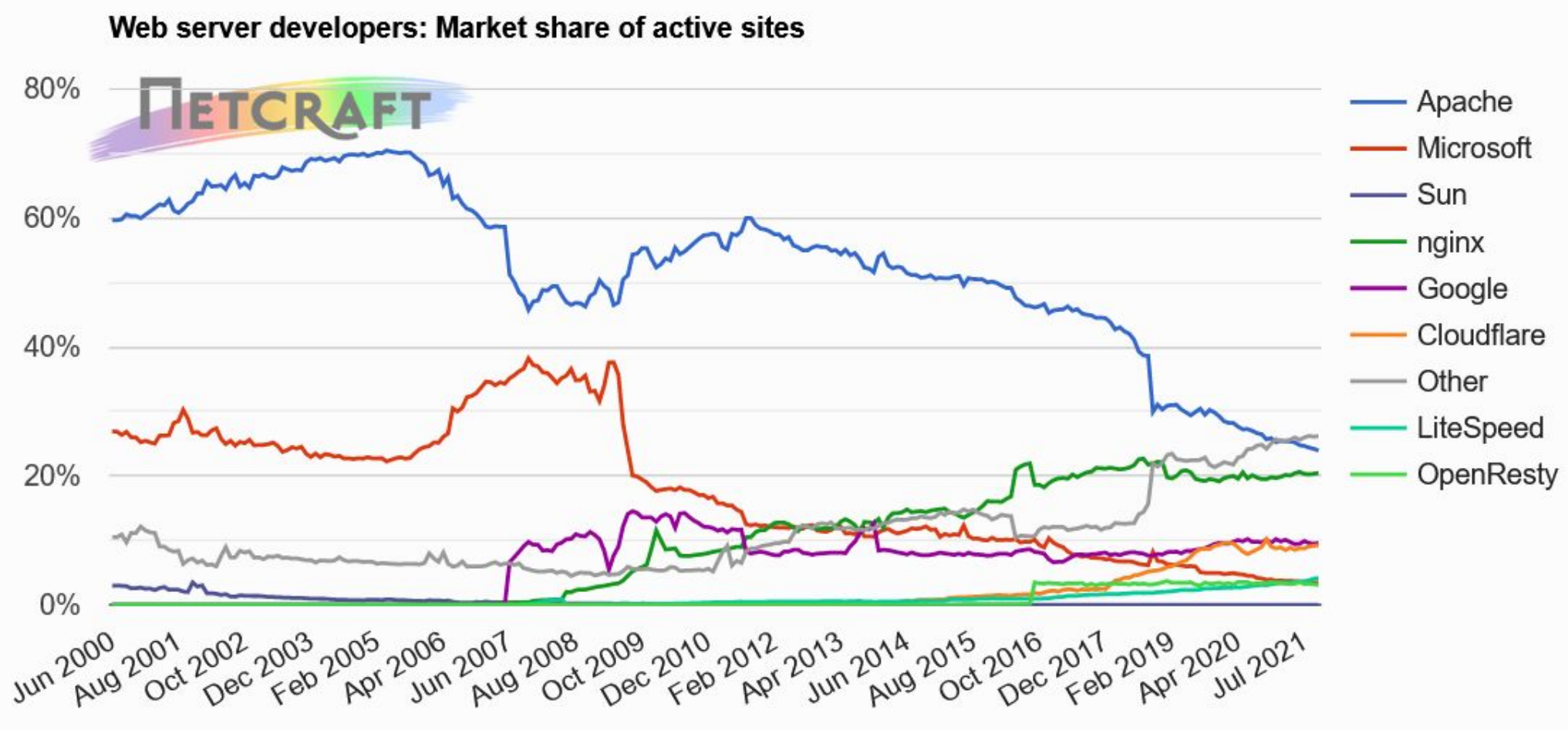
Timeline of Prior Related Research



Takeaways from Past Research

- Large scale security testing of WebSockets "in the wild" hasn't been publicly done before
- Research has been focused on the protocol level and proxy (mis)handling - but what about the server implementations?
- HTTP gets all the attention

HTTP Servers Market share



Source: https://news.netcraft.com/archives/2021/10/15/october-2021-web-server-survey.html

WebSockets Servers Market share

?

Common WebSocket Server Implementations

Name	Language	Repository	GitHub Stars (as of Nov 2021)
ws	JS	https://github.com/websockets/ws	17,200
Gorilla	Go	https://github.com/gorilla/websocket	15,700
uWebSockets	C++	https://github.com/uNetworking/uWebSockets	13,300
Java-WebSocket	Java	https://github.com/TooTallNate/Java-WebSocket	8,500
Cowboy	Erlang	https://github.com/ninenines/cowboy	6,500
Ratchet	PHP	https://github.com/ratchetphp/Ratchet	5,600
warp	Rust	https://github.com/seanmonstar/warp	5,500
WebSocket++	C++	https://github.com/zaphoyd/websocketpp	5,100
websocket-sharp	C#	https://github.com/sta/websocket-sharp	4,400
ws	Go	https://github.com/gobwas/ws	4,200
websockets	Python	https://github.com/aaugustin/websockets	3,700
libwebsockets	C	https://github.com/warmcat/libwebsockets	3,200

Part 3: New STEWS tool(s)

Who doesn't like free stuff?

Fresh out of the oven today!

1. STEWS repository: https://github.com/PalindromeLabs/STEWS
2. Whitepaper for STEWS and this talk: https://github.com/PalindromeLabs/STEWS/blob/main/ws-servers-paper.pdf
3. WebSockets Playground: https://github.com/PalindromeLabs/WebSocket-Playground
4. WebSockets Security Awesome: https://github.com/PalindromeLabs/awesome-websockets-security

Top Tools Lack WebSockets Custom Test Support

1. nmap: https://seclists.org/nmap-dev/2015/q1/134
2. Burp Suite (supports WebSockets, but not for extensions): https://forum.portswigger.net/thread/websockets-api-support-c8e1660b9f0ab
3. nuclei: https://github.com/projectdiscovery/nuclei/issues/539

STEWS

STEWS = Security Testing and Enumeration of WebSockets

Performs 3 key steps in WebSockets security testing:

1. Discovery
2. Fingerprinting
3. Vulnerability Detection

1. WebSockets Discovery

Why WebSocket endpoint discovery is difficult:

1. WebSockets use HTTP to start a connection, but observing HTTP alone does not indicate a WebSocket
2. Websites often start WebSockets using JavaScript, so WebSocket endpoints aren't always found by clicking links on a site (or simple spidering)
    a. Sometimes the main website is not linked to the WebSockets because the WebSockets endpoint is a standalone API
3. WebSockets may only exist at one specific URL path and at one specific port of the endpoint

1. WebSockets Discovery

Approaches to discovering WebSockets:

1. Finding WebSockets on a specific website
    a. Spider website HTML and search for WebSocket keywords in source code (downsides: false positives)
    b. Spider website and load all JavaScript and watch for HTTP 101 responses (downsides: loading all JS is slow)
2. Finding WebSockets on any website
    a. Use wordlist of common WebSocket endpoints and brute force a large list of websites (downsides: only testing wordlist endpoints)

1. WebSockets Discovery

Approaches to discovering WebSockets:

1. Finding WebSockets on a specific website
   a. Spider website HTML and search for WebSocket keywords in source code (downsides: false positives)
   b. Spider website and load all JavaScript and watch for HTTP 101 responses (downsides: loading all JS is slow)
2. Finding WebSockets on any website
   a. Use wordlist of common WebSocket endpoints and brute force a large list of websites (downsides: only testing wordlist endpoints)

Good for finding many WebSockets endpoints quickly

1. WebSockets Discovery

Difficulties in scalable WebSocket endpoint discovery:

1. Tools like masscan and zmap are fast at endpoint detection
   a. ...However, they work at the TCP/IP layer and we need to operate at the HTTP/WebSockets layer
2. Burp Suite's Turbo Intruder is fast at the HTTP layer
   a. ...However, Turbo Intruder documentation states "it's designed for sending lots of requests to a single host", not testing many hosts
3. ZGrab2 is a fast application-layer scanner
   a. ...However, requires some tweaks to support WebSockets requests

1. WebSockets Discovery

Acquiring large lists of URLs

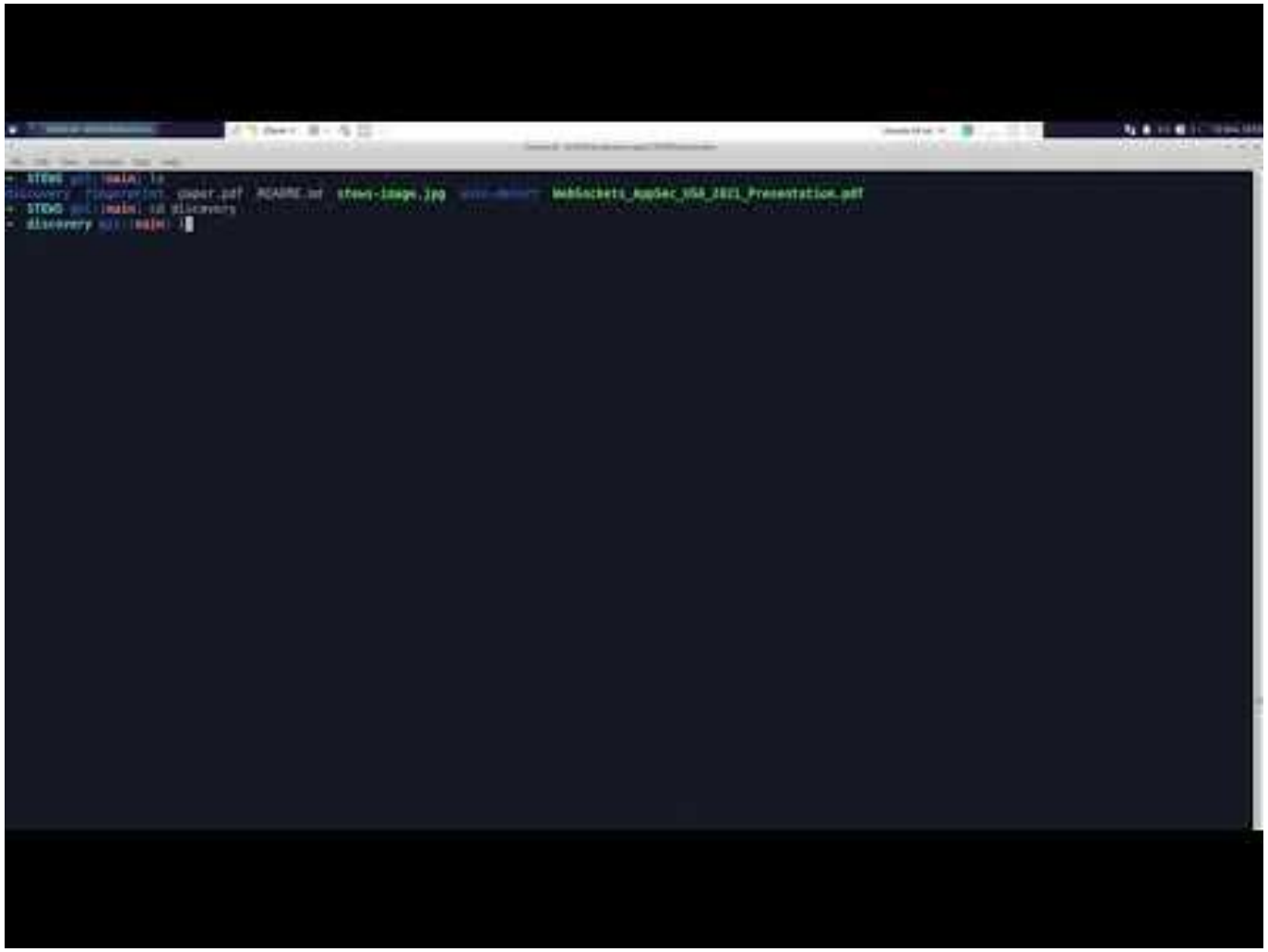
1. Googling "Top million URLs": https://www.letmegooglethat.com/?q=top+million+urls
2. Zone Files: https://czds.icann.org/home
   a. Zone Files are what DNS servers use for lookups
   b. Downside is that many URLs in zone file aren't active

1. WebSockets Discovery

Other difficulties:

- Large number of DNS lookups can be a bottleneck
  - Many DNS servers have rate limit
  - Using multiple DNS servers can help solution
- Obtaining wordlist of probable WebSocket paths to brute force requires manual effort
  - Found known WebSocket endpoints through random browsing, bug bounty reports, reading GitHub WebSocket repository issues

STEWS Discovery Demo



2. WebSockets Fingerprinting

"In theory there is no difference between theory and practice – in practice there is"

The challenge: to find implementation-level differences between WebSocket server implementations in order to identify them

2. WebSockets Fingerprinting

A few of the most popular WebSockets servers include:

- uWebSockets (C++)
- Gorilla (Go)
- ws (JavaScript)
- websockets (Python)
- Spring (Java)
- Websocket-sharp (C#)

But there's dozens of WebSockets server implementations

2. WebSockets Fingerprinting

To find WebSocket server identifying features, scripted a simple deterministic fuzzer to test different features of the WebSocket Server, such as:

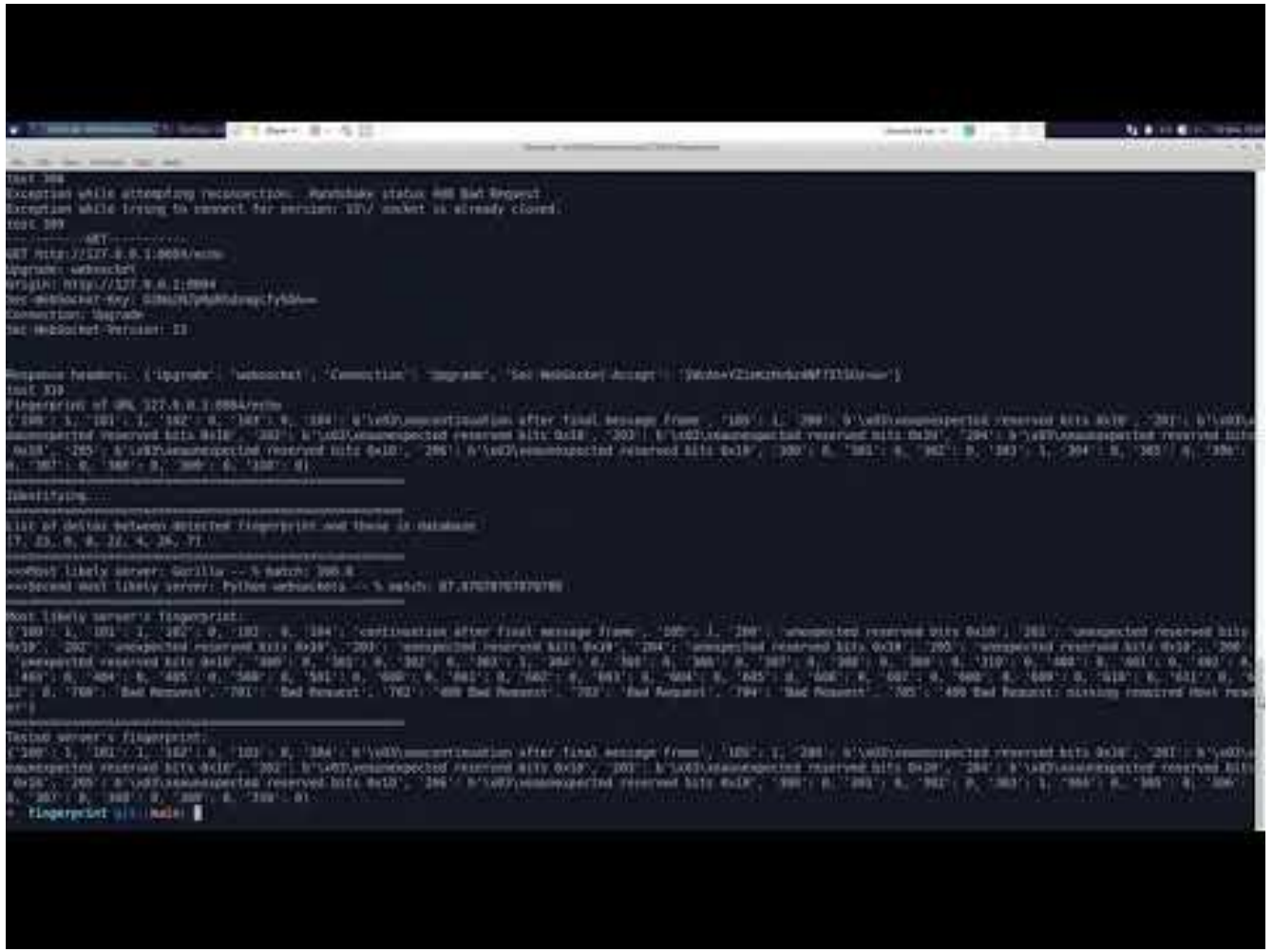
- Supported WebSocket Protocol Version Numbers
- Reserved and opcode bit support
- Verbose error messages
- Default maximum data length

2. WebSockets Fingerprinting

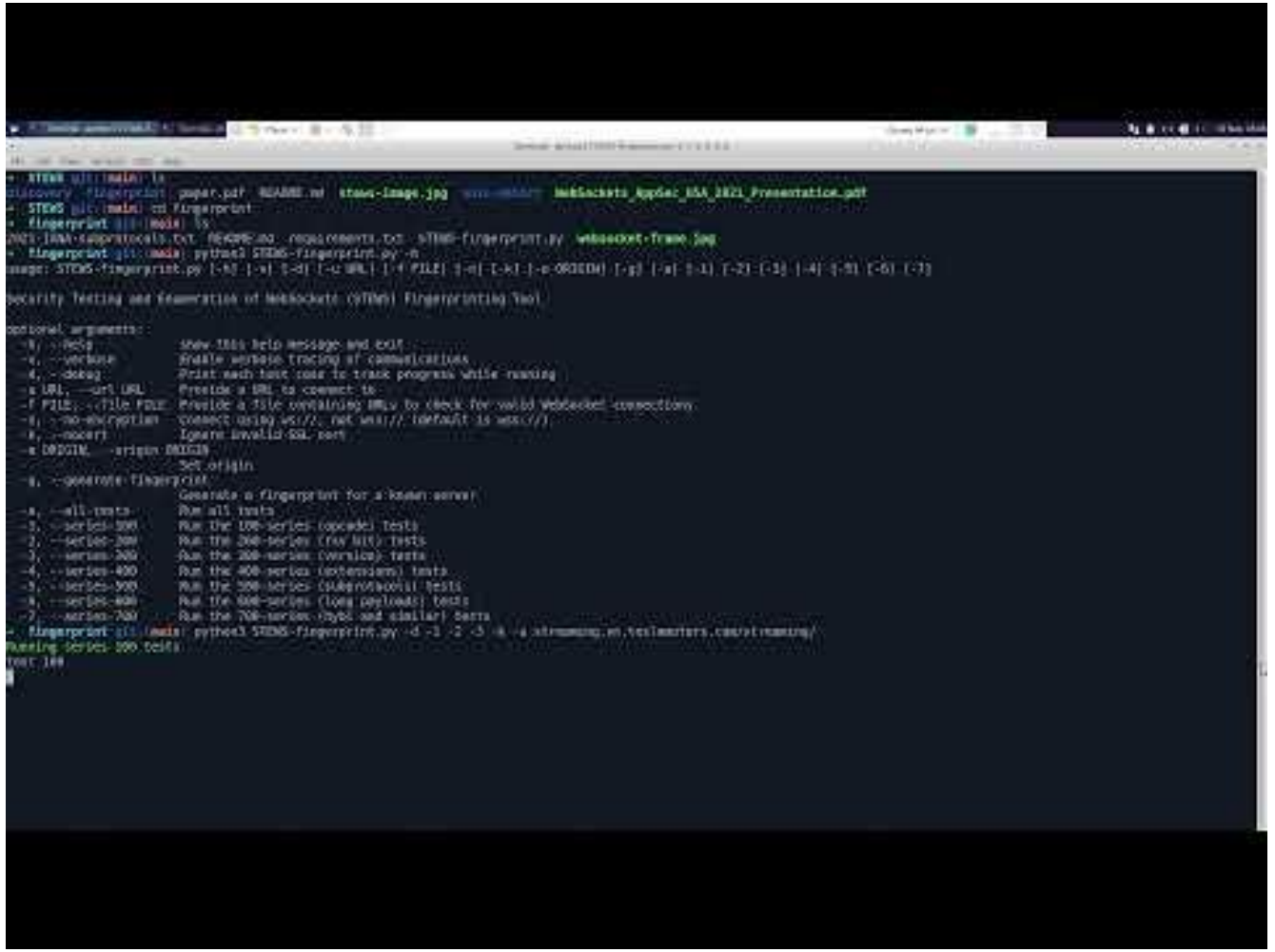
Differences from other fingerprinting tools:

- HTTP fingerprinters like wafw00f only handle 1 protocol, whereas WebSockets uses HTTP to negotiate the switch to WebSockets, meaning STEWS fingerprinting handles 2 protocols
-

STEWS Fingerprint Local Server Demo



STEWS Fingerprint Public Server Demo



3. WebSockets Vulnerability Detection

WebSocket servers have a few CVEs…

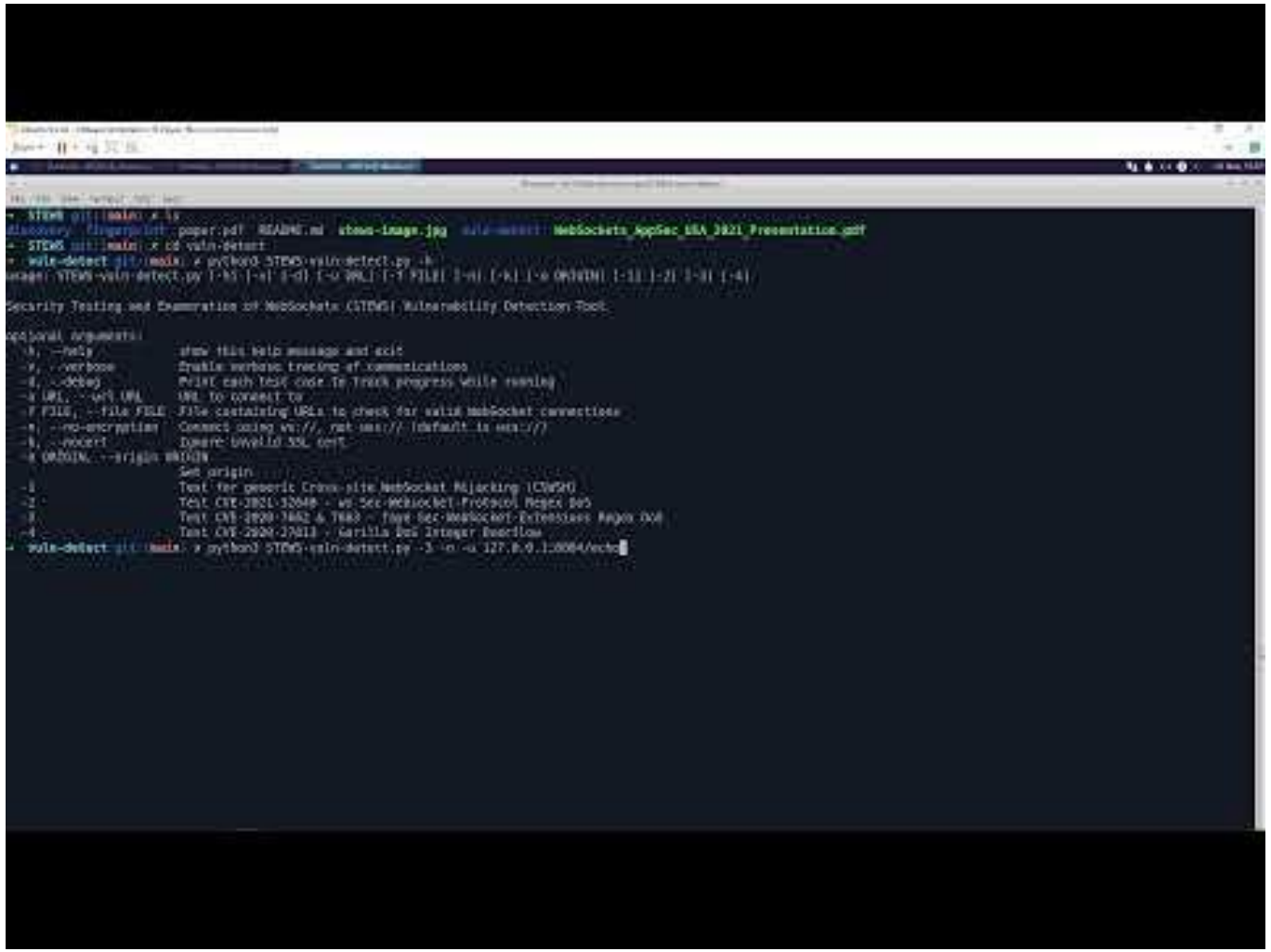
A longer list of WebSocket server CVEs found in [WebSocket Security Awesome]

CVE ID	Vulnerable package	Related writeup	Vulnerability summary
CVE-2021-42340	Tomcat	Apache mailing list	DoS memory leak
CVE-2020-36406	uWebSockets	Google OSS-Fuzz	Stack buffer overflow
CVE-2021-33880	Python websockets		HTTP basic auth timing attack
CVE-2021-32640	ws	GitHub Advisory	Regex backtracking Denial of Service
CVE-2020-24807	socket.io-file	Auxilium Security	File type restriction bypass
CVE-2020-15779	socket.io-file	Auxilium Security	Path traversal
CVE-2020-27813	Gorilla	Auxilium Security	Integer overflow
CVE-2020-11050	Java WebSocket	GitHub advisory	SSL hostname validation not performed
CVE-2020-15134	faye-websocket	GitHub advisory	Lack of TLS certificate validation
CVE-2020-15133	faye-websocket	GitHub advisory	Lack of TLS certificate validation
CVE-2020-7663	Ruby websocket-extensions	Writeup	Regex backtracking Denial of Service
CVE-2020-7662	npm websocket-extensions	Writeup	Regex backtracking Denial of Service
CVE-2018-1000518	Python websockets		DoS via memory exhaustion when decompressing compressed data
CVE-2018-21035	Qt WebSockets	Bug report	Denial of service due large limit on message and frame size
CVE-2017-16031	socket.io	GitHub Issue	Socket IDs use predictable random numbers
CVE-2016-10544	uWebSockets	npm advisory	Denial of service due to large limit on message size
CVE-2016-10542	NodeJS ws	npm advisory	Denial of service due to large limit on message size

3. WebSockets Vulnerability Detection

- Ideally the detection process of a CVE does not involve exploiting it, but often there is no other way
- STEWS vuln-detect includes checks for a few CVEs, though more should be added in the future:
  - CVE-2020-27813 (Gorilla DoS Integer Overflow)
  - CVE-2020-7662 & CVE-2020-7663 (faye Sec-WebSocket-Extensions Regex DoS)
  - CVE-2021-32640 (ws Sec-Websocket-Protocol Regex DoS)

STEWS Vuln Detect Demo



Summary

Part 1: WebSockets are like HTTP, but often examined less closely

Part 2: Minimal research done around WebSockets security and tools lack support

Part 3: STEWS toolset provides off-the-shelf tooling for discovery, fingerprinting, and vulnerability detection of WebSockets servers

Ideas for Future Research

1. Security of WebSockets subprotocols
2. Security of WebSocket Compression (RFC 7692)
3. Fast JavaScript-based spidering to discover WebSocket endpoints on single domain
4. Can other HTTP-type attacks be ported to WebSockets servers?

Over a dozen additional ideas listed in whitepaper

Recommended Additional Resources

PortSwigger WebSocket mini-CTF exercises:
https://portswigger.net/web-security/websockets

Mikhail Egorov's conference talk:
https://www.youtube.com/watch?v=gANzRo7UHt8

WebSockets RFC, RFC 6455:
https://datatracker.ietf.org/doc/html/rfc6455

Thank You!

Questions?

OWASP 2021

>irtual

APPSEC

THANK YOU!