RSA CONFERENCE 2013

USING HTML5 WEBSOCKETS SECURELY

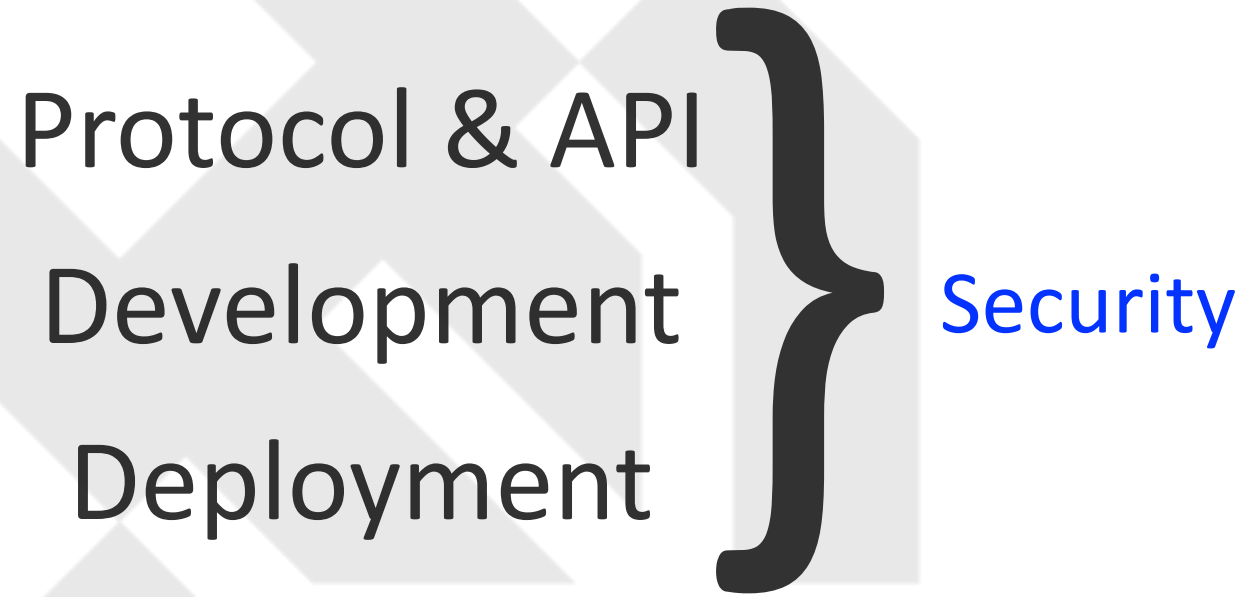
Mike Shema

Qualys, Inc.

Session ID: **ASEC-F41**

Session Classification: **Intermediate**

WebSockets



Forcing Persistence onto HTML4

...often at the server's expense of managing **one thread/ request**

...and restricted by the browser's **per-domain connection limit**

...while trying to balance an **efficient polling frequency**

...just to know when the server has some **data ready**.

Partial Solutions

Pseudo-bidirectional communication

web-socket-js -- The power of Flash's raw sockets with the benefits(?) of Flash's security

sockjs-client -- Pure JavaScript, choose your poison: long-polling, XHR, etc.

HTML5 Server-Sent Events

Properly-implemented long-polling

Content only flows from server → client

RFC 6455

The WebSocket Protocol enables **two-way communication** between a client running **untrusted code** in a **controlled environment** to a remote host that has opted-in to communications from that code.

Speak to Me

Protocol

Low overhead

Simple format

Content agnostic

HTTP compatible*

JavaScript API

Simple interface

Text or binary data

Origin security

Handshake

```
GET /?encoding=text HTTP/1.1
Host: echo.websocket.org
User-Agent: ...
Connection: Upgrade
Sec-WebSocket-Version: 13
Origin: http://www.websocket.org
Sec-WebSocket-Key: CjYoQD+BXC718rj3aiExxw==
```

16 random bytes

```
HTTP/1.1 101 Switching Protocols
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Accept: c4RVZSknSoEHizZu6BKl3v+xUuI=
```

SHA1(challenge + GUID)

Handshake

Establishes mutual agreement to speak WebSockets

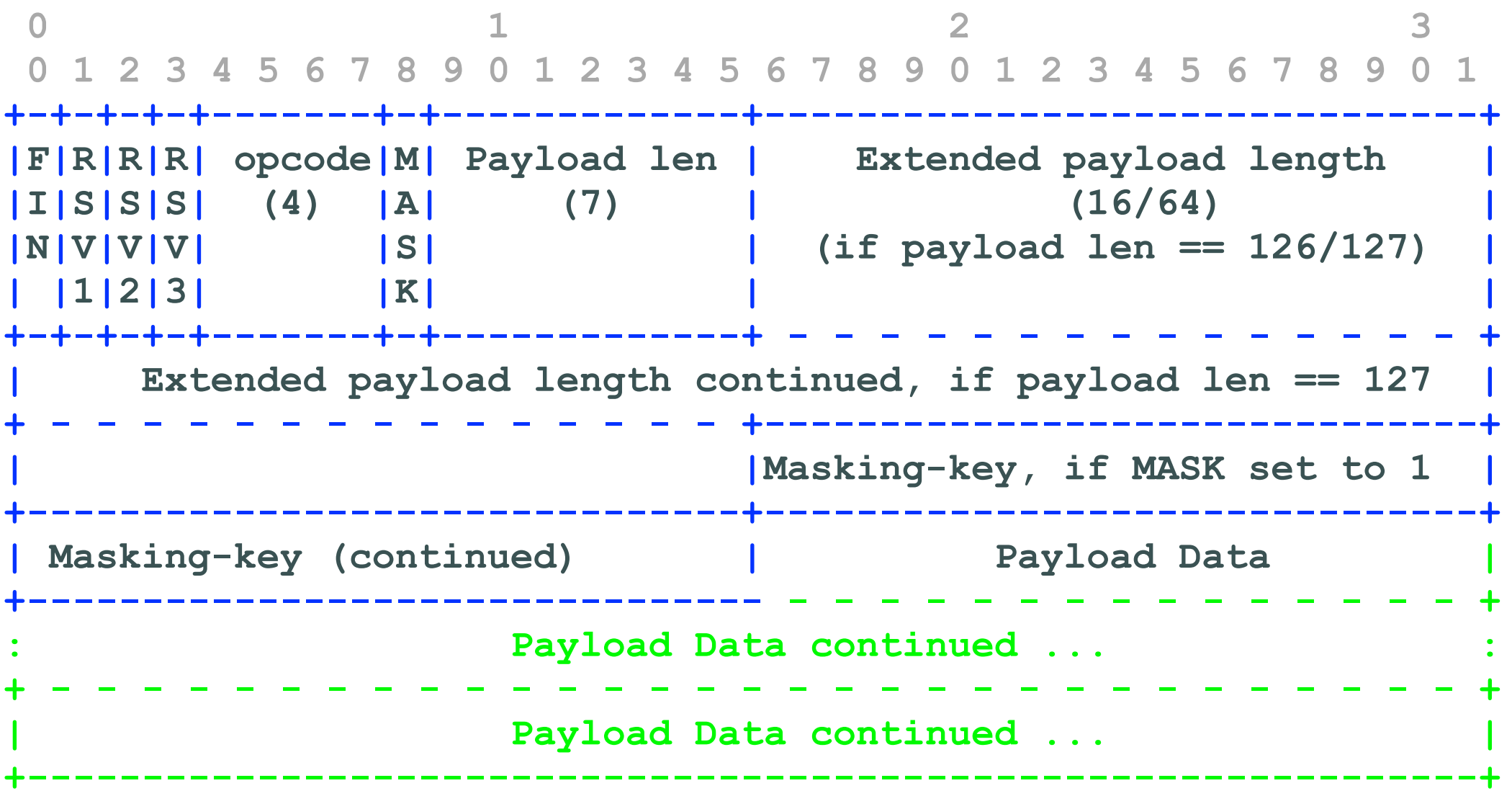
- Not intended to prove trust or identity

Prevents cross-protocol attacks

One handshake per Origin

- With (mostly) unlimited concurrent connections

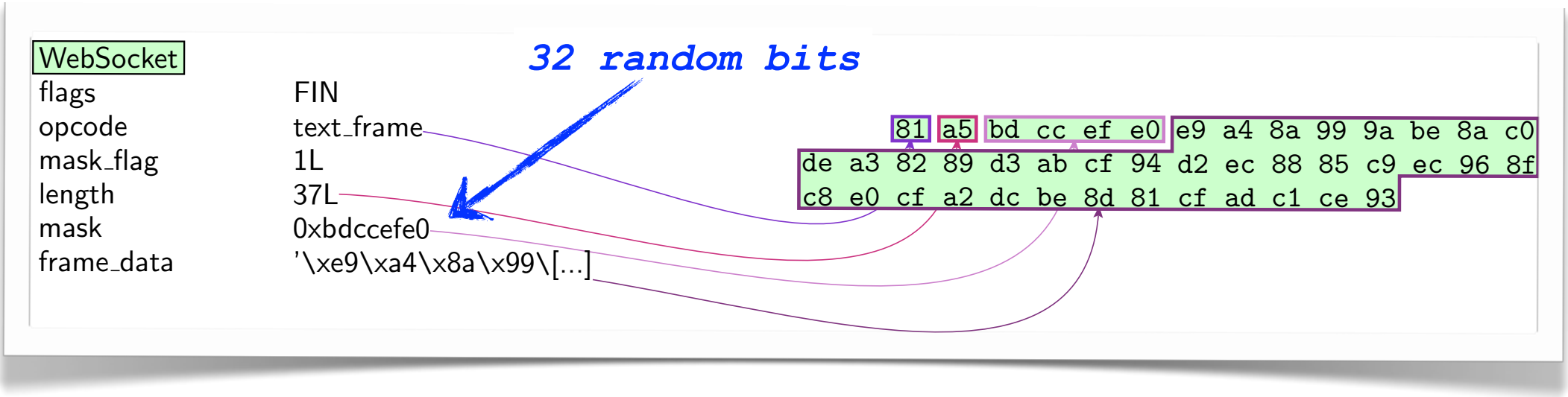
Data Frame Format



Variable Length Payloads

Decimal	Length (7 bits)	Extended Length (16- or 64-bit)
1	1 0 0 0 0 0 0	n/a
128	0 1 1 1 1 1 1	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
65535	0 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
65536	1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 . . .
2^64 - 1	1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 . . . 1 1 1 1 1 1 1 1
19	1 1 0 0 1 0 0	n/a
19	0 1 1 1 1 1 1	1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
19	1 1 1 1 1 1 1	1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . .

XOR Masks Data from Browser ➔ Server

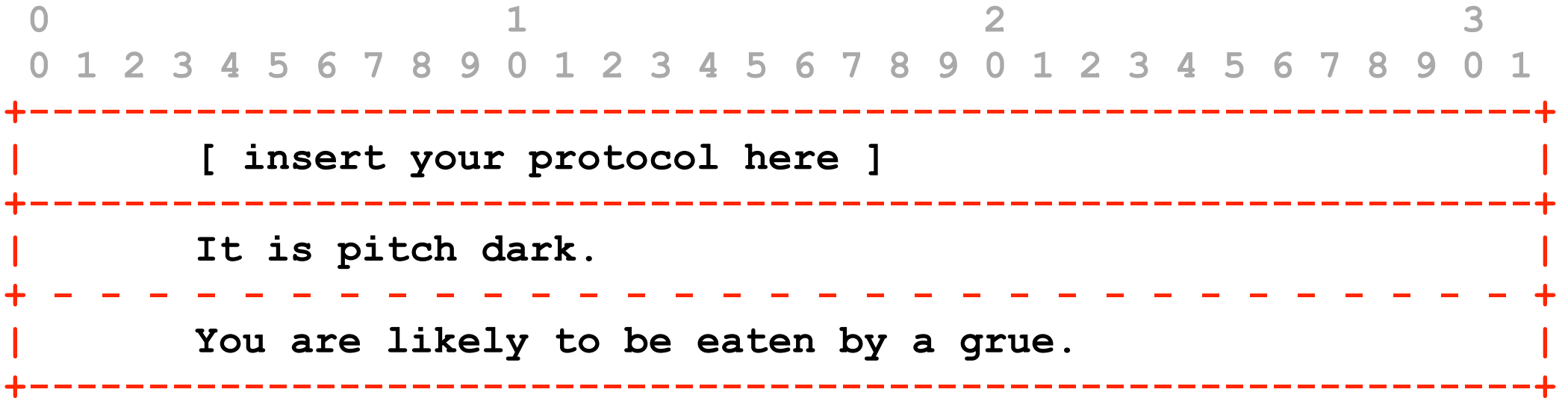



```
bd cc ef e0 bd cc ef e0 bd ...
e9 a4 8a 99 9a be 8a c0 de ...
T  h  e  y  `  r  e     c
```

Data Frame Security

Handshake headers provide security context

- Don’t lose this



Sending & Receiving Data

[Demo]

Basic WebSockets Security

Wherever WebSockets May Roam

Transport messages for chatty apps

Load HTML, JavaScript, CSS resources

Transport media (images, audio, video)

Tunnel a non-HTTP protocol

Browser Hacking

Mixed Content

If you can sniff **`http`** you can sniff **`ws`**.

If you can intercept or inject **`http`**, you can take over **`ws`**.

WebSocket API forbids using **`ws`** from resource obtained with **`https`**.

Denial of Service vs. ...

Client

One handshake at a time per Origin

Unlimited* WS connections to Origin vs. single-digit limits for HTTP connections

Server

Overwhelmed by poor connection/thread management

Attacks like Slowloris drain resources with persistent connections -- exactly what WebSockets rely on.

HTTP Headers

Handshake request includes `Origin`

- Should match the origin you expect, verify this on the server.
- Incorrect value may indicate forged requests from a victim's browser (e.g. CSRF).
- Spoofable -- not a reliable primary access control measure.

HTTP Headers

Content Security Policy

Restrict to JavaScript resources loaded from approved sources with `script-src`.

Restrict WebSocket (and XHR) connections to approved servers with `connect-src`.

X-WebKit-CSP:

```
script-src ‘self’
script-src https://static.cdn

connect-src ‘self’
```

(use ‘self’ for XHR)

```
connect-src wss://web.site

report-uri https://log.site
```

Hostile Clients

Overloaded JavaScript

```
var fuzz_replace = '';
var fuzz_append = '';

WebSocket.prototype._send = WebSocket.prototype.send;
WebSocket.prototype.send = function(data) {
    this._send(data);
    this.addEventListener('message', function(msg) {}, false);
    this.send = function(data) {
        if(fuzz_replace != '') {
            data = fuzz_replace;
        }
        else if(fuzz_append != '') {
            data = data + fuzz_append;
        }
        this._send(data);
    };
}
```

Browser Hacking

[Demo]

WebSocket Data Frame Security

Areas of Concern

Client

Executing in a hostile environment

Server

Implement the handshake, not a new web server

Protocol

Kerckhoff's Principle & Shannon's Maxim

WebSocket Server

Fingerprinting

Stability

Protecting Resources

Protocol Hardening Against...

Spoofing

Replay

Repudiation

Length manipulation

Defensive Handling of Payloads

Verify payload lengths

- Avoid buffer over-/under-runs

Avoid resource exhaustion

- Client declares 2GB length, only contains a few bytes

JavaScript Programming

"use strict"

Typed Arrays

- Strings are not binary containers

Server-side considerations

- !?

Examples

Epilogue

Wish You Were Here

- Security devices unaware of the protocol
  - WAF won't filter
  - IDS won't detect
- Data masking complicates protocol inspection
  - (and mix in compression, too!)

Points to Remember

Connection limits, data size limits

Security of tunneled protocol cannot rely on confidentiality within browser

Maintain session context

Client & Server

Restrict the browser's resources with Content Security Policy.

Connect with `wss:` scheme.

Security of data can't rely on secrecy of JavaScript.

Sec-WebSocket-Protocol: 13

The client is not trustworthy.

Verify the Origin header matches the value you expect.

Maintain coupling between the HTTP and WebSocket security contexts.

Apply

WebSockets solve connection problems, not security problems.

Remember basic security principles, especially for data frames' content.

"The new port 80" -- security devices have nonexistent(!?) awareness of the protocol.

Thank You!

Questions?

Mike @CodexWebSecurum

http://deadliestwebattacks.com

References

http://tools.ietf.org/html/rfc6455

http://dev.w3.org/html5/websockets/

http://dev.w3.org/html5/eventsource/

Tools

Browsers' Developer Tools

JavaScript

Zed Attack Proxy

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project