Hacking with WebSockets

Mike Shema

Sergey Shekyan

Vaagn Toukharian

December 2012

A Trip into HTML5

- WebSockets background
- Their appeal to developers
- Their appeal to attackers
- What makes them better

HTML4's Crude Solutions

Forcing persistence on a non-persistent protocol

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

...while dealing with the browser's per-domain connection limit

...and trying to figure out an efficient polling frequency

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

Almost WebSockets

- Forcing HTML5 on a non-HTML5 browser
  - **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 (http://www.w3.org/TR/eventsource/)
  - Properly-implemented long-polling
  - Content only flows from server → client

One Socket, Two Directions

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.

- RFC 6455

Speak to Me

- **Protocol (RFC 6455)**
  - Low overhead
  - Simple format
  - Content agnostic
  - HTTP compatible*
- **JavaScript API**
  - `.onmessage()`
  - `.send()`
  - Data as `String`, `Blob`, `ArrayBuffer`

Handshake Challenge

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==

base64(16 random bytes)

Handshake Response

HTTP/1.1 101 Switching Protocols

Upgrade: WebSocket

Connection: Upgrade

Proxy might remove this!

Sec-WebSocket-Accept: c4RVZSknSoEHizZu6BKl3v+xUul=

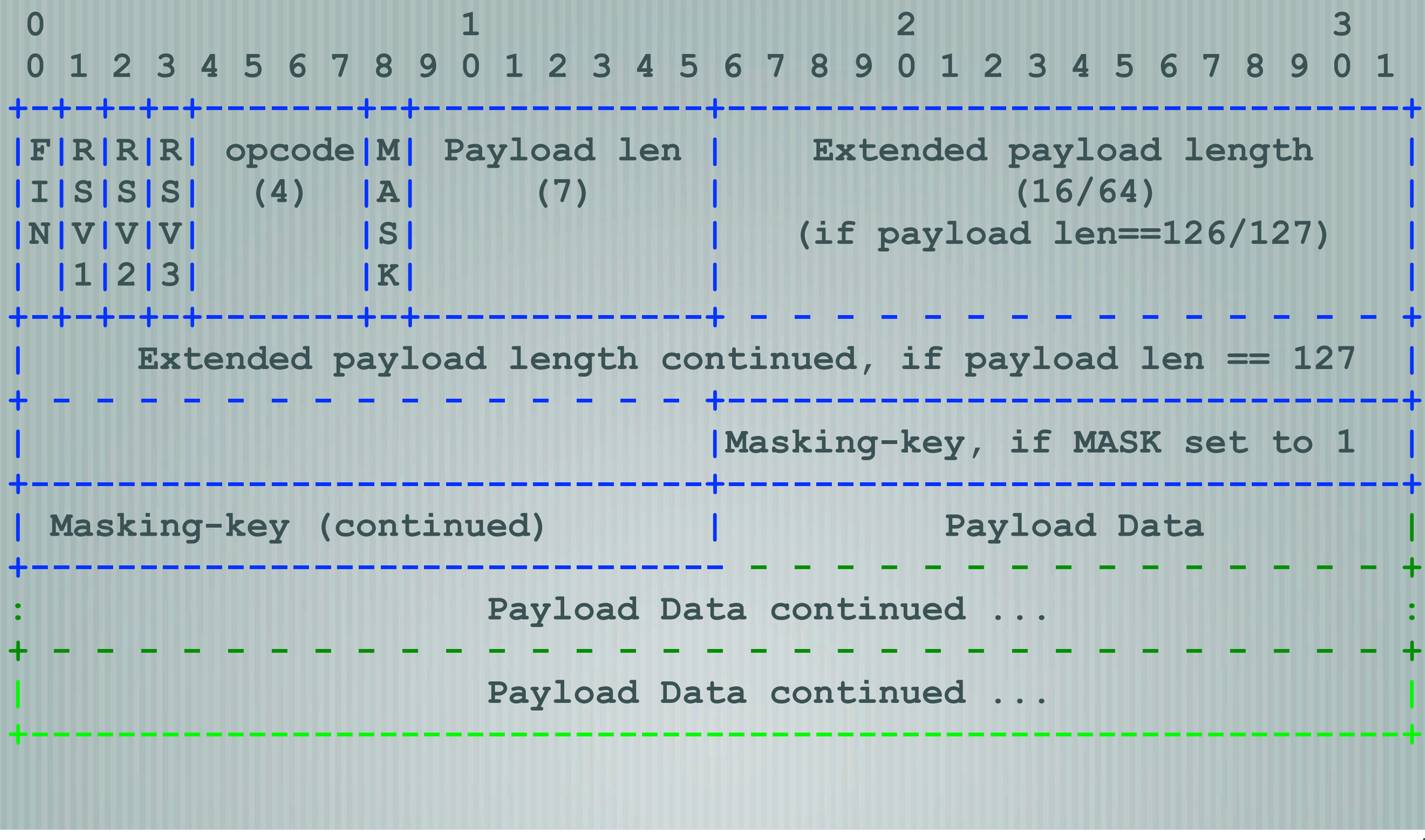
base64(SHA1(challenge + GUID)

[then the data frames begin]

HTTP Handshake

- Proves mutual agreement to speak WebSockets
  - Not intended to prove either trust or identity
- User Agent should not establish plaintext WebSocket (`ws:`) from "secure" resource (`https:`)
- Includes the Origin header
- Must complete before another connection may be established to the same origin

Data Frame Details

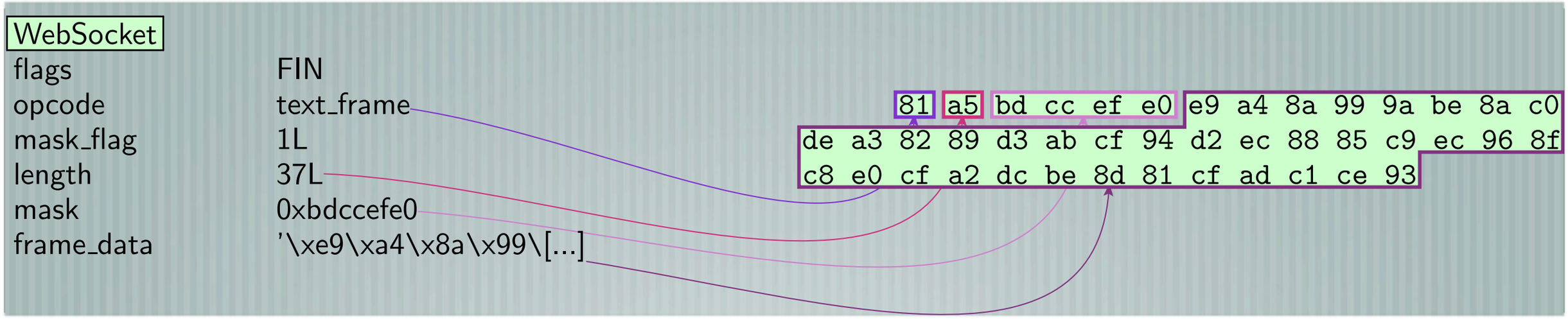


Variable Lengths

Decimal	Length (7 bits)	Variable 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 . . .
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 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 . . .

Masking Data

- 32-bit pseudo-random value, XOR byte by byte
- Prevent the browser from being leveraged for cross-protocol attacks, cache poisoning



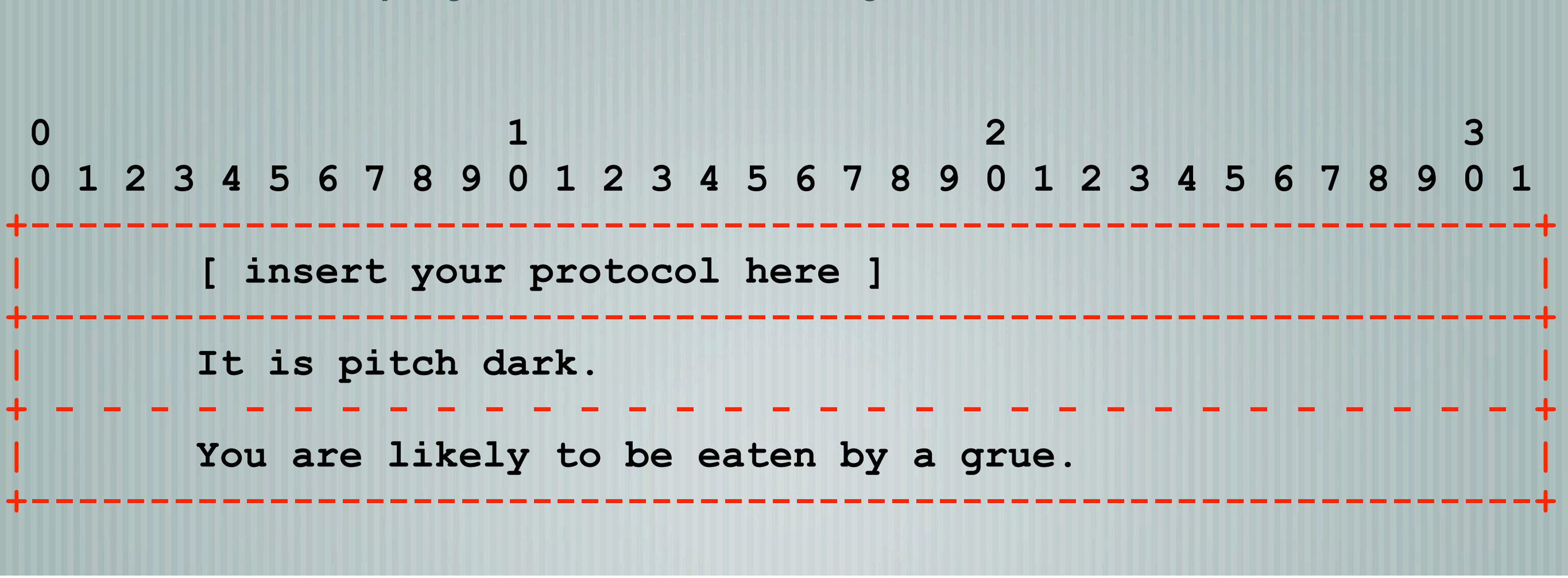
bd	cc	ef	e0	bd	cc	ef	e0	bd	...
e9	a4	8a	99	9a	be	8a	c0	de	...
	T	h	e	y	'	r	e		c

Design Choices

- Items transparent to JavaScript API
  - User Agents mask data to the server
  - "Ping" & "pong" frames for connection keep-alive
- User Agent should minimize details for certain kinds of connection failures to prevent "better" host/port scanning

Data Frame Security

- The handshake provides the headers to create a security context
  - Cookies, CORS, Origin, etc.
  - Decoupling this information is dangerous



What makes us worry

(Don't) Blame the Messenger

- WebSockets still fall victim to "old" threats
- WebSockets still have interesting things to discuss

Mixed content handling

- If you can sniff `http:` you can sniff `ws:`
- If you can intercept or inject, you can overtake `ws:` / `wss:`
- It should be impossible to mix `ws:` with `https:` by RFC
  - WebKit doesn't enforce this

Denial of Service - Client

- WebSockets connection limit is different than HTTP connection limit
- Malicious content can exhaust browser by grabbing max. allowed number of WebSocket connections

"..Yes, WebSocket is the first way to open an unlimited number of connections to a single server, so it indeed likely needs additional protection to prevent DOS attacks.
But we don't really have a way to implement this correctly..."

https://bugs.webkit.org/show_bug.cgi?id=32246

Denial of Service - Server

- Malicious content can create large number of WebSocket connections to victim WebSocket server
- Attacks like SlowLoris strive to maintain persistent connections thus draining server resources. WebSockets are naturally like that

Are Browsers OK?

- Still **no mixed content handling policy** implemented by WebKit-based
- Firefox still doesn't let WebWorkers create WebSockets
- Message sizes handled differently

Fuzzing WS

- Fuzzing WS data
  - Capturing real life data with JS
  - Fuzzing within a Browser
- Fuzzing WS frameworks
  - Fuzzing WS handshake
  - Fuzzing WS Headers

Fuzzing WS data with JS

```
var f_replace ='';
var f_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(f_replace !='') {
         data = f_replace;
      } else if(f_append !='') {
         data = data + f_append;
      }

      this._send(data);
   };
}
```

Fuzzing Frameworks

- Getting Crashes (not yet)
- Fingerprinting frameworks

Fingerprinting

WebSocket++

Successful handshake:
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Sec-WebSocket-Accept: Pzio3NY64M/GFfA/kK4WJpj2xY4=
Server: WebSocket++/0.2.0dev
Upgrade: websocket

Failed handshake:
HTTP/1.1 404
Server: WebSocket++/0.2.0dev

Fingerprinting

AutobahnPython

Successful handshake:
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Sec-WebSocket-Accept:fKrZviGIoYH4PrDbQ98Nvsbk2cU=
Server: AutobahnPython/0.5.9
Upgrade: WebSocket

Failed handshake:
HTTP/1.1 400 WebSocket version 12 not supported
(supported versions: 13,8,0)
Sec-WebSocket-Version: 13,8,0

Fingerprinting

Node.JS

Successful handshake:
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Sec-WebSocket-Accept:zSKX8MU5Omx1JXHacSpdN5a4ur4=
Upgrade: websocket

Failed handshake:
HTTP/1.1 426 Upgrade Required
Connection: close
Sec-WebSocket-Version: 13
X-WebSocket-Reject-Reason: Unsupported websocket client version: 12Only versions 8 and 13 are supported.

Looking for WS Security

- How to inspect WS traffic
- How to manipulate WS traffic?
- Are there browser plugins to help?
- Are there proxies that support WebSockets?

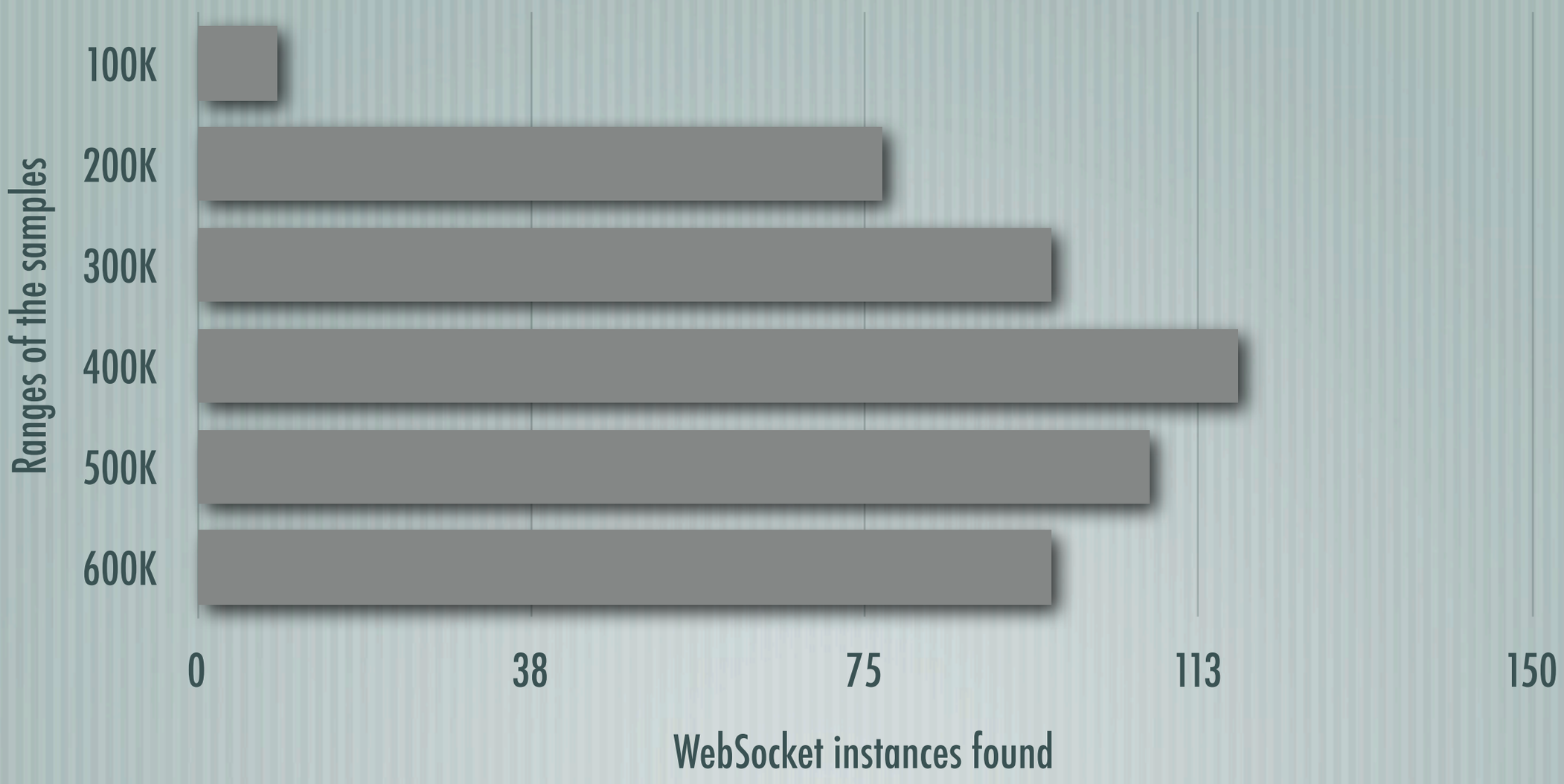
Tools

- WireShark
- Proxies(ZAProxy, Fiddler)
- Chrome Developer Tools
- overloaded WebSocket constructor and methods

Wish You Were Here

- Unawareness of WebSocket protocol by security devices (firewalls, IDS, IPS) makes them ineffective against malicious traffic
  - Masking inhibits identifying patterns in traffic
  - Missing auxiliary data type information makes it even harder
- Covert channels, command & control
  - Resurrect Loki (Phrack 49)
  - Sources of entropy: reserved flags, length representations, mask

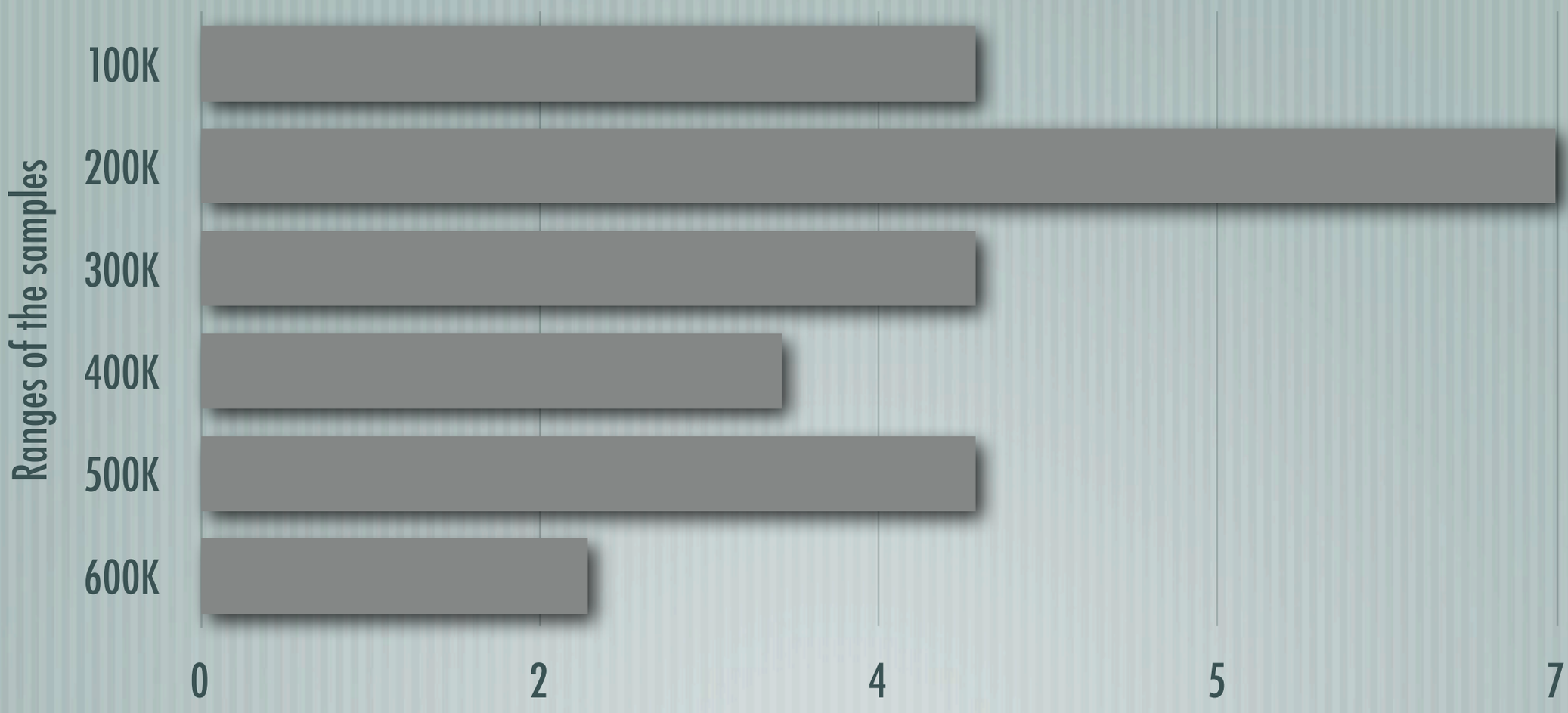
Not really...


Distribution of Alexa Top 600K websites that use WebSockets
Ranges of the samples
100K
200K
300K
400K
500K
600K
0
38
75
113
150
WebSocket instances found

Details?

- **0.15%** of websites use WebSockets on landing page.
- **Less than 4%** of captured WebSockets are using plain `ws` :
  - 95% of total WebSockets connect to a single vendor's customer support chat system
  - among remaining 5%, **less than 1% are using encryption**

True picture is...



What makes them better

Recommendations

- What it's good for
  - Time critical data delivery
  - Apps that require true bidirectional flow
  - Interactivity
  - Higher throughput
- Remember, it doesn't fix existing vulnerabilities

Deploying a Server

- Capacity planning & measurement to prevent self-inflicted DoS
- Verify the Origin header
- As always, assume the client is hostile – don't trust it
- Be careful when implementing the HTTP handshake
  - Create a single-purpose HTTP handler, not a pseudo-web server

Security of the tunneled protocol

- Beware of decoupling WebSocket session context from the HTTP session context
- Watch for protocol mistakes
  - using session cookies as chat IDs (visible to the recipient)
  - replay
  - spoofing
  - fragmentation, overlapping fragments
  - server-side buffer overflows, underflows

Remember Security Basics

- `wss` : means secure transport, not secure app
- Authn/Authz
- Session identifiers
- Server-side input validation
- Resource exhaustion
- Failure states

Summary

- WebSockets solve connection problems, not security problems.
- Basic security principles still apply, especially for data frames' content.
- "The new port 80" – security devices have poor (nonexistent!?) awareness of the protocol.

Q&A

Thank You!

Mike @CodexWebSecurum
Sergey @sshekyan
Vaagn @tukharian

References

- http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2008-June/015108.html
- http://www.ietf.org/mail-archive/web/tls/current/msg05593.html
- http://webtide.intalio.com/2011/09/cometd-2-4-0-websocket-benchmarks/

Interest in WebSockets?