They see me polling They hatin

WHO

WHAT

nodejitsu

@3rd-Eden

@3rdEden

WHERE

OMG, dude, polling is so 1995! Why no

WebSockets?

WebSockets?? Oh you mean

WebSuckets!

Browser supporting latest protocol

20+	12+	12.1+	6+	10+
RFC	RFC	RFC	RFC	RFC

Chrome for Android 18+	RFC
Firefox for Android 15+	RFC
Opera Mobile 12+	RFC

Browser supporting a protocol

4+ 4+ 11+ 4.2+ 10+

Chrome for Android 18+

Firefox for Android 15+

Opera Mobile 12+

Caution, feelings might get hurt

Trust me, I’m an engineer

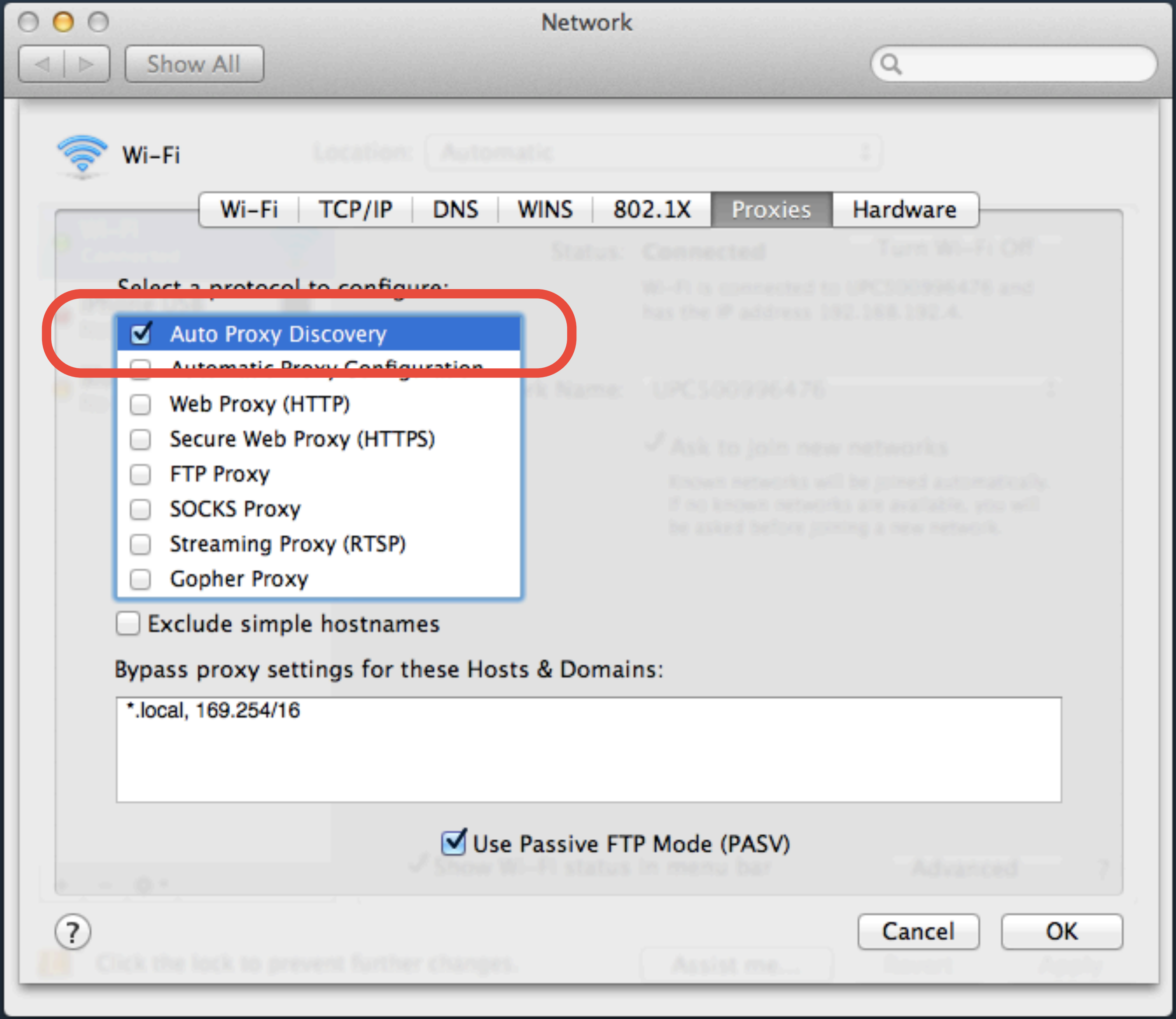
Using or detecting HTTP proxies crashes Safari < 5.1.4 and iOS Mobile WebKit

This causes full browser crashes or tab crashes. HTTP proxies cannot be detected easily.

```
if (
  // Target safari browsers
     $.browser.safari

  // Not chrome
  && !$.browser.chrome

  // And correct WebKit version
  && parseFloat($.browser.version, 0) < 534.54
) {
  // Don't use websockets
  return;
}
```



Network

Wi-Fi

Wi-Fi | TCP/IP | DNS | WINS | 802.1X | Proxies | Hardware

Select a protocol to configure:

- [✓] Auto Proxy Discovery
- [ ] Automatic Proxy Configuration
- [ ] Web Proxy (HTTP)
- [ ] Secure Web Proxy (HTTPS)
- [ ] FTP Proxy
- [ ] SOCKS Proxy
- [ ] Streaming Proxy (RTSP)
- [ ] Gopher Proxy

[ ] Exclude simple hostnames

Bypass proxy settings for these Hosts & Domains:

*.local, 169.254/16

[✓] Use Passive FTP Mode (PASV)

Writing to a closed WebSocket connection can cause a crash

This happens on Mobile Safari when returning to the page after backgrounding Safari or coming back from a different tab.

```
var ws = new WebSocket("wss://localhost:8080/");

ws.onmessage = function message(event) {
  // Wrap sends in a setTimeout out to allow the
  // readyState to be correctly set to closed
  setTimeout(function () {
    ws.send("Sup AmsterdamJS");
  });
};
```

```javascript
var ws = new WebSocket("wss://localhost:8080/");

ws.onmessage = function message(event) {
  // Wrap sends in a setTimeout out to allow the
  // readyState to be correctly set to closed. But
  // Only have this delay on mobile devices
  if (mobile) return setTimeout(function () {
    ws.send("Sup AmsterdamJS");
  });

  ws.send("Sup AmsterdamJS");
};
```

3G, 4G, LTE mobile connections.. dafuq

It’s not just one mobile provider, it’s a lot of them. They are either running reversed proxies or simply block WebSockets. Shame on you AT&T

```
var ua = navigator.userAgent.toLowerCase();

// Detect all possible mobile phones to filter out
// WebSockets
if (
     ~ua.indexOf('mobile')
  || ~ua.indexOf('android')
  || ~ua.indexOf('ios')
  || .. and more ..
) {
  // Don't use WebSockets
}
```

Pressing ESC in Firefox will close all active network connections.

Not only during page load, but also after page load. The issue remains the same. This is fixed in Firefox Nightly (20) all other version are affected.

```
$('body').keydown(function (e) {
  // make sure that you capture the `esc` key and
  // prevent it's default action from happening
  if (e.which === 27) e.preventDefault();
});
```

Be careful when sending UTF-8/16 to Node.js

This can cause WebSocket connection drops as V8 uses UCS encoding internally instead of modern UTF-16

```
var ws = new WebSocket("wss://localhost:8080/");

ws.onopen = function(event) {
  // encode and then unescape all messages that
  // contain utf 8 or user input.
  ws.send(unescape(encodeURIComponent(💩)));
};
```

↑

shitty emoji's

Firefox cannot connect using ws:// from a HTTPS secured server

It throws an “SecurityError: The operation is insecure.” error. Firefox 8+

Don’t use self signed certificates

Just don’t, some browsers give you no way of accepting them when using WebSockets. And you look like a cheap d*ck for not buying a proper cert

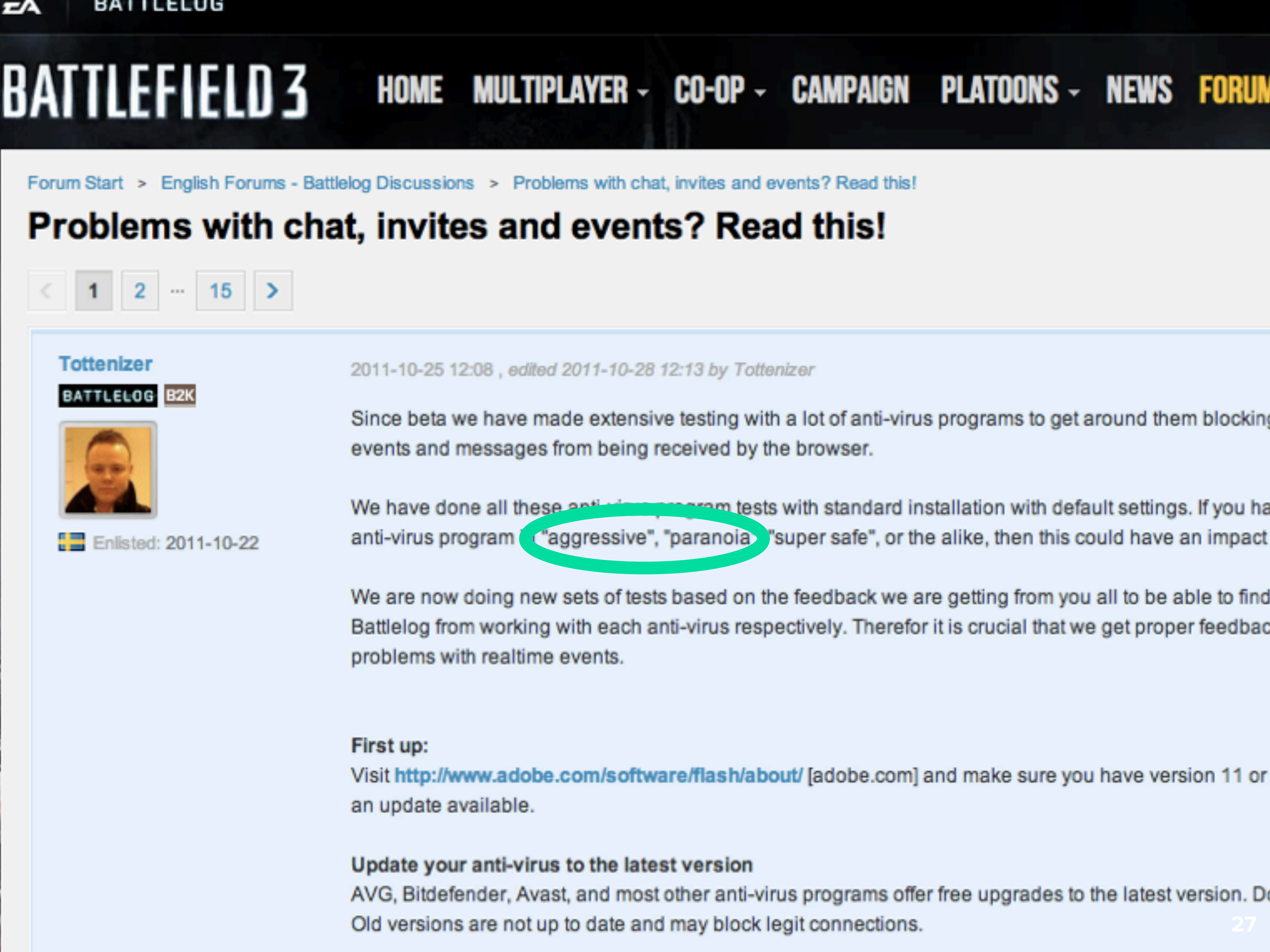
It can’t be worse, right?!

Debugging browser compatibility is nothing compared to debugging connection blocking

Connection blockage

Enterprise proxy usually block everything except ports: 80, 443, 843

Virus scanners on the other hand target port 80



Problems with chat, invites and events? Read this!

Tottenizer

2011-10-25 12:08 , *edited 2011-10-28 12:13 by Tottenizer*

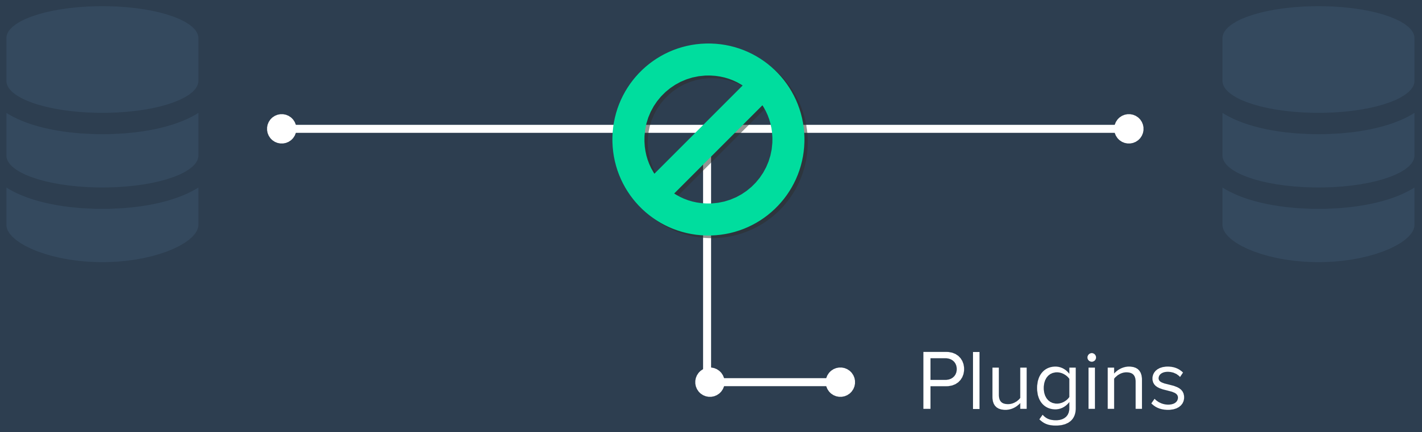
Since beta we have made extensive testing with a lot of anti-virus programs to get around them blocking
events and messages from being received by the browser.

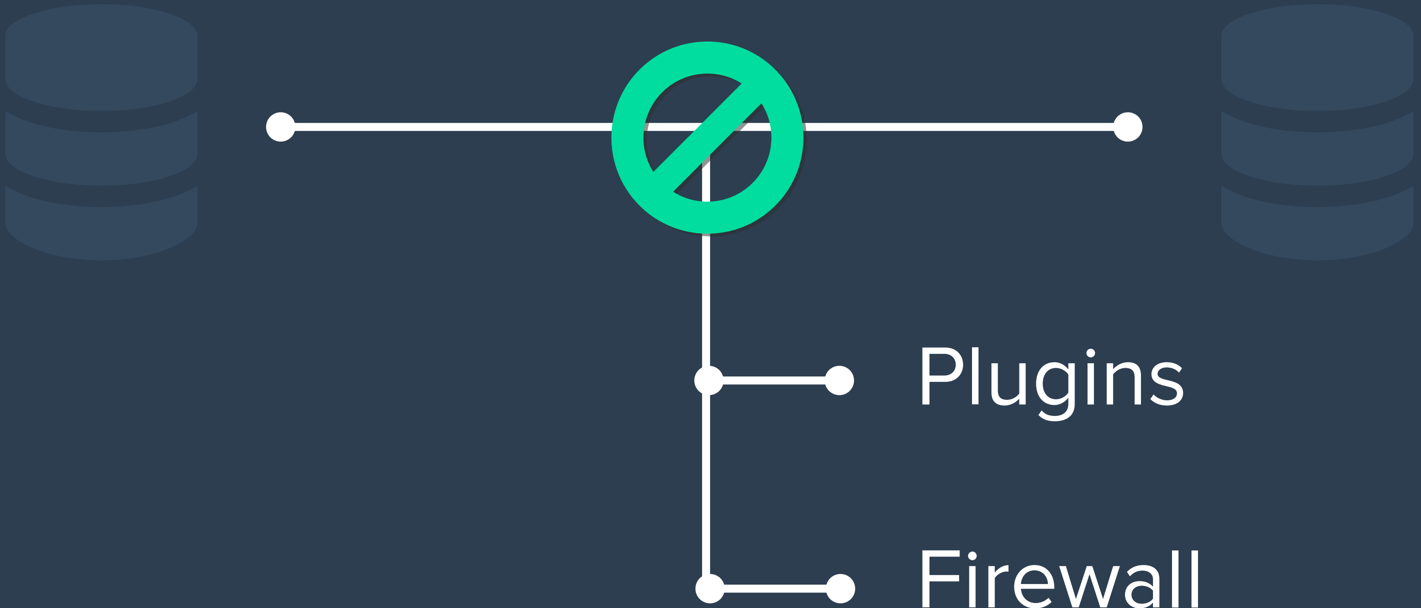
We have done all these anti-virus program tests with standard installation with default settings. If you ha
anti-virus program "aggressive", "paranoia" "super safe", or the alike, then this could have an impact

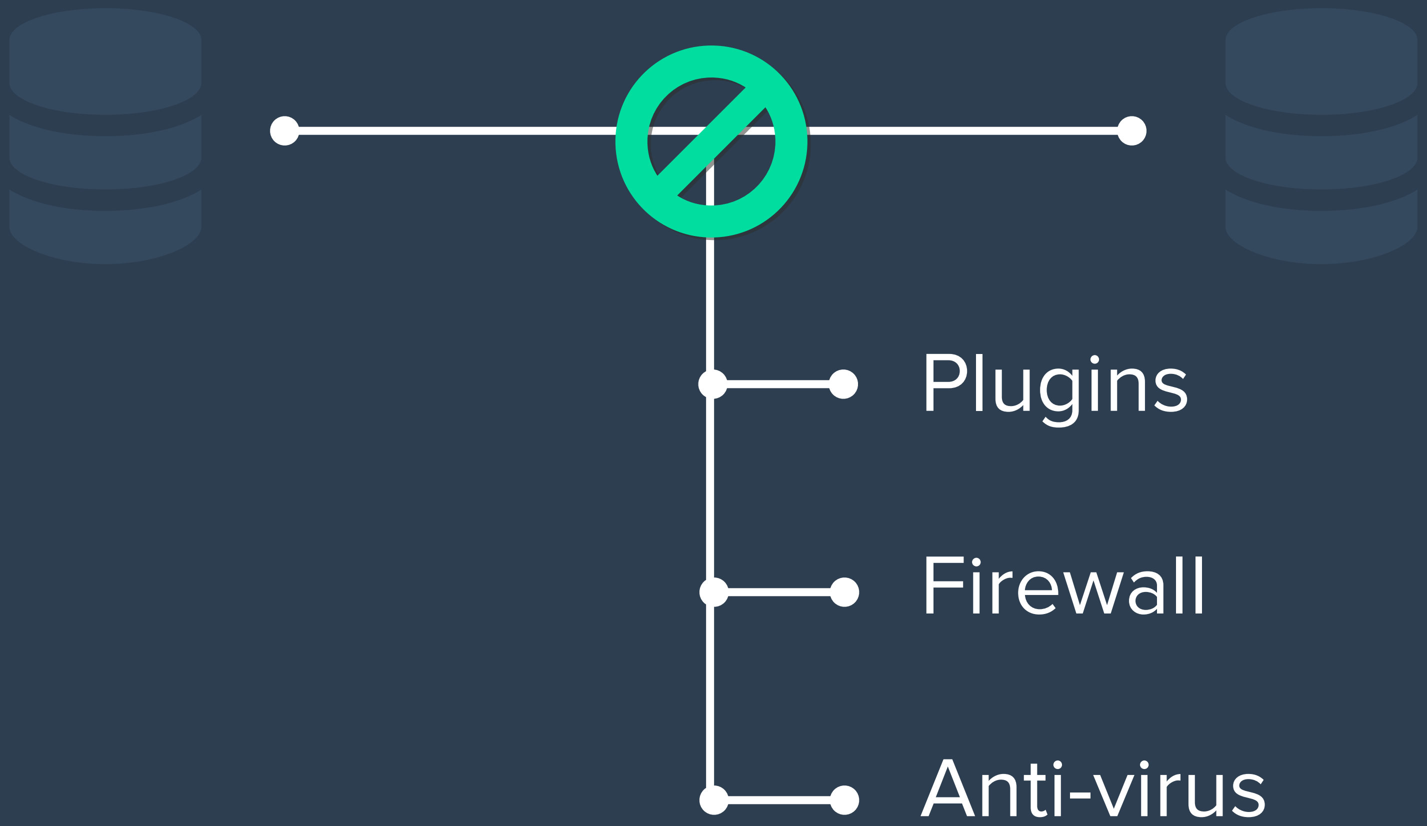
We are now doing new sets of tests based on the feedback we are getting from you all to be able to find
Battlelog from working with each anti-virus respectively. Therefor it is crucial that we get proper feedbac
problems with realtime events.

First up:
Visit http://www.adobe.com/software/flash/about/ [adobe.com] and make sure you have version 11 or
an update available.

Update your anti-virus to the latest version
AVG, Bitdefender, Avast, and most other anti-virus programs offer free upgrades to the latest version. D
Old versions are not up to date and may block legit connections.


Plugins


Plugins
Firewall


Plugins
Firewall
Anti-virus

AVG Anti-Virus

McAfee®

Blue Coat®

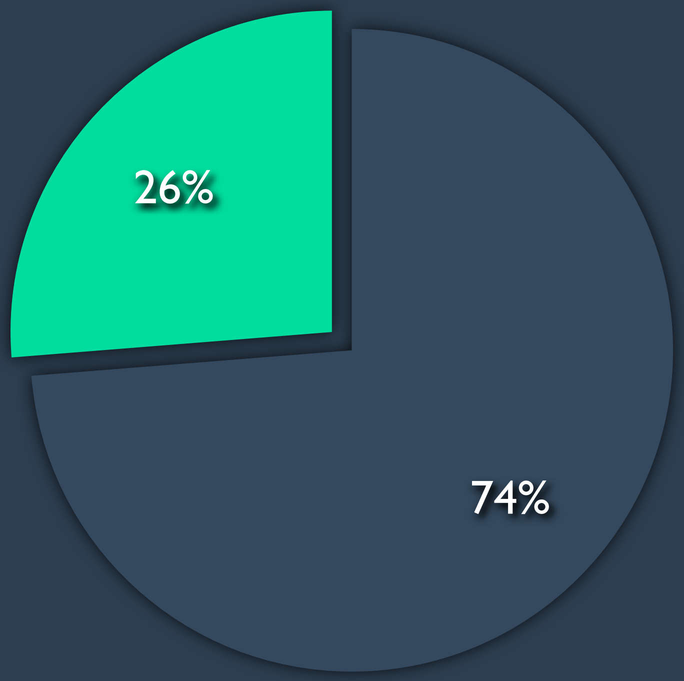
KASPERSKY lab

avast!® be free

These f*cks block JavaScript if it contains the word **ActiveX**

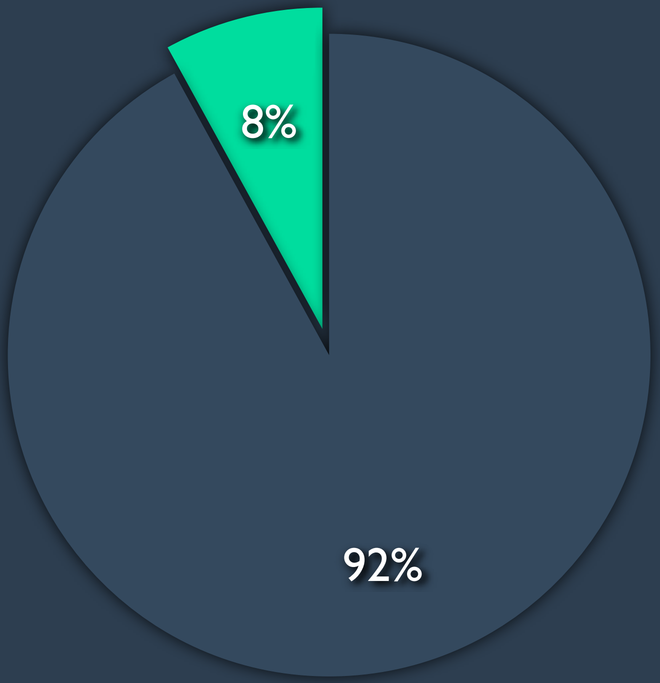
So..

What are we dealing with?

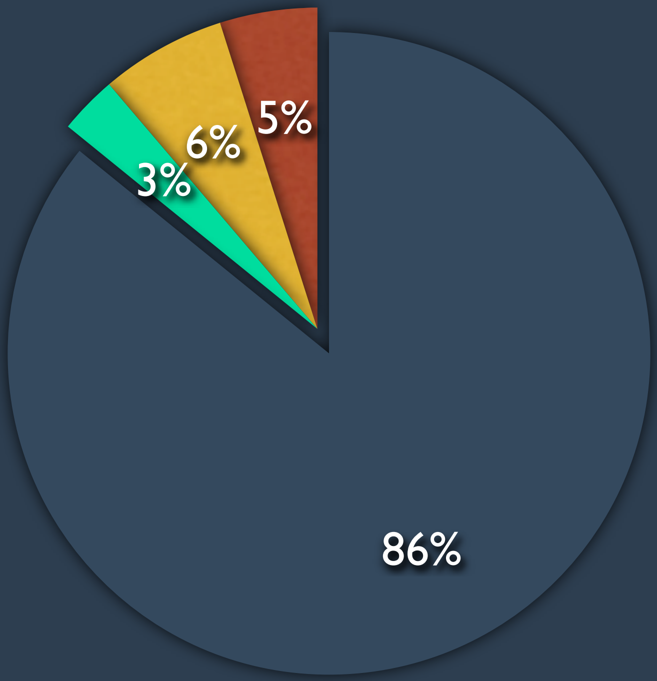


● Supported

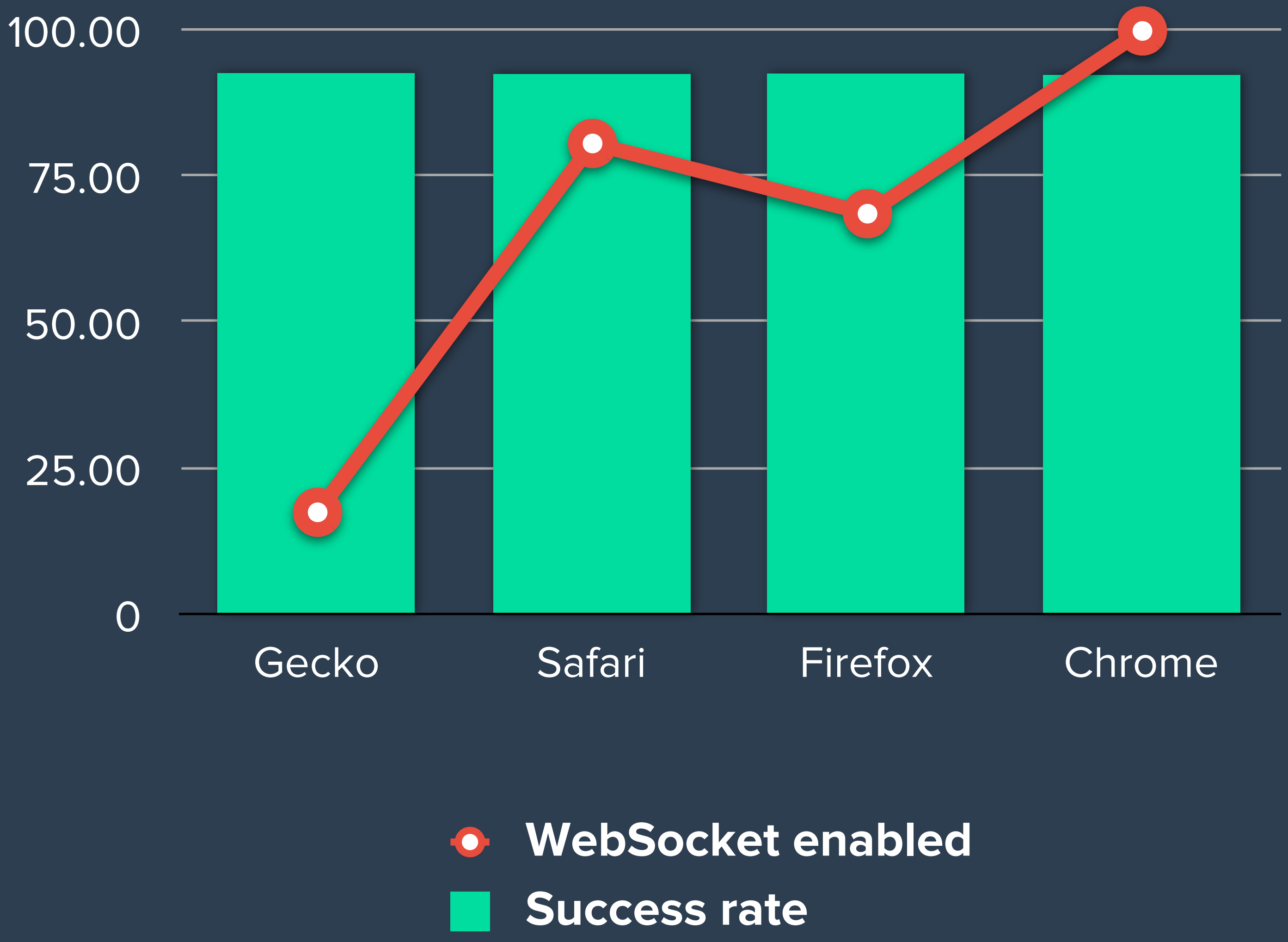
● Not supported

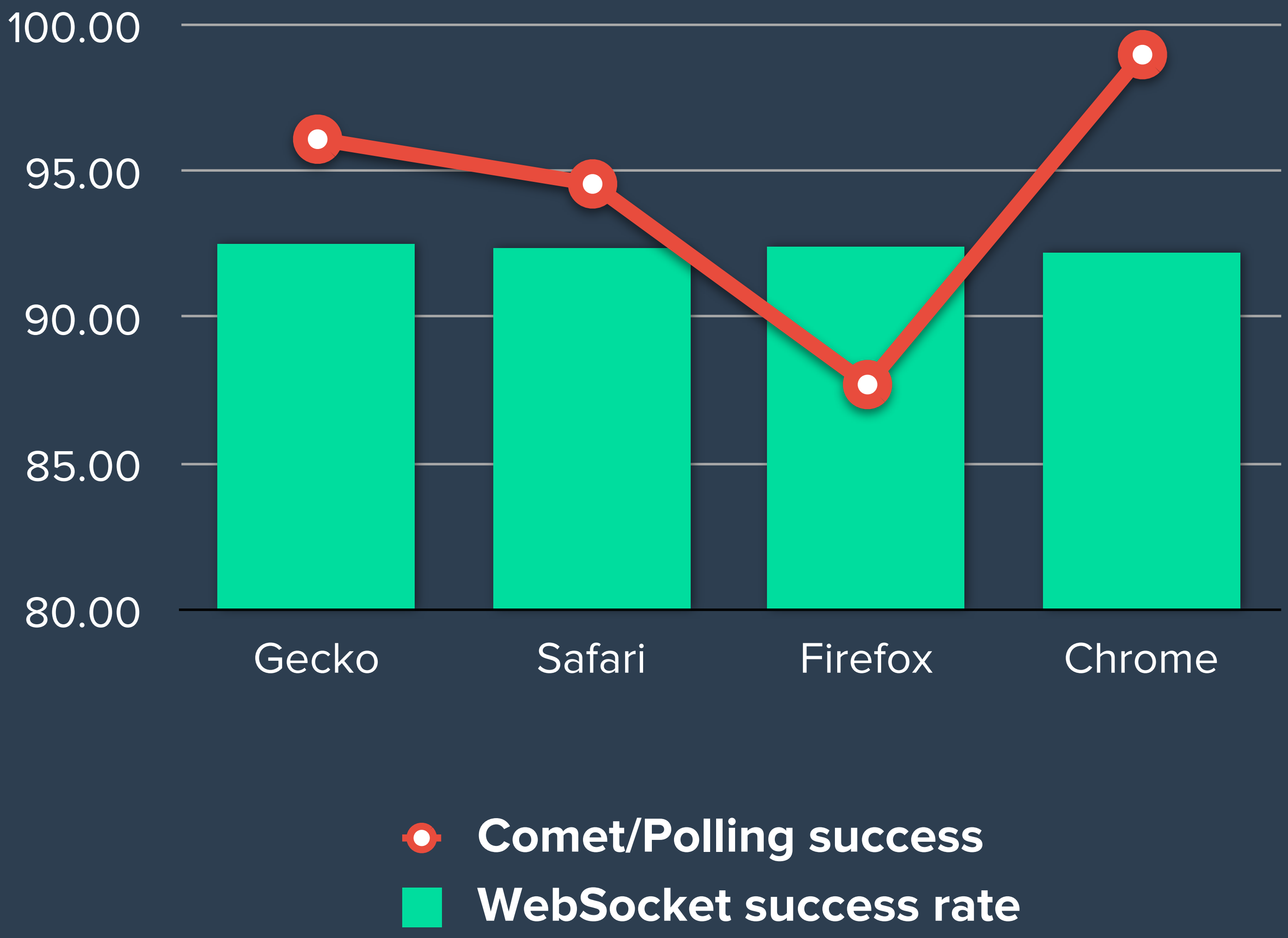


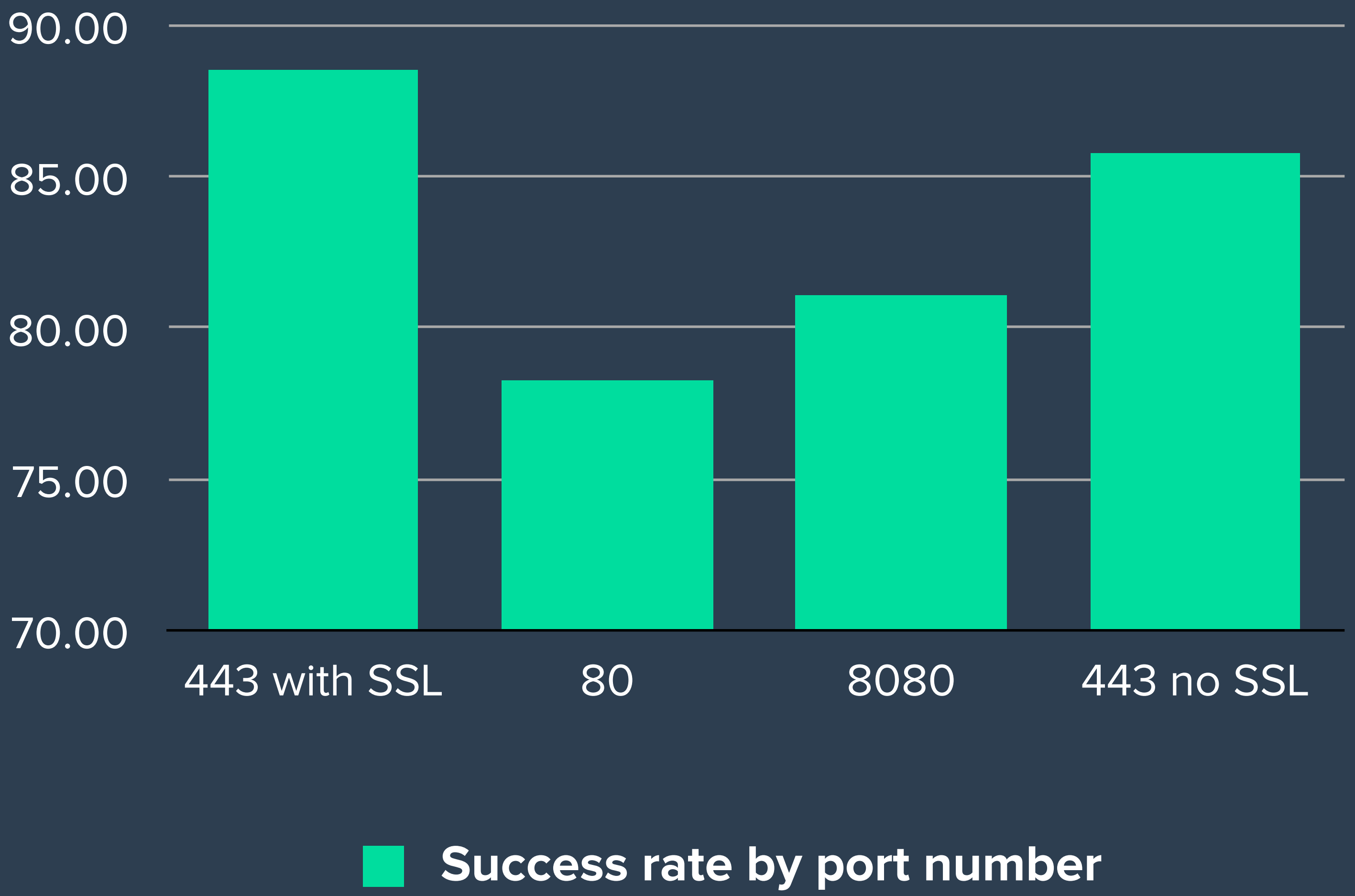
● Successful connection
● No connection



● No Proxy, Success
● With Proxy, Failed
● With Proxy, Success
● No Proxy, Failed


100.00
75.00
50.00
25.00
0
Gecko
Safari
Firefox
Chrome
WebSocket enabled
Success rate


100.00
95.00
90.00
85.00
80.00
Gecko
Safari
Firefox
Chrome
Comet/Polling success
WebSocket success rate


90.00
85.00
80.00
75.00
70.00
443 with SSL
80
8080
443 no SSL
🟩 Success rate by port number

And..

How can you deal with it?

Socket.IO

Socket.IO aims to make realtime apps possible in every browser and mobile device, blurring the differences between the different transport mechanisms. It's **care-free** real-time 100% in JavaScript.

488

Packages depend on Socket.IO

7,701

Stars on Github & **1, 878** on the client.

488

Forks on Github & **363** on the client.

4418

Users on Google Groups.

Big community

WebSocket

Supports old HIXIE drafts as well as the latest RFC specification.

FlashSocket

Fallback for browser that do not support WebSockets. A Flash file that emulates the WebSocket protocol so you can still bi-directional communication.

HTML File

Basically it’s a streaming iframe wrapped with some ActiveX magic. Sending data is done through XHR POST. Internet Explorer only, does not do cross domain.

XHR Polling

Long polling. Cross domain usage depends on the browser.

JSONP Polling

Injects small scripts in the page for fetching data and uses iframe’s and form posts to send the data to the server.

npm install socket.io --save

The *`--save`* tells npm to automatically add the installed version to your *`package.json`* file. Additionally you can also install the socket.io-client module if you want to connect to server from within node.js.

```
var io = require('socket.io').listen(8080);

io.sockets.on('connection', function (socket) {
  socket.on('another event', function (data) {
    // Client emitted a custom event
    socket.emit('custom event', data);
  });

  socket.on('disconnect', function () {
    // Socket disconnected
  });

  socket.send('hi there');

  // Automatic JSON encoding using a json flag
  socket.json.send({ foo: 'bar' });

  // Broadcast the message/event to every
  // connected user
  socket.broadcast.json.send({ foo: 'bar' });
});
```

```
var io = require('socket.io').listen(8080);
```

Creating: The listen method accepts either a HTTP server instance or it will create one for you with listens on the supplied port number

```
io.sockets.on('connection', function (socket) {
  socket.on('another event', function (data) {
    // Client emitted a custom event
    socket.emit('custom event', data);
  });

  socket.on('disconnect', function () {
    // Socket disconnected
  });

  socket.send('hi there');

  // Automatic JSON encoding using a json flag
  socket.json.send({ foo: 'bar' });

  // Broadcast the message/event to every
  // connected user
  socket.broadcast.json.send({ foo: 'bar' });
});
```

```
var io = require('socket.io').listen(8080);

io.sockets.on('connection', function (socket) {
  socket.on('another event', function (data) {
    // Client emitted a custom event
    socket.emit('custom event', data);
  });

  socket.on('disconnect', function () {
    // [illegible]
  });

  socket.send('hi there');

  // Automatic JSON encoding using a json flag
  socket.json.send({ foo: 'bar' });

  // Broadcast the message/event to every
  // connected user
  socket.broadcast.json.send({ foo: 'bar' });
});
```

Namespaces: the `io.sockets` points to the default namespace of your socket.io server. One server can have multiple namespaces or “endpoints”.

Flags: the *broadcast* and *json* are instructions to socket.io on how to send this message. But there are more:

- **json**: Automatically JSON encoding
- **broadcast**: Send message to every connected user except your self.
- **volatile**: Send message, we don't care if it get's lost in the transaction.
- **in(<room>)**: Send the message to everybody that is in this room.

```
socket.broadcast.json.send({ foo: 'bar' });
```

```
// Connect to a custom domain
var socket = io.connect('http://domain.com');

socket.on('connect', function () {
  // Socket connected
  socket.json.send({ foo: 'bar'});
});

socket.on('custom event', function (data) {
  // Server emitted a custom event
  socket.emit('another event', data);
});

socket.on('disconnect', function () {
  // socket disconnected
});

socket.send('hi there');
```

```
// Connect to a custom domain
var socket = io.connect('http://domain.com');
```

Crossdomain: When you don't supply it with a URL it will connect to page that loads the socket.io code and supply it with a custom domain to do cross domain connections.

Engine.IO

Engine.io is the implementation of transport-based cross-browser/cross-device bi-directional communication layer for Socket.IO. But it can also be used as standalone server.

```
var engine = require('engine.io')
  , server = engine.listen(80)

server.on('connection', function (socket) {
  socket.on('message', function () {
    // New message from the client
  });

  socket.on('close', function () {
    // Connection closed
  });

  socket.send('utf 8 string');
});
```

MIA: On the server side there are a couple of differences, it misses a lot of “features” that were build in. Like namespaces, rooms, automatic JSON encoding etc. But in return you get a really low level API

```
var engine = require('engine.io')
    server = engine.listen(80)

server.on('connection', function (socket) {
  socket.on('message', function () {
    // New message from the client
  });

  socket.on('close', function () {
    // Connection closed
  });

  socket.send('utf 8 string');
});
```

```
var socket = require('engine.io')('ws://localhost');
```

Component: The Engine.IO client is now component based. Component is a small JavaScript framework that brings node style dependencies and requires to the front-end.

MIA: Same as on the server side, it misses a lot of features like no events, json encoding, namespaces, authentication etc.

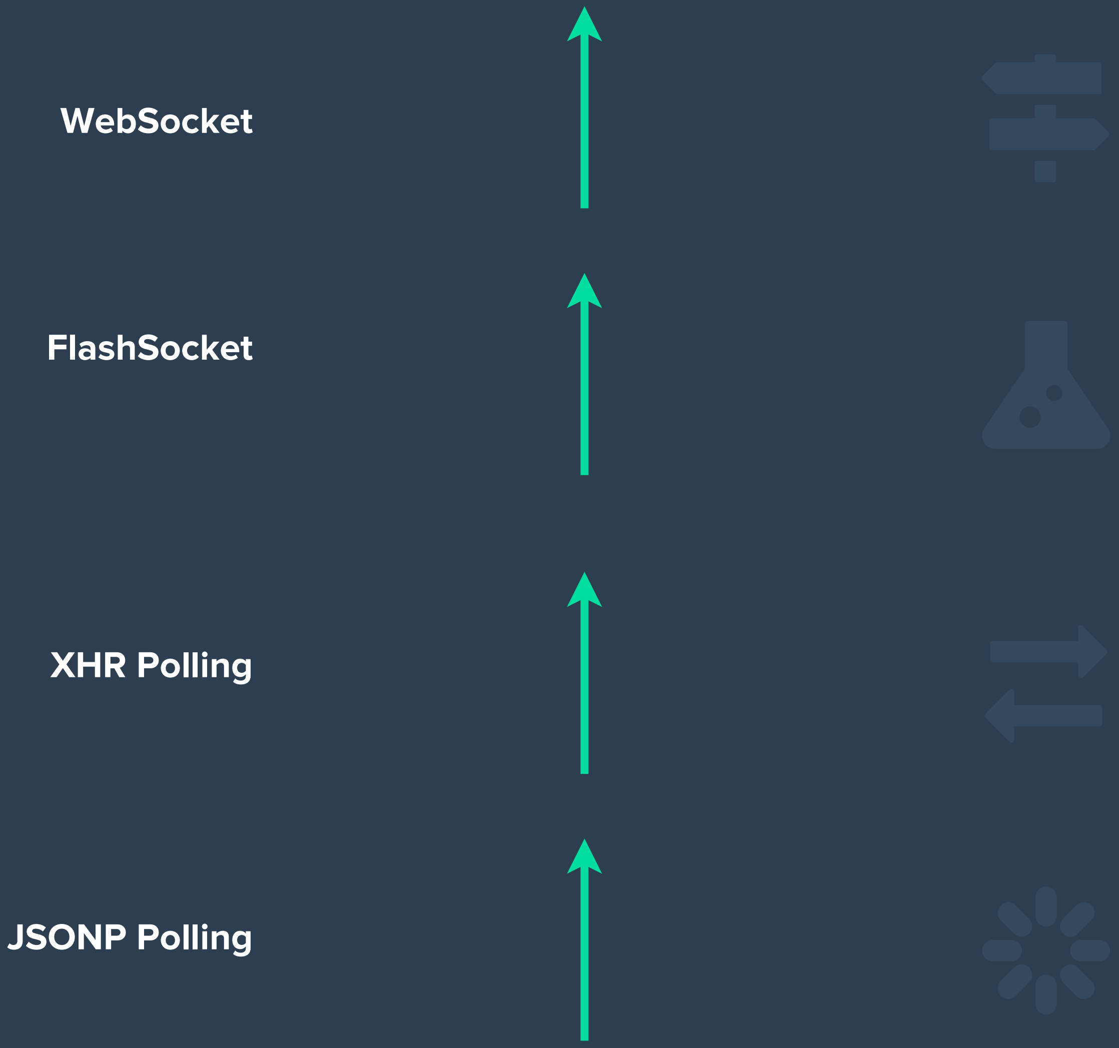
WebSocket

FlashSocket

HTML File

XHR Polling

JSONP Polling


WebSocket
FlashSocket
XHR Polling
JSONP Polling

Socket.IO 1.0

Socket.io 1.0 will be build on top of Engine.io and will supply the missing features that your used to in socket.io.

Key takeaways

Don’t use WebSockets on mobile

To much undetectable issues and polling works better for network switching and crappy networks.

Always use SSL

It makes you less vulnerable for connection blocking.

Upgrade from fallbacks transports

So your real-time connection works in every environment

QUESTIONS?

Talk nerdy to me