Why Web Security Is Fundamentally Broken

Jeremiah Grossman
Founder & Chief Technology Officer
Jeremiah Grossman

- Founder & CTO of WhiteHat Security
- International Presenter
- TED Alumni
- InfoWorld Top 25 CTO
- Co-founder of the Web Application Security Consortium
- Co-author: Cross-Site Scripting Attacks
- Former Yahoo! information security officer
- Brazilian Jiu-Jitsu Black Belt
WhiteHat Security

- Headquartered in Santa Clara, CA
- WhiteHat Sentinel – SaaS end-to-end website risk management platform (static and dynamic)
- Employees: 220+

Cool Vendor
Web Security

Rule #1: A website must be able to defend itself against a hostile client [browser].
Web Security

Rule #1: A website must be able to defend itself against a hostile client [browser].

Challenging, but possible to follow.
Web Security

Rule #2: A browser must be able to defend itself against a hostile website.
Web Security

Rule #2: A browser must be able to defend itself against a hostile website.

Impossible.
Today’s browsers make available to every website you visit:

Passive access to your operating system information, various system settings, browser type / version, installed add-ons & plug-ins, geographic location, websites currently logged-into, etc.
Today’s browsers make available to every website you visit:

Passive access to your operating system information, various system settings, browser type / version, installed add-ons & plug-ins, geographic location, websites currently logged-into, etc.

Give a website just 1 mouse-click — Then it gets access to:

Your name, where you live, where you’ve been, town you grew up in and went to school, martial status, photos, and in some cases, the browser’s auto-complete data and surfing history.
Today’s browsers make available to every website you visit:

Passive access to your operating system information, various system settings, browser type/version, installed add-ons & plug-ins, geographic location, websites currently logged-into, etc.

Give a website just 1 mouse-click — Then it gets access to:

Your name, where you live, where you’ve been, town you grew up in and went to school, martial status, photos, and in some cases, the browser’s auto-complete data and surfing history.

All browsers also allow a [malicious] website to:

Force your browser to send self-incriminating Web requests, hack your Intranet, auto-XSS / CSRF you on any website, etc.
The 2 Types of Browser Attacks

1) Attacks designed to escape the browser walls and infect the operating system with malware. (a.k.a. Drive-by-Downloads)

**Security:** Sandboxing, silent and automatic updates, increased software security, anti-phishing & anti-malware warnings, etc. [Enabled by default]

2) Attacks that remain within the browser walls and compromise cloud-based data. XSS, CSRF, Clickjacking, etc.

**Security:** SECURE Cookies, httpOnly, X-Frame-Options, Strict-Transport-Security, X-Content-Type-Options, Content Security Policy, EV-SSL, etc. [Opt-In by website, users can’t protect themselves]
Browser Interrogation

**Operating System and Browser Type via User-Agent Headers**
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.89 Safari/537.1

**Language setting, ActiveX support, and the Referer.**

```
GET / HTTP/1.1
Host: http://maliciouswebsite/
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.89 Safari/537.1
Accept-Language:en-US,en;q=0.8

<script>
if (navigator.language) {
    console.log(navigator.language);
}
</script>
```
Browser Interrogation (cont.)

**ActiveX**

```html
<script>
if(typeof(window.ActiveXObject)=="undefined"){
    console.log("ActiveX Unavailable");
} else {
    console.log("ActiveX Available");
}
</script>
```

**Referer**

```plaintext
GET / HTTP/1.1
Host: http://maliciouswebsite/
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5)
AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.89
Safari/537.1
Referer:http://searchengine/search?q=keywords
```
Virtualization Detection via Screen Dimensions

```javascript
<script>
var dimensions = {
  '320, 200': '',
  '320, 240': '',
  '640, 480': '',
  '800, 480': '',
  '768, 576': '',
  '854, 480': '',
  '1024, 600': '',
  '1152, 768': '',
  '800, 600': '',
  '1024, 768': '',
  '1280, 854': '',
  '1280, 960': '',
  '1280, 1024': '',
  '1280, 720': '',
  '1280, 768': '',
  '1366, 768': '',
  '1280, 800': '',
  '1440, 900': '',
  '1440, 960': '',
  '1400, 1050': '',
  '1600, 1200': '',
  '2048, 1536': '',
  '1680, 1050': 1,
  '1920, 1080': '',
  '2048, 1080': '',
  '1920, 1200': '',
  '2560, 1600': '',
  '2560, 2048': ''
};

var wh = screen.width + ', ' + screen.height;

if (dimensions[wh] != undefined) {
  console.log("Not virtualized");
} else {
  console.log("Operating in a virtualized environment");
}
</script>

Browser Interrogation (cont.)

**Identifying Installed Extensions and Add-Ons**

(CHROME)
```
<script src="chrome-extension://aknpkdffaafgjchaibgeefbgmgeghloj/manifest.json" onload="extensionDetected()">

(Firefox)
```
```
<script>
if (typeof uniquelyNamedObject !== 'undefined'){
    console.log("Add-On Present");
}
</script>
```
Common use-case:

```html
<img src="http://someotherwebsite/image.png">

<img src="http://someotherwebsite/image.png" onload="successful()" onerror="error()">
```

If the image file loaded correctly, the “successful” Javascript function executes. If some error occurred, obviously the “error” function executes.
Common use-case:

```html
<img src="http://someotherwebsite/image.png">

<img src="http://someotherwebsite/image.png" onload="successful()" onerror="error()">
```

If the image file loaded correctly, the “successful” Javascript function executes. If some error occurred, obviously the “error” function executes.

Login-Detection (via CSRF):

```html
<img src="http://someotherwebsite/loggedin.png" onload="loggedIn()" onerror="notLoggedIn()">
```

If the user is logged-in, the image file loads successfully, which executes the “loggedIn.” If they’re not logged-in, “notLoggedIn” is executed.
Authenticated Javascript/CSS

**Event Handler**
```
<script src="http://thirdparty/javascript.js"
onload="loggedin()" onerror="notloggedsin()"></script>
```

**Object Detection**
```
<script src="http://thirdparty/javascript.js"> </script>
<script>
if (typeof loggedInObject != 'undefined'){
    console.log ("Logged-In");
}
</script>
```

**CSS Object Detection**
```
<link rel="stylesheet" type="text/css" href="http://thirdparty/stylesheet.css" />
```
Authenticated IFRAMESs

iframe.contentWindow.length
<iframe id="login" src="http://thirdparty/profile/"></iframe>

<script>
if (iframe.contentWindow.length > 0) {
    console.log ("Logged-In");
}
</script>

XFO Detection
<iframe id="login" src="http://thirdparty/profile/"></iframe>
XFO Detection

<iframe id="login" src="http://thirdparty/profile/"/></iframe>

<script src="http://ajax.googleapis.com/ajax/libs/dojo/1.7.2/dojo/dojo.js"></script>
<script>
daringfireball.net/'];

function detect() {
    dojo.forEach(urls, function(url) {
        var iframe = dojo.create("iframe", { src: url, id: url });
        dojo.attr(iframe, "style", {display: 'none'});
        dojo.connect(iframe, "onload", function() {
            dojo.destroy(iframe);
            dojo.place(iframe, dojo.body());
        });
    });

    dojo.place(iframe, dojo.body());
    setTimeout(function () {
        var obj = dojo.byId(url);
        if (obj) {
            dojo.destroy(iframe);
            var entry = dojo.create("li", null, dojo.body());
            entry.innerHTML = "Yes: " + url;
        } else {
            var entry = dojo.create("li", null, dojo.body());
            entry.innerHTML = "No: " + url;
        }
    }, 3000);
});
</script>
Deanonymize (via Clickjacking)
Deanonymize (via Clickjacking)

Click For More Dancing Cats
Deanonymize (via Clickjacking)

Click For More Dancing Cats

"A mashup is a self-inflicted XSS attack."
Douglas Crockford
Deanonymize (via Clickjacking)

"A mashup is a self-inflicted XSS attack."
Douglas Crockford

http://mayscript.com/blog/david/clickjacking-attacks-unresolved
“Unless you've taken very particular precautions, assume every website you visit knows exactly who you are, where you’re from, etc.”

Jeremiah Grossman
Browser Intranet Hacking
Circa (2006)
Browser Intranet Hacking
Circa (2006)

<iframe src="http://192.168.1.1/" onload="detection()"></iframe>
Is My Web Browser Secure?

Saturday,
September 15
2012

Hello [Name],

Thank you for visiting us. Personal online security and privacy is extremely important and we want to help people protect themselves. What most don't know is how much sensitive information their Web browser is revealing, about THEM, with every website they visit. We'd like to show you exactly how much because who knows WHAT shady things others are doing!

DECLASSIFY

Computer
Cross-Site Scripting (XSS)
At least 55% of websites

+ Browser Auto-Complete = pwn

Cross-Site Request Forgery (CSRF)
At least 19% of websites

DNS Rebinding
Chrome Is Most Secure of the Top Three Browsers, Study Finds

By Katherine Noyes, PCWorld
Dec 9, 2011 12:02 PM

Even as Mozilla’s Firefox browser has been surrounded by uncertainty in recent weeks, Chrome seems to be having a very good month.

Not only did Google’s software officially surpass Firefox to assume the No. 2 position in market share last week, but today it was named the most secure of the top three browsers by security firm Accuvant.

“Both Google Chrome and Microsoft Internet Explorer implement state-of-the-art anti-exploitation technologies, but Mozilla Firefox lags behind without JIT hardening,” the company explains in a 100-page study.

Chrome’s plug-in security and sandboxing architectures, meanwhile, are “implemented in a more thorough and comprehensive manner,” making it “the
Chrome Is Not the No. 2 Browser, Study Finds

By Katherine Noyes, PCWorld

Even as Mozilla’s Firefox browser has been gaining weeks, Chrome seems to be having a tough time.

Not only did Google’s software overhaul fail to carry over to assume the No. 2 position in March’s report, but today it was named the most vulnerable of all browsers by security firm AccuScan.

“Both Google Chrome and Microsoft Internet Explorer 9 implement state-of-the-art anti-exploit technology,” says SentinelOne’s Robert Lipsey, behind without JIT hardening,” the company said.

Chrome’s plug-in security and sandboxing feature, implemented in a more thorough manner, was a major concern for the analyst.

Internet Explorer 9 is most secure browser

September 2011

August 2011

- IT-driven economy attracts £530 million broadband investment
- Learn more to earn more
- IT training – it’s in your hands
- IT employment levels hit a record high

Internet Explorer 9 is most secure browser

- Top 10 IT skills demanded by employers
- Apple losing its way with IT customer services?
- Hackers offered cash for shoring up Microsoft IT

Written by James West

Windows Internet Explorer 9 (IE9) is the best browser for blocking malware infections spread via social networks according to research undertaken by NSS Labs.

In its report, NSS Labs says that IE9 catches 99 per cent of malicious web-links spread through social networking sites such as Twitter and Facebook.

Demonstrating how far ahead IE9 is for IT security, Google Chrome came second capturing 13 per cent of threats, followed by Apple Safari 5 and Mozilla Firefox which both intercepted eight per cent, and Opera 11 blocking six per cent.

With the use of social media sites growing outside the personal sphere and into business applications, such as customer service and marketing, IT users and professionals alike need to be aware of just how vulnerable they are - the report stated that between 15,000 and 50,000 new malware programmes are being added to the web every day.

Latest figures from the European Union’s statistic office Eurostat found that almost one-third of internet users caught a virus or malware infection in 2010 resulting in a loss of either information or time rectifying the problem, with three per cent reporting financial loss due to internet attacks.
Possible Solutions?
Login-Detection

Idea: Do not send the Web visitors cookie data to off-domain destinations, destinations different from the hostname in the URL bar, along with the Web requests.
Login-Detection

**Idea:** Do not send the Web visitors cookie data to off-domain destinations, destinations different from the hostname in the URL bar, along with the Web requests.

**Breaks the Web**
Not sending cookies off-domain would break websites using multiple hostnames to deliver authenticated content. Breaks single-click Web widgets like Twitter “Follow,” Facebook “Like,” and Google “+1” buttons. Also breaks visitor tracking via Google Analytics, Coremetrics, etc.
Deanonymization

_Idea:_ Ban IFRAMEs entirely, or at least ban transparent IFRAMEs. Ideally, browser users should be able to “see” what they are really clicking on.
Deanonymization

_Idea:_ Ban _IFRAMEs_ entirely, or at least ban transparent _IFRAMEs_. Ideally, browser users should be able to “see” what they are really clicking on.

**Breaks the Web**

Millions of websites currently rely upon _IFRAMEs_, including transparent _IFRAMEs_, for essential functionality. Notable examples are Facebook, Gmail, and Yahoo! Mail.
Browser Intranet Hacking

Idea: Create a barrier in the browser between “public” and “private” networks by prohibit the inclusion of RFC-1918 on non-RFC-1918 websites.
Browser Intranet Hacking

Idea: Create a barrier in the browser between “public” and “private” networks by prohibit the inclusion of RFC-1918 on non-RFC-1918 websites.

Breaks the Web
Some organizations actually do include intranet content on public websites, for their employees, which does not violate RFC specification.
Browser Intranet Hacking

Idea: Create a barrier in the browser between “public” and “private” networks by prohibit the inclusion of RFC-1918 on non-RFC-1918 websites.

Breaks the Web
Some organizations actually do include intranet content on public websites, for their employees, which does not violate RFC specification.

Vulnerabilities are required by Web standards.
bigger problem

KNOWN “WONT-FIX” ISSUES
Browser vendor’s choice is simple:

Be less secure and more user adopted, or secure and obscure.

Browser War

= Trench Warfare
“[N]obody's breaking the web, dude. Not now, not ever.”
Dan Kaminsky to Jeremiah Grossman, December 21, 2010


- Opt-In security, by website owners
- Measurably low adoption rates
- Do not allow for Web users to protect themselves
Web browsers are NOT “safe.”
Web browsers are NOT “secure.”
Web browsers do NOT protect your “privacy.”
Web browsers are NOT “safe.”
Web browsers are NOT “secure.”
Web browsers do NOT protect your “privacy.”

A HELPFUL VENN DIAGRAM
What do we do now?

Geek meditation session.
1) Status Quo
2) .SECURE
3) Break the Web

...
Mobile Apps

Mini-browsers, where each site / app is isolated. No issues with Login Detection, Denonymization, etc.

“DesktopApps”

Custom browsers’ designed to automatically launch to a website and go no further.
Thank You!

“I Know…” series
http://blog.whitehatsec.com/introducing-the-i-know-series/

Blog: http://blog.whitehatsec.com/
Twitter: http://twitter.com/jeremiahg
Email: jeremiah@whitehatsec.com

I was not in your threat model.
1:53 PM Apr 28th via TweetDeck
Retweeted by 1 person