SecurityInnovation
THE SOFTWARE SECURITY COMPANY

HTML5 & JavaScript Security

Are the new features something to worry about?

Marcus Hodges
Principle Security Engineer

About Security Innovation

- **Authority in Application Security**
  - 10+ years research on vulnerabilities
  - First publicly published security testing methodology, adopted by Microsoft, Adobe, Symantec, SAP
  - Authors of 8 books
  - Application Security partner for Microsoft, Cisco, HP, IBM and Trustwave

- **Helping Organizations Reduce Risk by Securing Applications at the Source**
  - Integrate security at each phase of the SDLC
  - Build internal expertise and competency
  - Find, remediate and ***prevent*** vulnerabilities

symantec, IBM, Cisco, Microsoft, Fujitsu, hp, ING, Allstate, Sony, Ericsson, State Farm, Royal Caribbean, Dell, intel, Ryder, amazon.com, Go Daddy.com, Google, Nationwide, FedEx, Barclays, Texas Instruments

About Me

About myself

- Principal Security Engineer
- Microsoft's SDL Pro Network Practice manager and Trainer
- Hacker, developer, and avid contributor to the security community
- Python, Linux, RE, exploitation, CTF, crypto, math, …

Agenda

- **What is HTML 5.0?**
- **New Features/Security Implications**
  - Cross origin resource sharing
  - Content Security Policy
  - Web Storage & Web SQL
  - New Input and Event tags/attributes
  - Web workers
  - Sandboxing iframes
  - Geolocation
- **Conclusion**

What is HTML 5?

- A collection of new individual features
  - storage, video, threading, …
- Features that required plugins before
  - Flash, Silverlight, Java Applets, ...
- Will HTML 5 replace these plugins? Unknown.
  - Webcam? Audio capture?
- The spec is not complete but browsers are already implementing

Agenda

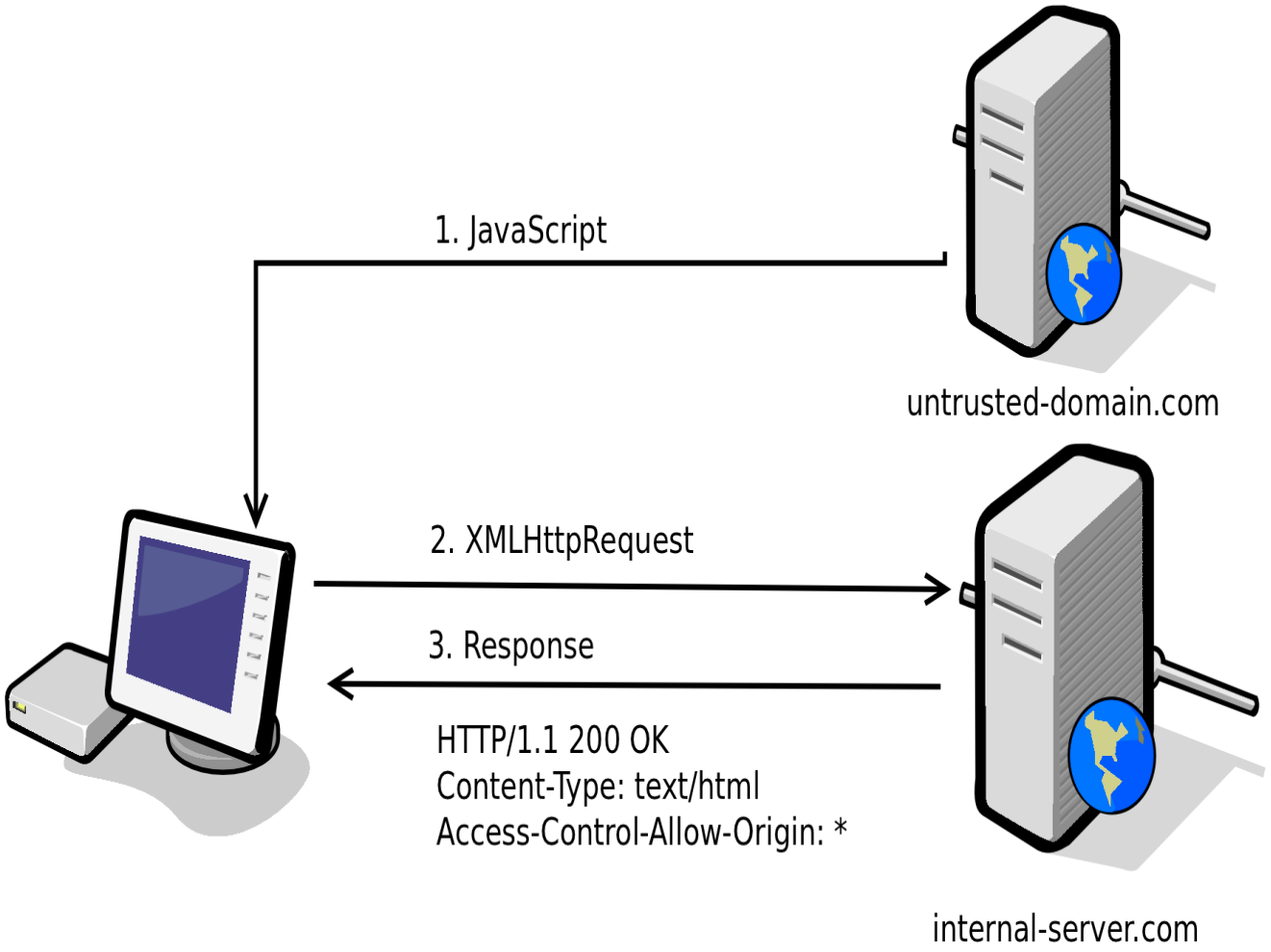
- What is HTML 5.0?
- New Features/Security Implications
  - Cross origin resource sharing
  - Content Security Policy
  - Web Storage & Web SQL
  - New Input and Event tags/attributes
  - Web workers
  - Sandboxing iframes
  - Geolocation
- Conclusion

Cross Origin Resource Sharing (CORS)

- HTML 5 relaxes cross domain restrictions!
  - HTML 4: Same Origin Policy
  - HTML 5: Configurable Policy (DIY)
- Like CrossDomain.xml from Flash, but for the whole web
- JavaScript can make requests to ANY domain
- Access-Control-Allow-Origin HTTP header

Defines whether JavaScript, from a given origin, is allowed to read the response from an XMLHttpRequest

Cross Origin Resource Sharing (CORS)



Cross Origin Resource Sharing (CORS)

Secure

```
HTTP/1.1 200 OK
Content-Type: text/html
Access-Control-Allow-Origin: http://resources.example.com
Access-Control-Max-Age: 86400
Access-Control-Allow-Methods: GET
```

Insecure

```
HTTP/1.1 200 OK
Content-Type: text/html
Access-Control-Allow-Origin: *
```

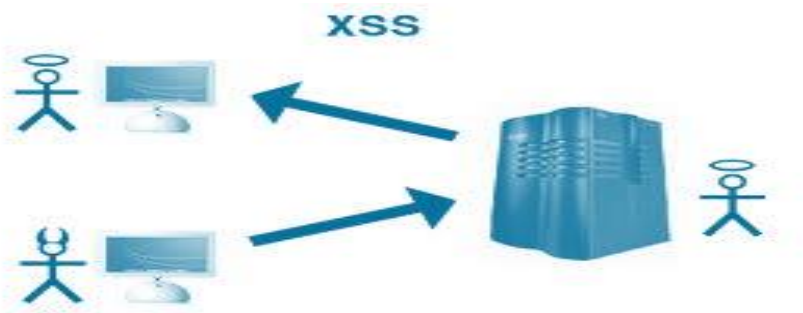
- CORS is a feature that can be used insecurely
- **At best, it is equivalent to the Same Origin Policy**
- Header injection attacks become more interesting

Cross Site Scripting (XSS)

- How can a browser protect against Cross Site Scripting (XSS)?
- XSS attacks exploit the browser's trust of content from the server
- Can the browser discern if <script> is from an attacker?
- There is no intrinsic separation of code from content

Is this script from the application or an attacker?

```
<a href="" onclick="alert('Hello world')">Link</a>
```



Content Security Policy (CSP)

- Server can whitelist which domains are a valid source for an executable script
- All other sources of JavaScript are disabled

```
<script> alert(‘inline scripts are disabled'); </script>
<img src="csp.png"
  onload="alert('event-handling attributes disabled');" />
```

- Requires adhering to development best practices
- Unobtrusive JavaScript (separate behavior from structure/content)

Content Security Policy (CSP)

- X-Content-Security-Policy HTTP header
- Example: Allow content from a trusted domain and all subdomains

```
X-Content-Security-Policy: default-src 'self' *.example.com
```

Unobtrusive JavaScript

Structure

```
<head>
<script type="text/javascript" src="https://js.example.com/script.js"></script>
…
 <a href="">Link</a>
```

Behavior

```
$("a").click(function() { alert("Hello world!"); });
```

As opposed to this

```
<a href="" onclick="alert('Hello world')">Link</a>
```

CORS & CSP

- CORS and CSP work together
- CORS allows XMLHttpRequest to go to any domain
- CSP provides a way to whitelist for which domains requests are allowed
- CORS provides a way to whitelist which responses can be read
- CSP protects the content from potentially malicious JavaScript in the response

Web Storage & Web SQL

- More data can be stored on the client
- Increases compliance woes if not used properly (PCI, HIPPA)
- Two types
  - Web Storage (key-value pair)
    - localStorage
    - sessionStorage (not covered)
  - Web SQL

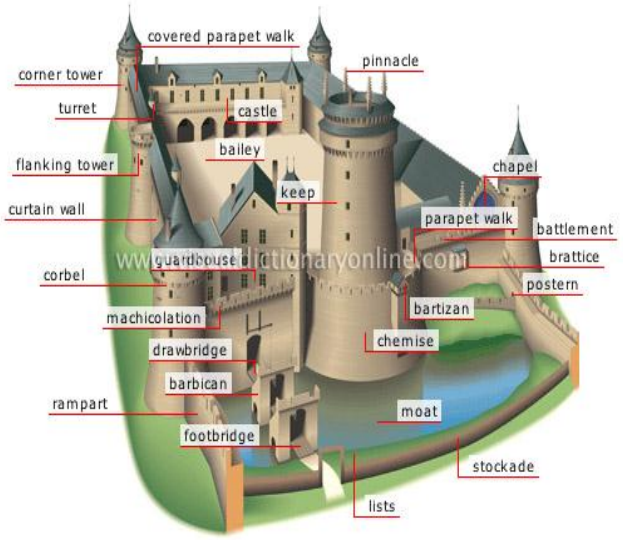
Web Storage & Web SQL

- That was then
  - Cookies were the only form of local storage
  - Relatively limited
  - Sent with every request
  - Attackers stole them with XSS and leveraged them for CSRF
- This is now
  - Lots of space to persist data!
  - Developers control when it's sent to the server
  - Data does not expire, unlike cookies
  - Protected by the Same Origin Policy

Web Storage & Web SQL

Increased Attack Surface

- More assets to steal or manipulate from JavaScript
- Client-side SQL injection
- Privacy impacts
- Cross-directory attacks (shared domains)
- Shared computers are at greater risk (or physical access)
- Persistent attacks vectors in the client



Web Storage & Web SQL

Privacy Concerns

- Users have no control or knowledge over what type of data is stored
- Users can be tracked with persistent tokens that never expire (ever cookie)
- Local Storage is not deleted when the history is cleared (user agent implementation specific)

Cross-Directory Attacks

- Shared domains share data!
- One database / dictionary per domain
  - godaddy, posterous, blogspot, github, microsoft
- There is no feature to restrict access by pathnames

Web SQL

SQL Injection

- Client needs to verify data before processing
- **executeSql()**
- Parameterized Queries: Use the **?** placeholder feature
- Never construct SQL statements dynamically
- **LOAD DATA INFILE** will be a security risk (if implemented)

Correct:

executeSql('SELECT title, author FROM docs WHERE id=?', [id], …

New Tags and Attributes

- Mixed security impact
- Automatic client-side validation
- New native controls in the browser
- New event attributes for JavaScript injection
- All increase the potential for new browser bugs

New Form Input Attributes

- **form**: Associate orphaned controls with a specific form
- **formaction**: Allows changes to where the form content is submitted
- **formenctype**: Changes the form data's encoding type
- **formmethod**: Changes a GET into a POST method and vice versa
- **formnovalidate**: Turns off validation in a form
- **formtarget**: Changes where the action's response is displayed

```
<form id="newHTML5Form">
    <input type="text" value="myValue">
</form>
<input type="submit" form="newHTML5Form" />
```

New Event Attributes

- New events
  - **onerror**
  - **oninput**
  - **onloadstart**
  - **onscroll**
  - ...
- All present new JavaScript injection targets

```
<video  onerror="javascript:alert(1)">
<audio  onerror="javascript:alert(2)">
<canvas onerror="javascript:alert(3)">
```

New Form Input Types

- **Email** – Not RFC compliant but can overload Regex
- **Search** – Just a textbox that you're supposed to style differently
- **Telephone** – Does not enforce a particular syntax
- **URL** – No automatic validation or encoding
- **DateTime** – Several types, all slightly different, may display a native control
- **Number** – A valid floating point number
- **Range** – May be displayed as a slider, return number unless it doesn't
- **Color** – May display a color picker. May display a textbox if unsupported

Note:

- All user input must still be **validated on the server**
- New types may mislead developers to **validate on the client**

Web Workers

- Allows for heavy in-browser processing (threads)
- Introduces race conditions to your browser
- Combined with Web Sockets and CORS creates perfect platform for:
  - DDoS attacks
  - Botnets
  - Reverse Shells
  - Distributed Rainbow Tables

Sandboxing iframes

By default, the sandbox enables extra restrictions

- Content is treated as being from a unique origin
- Cannot access DOM of parent page
- Forms, scripts, and plugins are disabled
- Links cannot target other browsing contexts
- Cannot access cookies or Local Storage

<iframe sandbox="">

Sandboxing iframes

Values of the sandbox attribute REMOVE restrictions (danger!)

- **allow-same-origin**: treat iframe content as being from same origin as containing document
- **allow-top-navigation**: iframe content can load content from containing page
- **allow-forms**: allow form submission
- **allow-scripts**: allow script execution

<iframe sandbox="allow-same-origin allow-scripts">

GeoLocation

- Websites can identify a user's physical location
- Does not require GPS (wifi, IP address, cell towers, …)
- Coordinates accessible via JavaScript (and XSS)
  - Latitude
  - Longitude
  - Heading
  - Speed

GeoLocation

Cross domain user tracking

- Websites can track and share your physical heuristic
- Physical heuristics can be correlated with
  - Sessions
  - Users
  - Ever cookie
- Combine with Web Workers for real-time user tracking

GeoLocation

- Anyone with a motive now knows exactly
  - Who you are
  - Where you are
  - What you are doing
  - Who you are doing it with
- Probably the scariest thing since 1984
  - If they hack my computer – they have access to my data
  - If they have my location – they have access to ME

Privacy is Dead. And we have killed it.

Agenda

- **What is HTML 5.0?**
- **New Features/Security Implications**
  - Cross origin resource sharing
  - Content Security Policy
  - Web Storage & Web SQL
  - New Input and Event tags/attributes
  - Web workers
  - Sandboxing iframes
  - Geolocation

➢ **Conclusion**

Conclusion

Soon, your **web browser** will be able to:

- Execute code (on or offline)
- Interact with devices (webcam, microphone, GPS,…)
- Play multimedia and accelerate 3D applications
- Open TCP connections (to arbitrary hosts)
- Read and write to local storage (databases and files)
- Spawn threads for parallel computing (workers)

I've seen these features before…

Conclusion

Oh yeah! It's called an **operating system**!

(And we haven't seen any vulnerabilities in those…)

Conclusion

The Downside

- HTML 5 adds a lot of features, but no real security improvements
- Requires more developer effort and configuration than before
- There are going to be a lot of new browser vulnerabilities
- and differences between user agent implementations
- XSS gets new toys
- Privacy is dead

The Upside

- Standardizes features that are currently hacked together
- CSP promises real security improvements (so support it!)

Security Innovation Solutions

TeamProfessor
Computer Based Training

- 45 Technical & Awareness courses
- Secure Application Design, Coding, Testing
- .NET, Java, C/C++, C#, PHP, Mobile, OWASP, PCI, Database

TeamMentor
Secure Development Standards

- 3,500 searchable assets (how-to's, checklists, attacks, principles)
- Filter by language, development phase, asset type, etc

Professional Services

- Application Assessment (code and as-built level)
- SDLC Gap Analysis
- Instructor-Led Training

Marcus Hodges
mhodges@securityinnovation.com