

# Defending Computer Networks

## *Lecture 16: Web Drive-bys*

Stuart Staniford

Adjunct Professor of Computer Science

# Logistics

- Project Survey
  - Milestone 1 due Fri Nov 1st

# Assigned Reading

- Javascript Tutorial
  - <http://www.w3schools.com/js/>
  - Recommend you play around with at least the basics to get a feel for it

# Latest News

## **DARPA ANNOUNCES CYBER GRAND CHALLENGE**

October 22, 2013

*First-of-its-kind cyber defense tournament seeks to drive automation revolution in information security*

What if computers had a “check engine” light that could indicate new, novel security problems? What if computers could go one step further and heal security problems before they happen?

To find out, the Defense Advanced Research Projects Agency (DARPA) intends to hold the Cyber Grand Challenge (CGC)—the first-ever tournament for fully automatic network defense systems. DARPA envisions teams creating automated systems that would compete against each other to evaluate software, test for vulnerabilities, generate security patches and apply them to protected computers on a network. To succeed, competitors must bridge the expert gap between security software and cutting-edge program analysis research. The winning team would receive a cash prize of \$2 million.

“DARPA’s series of vehicle Grand Challenges were the dawn of the self-driving car revolution,” said **Mike Walker**, DARPA program manager. “With the Cyber Grand Challenge, we intend a similar revolution for information security. Today, our time to patch a newly discovered security flaw is measured in days. Through automatic recognition and remediation of software flaws, the term for a new cyber attack may change from zero-day to zero-second.”

# Key Dates on DARPA Grand Challenge

- Jan 14<sup>th</sup>, 2014 – Proposals Due
- Jun 2014 – Funded work begins
- Jun 2015 – Qualification event
- Jul 2016 – Final Event

# Main Goals for Today

- Start on web-client attacks
  - Learn a little bit of javascript
    - Large language which cannot possibly cover in detail
    - Just enough to understand some uses in attacks

# Web Drive-By Download Attacks

- Became big around 2005-2007.
- Have been the main action since then.
- Attacker response to firewall/IPS technology.
  - Largely circumvents those defenses.
  - Took a while to develop useful defenses.
    - Still a very active arms-race.

# Two Main Schemas

- Scan/Compromise legit websites
  - Eg SQL Injection attacks
  - Insert <iframe>s into site
  - Iframes include content from an exploit server
- Malverts
  - Malicious ads bought through chains of middle men
  - Redirects to malicious content (often swf (Flash))



# Either Way

- Exploit server runs an exploit kit
- Exploit kit tests nature of browser/plugins
  - Java
  - PDF
  - Flash
- Picks one or more exploit objects
- Takes control of browser/plugins
- Installs malware/trojans
- Command and control for instructions
- Exploit kits often have extensive management infrastructure
- Profit!

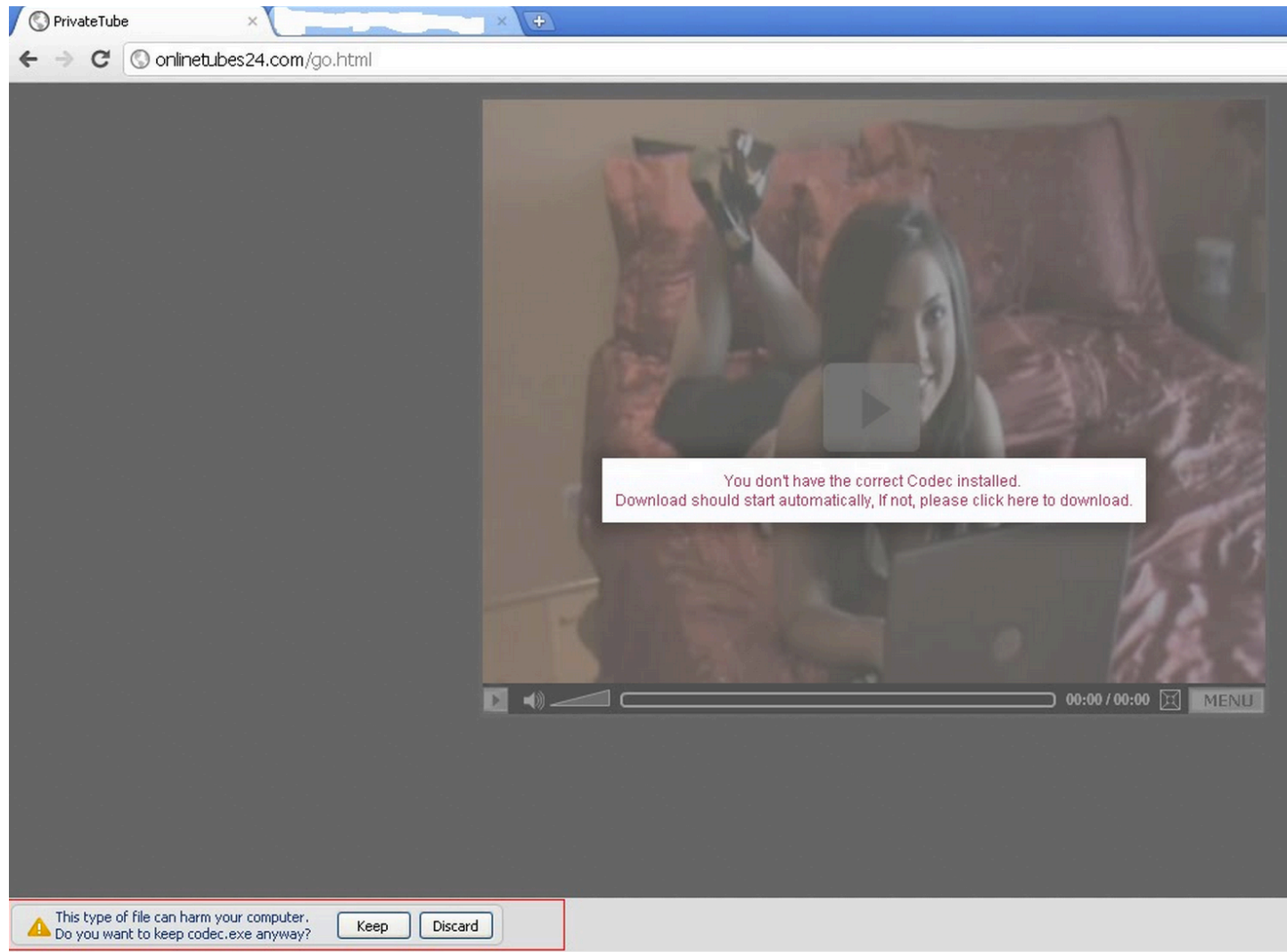
# Also, social engineering

- Trick humans into installing/running malware
- Also works pretty well
- Let's look at a few examples

# Fake AV



# More Social Engineering




<http://research.zscaler.com/2011/12/fake-video-codecs-still-going-strong.html>

# And More

Home / Downloads / Adobe Flash Player /

## Adobe Flash Player

 Adobe Flash Player 11.2.181.25 (1.95 MB)

[Do you have a different operating system or browser?](#)

By clicking the Download now button, you acknowledge that you have read and agree to the [Adobe Software Licensing Agreement](#).

[Download now](#)

Please note, depending on your settings, you may have to temporarily disable your antivirus software.

**RESOURCES**

- [Learn more about Flash Player](#)
- [Flash Player system requirements](#)
- [Distribute Flash Player](#)

**GFI**

<http://www.threattracksecurity.com/it-blog/fake-adobe-flash-updates-resurfaces-in-the-web/>

# Javascript

- Javascript is programming language of www
  - Tightly integrated with html
  - Interpreted in browser
  - Provides dynamic behavior to web pages
  - C-ish syntax
- Used very heavily in web attacks
  - Sometimes directly in exploit
  - Other times just for obfuscation
  - Javascript is pretty much a security nightmare
  - Can't do much with web attacks without learning at least some javascript
- History:
  - Developed by Netscape (1995)
  - Very limited connection with java

# Inclusion in HTML

- `<script> js – blah - blah </script>`
  - Technically should be
  - `<script language = “javascript”>`
- `<script src = “foo.js”>`
- These are interpreted/run at page load time
- In tag attributes:
  - `<button type="button" onclick="myJSFunc()">Button Name</button>`
  - onmouseover, onkeypress, dozens more events that can trigger interpretation/execution of additional js

# Some basics of syntax

- Variable declarations
  - `var x;` // Now x is undefined
  - `var x = 5;` // Now x is a Number
  - `var x = "John";` // Now x is a String
- Loose dynamic typing a la Perl etc
- All the usual C operators: `+`, `-`, `++`, `&&`, ...
- `+` on strings is concatenation
  - `"foo" + "bar" == "foobar"`
  - `"foo"+5 == "foo5"`



# JavaScript Arrays

- `var cars=["Saab","Volvo","BMW"];`
  - `cars[0] == "Saab"`
  - `cars.length == 3`
- Arrays can be returned from functions and passed to functions

# Control Structures

- `if(i<5) {foo code} else {bar code}`
- `for (var i=0;i<N;i++) { blah; blah;}`
- `while (i < 5) {blah; blah;}`
- `switch(n) {`
  - `case 1: blah;break;`
  - `case 2: blah; break;`
  - `default: blah}`

# Javascript Functions

```
function myFunction(var1,var2)  
{  
some code  
}
```

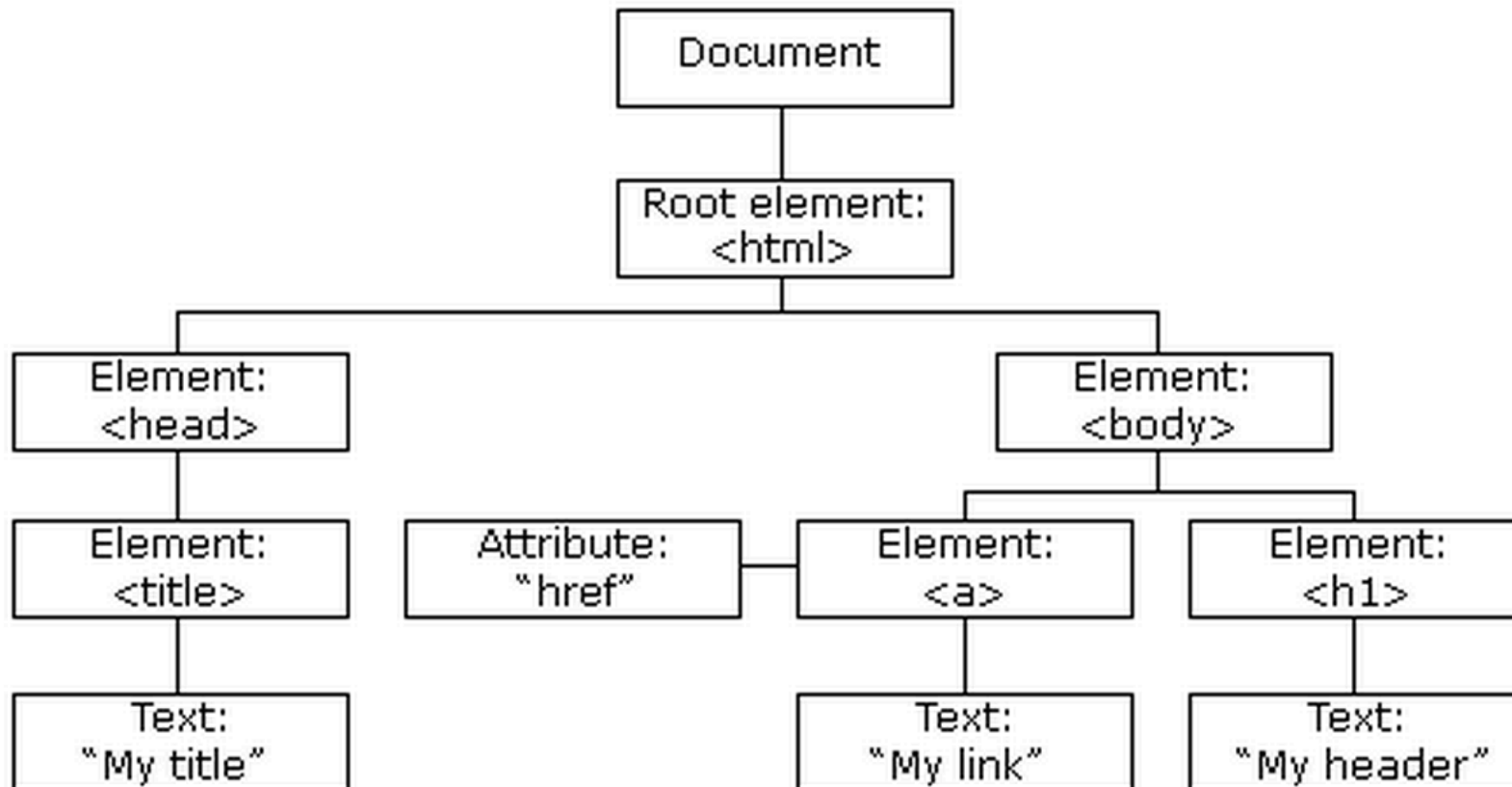
...

```
<blah onclick="myFunction('foo', 7)">
```

# Object Orientation in JS

- Objects are like hashes/dictionaries
- `var person={firstname:"John", lastname:"Doe", id:5566};`
  - `person.id==5566`
- Everything is an object, and many standard methods available
  - `var foo = "bar";`
  - `foo.length == 3`
  - `foo.substring(0,1) == "ba"`

# Document Object Model



[http://www.w3schools.com/js/js\\_htmlDOM.asp](http://www.w3schools.com/js/js_htmlDOM.asp)

# JS Dom Access

- JS can change all the HTML elements in page
- JS can change all the HTML attributes in page
- JS can change all the CSS styles in the page
- JS can react to all the events in the page
- Browser will show changes
  - Allows for dynamic control of page

# Accessing the DOM from JS

- Given `<p id="intro">Hello world.</p>`
  - `var x=document.getElementById("intro");`
  - `var y = document. getElementsByTagName("p")`
    - y is now an array of all the `<p>` elements
    - `for(var i=0; i<y.length; i++)...`
  - `x.innerHTML = "Goodbye."`
    - Will replace "Hello world" with "Goodbye"
  - `document.createElement("p");`

# Let's Have a Quick Play

- <http://www.w3schools.com/js/>



# Sample Browser Exploit

- This is a famous IE exploit used as 0day
  - To compromise Google and many others
  - By Chinese PLA
- We will walk through
- <http://www.exploit-db.com/exploits/11167/>

# Protecting Yourself

- Up-to-date
  - OS
  - Browser
  - Plugins
- \*BSD > Linux > Mac OS > Windows
  - Not inherently more secure, just less attacked
- Click-to-play
  - <http://krebsonsecurity.com/2013/03/help-keep-threats-at-bay-with-click-to-play/>
- AV (sort of)

## National Cyber Awareness System

### Vulnerability Summary for CVE-2010-0249

**Original release date:** 01/15/2010

**Last revised:** 07/13/2013

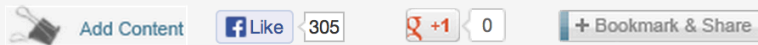
**Source:** US-CERT/NIST

#### Overview

Use-after-free vulnerability in Microsoft Internet Explorer 6, 6 SP1, 7, and 8 on Windows 2000 SP4; Windows XP SP2 and SP3; Windows Server 2003 SP2; Windows Vista Gold, SP1, and SP2; Windows Server 2008 Gold, SP2, and R2; and Windows 7 allows remote attackers to execute arbitrary code by accessing a pointer associated with a deleted object, related to incorrectly initialized memory and improper handling of objects in memory, as exploited in the wild in December 2009 and January 2010 during Operation Aurora, aka "HTML Object Memory Corruption Vulnerability."

# createEventObject

## createEventObject method (document)



### Browser support:



Creates an [event](#) object for message sending.

After a new [event](#) object is created, it can be dispatched with the [fireEvent](#) method. The necessary properties of the newly created [event](#) object must be set before dispatching. Use an existing [event](#) object for the first parameter of the **createEventObject** method to copy its properties. The [cancelBubble](#), [returnValue](#), [srcElement](#) and [type](#) properties need not be set, the [fireEvent](#) method overrides these properties before dispatching.

In Firefox, Opera, Google Chrome, Safari and Internet Explorer from version 9, use the [createEvent](#) method to create a new [event](#) object.

Although Internet Explorer 9 supports the **createEventObject** method, but the [createEvent](#) method is cross-browser and it provides more complex functionality, so it is preferred to use.

### Syntax:

```
object.createEventObject (eventObjToClone);
```

You can find the related objects in the [Supported by objects](#) section below.

<http://help.dottoro.com/ljhlvomw.php>