Michael Sutton

VP, Security Research

A WOLF IN SHEEP'S CLOTHING

The Dangers of Persistent Web Browser Storage

Twitter Questions: @zscaler_sutton

ZSCALER™

Who Am I?

Company

- Zscaler – SaaS solution for web browser security
- VP, Security Research

Background

- SPI Dynamics – acquired by HP
- iDefense – acquired by VeriSign

Research

- Web security
- Client-side vulnerabilities
- Fuzzing

Overview

Background

Data Privacy

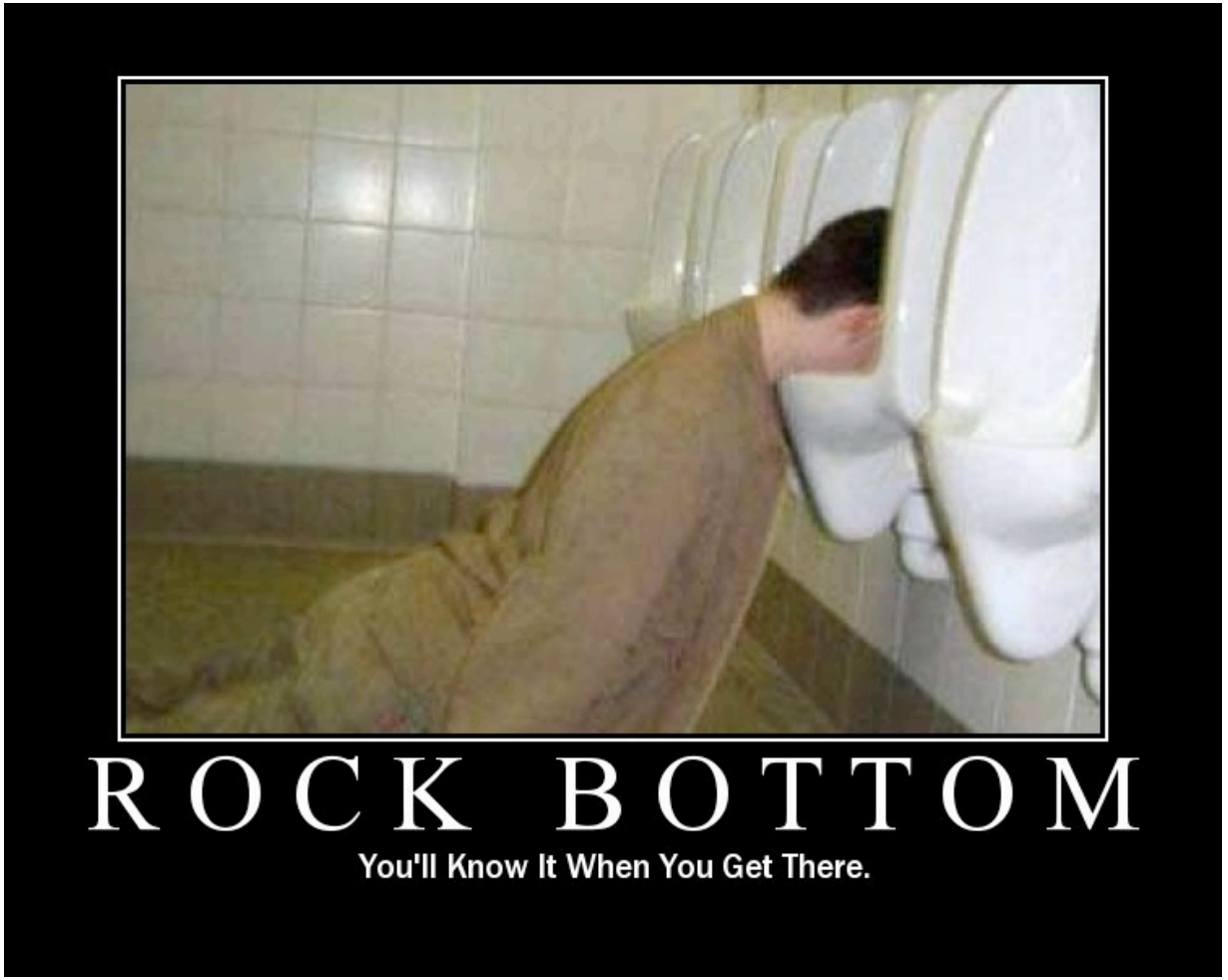
- HTTP Cookies
- Flash Local SharedObjects

Data Integrity and Confidentiality

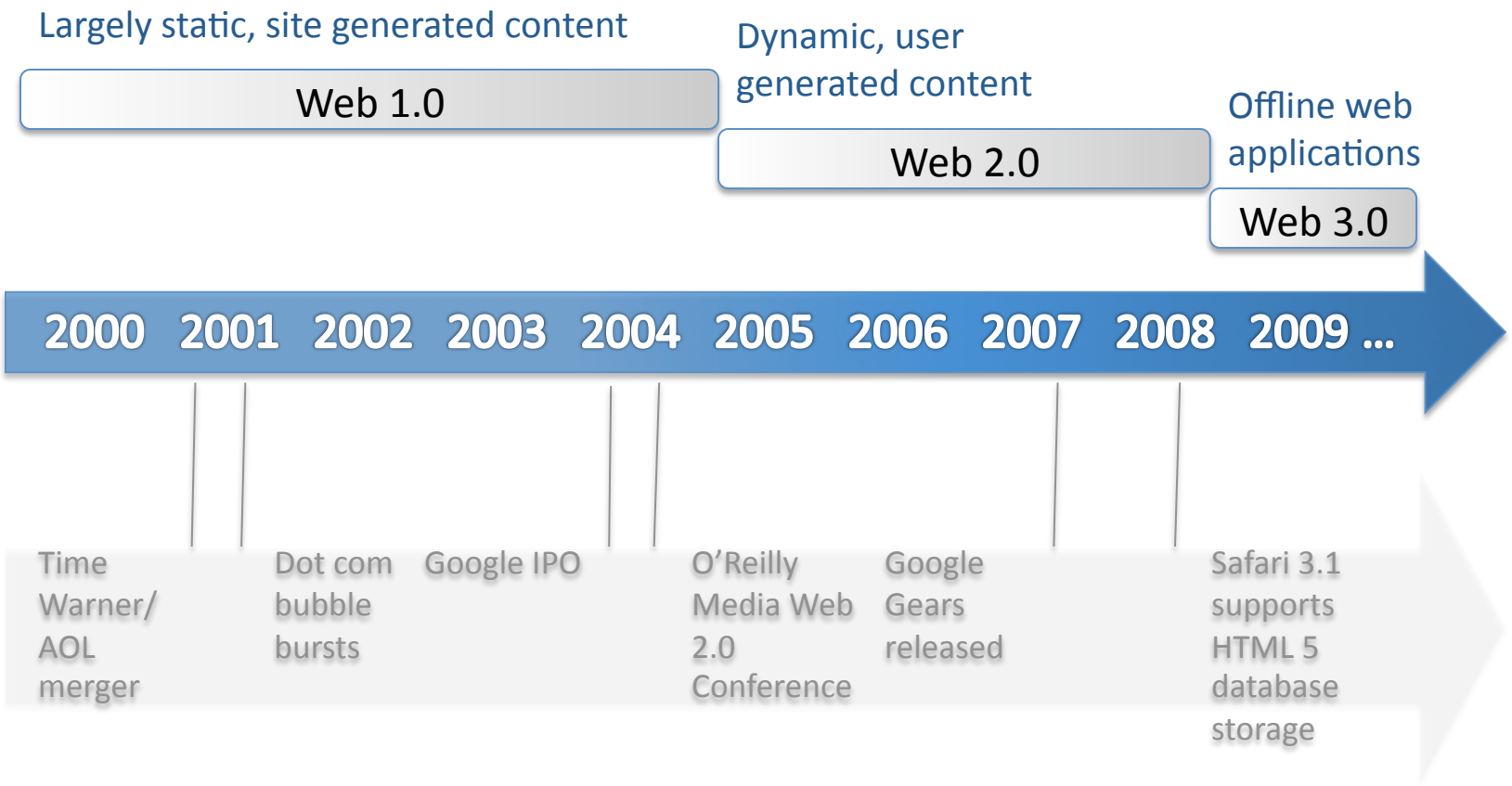
- Gears
- HTML 5 Structured Client Side Storage

Future

Background



Evolution of Web Applications



Browser Storage

HTTP Cookies

- Initially supported by Mosaic Netscape v0.9 beta – released Oct. 13, 1994
- Internet Explorer v2.0 support in Oct. 1995
- Primarily used for personalization/tracking
- RFC 2109 recommends minimum storage capacity of 4KB per cookie

Flash Local Shared Objects

- First introduced in Flash Player 6.0
- User controlled settings to manage 'Flash cookies' introduced in Flash Player 8.0
- Default storage capacity of 100KB

(Google) Gears

- Launched May 31, 2007
- Full local relational database

HTML 5 Database Storage

- Supported by Safari 3.1, released March 18, 2007
- Full local relational database

HTTP Cookies

HTTP Cookies

Origin

- Mosaic Netscape v0.9 beta – Oct. 13, 1994
- Patented by Netscape in 1995

Purpose

- Primarily used for tracking
- Allow sites to identify a combination of user, browser and computer

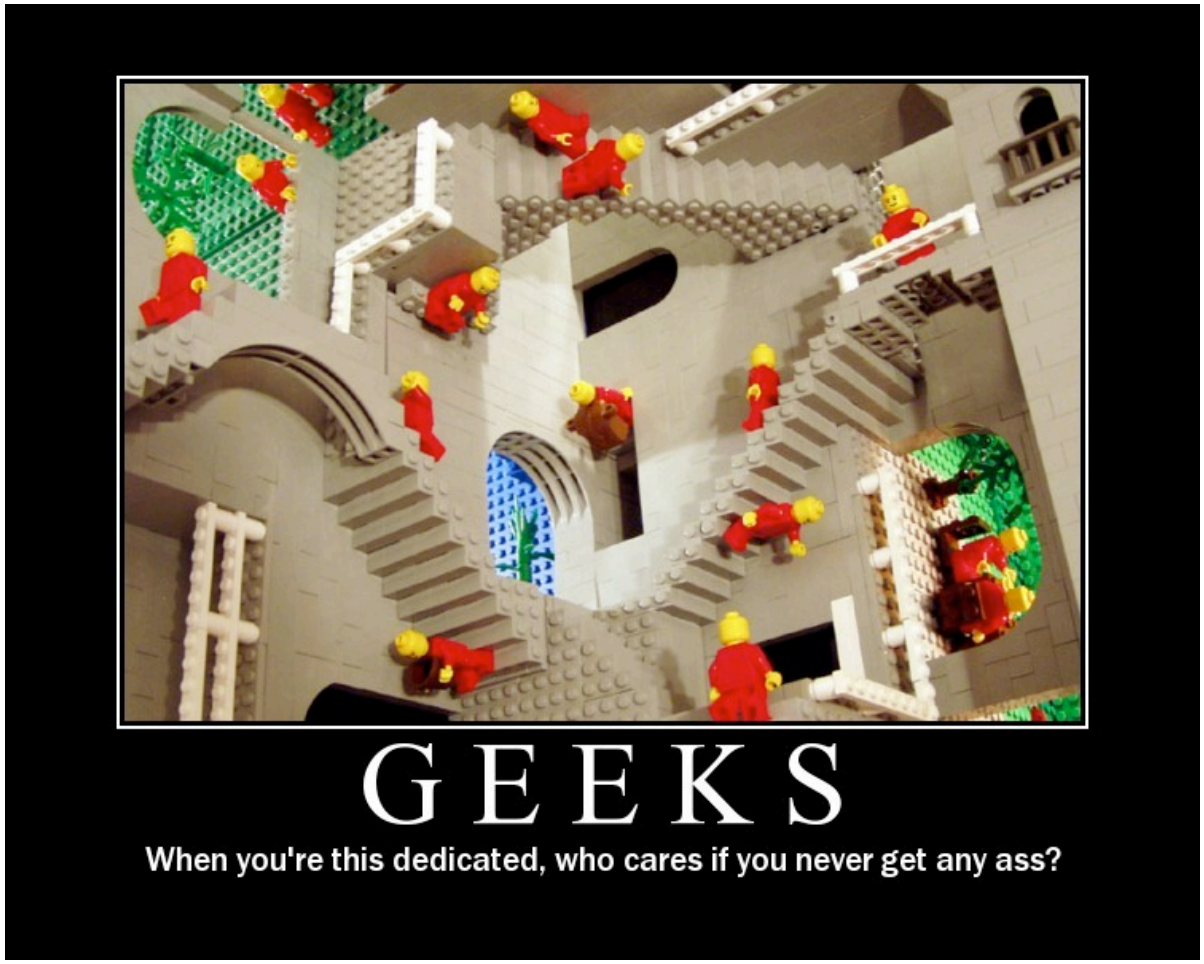
Details

- Restricted by same origin policy
- RFC 2109 - HTTP State Management Mechanism
  - At least 4096 bytes per cookie
  - At least 20 cookies per unique host
- Controllable expiration

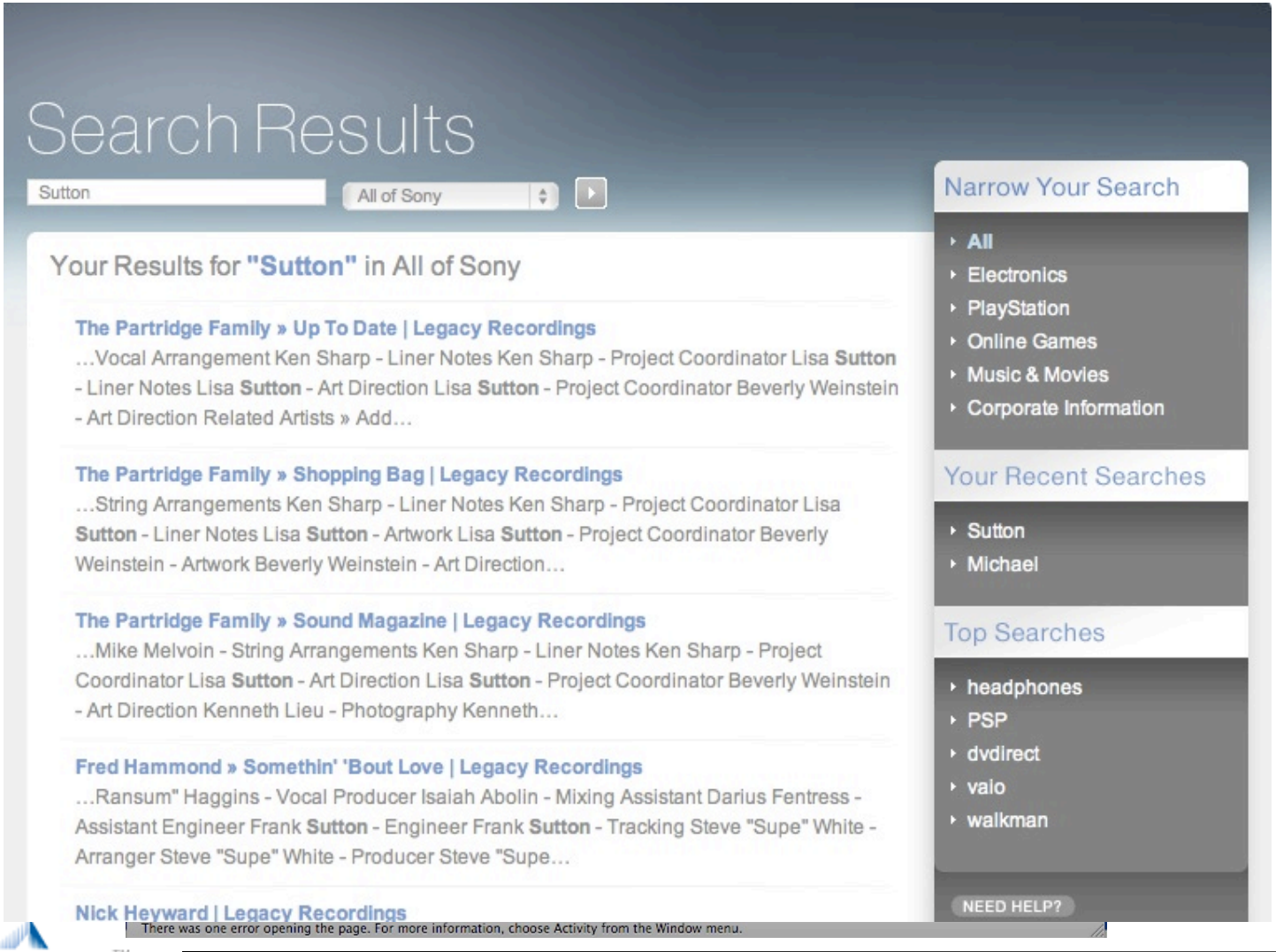
Abuse

- Cookie hijacking
- Cookie poisoning

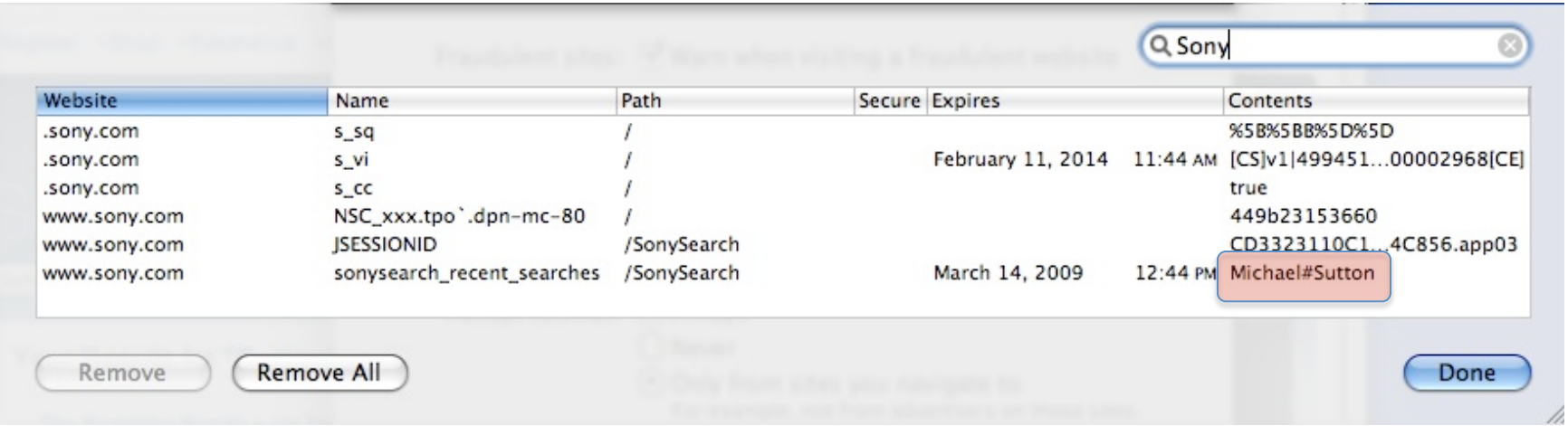
Persistent csXSS



Sony Search



Sony Persistent csXSS



Sony Persistent csXSS

Persistent csXSS

Unique Aspects

- Persistent only on client
- Automatically triggered whenever page is revisited

Attack Potential

- Leverage for user-specific XSS attacks
  - Not possible with traditional persistent XSS
- Inform attacker whenever you've returned to a site
  - Timing is an issue with attacks such as CSRF

Prevalence

- Surprisingly common, especially on sites which feature a search history

Flash Local SharedObjects

Flash LSOs

Origin

- Flash Player 6.0 – March 2002
- Flash Player 8.0 - User controlled settings to manage 'Flash cookies'

Purpose

- Primarily used for tracking/default settings
- Larger capacity permits use for additional purposes
- Popular – my laptop currently has LSOs from 102 domains – all from regular browsing

Details

- Default storage of 100K → can be unlimited
- No expiration
- Difficult to delete – not tied to browser caches

Abuse

- Cookie hijacking
- Cookie poisoning
- Data leakage

What's Stored in Flash LSO's?

Tracking Identifiers

- Most common

Configuration Settings

- Typical on audio/video streaming sites

Authentication Credentials

- Pandora (Encoded password)

Easter Eggs

- *"Hey. You've just found another easter egg. Congrats - you gained nothing :)!"*
  - Portal – Flash game by Armor Games

SharedObject Sandboxing

Programming Adobe ActionScript 3.0 for Adobe Flash

SharedObjects

Flash Player provides the ability to use shared objects, which are ActionScript objects that persist outside of a SWF file, either locally on a user's file system or remotely on an RTMP server. Shared objects, like other media in Flash Player, are partitioned into security sandboxes. However, the sandbox model for shared objects is somewhat different, because shared objects are not resources that can ever be accessed across domain boundaries. Instead, shared objects are always retrieved from a shared object store that is particular to the domain of each SWF file that calls methods of the SharedObject class. Usually a shared object store is even more particular than a SWF file's domain: by default, each SWF file uses a shared object store particular to its entire origin URL. evening. "We worked quickly to implement a fix for the issue recently reported in Orkut. We also took steps to help prevent similar problems in the future. Service to Orkut was not disrupted during this time."

Flash LSO Storage Locations

Windows XP

- $user\Application Data\Macromedia\Flash Player\#SharedObjects.

Windows Vista it is in each user's

- $user\AppData\Roaming\Macromedia\Flash Player\#SharedObjects.

Mac OS X

- ~/Library/Preferences/Macromedia/Flash Player/#SharedObjects.

Linux

- /home/$user/.macromedia/Flash_Player/#SharedObjects.

LSO Files

Format

- Binary files
- *.sol extension
- Store text data

SharedObject readers

- FD3
- SOLReader

User Control

- Website Storage Settings in Flash Player Settings Manager
- Firefox add-ons – Objection, Better Privacy

Reading/Writing From/To Flash Cookies

Limitations

- Same origin policy
- Origin determined by path
  - Sites can write LSO's at a predefined level (e.g. `SharedObject.getLocal("zscaler", "/")`)

Requirements

- Ability to upload SWF files
  - Increasingly common on Web 2.0 sites
- Victim must visit site with uploaded content

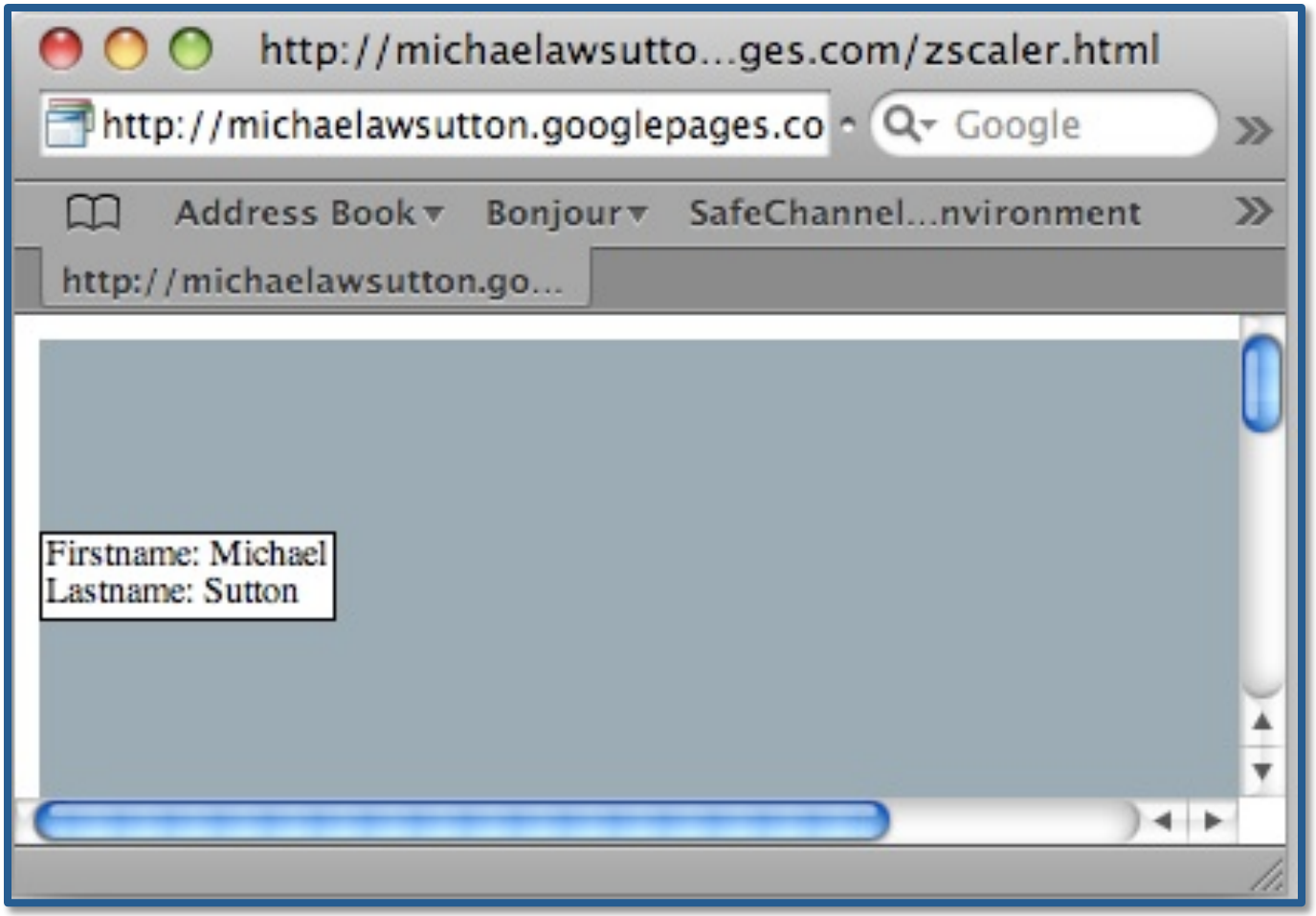
Writing To a Flash Cookie

```
package {

        import flash.net.SharedObject;
        import flash.display.Sprite;

        public class zscaler extends Sprite {
                private var user:SharedObject;
                private var firstname :String;
                private var lastname:String;
                public function zscaler() {

                        user = SharedObject.getLocal("zscaler");
                        firstname = "Michael";
                        lastname = "Sutton";

                        user.data.firstname = firstname;
                        user.data.lastname = lastname;

                        user.flush();
                }
        }
}
```

Reading From a Flash Cookie

```
...
public function zscaler() {
        var label:TextField;

        user = SharedObject.getLocal("zscaler");

        firstname = user.data.firstname;
        lastname = user.data.lastname;

        label = new TextField();
        label.autoSize = TextFieldAutoSize.LEFT;
        label.background = true;
        label.border = true;
        label.text = "Firstname: " + firstname + "\nLastname: " + lastname;

        addChild(label);

        user.flush();
}
...
```

Reading From a Flash Cookie



Pros/Cons of Flash Cookies

Pros

- Model increases complexity of cookie stealing
- Sandboxing limits scope of attacks – similar to HTTP cookies

Cons

- Greater default storage capacity (100KB) – increases likelihood that storage will be used for sensitive data
- Difficult to delete
- No expiration

(Google) Gears

Gears

Origin

- Launched as Google Gears on May 31, 2007
- 'Google' dropped from project title on 1st anniversary

Purpose

- Initial – "offline-enabling applications"
- Overall – "close the gap between web apps and native apps by giving the browser new capabilities"

Details

- Primary components:
  - LocalServer – Local HTTP/HTTPS capable server for delivering content
  - Database – Local implementation of SQLite relational database for storing content
  - WorkerPool – Run resource intensive JavaScript in the background to improve performance

Abuse

- Data confidentiality
- Data integrity

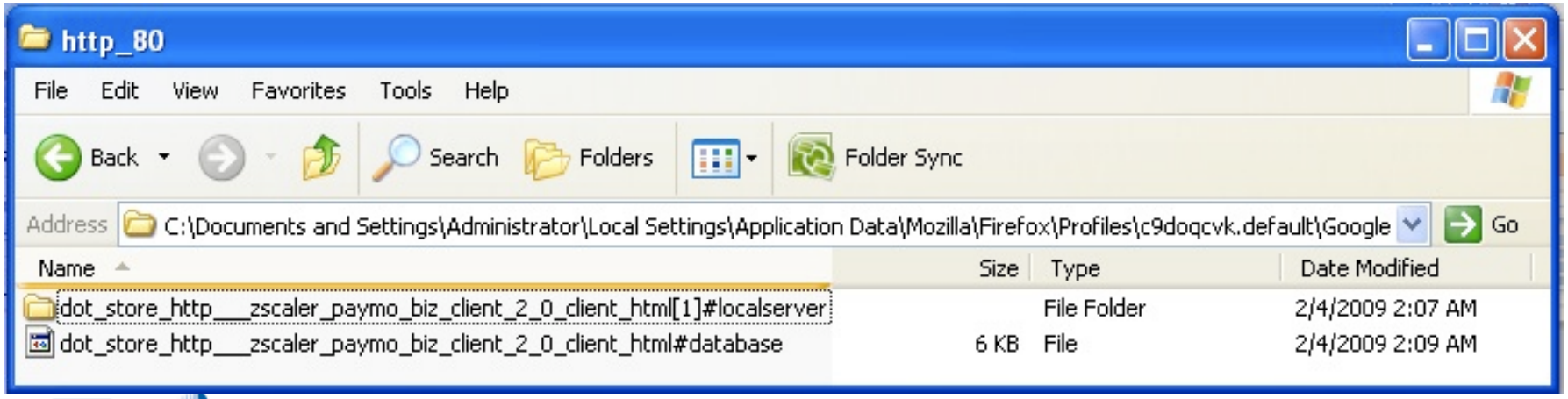
Gears Activation

Allow

- User must permit Gears access

Install

- SQLite database installed on local file system



Gears Storage Locations

Windows XP

- **Internet Explorer**: C:\Documents and Settings\<user>\Local Settings\Application Data\Google\Google Gears for Internet Explorer
- **Firefox**: C:\Documents and Settings\<user>\Local Settings\Application Data\Mozilla\Firefox\Profiles\{PROFILE}.default \Google Gears for Firefox
- **Google Chrome**: C:\Documents and Settings\<user>\Local Settings\Application Data\Google\Chrome\User Data\Default \Plugin Data\Google Gears

Windows Vista

- **Internet Explorer**: C:\Users\<user>\AppData\LocalLow\Google\Google Gears for Internet Explorer
- **Firefox**: C:\Users\<user>\AppData\Local\Mozilla\Firefox\Profiles\{PROFILE}.default\Google Gears for Firefox
- **Google Chrome** C:\Users\<user>\AppData\Local\Google\Chrome\User Data\Default\Plugin Data\Google Gears

Mac OS X:

- **Firefox**Users/<user>/Library/Caches/Firefox/Profiles/{PROFILE}.default/Google Gears for Firefox
- **Safari**: ~/Library/Application Support/Google/Google Gears for Safari

Linux

- **Firefox**: <user>/.mozilla/firefox/{PROFILE}.default/Google Gears for Firefox

Windows Mobile

- **Mobile Internet Explorer**: \Application Data\Google\Google Gears for Internet Explorer

csSQLi

csSQLi

Definition

- Ability to read/write to/from a database stored on a client machine

Facilitator

- Browser databases are accessed via JavaScript
- XSS on a vulnerable site can expose any web browser to csSQLi, regardless of patch level

Targets

- Gears
- HTML 5

A Big Thank You To Paymo.biz

Timeline

- Feb 4 – Vulnerability reported to Paymo.biz
- Feb. 5 – Initial response requesting additional information
- Feb. 5-9 – Additional Correspondence
- Feb. 9 – Fix implemented

Thank You

- Paymo went out of their way quickly respond to the reported vulnerability in order to protect their clients. They were gracious and a pleasure to work with. Web application vendors everywhere can learn from their example.
- …and they offered a free year of service! How's that for gratitude.

Paymo Injection Point

```
<h2>SQLi</h2>
<p><strong>Client</strong>
<a href="/clients/view/?id=16392">Default Client</a></p>

<p>***injection_point***</p>

<div style="float: left; padding-bottom: 10px;">
```

Injection point

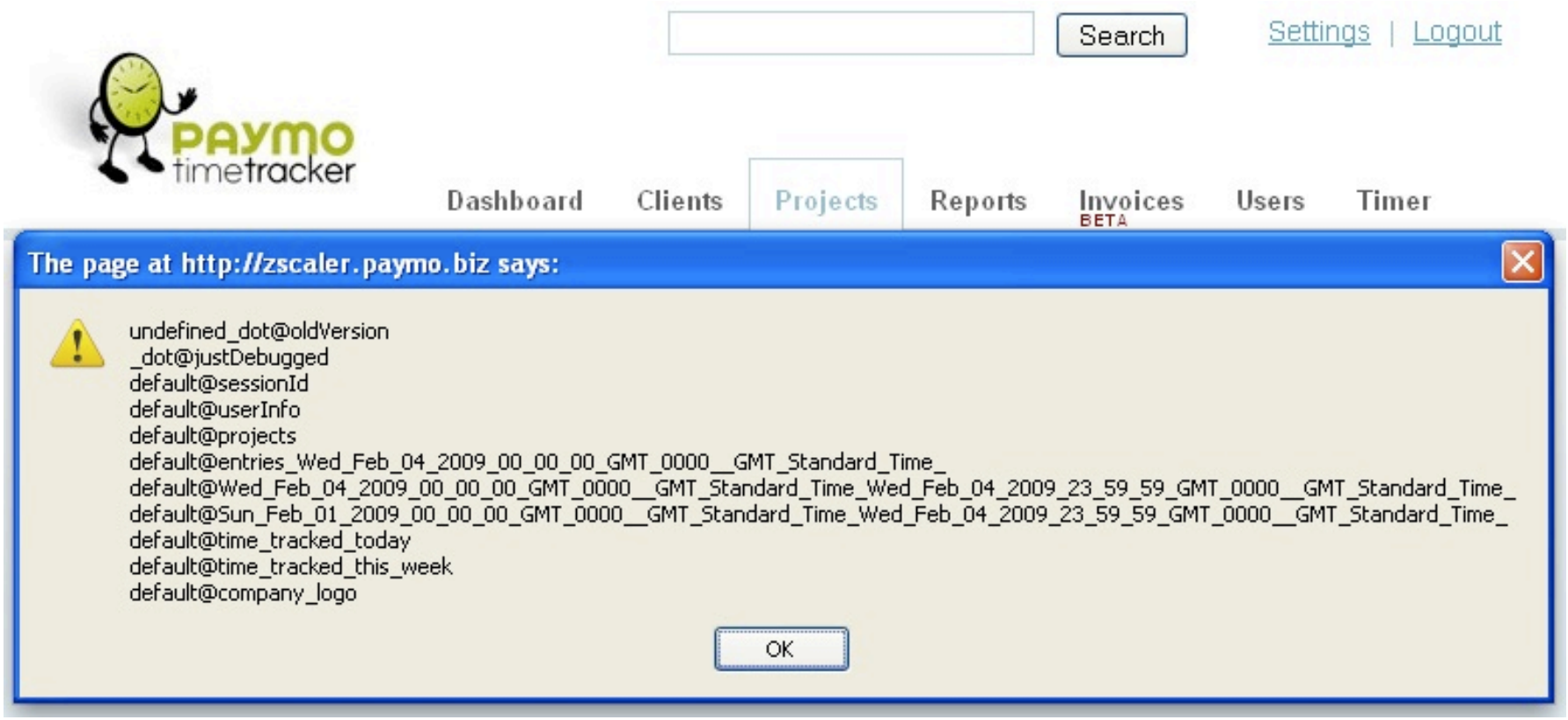
- Within paragraph tag
- Tag will need to be closed </p>

Read Paymo Data

```
1 </p>
  <script type="text/javascript"
2   src="http://code.google.com/apis/gears/gears_init.js"></script>
  <script type="text/javascript">
  var db = google.gears.factory.create('beta.database');
3 db.open('dot_store_http___zscaler_paymo_biz_client_2_0_client_html');
  var data;
4 var rs = db.execute('SELECT * FROM __DOJO_STORAGE');
  while (rs.isValidRow()) {
    data = data + (rs.field(0) + '@' + rs.field(1));
    data = data + '\n';
    rs.next();
  }
  alert(data);
  rs.close();
  </script>
  <p>
```

1. Close paragraph tag
2. Include Gears API
3. Open existing local database
4. Execute SQL query

Paymo csSQLi



Gears csSQLi

BulitIn SQLi Protection

- Secure → db.execute('insert into MyTable values (?)', data);
- Insecure → db.execute('insert into MyTable values (' + data + ')');

Meaningless if a site is vulnerable to XSS

- 67% of sites likely to have XSS [Whitehat Security – December 2008]

SQLi vs csSQLi

SQLi

- Identify database structure through verbose error messages or brute force
- Online attacks
- SQL statement must be vulnerable

csSQLi

- Database structure is readily accessible
- Online and offline attacks
- XSS makes any site vulnerable, regardless of SQL syntax

csSQLi vs Cookie Theft

Question

- Couldn't I access the same information by stealing a user's cookie and accessing their online data?

Answer

- Cookie theft does not guarantee data access
  - Site may not use cookies for authentication
  - Additional ACLs (i.e. IP source address) would prevent access
  - Session credentials have expired or user has logged out
- Offline data does not have to mirror online data

Verdict

- No

Sites Using Gears

Pros/Cons of Gears

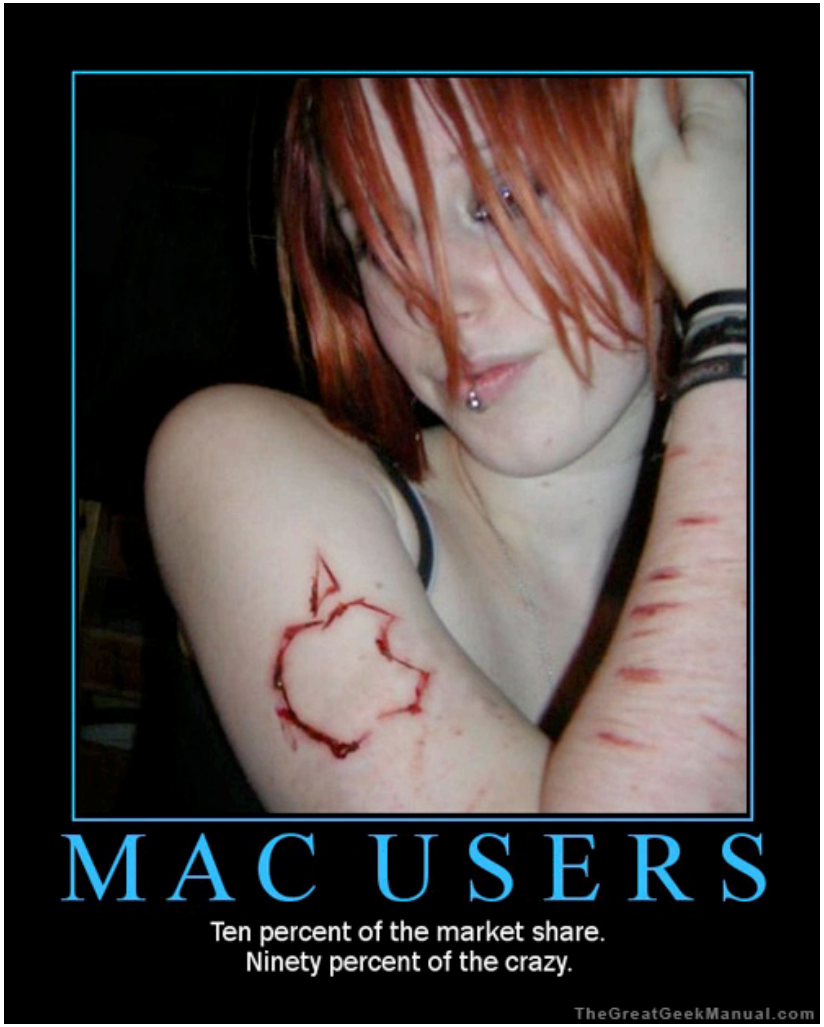
Pros

- Requires explicit user acceptance
- Has built in protections for vulnerabilities such as SQLi

Cons

- Despite default protections, being JavaScript based, it is open to attack should injection flaws such as XSS exist in the host application
- Implementing a secure technology on an insecure site invalidates the built in protections
- Increases the attack surface
  - csSQLi is a reality - Data can be remotely accessed from a local relational database

HTML 5
Structured Client Side Storage



HTML 5

Origin

- WHATWG began work on specification in 2004
- W3C published first public working draft Jan. 22, 2008

Purpose

- New markup, APIs, error handling, etc.
- Includes section on *Structured Client-Side Storage*

Details

- Session Storage – Similar to HTTP session cookies with greater flexibility
- Local Storage – Similar to HTTP persistent cookies with greater flexibility
- Database Storage – Local relational database

Abuse

- Data confidentiality
- Data integrity

HTML 5 Browser DB Support

- Internet Explorer 8
  - Supports session storage and local storage, not database storage
- Firefox
  - Supports session storage and local storage, not database storage
- Safari 3.2x
  - Full support
- Opera
  - No HTML 5 support
- Chrome
  - "Despite using the latest branch of…the local database features didn't make it into Chrome's first release… Chrome's isolated sandbox system…would break the built-in WebKit database functionality…" [monkey_bites]

HTML 5 Database Storage Locations

Mac OS X

- /Users/[username]/Library/Safari/Databases

Others

- Currently, Webkit based browsers are the only ones supporting HTML Database Storage

HTML 5 csSQLi

Resources

- Paper by Alberto Trivero describes potential abuse of HTML 5 structured client side storage
- http://trivero.secdiscover.com/html5whitepaper.pdf
- Various issues covered including csSQLi via XSS
  - Same overall issue as demonstrated in Paymo.biz example

Gears vs. HTML 5

- Blog postings from Google indicate a desire to ultimately make Gears compatible with the HTML 5 specification

Comparison of Local Storage Technologies

	HTTP Cookies	Flash LSOs	Gears	HTML 5
Explicit Acceptance	No	No	Yes	No
Storage Limit	4KB	Unlimited (100KB default)	Unlimited	Unlimited
Expiry	Custom	Never	Never	Never
File Format	Text	Binary	Binary (SQLite)	Binary (SQLite)
Deployment	Universal	Near universal	Minimal	Beta only

How Gears and HTML 5 Change the Game for Attackers

Offline

- Targets can be attacked regardless of current Internet connectivity
  - e.g. Offline - Phishing email read while from Gmail, linked clicked and Gears enabled application attacked

Open

- No need to determine data structure for SQLi – everyone has it

Attack surface

- Potentially confidential data moves from a single, centralized location (server) to potentially millions of individual locations (client)
- All targets (clients) can be attacked from one location (web app w/ XSS vuln.)

Predictions

Adoption

- Expect increased adoption of Gears thanks to favorable exposure from Gmail integration
- HTML 5 and Gears are unlikely to compete – Google has already expressed a desire to make Gears compatible with the HTML 5 specification

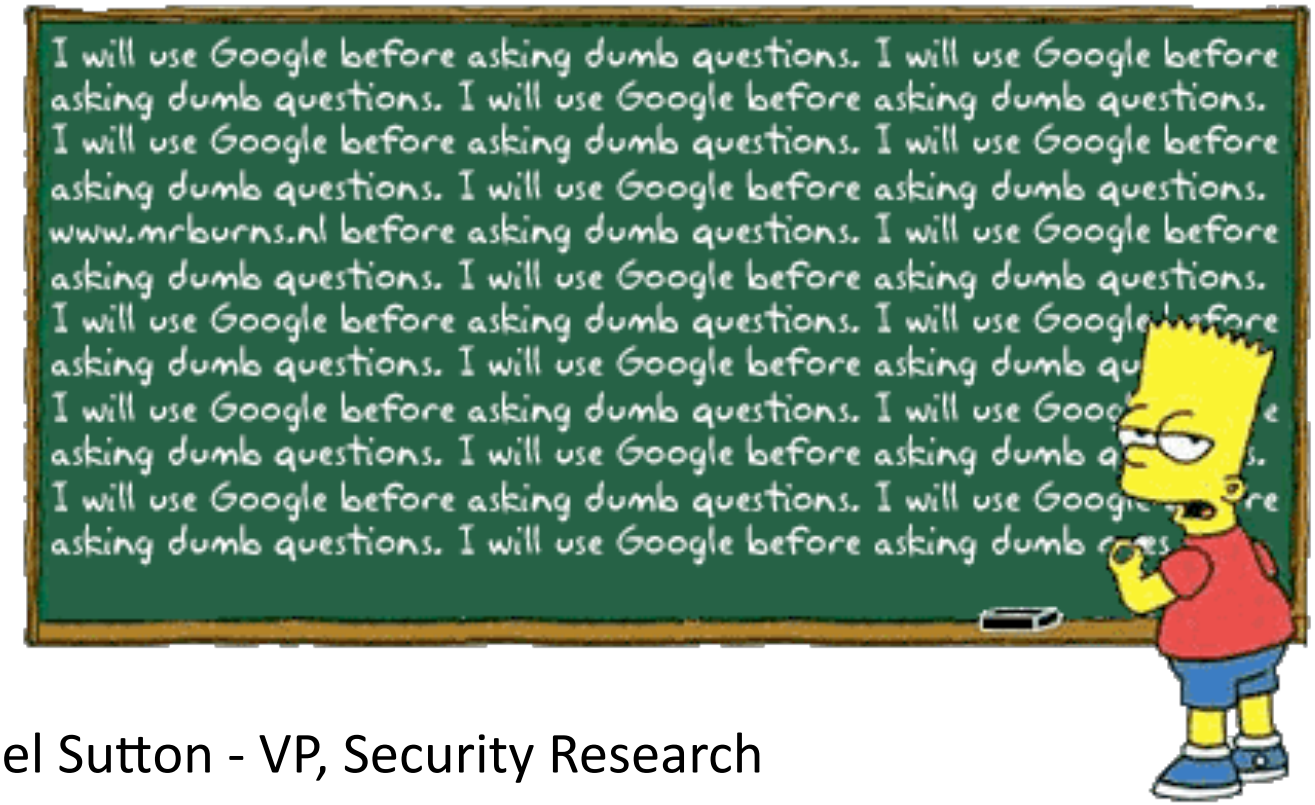
Vulnerable Sites

- Sites will continue to push the limits of widely adopted technologies such as HTTP cookies and Flash LSOs, resulting in exploitable vulnerabilities
- A significant portion of sites adopting local database technologies will have injection flaws that leave them open to attack

Attacks

- Attack prevalence will increase in proportion to adoption rates

Questions?



Michael Sutton - VP, Security Research

http://research.zscaler.com

Michael.Sutton@zscaler.com