



# The Malicious Web

Resurgence and emergence of browser exploits.

By Mike Shema <mshema@qualys.com>

Security Research Engineer, Qualys Inc.

April 23<sup>rd</sup>, 2008



# Agenda

- Highlight current state of web security
- Review the current state of browser security
- Examine some different web-related attacks.
- Identify current methods for protecting the browser and hardening web applications
- Highlight future areas of interest

# Web Security

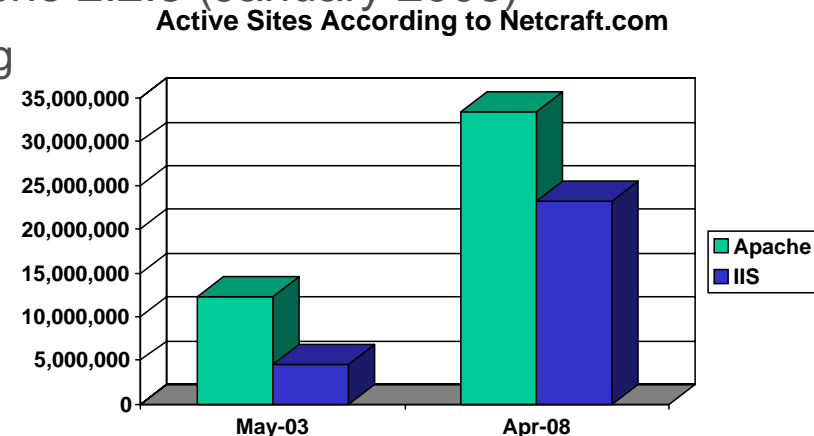
- Web application (in)security continues to grow.
  - Web-related vulnerabilities pop up on Bugtraq daily. (<http://www.securityfocus.com/bid/>)
  - Web-related attacks are large and expensive to investigate, react, and resolve.
  - Web security becomes a requirement of PCI in 2008.
- Common focus on threats to web applications.
  - OWASP Top 10
  - WASC Threat Classification
- What is a web application attack?
  - SQL injection, cross site scripting, request a link, deliver some HTML, visit a web page, post a comment

# Web Security

- Web security expands its grasp
  - Attackers target large properties: MySpace, Google, Yahoo!
  - Researchers target application engines: Month of PHP bugs, uninitialized variables in Flash applications.
  - Exploits target browsers: malicious JavaScript
- XSS remains a significant problem.
  - Original CERT advisory February 2000 (<http://www.cert.org/advisories/CA-2000-02.html>)
  - USENET references to “malicious html” and “malicious javascript” as far back as 1996
    - comp.security.unix post on March 1996: <http://tinyurl.com/2s593m>
    - Entertaining discussion of JavaScript: <http://tinyurl.com/2g2476>

# Web Security

- *Reported web server vulnerabilities have decreased.*
  - IIS 6.0 released April 2003
    - MS06-034 (specially-crafted ASP file could cause buffer overflow)
    - No resurgence of Code Red or Nimda style vulnerabilities
  - Apache 2.0.45 (March 2003) to Apache 2.0.63 (January 2008)
    - 40 security bugs according to changelog
    - 24 specific to core or mod\_ssl
  - Apache 2.2.0 (November 2005) to Apache 2.2.8 (January 2008)
    - 13 security bugs according to changelog
    - 2 specific to core or mod\_ssl
- And the number of servers continues to grow significantly.



# Desktop & Browser Security

- Desktop applications moving to the web faster than desktop security.
- Strong bulwarks between the browser and the filesystem.
- Same Origin Rule intended to separate data access within the browser.
  - Code within a domain is executed with the assumption of trust.
- Browsers present forensic challenges.
- Much easier to write HTML & JavaScript than to craft a reliable buffer overflow.

# A Brief History of Desktop Security

- User and file privileges
- Event logging
- Signature-based virus detection
- Host-based firewall
- Behavioral-based virus detection
- Host-based intrusion detection system
- NAC / NAP (don't let the device onto the network unless it's secure)

# A Brief History of Browser Security

- Same Origin Rule
- Restrict Java applet access to the localhost
- Block access to specific ports (some browsers)
  - ...because of security concerns
- Block pop-ups
  - ...because of obnoxious advertising
- Block third-party cookies
  - ...because of advertising and privacy concerns
- Block web bugs (1x1 images, etc.)
  - ...because of advertising and privacy concerns
- Compare URLs with known phishing sites
  - Send your complete browsing behavior to a third-party...raises privacy concerns
- Security plugins (NoScript, etc.)

# Threats Evolve

- **Financial motivation**
  - More information with value
  - Game accounts, Ebay accounts, bank accounts
- **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
- **Increased potential for targeted attacks**
  - Exploit a victim's social network
  - Exploit a site's business logic
- **Exploit the trust between the server and browser**
  - Thrive on the increase in user-generated content

# Attacks Adapt

- Bring the exploit to victim rather than bring the victim to the exploit.
- “Web 2.0”: More business logic and capabilities moved to the web browser.
  - Consolidation of personal, business, or financial data into a web application.
- Social networking as an enabler for non-technical attacks.
  - ID theft, bullying, fraud, graft, libel
- Target the web browser
  - Cross-platform uniform exploit environment

# 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
- **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!

# Persistent Browser Problems

- Assumption of trust in HTML and JavaScript (no “signed” content)
  - SSL enables trust of identity, but not content (or intent).
- No separation of UI generation and data manipulation
  - JavaScript can affect all aspects of DOM
  - Leads to exploits like XSS, phishing, social engineering
  - Coarse data access controls (yes/no)
- Few restrictions on pulling together inter-domain content, no “trusted peers” for a domain.
  - Some exceptions for images and cookies, due to spam and advertisers
  - DNS load balancing, third-party content servers, open nature of the web make this an extremely difficult problem.

# Browser Engine Attack Techniques

- Same Origin Defeat
  - Break the domain access restrictions
- HTML parsing idiosyncrasy
  - E.g. SAMY, Firefox unclosed `<script>`
- DOM injection
  - Create, modify, delete elements
  - e.g. create `<img>` elements to determine live hosts/ports, modify event handlers to insert keylogger
- Namespace infection
  - Inject JavaScript into the page, unaffected by SOR
  - e.g. modify serialized values, affect application logic

# Browser Engine Attack Techniques (cont'd)

- Browser instrumentation (Cross-Site Request Forgery)
  - Inject HTML or JavaScript that conducts pre-packaged requests to a third party.
  - Doesn't have to bypass SOR
- Information leakage via inference
  - Timing (e.g. DNS resolution, response time for <img> elements to determine live hosts/ports)
  - Content inspection (e.g. access font colors to determine browser history)
- Ambiguous content-type
  - JavaScript inside PDF, file metadata
- Insecure plug-in
  - Buffer overflow
  - Uninitialized variables in Flash applications.  
(<http://tinyurl.com/6bbh2h>)

# What do these attacks look like?

- Review some examples to see where vulnerabilities exist and how they are exploited.

# Recent Examples

- **Graft & Gullibility**
  - How to collect usernames and passwords.

# Recent Examples

- **Session Fixation & Stock Inflation**
  - Buy stocks using someone else's account.

# Session Fixation & CSRF

- Victim receives an e-mail with a legitimate link to the trading site:  
<https://site/login.cgi?sid=65531>

**x.y.72.13** --> /trade.cgi?sid=655321&shares=1000&stock=FOO

Redirect to /login.cgi <-- server

**x.y.72.13** --> /trade.cgi?sid=655321&shares=1000&stock=FOO

Redirect to /login.cgi <-- server

**x.y.72.13** --> /trade.cgi?sid=655321&shares=1000&stock=FOO

Redirect to /login.cgi <-- server

**a.b.101.92** --> /login.cgi?sid=655321

Redirect to /welcome.cgi?sid=655321 <-- server

**x.y.72.13** --> /trade.cgi?sid=655321&shares=1000&stock=FOO

Trade executed <-- server

# Recent Examples

- **Filters & Forwarding**
  - Backdoor a web-based e-mail account.

# Recent Examples

- Rentals & Recognition
  - Correlate datasets to de-anonymize users.

# Wildly Different Vulnerabilities

- Greed
- Session fixation
- Cross-site request forgery
- Lack of input validation
- Manipulation of business logic
- Data fingerprints
- Insufficient anonymity

# Where Are The Worms?

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

# Where Are The Worms?

- Attacks like Nimda, Code Red or SQL Slammer haven't been repeated in a while.
  - Inefficient in terms of time and benefit for a financial purpose.
- Exploit preferences seem to fall to the lowest common denominator.
  - Financially successful attacks don't require sophistication.
  - Infecting a single popular, trusted site with XSS (or a virus, etc.) ensures that the payload will spread to many victims.

# May Be More Likely To See

- Thick-client bots
- Anti-analysis
  - Code obfuscation
  - De-obfuscation traps
- Anti-detection
  - Detect browsers vs. automated tools
  - Geo-based content
  - Identify browsers in virtualized systems (e.g. HoneyMonkey project)

# Proactive Countermeasures

- Prevent the initial compromise
- Web application hardening
  - Prevent unexpected HTML injection
  - Identify areas where user-generated content is permitted
    - Pre-inspect content
    - Quarantine content
    - Enforce content-type interpretation
- Continuous monitoring of the site for infection.
- 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 only detects a subset of known Trojans, exploits
  - 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.**
- **Disable or restrict capabilities**
  - Firefox NoScript

# Upcoming Areas of Interest

- The uneasy relationship between web application capabilities and browser security ensures a profitable future.
- HTML5 Client-side database storage
  - More user-data accessible to malicious scripts, XSS, CSRF
- HTML5 Cross-document messaging
  - Intentional relaxation of the Same Origin Rule
- IE8 XDomainRequest()
  - Asynchronous cross-domain requests
- The network becomes the computer (again)
  - Vulnerabilities without borders

# Summary

- The web browser continues to bear more and more functionality that used to be relegated to desktop applications -- but the browser security model hasn't kept pace.
- Attackers placing more focus on compromising trusted sites rather than lure victims to fake sites.
- Social networking, Web 2.0, and similar concepts place more and more personal data only a browser request away.

Thank you!

# Questions

