

Sunera LLC

Web Application Security Management & Best Practices Part II

Agenda

- Frequently Encountered Vulnerabilities
- Frequently Misunderstood Mitigations
- Tools for Testing Web Applications
- Resources

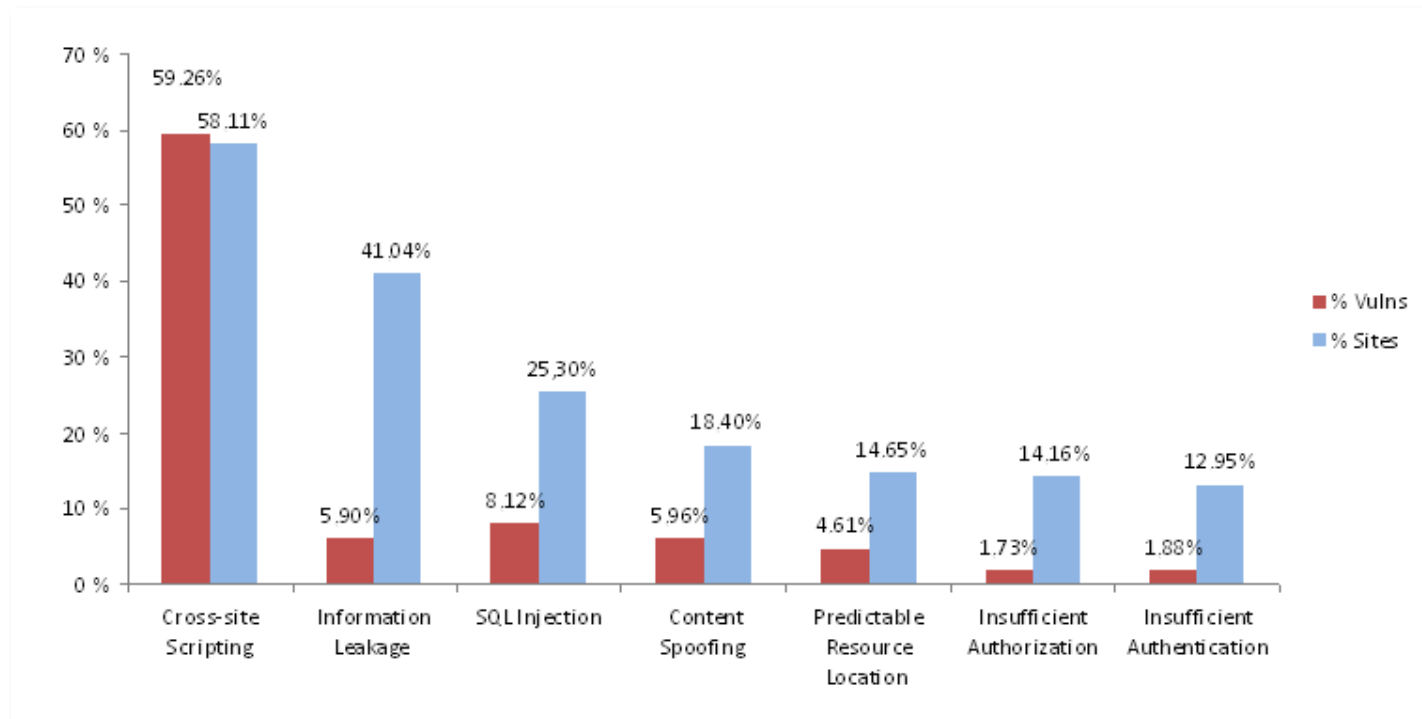
Frequently Encountered Vulnerabilities

Frequently Encountered Vulnerabilities

- Cross-Site Scripting
- Failure to Restrict URL Access
- Misuse of HTTP POST / GET Methods
- Improper Error Handling

Cross-Site Scripting

- Cross-Site Scripting of varying impacts is found in an overwhelming majority of penetration tests



¹ <http://www.webappsec.org/projects/statistics/>

Cross-Site Scripting

- Cross-Site Scripting attacks are limited by the point of reflection, the application's attack surface, and the creativity and skill of the attacker
 - Attacks can be bi-directional, so instances of XSS on “internal-only” applications are not exempt from proper remediation
- End result – Cross-Site Scripting and input injection in any form is a dangerous vulnerability to leave unmitigated
 - There is no “acceptable” input injection

Failure to Restrict URL Access

- URL Access should be restricted in order to prevent users with malicious intent from directly enumerating protected content
- Lessons Learned:
 - Ensure effective data categorization to avoid mistakenly publishing sensitive content
 - Protect resources with strong authentication and session management mechanisms
 - Don't rely on security through obscurity to protect resources

Misusing HTTP POST and GET

- Use POST to submit forms
- Use GET to access resources
 - NEVER use GET to authenticate users as it leaves a residual trace of all users in the web server access logs (not to mention web application proxy tools)

Improper Error Handling

- While often used to assess web sites (especially in the case of Mozilla Firefox) , malicious users are not limited to using web browsers to attack web sites
- A web application proxy is commonly used to test web applications
 - Manually inspecting all HTTP Requests / Responses sometimes reveals hidden error conditions that are not exposed to the browser
 - Commonly occurs when testing AJAX applications, as the presentation layer may ignore improperly formatted, incomplete, or erroneous JSON datasets¹
 - Allows for manual inspection and modification of HTTP Requests to circumvent any rudimentary client-side JavaScript input validation
- Generic and consistent error handling is the best approach

¹ Many examples can be found in Billy Hoffman and Bryan Sullivan's book "Ajax Security" published by Addison-Wesley

Frequently Misunderstood Mitigations

Frequently Misunderstood Mitigations

- Input Validation
 - Confusion with input validation routines
- Stored Procedures
 - Poor query construction without datatype safety
- Error Handling
 - Visible error conditions being handled, hidden conditions still present, especially in Web 2.0 Applications
- Secure Sockets Layer
 - SSLv2 and Weak Ciphers commonly found

Input Validation and Output Encoding

- Input Injection (Cross-Site Scripting / SQL Injection)
 - Blacklist vs. Whitelist Input Validation
 - Attempting to only filter out dangerous characters instead of only accepting a pre-determined pattern
 - Commonly encountered examples include:
 - » User supplied form data from web applications
 - » Blogs where only a subset of HTML tags are allowed
 - » HTML text areas for web-based email applications
 - Incomplete HTML output encoding and parameter filtering (XSS Only)

Input Injection and Regular Expressions

- Regular Expressions or “Regex’s” are a specialized language used to describe textual patterns
- Many, if not all, programming languages have various API’s used to support Regex’s
- Allows for matching characters based on classes or literals
- Allows decision logic on string matches, substitution/replacement, and frequency of occurrence of specific characters
- Widely understood as terse and difficult, but tools are available to simplify this learning curve
- Definitive Guide: “Mastering Regular Expressions” by Jeffrey Freidl, published by O’Reilly

Zip Code Regex

`^\d{5}(?:-\d{4})?$`

Zip Code Regex: Step 1

$\wedge \backslash d \{ 5 \} (? : - \backslash d \{ 4 \}) ? \$$

- \wedge match at the beginning of the string, ensuring no text is prepended to our string

Zip Code Regex: Step 2

`^\d{5}(?:-\d{4})?$`

- `^` Match at the beginning of the string, ensuring no text is prepended to our string
- `\d{5}` Match exactly 5 numeric characters

Zip Code Regex: Step 3

`^\d{5}(?:-\d{4})?$`

- `^` Match at the beginning of the string , ensuring no text is prepended to our string
- `\d{5}` Match exactly 5 numeric characters
- `(?:-\d{4})?` Make the contents grouped within the parenthesis, a literal '-' with exactly 4 numeric characters, 'optional' to add flexibility to successfully match both short and extended postal codes

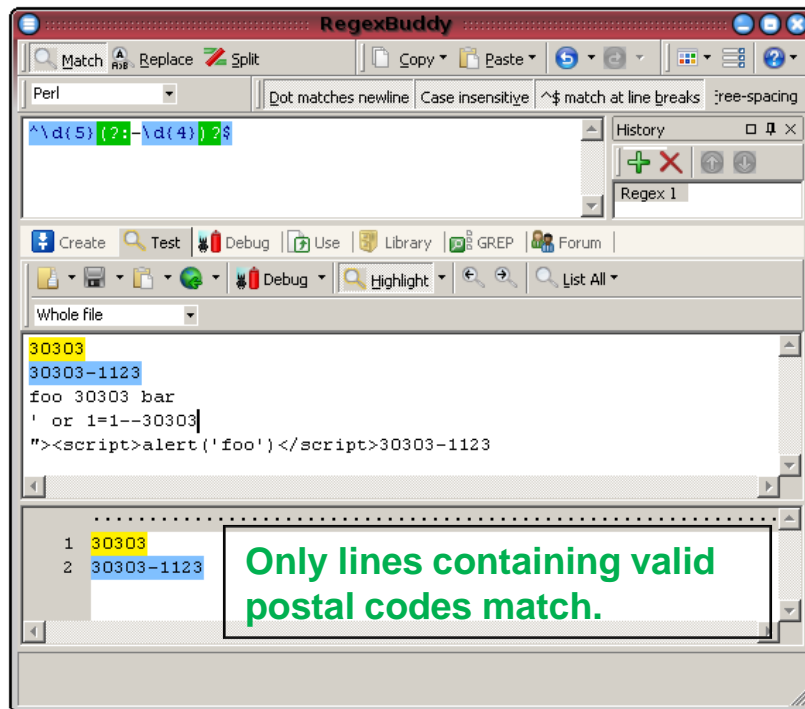
Zip Code Regex: Step 4

`^\d{5}(?:-\d{4})?$`

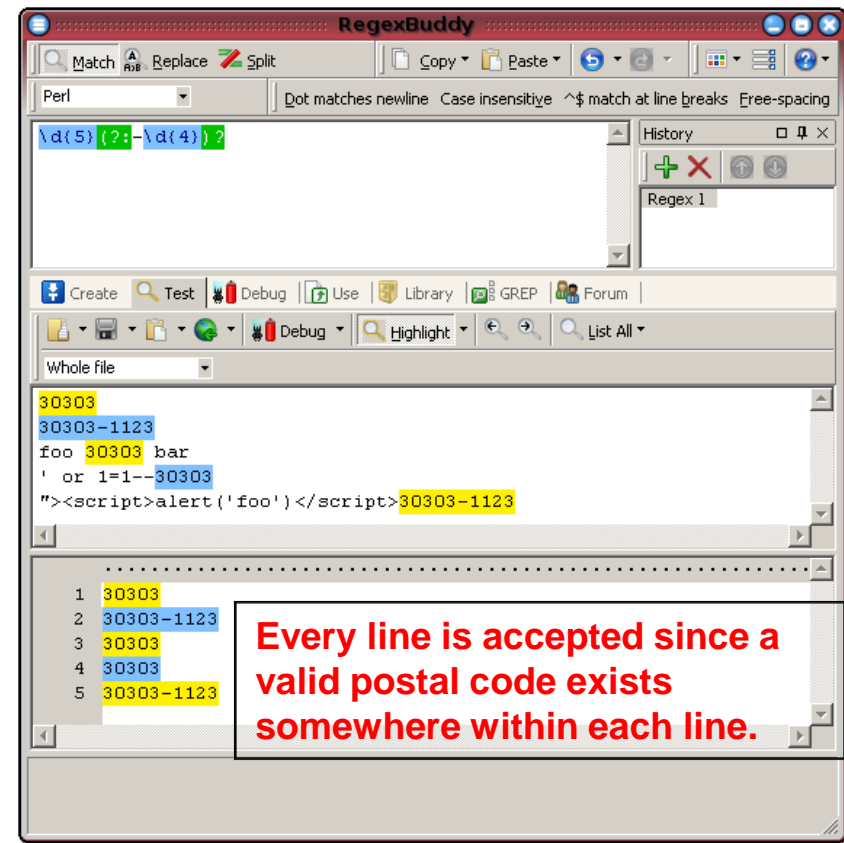
- `^` Match at the beginning of the string, ensuring no text is prepended to our string
- `\d{5}` Match exactly 5 numeric characters
- `(?:-\d{4})?` Make the contents grouped within the parenthesis, a literal '-' with exactly 4 numeric characters, 'optional' to add flexibility to successfully match both short and extended postal codes
- `$` Terminate our match to ensure no trailing characters are present

Zip Code Regex: Step 5

`^\d{5}(?:-\d{4})?$`



`\d{5}(?:-\d{4})?`



Stored Procedures and String Concatenation

- Preventing SQL Injection requires a “Defense in Depth” Strategy
 - 1st layer: Whitelist input validation coupled with Blacklist input validation for format exceptions
 - 2nd layer: Capabilities based access controls to perform SQL queries
 - 3rd layer: Parameterized Queries enforcing Datatype Safety
- Avoid using string concatenation!

String Concatenation

■ Bad: String Concatenation

```
String sql = "select * from users where userid="+nUserId;
```

■ Good: Parameterized Queries

```
SqlCommand cmd = new SqlCommand(
    "select * from users where userid = @UserId"
    , conn
);
SqlParameter param = new SqlParameter(
    "@UserId", userId, SqlDbType.Int
);
cmd.Parameters.Add(param);
```

Appropriate Blacklist Validation

- What if an apostrophe is required?
 - Subsequent validation can be performed that further restricts the use of the 'single-tick' character to make sure there is only one instance or the input string matches a specific character pattern (i.e. Regular Expressions)
- There are ways to make exceptions to input validation without failing open.
 - Blacklist validation should never be used alone, but using blacklist validation to further restrict content after whitelist input validation is acceptable

Secure Sockets Layer

■ Secure Sockets Layer

- Proper SSL Protocol/Cipher use requires registry tweaks, not merely by checking “Require 128-bit SSL” in IIS Manager
- Explicitly disable <128 bit cipher strengths in the registry
- Explicitly disable SSLv2 Protocol in the registry

■ Sample Command line for quick testing:

- `openssl s_client -connect 192.168.1.1:443 -cipher [LOW | MEDIUM | HIGH]`
- `openssl s_client -connect 192.168.1.1:443 -ssl2`
- Protocol and Cipher queries can be combined
- If successful, then selected ciphers/protocols are implemented

Tools For Testing Web Applications

First Tool: Knowledge

- The more you know about the following subjects the better you will be at performing web application security assessments
 - Web Languages
 - Perl / Python / java web languages (JSP/Servlets) / JavaScript / PHP / ASP.Net
 - Markup / Presentation Language
 - HTML / XML / Cascading Stylesheets
 - RFC2616, HTTP Specification
 - Naturally inquisitive, patient, and logic-oriented

Tools For Testing Web Applications

■ Automated Scanners

– Commercial

- BurpScan (part of BurpSuite Pro)
- Cenzic Hailstorm
- HP WebInspect
- IBM AppScan
- WhiteHat Sentinel

– Freeware

- GrendelScan
- Nikto, Wapiti

Tools For Testing Web Applications

■ Source Code Analyzers

- Fortify Software
- Ounce Labs

■ Mozilla Firefox (with Addons)

- FireBug
- Cookie Editor
- Tamper Data
- Switch Proxy
- Web Developer Toolbar
- Greasemonkey

Tools For Testing Web Applications

■ Fuzzers

- Burp Intruder
- Webscarab
- SPIKE

■ Decompilers

- Jad – Java Decompiler
- Flare / Flasm – Flash object decompiler/recompiler
- Reflector - .Net managed code disassembler

Tools for Testing Web Applications

■ Independent “Purpose-Driven” Tools

- THCSSLCheck
- Server Analyzer / SQL Injector (Part of HP WebInspect security toolkit)
- ‘openssl’ command with the ‘-cipher <strength>’ argument
- DirBuster, SQLMap, Hydra, Nikto, netcat

■ Web Application Proxies

- BurpSuite Pro*
- Fiddler
- Paros*
- WebScarab*

Recommended Reading

1. Stuttard, Dafydd and Marcus Pinto. **The Web Application Hackers Handbook: Discovering and Exploiting Security Flaws**. Indianapolis: Wiley, 2008
2. Gourley, David and Totty, Brian. **HTTP: The Definitive Guide**. Sebastopol: O'Reilly & Associates, Inc. 2002
3. Freidl, Jeffrey. **Mastering Regular Expressions: Second Edition**. Sebastopol: O'Reilly & Associates, Inc. 2002
4. Long, Johnny. **Google Hacking for Penetration Testers**. Rockland: Syngress Publishing, Inc. 2005
5. Hoffman, Billy and Sullivan, Bryan. **AJAX Security**. Boston: Pearson Education, Inc. 2008
6. Scambray, Joel and Shema, Mike and Sima, Caleb. **Hacking Web Applications Exposed: Second Edition**. San Francisco: The McGraw-Hill Companies. 2006
7. Grossman, Jeremiah and Hansen, Robert and Petkov, Petko and Rager, Anton and Fogie, Seth. **XSS Attacks: Cross Site Scripting Exploits and Defense**. Burlington: Syngress Publishing, Inc. 2007
8. Litchfield, David and Anley, Chris and Heasman, John and Grindlay, Bill. **The Database Hacker's Handbook**. Indianapolis: Wiley, 2005
9. Flanagan, David. **JavaScript: The Definitive Guide**. Sebastopol: O'Reilly & Associates, Inc. 2006
10. Hemenway, Kevin and Calishain, Tara. **Spidering Hacks**. Sebastopol: O'Reilly & Associates, Inc. 2004

Recommended Websites


- ZScaler Research
 - <http://research.zscaler.com>
- Open Web Application Security Project
 - <http://www.owasp.org>
- Web Application Security Consortium
 - <http://www.webappsec.org>
- Jeremiah Grossman
 - <http://jeremiahgrossman.blogspot.com/>
- GNU Citizen
 - <http://www.gnucitizen.org>
- Dark Reading
 - <http://www.darkreading.com/>

Contact Information

- For additional information on Sunera's security services, visit our website at www.sunera.com
- Or contact the following Sunera representatives:

Andrew Cannata, CISSP, CISM
Managing Director
Information Security & Network Services
acannata@sunera.com

Joe Sechman, CISSP, CISA
Manager
Information Security & Network Services
jsechman@sunera.com



Sunera South Florida Office Address:
3350 SW 148th Avenue, Suite 210
Miramar, FL 33027