

Injection Flaws Overview

Where Are We? What Are We Talking About?



What Are We Talking About?

- Injection flaws Weakness in an application whereby foreign input subverts the otherwise legitimate use of a subsystem.
- Different subsystems == different flaws
 - □ RDBMS: SQL Injection
 - □ Web browser: Cross-Site Scripting (XSS)
 - □ OS Shell: Command Injection
 - ☐ "Self-same" subsystems (SSI injection, evil eval)
- Our focus: two most prevalent external subsystem injection attacks



SQL Injection

SQL Injection is an attack technique used to exploit web sites that construct SQL statements from user-supplied input.

- Web Application Security Consortium

http://www.webappsec.org/projects/threat/classes/sgl_injectjon.shtml

- Long history
 - □ 1998 debut (Phrack 54)
 - □ In bloom in late 2001 Stream of advisories: QDAV-2001-7-2, RUS-CERT 2001-08:01, RUS-CERT 2001-09:01, ...
- Diverse impact
 - □ Data C.I.A. attacks, application subversion
 - Server compromise: remote command execution, Java class uploads, stored procedure overflows, &c.



XSS (a.k.a. HTML Injection)

Cross-site scripting allows a bad guy to trick an innocent guy into running code the bad guy wrote.

- Lincoln Stein quoted in "Cross-site scripting' tears holes in Net Security," <u>USA</u> Today. 30 Aug 2001.

- On all radars in late 1999.
 - ☐ Famed CERT advisory in Feb. 2000
 - □ References back to 1997
- Low-hanging fruit
 - □ Passé, damning or both?
 - □ Attacks the mediation between app and end-user



Conventional Wisdom

- Injection flaws are "Input Validation Errors"
 - ☐ Commonsense, pervasive taxonomy
 - SecurityFocus BID classifications
 - Language of vendor advisoriese.g., CAN-2005-0040, CAN-2005-1525 advisories
 - □ Old, familiar refrain: *Validate your input!*
- Emphasizes incoming trust boundary Protect the (application) perimeter
- Let's look more closely at validation

Rethinking Input Validation



Validation Missteps (1/2)

- Undesirable dataset restrictions
 - □ "DOMAIN\username" "What if x' < y?"
 - □ Predictable debate: whitelist or blacklist?
- Regex rex
 - □ Regexes get complicated quickly
 - □ Cargo-cult practices
- Validating the wrong thing at the wrong time
 - □ Validation != canonicalization http://doof.us/vuln.cgi?input=%3Cscript%3E...
 - ☐ Data type constraints v. output metacharacters



Validation Missteps (2/2)

- Pre-encoded attack
 - □ XSS Filter Bypass

e.g.,

- □ RDBMS Encoding Schemes
 - Quoting/escaping conventions
 - Hex literals and interpolated variables
- ☐ Attackers know your output better than you do!
- Lacking precognition
 - □ Deferred, second-order attacks
 - □ Output subsystem variety, changes
 - □ Input sources change: \$_REQUEST or \$_GET ?



Validation In Practice

- Often honored in breach
 - □ Validation is inspectional
 - ☐ Few have patience, error-handling
 - □ What then? Mogrify!
 - "The best practice is to strip special characters..."
 - "3 models for data validation: accept, reject, sanitize."
 - "s/the_bad_stuff//;"
- Client-side practices migrated server-side
 - □ A step forward
 - □ Focused on obvious trust boundary, attacker-facing
 - ☐ Divorced from vulnerable subsystems
- Conflation of terms: stripping, sanitizing, filtering, validating



Turning It Around...

- Not input, but output Last chance to defend.
- Not validation, but presentation Form of output, rather than content.
- Desirable characteristics
 - ☐ Discrete, tactical coding practices
 - □ Opaque data handling
 - □ API abstraction conceals mechanics
 - □ Remember format string vulnerabilities?



Defense Techniques

Lessons Learnt From Layer 6



SQL Injection: Bind Variables

- Related terms: placeholders, prepared statements, precompiled query plans, parameterized queries
- Proper syntactical interpolation:

```
SELECT secret FROM tbl WHERE user=?;
```

- '?' becomes safe SQL scalar, whether it is "jdoe" or "' OR 1=1"
- Variations and Limitations
- Database-independent APIs preferred



Bind Variables: JDBC

■ JDBC 3.0

http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/
PreparedStatement p =
 con.prepareStatement("SELECT secret FROM
tbl WHERE user=? AND passhash=?");
p.setString(1, username);
p.setString(2, hashedpass);

Allows named parameters for CallableStatement (stored procedure) execution.



Bind Variables: ODBC

ODBC

http://msdn.microsoft.com/
library/en-us/odbc/htm/odbcabout_this_manual.asp

```
r = SQLPrepare(hstmt, "SELECT * FROM tbl WHERE user=? AND passhash=?", SQL_NTS); SQLEindParameter(hstmt, 1, SQL_PARAM_INPUT, SQL_C_CHAR, USERNAME_LEN, 0, szUser, 0, &cbUser); SQLEindParameter(hstmt, 2, SQL_PARAM_INPUT, SQL_C_CHAR, HASHPASS_LEN, 0, szPass, 0, &cbPass); SQLExecute(hstmt);
```

Other ODBC interfaces more tolerable.



Bind Variables: Perl

■ Perl DBI

http://dbi.perl.org

```
my $stmt = $dbh->prepare("SELECT secret
FROM tbl WHERE user=? AND passhash=?");
$stmt->execute($username, $hashedpass);
```

■ Individual drivers may expose named placeholder functionality (e.g., DBD::Pg).



Bind Variables: PHP

PFAR DB API

http://pear.php.net/package/DB

```
$sth = $db->prepare("SELECT * FROM tbl
WHERE user=? AND passhash=?");
$params = array($username, $hashed);
$sth->execute($params);
```

ADOdb

http://adodb.sourceforge.net/ Well-regarded 3rd-party abstraction library.



Bind Variables: Python

■ DB API 2.0

http://www.python.org/peps/pep-0249.htm

```
userInp = { 'user':..., 'hashed':...}
c = dbconn.cursor()
c.execute('SELECT * FROM tbl WHERE
user=%(user)s AND passhash=%(hashed)s',
userInput)
```

 Specification allows multiple styles of parameter markers.



XSS: Output Encoding (1/2)

■ Simple Example

Double Choco Latte BID 12894

```
diff -ruN dcl-0.9.4.3/inc/functions.inc.php \
    dcl-0.9.4.4/inc/functions.inc.php
--- dcl-0.9.4.3/inc/functions.inc.php 2005-03-23 19:38:12 -0600
+++ dcl-0.9.4.4/inc/functions.inc.php 2005-03-23 22:02:50 -0600
@@ -173,14 +173,14 @@
    import($class);
    if (!class_exists($class))
    {
        print('Invoke could not find class: ' . $class);
        print('Invoke could not find class: ' . htmlspecialchars($class));
        return;
    }
}
```



XSS: Output Encoding (1/2)

- Long-known, little advocated
- Encode just prior to output
 - □ Everything and everywhere
 - ☐ Fix Content-Type, double-quote attributes
- Templating system URI and HTML encoding.
- Remarks on HTML-permissive CMS



Concluding Remarks

- Bind variables; output encoding
- Bake into coding guidelines, QA processes and documentation
- Bind variables; output encoding
- Identify trust boundaries
- Bind variables; output encoding
- Validate your input!
 - □ Be precise in terminology
 - □ What is being validated, where and why?



Questions?

After the conference:

mjp-bh05@securepipe.com



Selected Materials (1/2) – Caveat lector!

- Anley, Chris. "Advanced SQL Injection In SQL Server Applications." NGSSoftware Insight Security Research papers. Jan 2002.
- Anley, Chris. "(more) Advanced SQL Injection." NGSSoftware Insight Security Research papers. Jun 2002.
- CERT/CC. "CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests." Advisory CA-2000-02. 3 Feb 2000.
- CERT/CC. "Understanding Malicious Content Mitigation for Web Developers." CERT Tech Tips. 2 Feb 2000.
- Cgisecurity.com. "The Cross Site Scripting FAQ." Cgisecurity.com article. Aug 2003.
- Clover, Andrew. "Re: GOBBLES SECURITY ADVISORY #33." Bugtraq mailing list post. 11 May 2002. Message-ID <20020511150446.A2580@doxdesk.com>
- Endler, David. "The Evolution of Cross-Site Scripting Attacks." iDEFENSE, Inc. whitepaper. 20 May 2002.
- Foster, James C. "Defense tactics for SQL injection attacks." SearchSecurity.com Tips & Newsletters, Network Security Tactics. 21 Mar 2005.
- Friedl, Stephen J. "SQL Injection Attacks By Example." Unixwiz.net Tech Tips. Last Mod. 13 Jan 2005.
- GOBBLES "GOBBLES SECURITY ADVISORY #33." Bugtraq mailing list post. Mod./Fwd. Dave Ahmad. 11 May 2002. Message-ID <Pine.LNX.4.43.0205100832290.18396-100000@mail.securityfocus.com>



Selected Materials (2/2) - Caveat lector!

- Hu, Deyu. "Preventing Cross-Site Scripting Vulnerability." GIAC GSEC Practical. 4 May 2004.
- Huseby, Sverre H. "Incompatible Parameter Parsing." Huseby technical paper. 29 Apr 2005.
- informIT. "Unexpected Input." informIT <u>Security Reference Guide</u>, "Web Application Security" chapter. 26 May 2005.
- Lee, Paul S. "Cross-site scripting: Use a custom tag library to encode dynamic content." IBM developerWorks. 1 Sep 2002.
- Microsoft Corporation. "Web Security Threats and Countermeasures." Microsoft Patterns and Practices Collection. Jan 2004.
- Monsch, Jan P. et. al. "Re: ISA Server and SQL Injection." WebApp Sec mailing list. 3 Mar 2005.
 Archived at: http://www.networksecurityarchive.org/cgi-bin/mesg.cgi?a=Web-App-
 - Ollmann, Gunter. "Second-order Code Injection Attacks." NGSSoftware Insight Security Research papers. Nov 2004.
- OWASP. "A Guide to Building Secure Web Applications and Web Services." The Open Web Application Security Project. 6 Jun 2005.
- Rain Forest Puppy. "NT Web Technology Vulnerabilities." <u>Phrack Magazine</u> Vol. 54. 25 Dec 1998.
- Spett, Kevin. "SQL Injection: Are your Web applications vulnerable?" SPI Dynamics, Inc. whitepaper. 2002.