Epic Facepalm

Spectacular Appsec Failures Since About This Time Last Year
Will
John
Overview
SQL Injection
Sony Woes Continue With SQL Injection Attacks
IT Security & Network Security News

Nokia Shuts Down Forums After SQL Injection Exposes Developer Info
SQL injection blamed for widespread DNS hack
SELECT *
FROM xaction
WHERE payee = 'Cyber Security Symposium'
ORDER BY paid_on DESC;
String payee = request.getParameter("payee");
String sql = "SELECT * 
  + "FROM xaction 
  + "WHERE PAYEE = ' 
  + payee 
  + "' ORDER BY paid_on DESC;";
rs = stmt.executeQuery(sql);
String payee = request.getParameter("payee");
String sql = "SELECT * "
    + "FROM xaction "
    + "WHERE PAYEE = '"
    + payee
    + "' ORDER BY paid_on DESC;"
rs = stmt.executeQuery(sql);
String payee = request.getParameter("payee");
String sql = "SELECT * "
+ "FROM xaction "
+ "WHERE PAYEE = '"" + payee + "' ORDER BY paid_on DESC;"
rs = stmt.executeQuery(sql);
String payee = request.getParameter("payee");
String sql = "SELECT * 
      + "FROM xaction " 
      + "WHERE PAYEE = '" 
      + payee 
      + '' ORDER BY paid_on DESC;"
rs = stmt.executeQuery(sql);
String payee = request.getParameter("payee");
String sql = "SELECT * 
+ "FROM xaction 
+ "WHERE PAYEE = '");
+ payee
+ " ORDER BY paid_on DESC;"
rs = stmt.executeQuery(sql);
String payee = request.getParameter("payee")
String sql = "SELECT * "
+ "FROM xaction "
+ "WHERE PAYEE = "
+ payee
+ " ORDER BY paid_on DESC;"
rs = stmt.executeQuery(sql);
Cyber' OR '1' = '1
SELECT *
FROM xaction
WHERE payee = 'Cyber' OR '1' = '1'
ORDER BY paid_on DESC;
String payee = request.getParameter("payee");
String sql = "SELECT * "
    + "FROM xaction "
    + "WHERE PAYEE = ?" 
    + ORDER BY paid_on DESC;"
stmt = new PreparedStatement(sql);
stmt.setString(1, payee);
rs = stmt.executeQuery();
String payee = request.getParameter("payee");
String sql = "SELECT * 
  + "FROM xaction 
  + "WHERE PAYEE =?" 
  + ORDER BY paid_on DESC;"
stmt = new PreparedStatement(sql);
stmt.setString(1, payee);
rs = stmt.executeQuery();
String payee = request.getParameter("payee");
String sql = "SELECT * "
  + "FROM xaction "
  + "WHERE PAYEE = ?"
  + " ORDER BY paid_on DESC;"
stmt = new PreparedStatement(sql);
stmt.setString(1, payee);
rs = stmt.executeQuery();
Parameterized Query

Notes

• Most languages have them in some form
• Placeholder text may vary
• PHP/MySQL, use MySQLi driver
• Even available in ADO
• Often improves performance
• Blacklist “validation” insufficient
• Some things take extra work:
  \WHERE payee LIKE '%\%' + ? +
• Use indirect reference for columns
About NoSQL

• Injection works exactly the same way
• Except you’ll use JSON, XPATH, etc.
• Problem with mixing control channel and data channel
• Welcome back to 1971
Direct Object Reference

Turned to Horizontal Privilege Escalation
Citigroup hack exploited easy-to-detect web flaw
Brute force attack exposes 200,000 accounts
Direct Object Reference

- Object ID (primary key value, etc.) passed to UI to refer back to that object
- Browser in control of the value
- User not checked for “entitlement” to that object ID

/view_account?account_id=1234

SELECT *
FROM account
WHERE account_id = ?; -- NOTE: NO SQLI
Preventing Direct Object Reference

- Check “entitlement” along with the query that makes the request

```
SELECT * FROM account WHERE account_id = ? AND owner_name = ?
```

(get owner_name from session, NOT cookie)

- Check “entitlement” with a separate query

- Don’t send direct objects - use indirect ones
Indirect Object Reference

- // When rendering the list of accounts the user can manipulate
  session.setAttribute("ACCOUNTS", array_of_accounts);

  // When the user posts back
  id = request.getParameter("id");
  accts = session.getAttribute("ACCOUNTS");
  acct = accts.get(id);
Indirect Object Reference: Error Checking

- XSRF checking since all URL’s have predictable ID’s (1, 2, 3, etc.) - could be handled by using XSRF-token style IOR’s

- Type checking on incoming ID (checking it’s an integer or properly formatted XSRF token)

- Ensure the incoming ID is within the bounds of the list of objects

- Ensure the list of objects is in the session already
Clickjacking
Facebook "clickjacking" spreads across site

New Facebook Clickjacking Attack Uses Justin Bieber as Bait [WARNING]

June 02, 2010 by Christina Warren

172
Clickjacking

• Target site can be loaded in an iframe
• Target site has something that requires clicking
• Attacker uses a “lure” on their site
• Positions real target immediately above lure
• Makes iframe invisible with higher z-index
• User thinks they’re clicking the lure, but they’re clicking the target (with their cookies, etc.)
Clickjacking Uses

- XSRF token evasion
- “Like” button on Facebook
- Useful for spreading malware
- Enabling the camera
- Ad click fraud
Clickjacking Code

```javascript
vuln {
  position: absolute;
  top: 43 px;
  left: 97 px;
  opacity: 0.0;
  filter: alpha(opacity=0);
  z-index: 100;
}
...
<iframe src="http://facebook.com/comment.php" class="vuln"></iframe>
```
Preventing Clickjacking

• X-FRAME OPTIONS
  • Most newer browsers
  • SAMEORIGIN or DENY
  • Easier than frame busting
  • Can be done in META in HEAD

• Framebusting
  • Lots of ways around it
  • Best approach for now is at
Wrapping Up

• SQL Injection in 2004 OWASP Top 10 (A6)

• Direct Object Reference in 2004 OWASP Top 10 (A2)

• Clickjacking - you’re welcome

• All of these can be found with
  • Basic webapp hacking
  • First two with static analysis (third, with standards and customization)
Thanks