Web Application Attacks
What can an attacker do and just how hard is it?

By Damon P. Cortesi <damon.cortesi@ioactive.com>
IOActive, Inc.
Comprehensive Computer Security Services
www.ioactive.com
cortesi:~ damon$ whoami

- Systems and Security Administrator (Windows, Active Directory, Linux)
- Penetration Tester
- Source Code Reviewer
- PCI Assessor
- Breaker of Web Applications
- Destroyer of Quad Dual-Core Servers
Web Application Attacks

• SQL Injection
• Blind SQL Injection
• Authorization Bypass (Parameter Manipulation)
• Cross-Site Scripting

• Overview and Live Demonstration
  – What exactly can a “hacker” do?
SQL Injection

• Due to lack of input validation and string concatenation techniques:
  strQuery = “SELECT username, password FROM users “ & _
  WHERE username = ‘’ & Request.QueryString(“user”) & ‘’
• Basic forms are easily identifiable and exploitable via ODBC driver error messages
  – I am sometimes known as Capt. “Just-a-Tick”
• Can be used to:
  – Enumerate or modify database contents
  – Compromise the database host operating system
• What’s needed to exploit?
  – A web browser…but numerous tools exist to make life easy
• Exploits standard, required functionality
• Bypasses the network firewall
DEMO!

- Basic error-based SQL Injection
- Demonstration of tools such as Absinthe and PRIAMOS

- Executive-level Impact – Which one sounds worse?
  - Your website is SQL-injectable…
  - Here’s a list of your clients including social security numbers
Blind SQL Injection

• Same root cause as regular SQL Injection, but data extraction methods are more advanced
  – No error-based extraction
  – Can only obtain a “True” or “False” result from SQL Queries

• Well that’s not so bad, right?

• `SELECT username, password FROM users WHERE username = 'user' and (SELECT SUBSTRING(@@SERVERNAME,1,1)) > 'M'` —
  – Programmatically determine a value based on comparisons of individual characters!

• Wow…that would take a while, wouldn’t it?
  – Yes, but that’s what automated tools are for!
DEMO!

• Basic state-based Blind SQL Injection
• Demonstration of Absinthe in “blind” mode

• Beyond data extraction
  – Demonstration of system compromise via xp_cmdshell
SQL Injection Mitigations

- Validate, Validate, Validate!
  - Front-end Validation – User Experience ONLY
  - Back-end Validation – Server-side validation of data
    - Strongly-typed variables
    - White-list techniques as opposed to black-listing “bad” characters
  - Database enforcement of the above validation
    - Typed columns, limits to field lengths
    - …everything is NOT an nvarchar with MAX length
  - Data encoding
- Stored procedures or Parameterized Queries
- Low-privileged accounts for database and web server
- Egress Filtering
Authorization Bypass

• OK – so you’ve put generic error messages in place and you’re using stored procedures…what can an attacker do now?
• Have you put an authorization layer in place?
  – http://hostname/creditinfo.asp?customerId=5637
  – Should I have access to customerId 5637?
  – What about 5638? Or 5639? Or 2567?

• Unfortunately, this is extremely common and again requires NO advanced techniques…just a simple modification of a predictable number

• Quick demo
Cross-Site Scripting

- Root cause same as SQL Injection – Input Validation!
- HTML characters accepted as input and re-displayed to page without encoding:
  - Damon”><script>alert(‘IOA’)</script> is NOT a valid username!
    1. This shouldn’t be accepted in the first place
    2. If it is accepted, does it need to be re-displayed?
    3. Any HTML characters should be encoded in output
      - “ becomes &quot;
      - > becomes &gt;
- Can be used to:
  - Execute client-side code, such as arbitrary JavaScript
  - Steal cookies, credential information, alter any aspect of user experience
DEMO!

- Basic Cross-Site Scripting (XSS)
- Persistent XSS

- Beyond alert dialogs
  - Escalation to administration using XSS
  - Using XSS to compromise a client system via browser exploits
Current Threats

• Technology and Development Environments improving
  – Difficult to make this demo work in ASP.NET!

• Today’s threats
  – Logic errors and authorization
  – Poor crypto implementation
    • SSL/TLS used only for transport security, not for other benefits
    • Web services using passwords instead of authenticating certs
  – File handling issues (arbitrary read/write)
  – Still…input validation!
  – In-depth manual review of complex web applications still required
    • Automated web app scanners have matured, but not enough
  – Applications still not designed with security ingrained in the process
SQL Injection Tools

• Commercial
  – SPI Dynamics Toolkit

• Free/Open Source
  – Abinsthe - www.0x90.org/releases/absinthe/
  – Bsqlbf - http://www.unsec.net/download/bsqlbf.pl
  – BobCat - http://www.northern-monkee.co.uk/projects/bobcat/bobcat.html
  – PRIAMOS - www.priamos-project.com
  – SQLiX - www.owasp.org/index.php/Category:OWASP_SQLiX_Project
  – Sqlmap - sqlmap.sourceforge.net
  – Sqlninja - sqlninja.sourceforge.net
**XSS in “Web 2.0”**

- AJAX and Dynamic Applications require the use of JavaScript
- Greater functionality in applications = greater functionality for attackers!
- Expose API’s - Again lack of authorization!

- Jikto - JavaScript port of Nikto for distributed web vulnerability scanning using cross-site scripting as a distribution method
  - Ie: XSS Bot-net
- JavaScript “Attack Toolkits” being released
  - BeEF - Browser Exploitation Framework
- New Classes of JavaScript attacks being revealed

- Unfortunately, also no legal avenues to report web application vulnerabilities to raise awareness
  - Unknown how many web applications in the wild are vulnerable

Copyright IOActive, Inc. 2007