Boston OWASP

You Say Tomayto and I Say Tomahto
Talking to Developers About Application Security

Scott Matsumoto
Principal Consultant
smatsumoto@cigital.com
What Security Says | What Developer's Hear

- There are XSS vulnerabilities all over your application.

- This is really serious problem. You really need to fix it.

- The problem with XSS is that you can hijack someone's session or steal their credentials.

- Blah, XSS, blah application.

- This is a really serious problem. You need to make a change that touches hundreds of files.

- "Noone will ever do that"
The Dichotomy Is Inherent

- Programmers are constructive in nature
  - Define and follow rules
  - Know what the program inputs expects, so they give it what it expects
  - Know intimate details about their application domain
  - Don't like to fix bugs (like to make features)

- Security-ers are destructive
  - Break rules
  - Try to give programs data that is interpreted as code
  - Know intimate details about the technology platform
  - Only find bugs
Describing the Problem

Be The Developer
Attack Versus Vulnerability

- Buffer overflow and SQL Injection the attack and the vulnerability are very close

- How do you describe XSS?

- How do you demonstrate XSS?
Make Effective Use of Sandwich

- Makes use of an alternate web site to host the attack
  - Alter content that is presented to user
  - Send information to evil host
  - Trick user into divulging private information
- Two primary varieties:
  - Reflected XSS makes use of a *phishing* attack with crafted links
  - Stored XSS where attacker stores malicious content on server
Developers Need to See The Code

- URL
  p.jsp?banner=<script>evil javascript</script>

- Generated Page
  <H1>Welcome</H1>

- Even this pseudo code isn't enough
The Code Has to Run

- A more complete attack is more compelling
- The developer must recognize the code as being "his code"
Thinking About The Solution
Will Kruse's Top 12 Myths

- Hidden form fields, cookies, and headers are immutable.
- Authentication = Authorization.
- Encoding (URL, Base64, etc) = encryption.
- JavaScript validation is the way to protect the server (much less important myth but good for discussion: JavaScript validation has no place ever).
- Attackers can't pull sensitive data out of binaries, flash files, HTML or JS.
- Stored Procedures prevent SQL injection.
- Thick clients are secure against SQL injection.
- Black-lists are fine; white-lists are for paranoids.
- Encryption solves everything. Just encrypt it.
- Encryption protects message integrity.
- Tool X will solve all my problems.
Our Guidance Is Perfect - Not

- Clearly define trust boundaries and validate all input values
- Validate all input lexically and grammatically
  - Combine white-lists and black-list
- Encode user provided value into safe format before use
- Log all input validation failures
Input Validation Versus Output Encoding

- What do you recommend first for XSS?
  - Input Validation
  - Output Encoding

- Why Input Validation is excellent?

- Why does Input Validation suck?

- Why Output Encoding is excellent?

- Why does Output Encoding suck?
Thank you for your time.