Unforgivable Vulnerabilities

Steve Christey
The MITRE Corporation
August 2, 2007
Introduction

- Vulnerabilities are a fact of life

- Many vulnerabilities simply shouldn’t be in software anymore

- Everything’s obvious to smart people like us!

- How to identify the worst of the worst?

- What issues should give pause to consumers, and nightmares to vendors?

- Raw vulnerability numbers don’t tell the whole story
Typical Vulnerability History of a Product

1. Obvious types in critical functionality

2. Incomplete fixes, closely related vectors

3. Variants of common vulnerability types

4. Limited environments, platforms, configs

5. Elimination of most common types

6. Rare or novel types and attacks

7. Unique types or attacks, extensive expert analysis

- ActiveX, Joe Schmoe SW
- Image and Document Processors
- High-profile network servers
Criteria for an “Unforgivable” Vulnerability

- Precedence: Many have made the same mistake
  - Required

- Documentation: The mistake is well-documented
  - Required

- Obviousness: The attacks are obvious

- Attack Simplicity: The manipulations are very simple
  - 2 of 3 Required

- Found in Five: Able to be found with 5 minutes of effort
We hold these vulnerabilities to be self-evident...
The Lucky 13

**Buffer overflow**
- User/Password
- Filenames
- Common Commands
  - (CWE-120)

**XSS**
- User/Password
- Body, subject, title, to, from
  - (CWE-79)

**Remote File Inclusion**
- *template=http://example.com/c99*
- Any include/require that interpolates
  - $_GET, $_POST, etc.
  - (CWE-98)

**SQL Injection**
- ’ OR 1=1
- User/Password id or other numeric field
  - (CWE-89)

**Directory Traversal**
- “../..” or “/full/path”
- Get/Send Command
- File sharing
  - (CWE-23)

**-rwxrwxrwx myprog**
- Executables
- Libraries
- Configuration Files
  - (CWE-276, 279)

**World-Writable Files**
- (CWE-276, 279)
The Lucky 13 (Continued)

- **http://example/admin/script.cgi**
  - Admin functionality
  - Library code with executable extensions

- **Direct Request**
  - (CWE-425)

- **Auth bypass**
  - (CWE-472)

- **User**: psychyore
  - **Pass**: psychyore

- **Length**: 0xffffffff
  - **Width**: 0xffffffff

- **Cipher**
  - (CWE-327)

- **Substitution**

- **Hard-coded Pass**
  - (CWE-259)

- **Symlink Following**
  - (CWE-61)

- **Integer overflow**
  - (CWE-190)

- **Log files**
  - Temporary files
  - Command-line args

- **Help**
  - Selected from privileged Windows executable

- **Integer overflow**
  - (CWE-190)

- **Auth bypass**

- **tebj lbhe bja pelcgb**
Shall We Play a Game?

Do you have any more nominations?

Is the research community effectively getting the message across?

How long will it take one of these problems to show up in your favorite software?

How many Black Hat talks in new technologies demonstrate these problems?
VAAL-idation

- **Vulnerability Assessment Assurance Levels (Litchfield)**
  - Level of confidence: “how secure is software X?”
  - Based on an audit of a product
  - Depth of analysis: “how much effort was put into analysis?”
  - VAAL != Common Criteria

- **Move away from those pesky vulnerability counts!**

- **Communicate to consumers and each other**

  The professional research community needs to stop pretending that basic research is magic and become more consumer-friendly. This means metrics.
Proposed VAAL Dimensions

- **Access Constraints**
  - Privileges/restrictions needed for access
- **Feature Frequency**
- **Potential Severity**
- **Novelty**
  - How new/unusual is the vuln/attack?
- **Vector Depth**
  - How “close together” are the entry point and the vulnerability?
- **Manipulation Complexity**
  - &lt;SCRIPT&gt; or RSnake head-scratcher?
- **Ubiquity**
  - Configuration, Platforms, Environments
- **Level of Effort**
  - Shhhh, never let them see you sweat
Unforgivable Vulnerabilities in VAAL-speak

- Low access constraints
- Very high feature frequency
- Very low novelty
- Low manipulation complexity
- Low level of effort

- Not directly applicable
  - Potential severity
  - Vector depth
  - Ubiquity
Related Work

- Attack Surface Measurement (Howard, Manadhata, and Wing)
- Threat Modeling (*Trike*, *STRIDE*, Snyder)
- SAMATE: Software Assurance Metrics and Tool Evaluation (NIST)
- Security Quality Score (Wysopal/Veracode)
- CVSS: Common Vulnerability Scoring System (FIRST)

See paper for details and more references
Questions?