More Repeatable Vulnerability Assessment
An introduction

Helén Svensson
Agenda

- Background
- Introduction to the following “aspects”
  - CVE/CWE/CAPEC
  - CWE/SANS TOP 25
  - ISO technical paper on Vulnerability Analysis
- How to apply the above “aspects” in the scheme
  - Education
  - Guidelines
- Expectations on evaluators
- Expectations on developers
- Questions?
Background

• **CSEC vill skapa verktyg för**
  – Enhetligt arbetssätt kring sårbarhetsanalyser
  – Gemensam grund för kompetensutveckling

• **Flera länder tittar på att nytta CVE/CWE/CAPEC vid framtagning av PP**

• **Dagens presentation**
  – visar ett möjligt sätt att använda CVE/CWE/CAPEC som ett verktyg för sårbarhetsanalys och gemensam grund för kompetensutveckling
  – Denna presentation är ”food-for-thought” - CSEC har inte något krav på att ITSEF ändrar arbetssätt baserat på denna presentation
  – Ska ses som information och diskussionsunderlag
**CVE**

- **Common Vulnerabilities and Exposures**
- **Goals:**
  - Uniquely naming every publicly known information security vulnerability and exposure.
  - Injecting CVE names into security and vendor advisories.
  - Establishing CVE usage in information security products as common practice.
  - Having CVE usage permeate policy guidelines about methodologies and purchasing, included as requirements for new capabilities, and introducing CVE into training, education, and best practices suggestions.
  - Convincing commercial software developers to use CVE names in their fix-it sites and update mechanisms.
CVE Identifiers

- Each CVE Identifier includes:
  - CVE Identifier number (i.e., "CVE-1999-0067").
  - Indication of "entry" or "candidate" status.
  - Brief description of the security vulnerability or exposure.
  - Any pertinent references (i.e., vulnerability reports and advisories or OVAL-ID).
Today

- **CVE:**
  - Is the industry standard for vulnerability and exposure names
  - Provides a common language for security products and services
  - Provides a baseline for evaluating the coverage of tools and services

- **Result:** better coverage, easier interoperability, enhanced security.
CWE

• Common Weakness Enumeration
• Launched March 2006 with Draft 1
• Establishes acceptable definitions and descriptions of these common weaknesses

• Objective:
  – help shape and mature the code security assessment industry and also dramatically accelerate the use and utility of software assurance capabilities for organizations in reviewing the software systems they acquire or develop.

• Result:
  – A common language for describing weaknesses
  – A standard measuring stick for tools targeting these vulnerabilities
  – A baseline standard for weakness identification, mitigation and prevention efforts
CWE Structure

- Description
- Alternate terms
- Time of introduction
- Applicable platforms
- Common consequences
- Likelihood of exploit
- Enabling Factors for Exploitation
- Detection Methods
- Demonstrative Examples
- Observed Examples (CVEs)
- Potential Mitigations
- Background Details

- Weakness Ordinalities
- Relationships
- Causal Nature
- Taxonomy Mappings
- Related Attack Patterns (CAPECs)
- References
- Content History
CWE/SANS Top 25 1(2)

- **Top 25 Most Dangerous Software Errors**
  - CWE-89  Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
  - CWE-78  Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
  - CWE-120  Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
  - CWE-79  Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
  - CWE-306  Missing Authentication for Critical Function
  - CWE-862  Missing Authorization
  - CWE-798  Use of Hard-coded Credentials
  - CWE-311  Missing Encryption of Sensitive Data
  - CWE-434  Unrestricted Upload of File with Dangerous Type
  - CWE-807  Reliance on Untrusted Inputs in a Security Decision
  - CWE-250  Execution with Unnecessary Privileges
  - CWE-352  Cross-Site Request Forgery (CSRF)
CWE/SANS Top 25 2(2)

- CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- CWE-494 Download of Code Without Integrity Check
- CWE-863 Incorrect Authorization
- CWE-829 Inclusion of Functionality from Untrusted Control Sphere
- CWE-732 Incorrect Permission Assignment for Critical Resource
- CWE-676 Use of Potentially Dangerous Function
- CWE-327 Use of a Broken or Risky Cryptographic Algorithm
- CWE-131 Incorrect Calculation of Buffer Size
- CWE-307 Improper Restriction of Excessive Authentication Attempts
- CWE-601 URL Redirection to Untrusted Site ('Open Redirect')
- CWE-134 Uncontrolled Format String
- CWE-190 Integer Overflow or Wraparound
- CWE-759 Use of a One-Way Hash without a Salt
CWE/SANS 16 "On the Cusp" weaknesses

- Weaknesses that almost made it to the CWE/SANS TOP25
  - CWE-770: Allocation of Resources Without Limits or Throttling
  - CWE-129: Improper Validation of Array Index
  - CWE-754: Improper Check for Unusual or Exceptional Conditions
  - CWE-805: Buffer Access with Incorrect Length Value
  - CWE-838: Inappropriate Encoding for Output Context
  - CWE-330: Use of Insufficiently Random Values
  - CWE-822: Untrusted Pointer Dereference
  - CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')
  - CWE-212: Improper Cross-boundary Removal of Sensitive Data
  - CWE-681: Incorrect Conversion between Numeric Types
  - CWE-476: NULL Pointer Dereference
  - CWE-841: Improper Enforcement of Behavioral Workflow
  - CWE-772: Missing Release of Resource after Effective Lifetime
  - CWE-209: Information Exposure Through an Error Message
  - CWE-825: Expired Pointer Dereference
  - CWE-456: Missing Initialization
Top 25 – CWE Details

• Summary

CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

<table>
<thead>
<tr>
<th>Summary</th>
<th>Weakness</th>
<th>Prevalence</th>
<th>Consequences</th>
<th>Remediation Cost</th>
<th>Ease of Detection</th>
<th>Attack Frequency</th>
<th>Attacker Awareness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td></td>
<td>Code execution, Denial of service, Data loss</td>
<td>Low</td>
<td>Easy</td>
<td>Often</td>
<td>High</td>
</tr>
</tbody>
</table>

Discussion

Buffer overflows are Mother Nature's little reminder of that law of physics that says: if you try to put more stuff into a container than it can hold, you're going to make a mess. The scourge of C applications for decades, buffer overflows have been remarkably resistant to elimination. However, copying an untrusted input without checking the size of that input is the simplest error to make in a time when there are much more interesting mistakes to avoid. That's why this type of buffer overflow is often referred to as "classic." It's decades old, and it's typically one of the first things you learn about in Secure Programming 101.
Top 25 – CWE Details

- Prevention and Mitigations

**Prevention and Mitigations**

**Requirements**
Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
For example, many languages that perform their own memory management, such as Java and Perl, are not subject to buffer overflows. Other languages, such as Ada and C#, typically provide overflow protection, but the protection can be disabled by the programmer.

Be wary that a language's interface to native code may still be subject to overflows, even if the language itself is theoretically safe.

**Architecture and Design**
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
Examples include the Safe C String Library (SafeStr) by Messier and Viega, and the Strsafe.h library from...

Other inputs...

**Architecture and Design, Operation**
Run your code using the lowest privileges that are required to accomplish the necessary tasks. If possible, create isolated accounts with limited privileges that are only used for a single task. That way, a successful attack will not immediately give the attacker access to the rest of the software or its environment. For example, database applications rarely need to run as the database administrator, especially in day-to-day operations.
Top 25 – CWE Details

- Related CWEs
  - **Related CWEs**
    - **CWE-129**: Improper Validation of Array Index
    - **CWE-131**: Incorrect Calculation of Buffer Size

- Related CAPECs
  - **Related Attack Patterns**
    - CAPEC-IDs: [view all]
    - 8, 9, 10, 14, 24, 42, 44, 45, 46, 47, 67, 92, 100
CAPEC

- Common Attack Pattern Enumeration and Classification
- Developed by MITRE
- Initial release March/April 2007

- **Goal:** “Representing the attacker’s perspective in a formalized and constructive way to provide expert-level understanding and guidance to software development personnel of all levels as to how their software is likely to be attacked, and thereby equip them to build more secure software”

- List of patterns employed by attackers when compromising systems
- Generated from in-depth analysis of specific real-world exploit examples
CAPEC Structure

• Through analysis of observed exploits, the following typical information is captured for each attack pattern:
  – Identifying Information
    • Pattern ID
    • Pattern name
  – Describing Information
    • Description
    • Related weaknesses
    • Related vulnerabilities
    • Methods of attack
    • Examples – instances
    • References
  – Prescribing Information
    • Solutions and mitigations
CAPEC Structure

- Scoping and Delimiting Information
  - Typical Severity
  - Typical Likelihood of Exploit
  - Attack Prerequisites
  - Attacker Skill or Knowledge Required
  - Resources Required
  - Attack Motivation-Consequences
  - Context Description
  - Purpose
  - CIA Impact
  - Technical Context
  - Keywords
  - Pattern Abstraction Level

- Administrative Information
  - Source
CAPEC Structure

- **Supporting Schema Elements**
  - Describing Information
    - Injection Vector
    - Payload
    - Activation Zone
    - Payload Activation Impact
  - Diagnosing Information
    - Probing Techniques
    - Indicators-Warnings of Attack
    - Obfuscation Techniques
  - Enhancing Information
    - Related Attack Patterns
    - Relevant Security Requirements
    - Relevant Design Patterns
    - Relevant Security Patterns
    - Related Security Principles
    - Related Guidelines
Refining software vulnerability analysis under ISO/IEC 15408 and ISO/IEC 18045

• Technical report covering vulnerability assessment based on CWE and CAPEC

• Usage
  – in conjunction with and as an addendum to CC
  – Does not intend to satisfy the full range of requirements under the AVA_VAN family
  – Does not intend to restrict the activities performed by evaluators
  – (Intend to provide a minimal baseline of AVA_VAN evaluation)

• Current status
  – Draft technical report
  – Currently undergoing a 3-month DTR ballot
  – P-members of JTC 1 and SC 27 are requested to submit their votes by 2012-05-01
  – Is on the agenda for the ISO/IEC SC27 meetings in Stockholm in May…
Refining software vulnerability analysis under ISO/IEC 15408 and ISO/IEC 18045

• Syfte
  – add refinement and clarification of the “Potential vulnerability identification from public sources” (AVA_VAN.1.2E/2.2E/3.2E/4.2E) and “Penetration testing” (AVA_VAN.1.3E/2.4E/3.4E/4.4E) evaluator actions
  – to objectively search for, identify, filter and test potential vulnerabilities utilizing international ad hoc standard resources for software weaknesses and attack patterns.
  – The set of relevant software weaknesses and attack patterns identified through this guidance represent a minimal set for analysis under the AVA_VAN assurance family in an ISO/IEC 15408 evaluation.
  – Additional weaknesses and attack patterns may be determined relevant
  – relevant weaknesses and attack patterns identified and tested for during development can provide an head start template for a TOE-specific set of relevant weaknesses and attack patterns for use in the security evaluation
Existing structured assurance cases

- assurance case
  - structured set of claims, arguments and a corresponding body of evidence to demonstrate that a system satisfies specific claims with respect to its security properties

- If a structured assurance case exists for the TOE then the CWEs and CAPECs identified in the structured assurance case should be considered the relevant set for the IT security evaluation.

- If multiple structured assurance cases exists for the TOE then the CWEs and CAPECs identified in the most TOE-specific structured assurance case should be considered the relevant set for the IT security evaluation.
Identify relevant weaknesses and attack patterns from public sources

- **Identify initial set**
  - This is performed using CWE and CAPEC
  - to enumerate a very broad set of software weaknesses and attack patterns that may be relevant across a wide range of TOE contexts.

- **Filter initial set**
  - To bound the set of potential vulnerabilities to a reasonable scope for the IT security evaluation, the initial set of potentially relevant weaknesses and attack patterns identified
Initial set - CWE

- CWE weaknesses where the following criteria are all true:
  - CWE weaknesses with a **minimum adequate level** of defined detail:
    - Weakness_Abstraction is defined and equal to "Base" or "Variant"
    - Applicable_Platforms is defined
    - Detection_Methods is defined
    - Related_Attack_Patterns is defined
  - CWE weaknesses whose technical context factors are relevant to the TOE and its operational environment

- Ensure that the initial set contains CWE weaknesses that are identifiable from TOE-relevant CVE vulnerabilities.
Initial set - CAPEC

- CAPEC attack patterns where the following criteria are all true:
  - CAPEC attack patterns with a minimum adequate level of defined detail:
    - Pattern_Completeness is defined and equal to "Complete"
    - Pattern_Abstraction is defined and equal to "Standard" or "Detailed"
    - Attack_Execution_Flow is defined
    - Technical_Context is defined
    - Related_Weaknesses is defined
  - CAPEC attack patterns whose technical context factors are relevant to the TOE and its operational environment

- Check the initial set for any CAPEC entries referenced by identified CWEs but not in the initial set and the revers. If found, consider to include them in the initial set.
Filter initial set

- **Filter relevant weaknesses**
  - Filter out CWE weaknesses which do not contain Detection_Method schema elements specifying automated or black box forms of analysis.
  - Filter out CWE weaknesses which are not relevant due to measures in the operational environment, either IT or non-IT, preventing exploitation of the potential vulnerability in that operational environment. The evaluator must clearly record the specific reasoning involved for each weakness excluded.
Filter initial set

• Filter relevant attack patterns
  – Filter out CAPEC attack patterns whose intent and nature of impact are not relevant to the security sensitivity and critical security properties of the TOE.
  – Filter out CAPEC attack patterns which are not relevant due to measures in the operational environment, either IT or non-IT, preventing effective implementation of the attack pattern in that operational environment. The evaluator must clearly record the specific reasoning involved for each attack pattern excluded.
Penetration testing

- **Documentation of the test cases**
  - identification of the weakness (CWE) that the TOE is being tested for
  - identification of the attack pattern (CAPEC) the test case is instantiating
  - the detailed attack execution flow being carried out in the test case
Possible usage of CWE in the scheme: Education

• CWE/SANS TOP 25
  − Goal – the evaluator shall be able to explain in detail how the 25 CWEs works for example
    • how they are introduced
    • the environment in which they are relevant
    • How they are exploited by an attacker
    • how they can be identified during the evaluation
  − self-study
  − A test will be provided by CSEC
Guideline - effective vulnerability assessment

• A guideline is under development
  – ISO technical paper on Vulnerability Analysis
    • CWE/CAPEC should be used during the evaluation
  – Use of other relevant sources for Vulnerability Analysis
    • CVE should be used
    • Product specific sources of information should be used when applicable, examples:
      – Will contain examples to illustrate the use of CVE/CWE/CAPEC and other sources

• The evaluator shall take the guideline into account during AVA-VAN

• The certifier will use the guideline as part of the Technical oversight
Expectations on Evaluators

• Continually work on the ability to perform vulnerability analysis
  – Understand the weaknesses described in CWE/SANS TOPS 25
  – Preferably also understand the weaknesses described in CWE/SANS 16 "On the Cusp"
  – Also use other sources for information about weaknesses

• Use relevant public sources during evaluations to identify potential vulnerabilities
  – the guideline - effective vulnerability assessment will be a starting point for this
Developers

- Awareness of the weaknesses and vulnerabilities that could be introduced during a product's life cycle is essential when developing a secure product.

- CWE/CAPEC and other sources can be useful to developers during:
  - Education of the developers
  - Design
  - Implementation
  - Testing:
    - Use CWE as a “checklist”
    - Find related weaknesses that might be present as well
    - Links to attack vectors (CAPECs), aka “how to attack”
    - Can be used as a metric for reporting found vulnerabilities
Questions?