HS SEDI
Homeland Security Systems Engineering and Development Institute

Understanding How They Attack Your Weaknesses: CAPEC

Sean Barnum
MITRE

Homeland Security

The Long-established Principal of "Know Your Enemy"

- "One who knows the enemy and knows himself will not be endangered in a hundred engagements. One who does not know the enemy but knows himself will sometimes be victorious. Sometimes meet with defeat. One who knows neither the enemy nor himself will invariably be defeated in every engagement."

- Chapter 3: "Planning the Attack"
  - The Art of War, Sun Tzu

The Importance of Knowing Your Enemy

HS SEDI
Homeland Security Systems Engineering and Development Institute

- **An appropriate defense can only be established if you know how it will be attacked**

- **Remember!**
  - Software Assurance must assume motivated attackers and not simply passive quality issues
  - Attackers are very creative and have powerful tools at their disposal
  - Exploring the attacker's perspective helps to identify and qualify the risk profile of the software

What are Attack Patterns?

- Blueprint for creating a specific type of attack
- Abstracted common attack approaches from the set of known exploits

- Capture the attacker's perspective to aid software developers, acquirers and operators in improving the assurance profile of their software

Leveraging Attack Patterns Throughout the Software Lifecycle

- Guide definition of appropriate policies
- Guide creation of appropriate security requirements (positive and negative)
- Provide context for architectural risk analysis
- Guide risk-driven secure code review
- Provide context for appropriate security testing
- Provide a bridge between secure development and secure operations

Common Attack Pattern Enumeration and Classification (CAPEC)

HS SEDI
Homeland Security Systems Engineering and Development Institute

- **Community effort targeted at:**
  - Standardizing the capture and description of attack patterns
  - Collecting known attack patterns into an integrated enumeration that can be consistently and effectively leveraged by the community
  - Gives you an attacker's perspective you may not have on your own

- **Excellent resource for many key activities**
  - Abuse Case development
  - Architecture attack resistance analysis
  - Risk-based security/Red team penetration testing
  - Whitebox and Blackbox testing correlation
  - Operational observation and correlation

- **Where is CAPEC today?**
  - http://capec.mitre.org
  - Currently 386 patterns, stubs, named attacks


CAPEC – Common Attack Pattern Enumeration and Classification (CAPEC)
http://capec.mitre.org/
Google
CAPEC
Common Attack Pattern Enumeration and Classification
A Community Knowledge Resource for Building Secure Software
Search by ID:
Go
CAPEC List
Full CAPEC Dictionary
Methods of Attack View
Reports
About CAPEC
Documents
Resources
Community
Related Activities
Collaboration List
News & Events
Calendar
Free Newsletter
Contact Us
Search the Site
Building software with an adequate level of security assurance for its mission becomes more and more challenging every day as the size, complexity, and tempo of software creation increases and the number and the skill level of attackers continues to grow. These factors each exacerbate the issue that, to build secure software, builders must ensure that they have protected every relevant potential vulnerability; yet, to attack software, attackers often have to find and exploit only a single exposed vulnerability. To identify and mitigate relevant vulnerabilities in software, the development community needs more than just good software engineering and analytical practices, a solid grasp of software security features, and a powerful set of tools. All of these things are necessary but not sufficient. To be effective, the community needs to think outside of the box and to have a firm grasp of the attacker's perspective and the approaches used to exploit software.
Attack patterns are a powerful mechanism to capture and communicate the attacker's perspective. They are descriptions of common methods for exploiting software. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples.
To assist in enhancing security throughout the software development lifecycle, and to support the needs of developers, testers and educators, the Common Attack Pattern Enumeration and Classification (CAPEC) is sponsored by the Department of Homeland Security as part of the Software Assurance strategic initiative of the National Cyber Security Division. The objective of this effort is to provide a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy. This site now contains the initial set of content and will continue to evolve with public participation and contributions to form a standard mechanism for identifying, collecting, refining, and sharing attack patterns among the software community.
Release 1.6 Available
Page Last Updated: February 07, 2011
MITRE
CAPEC is a Software Assurance strategic initiative co-sponsored by the National Cyber Security Division of the U.S. Department of Homeland Security.
This Web site is sponsored and managed by The MITRE Corporation to enable stakeholder collaboration.
Copyright 2011, The MITRE Corporation. CAPEC and the CAPEC logo are trademarks of The MITRE Corporation.
Contact capec@mitre.org for more information.
Privacy policy
Terms of use
Contact us
Member of Making Security Measurable
Done

What do Attack Patterns Look Like?

HS SEDI
Homeland Security Systems Engineering and Development Institute

- **Primary Schema Elements**
  - Identifying Information
    - Attack Pattern ID
    - Attack Pattern Name
  - Describing Information
    - Description
    - Related Weaknesses
    - Related Vulnerabilities
    - Method of Attack
    - Examples-Instances
    - References
  - Prescribing Information
    - Solutions and Mitigations
  - Scoping and Delimiting Information
    - Typical Severity
    - Typical Likelihood of Exploit
    - Attack Prerequisites
    - Attacker Skill or Knowledge Required
    - Resources Required
    - Attack Motivation-Consequences
    - Context Description

- **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

Attack Pattern Description Schema Formalization

HS SEDI
Homeland Security Systems Engineering and Development Institute

Description

- **Summary**
- **Attack_Execution_Flow**
  - Attack_Phase$^{1..3}$ (Name(Explore, Experiment, Exploit))
    - **Attack_Step$^{1..*}$**
      - **Attack_Step_Title**
      - **Attack_Step_Description**
      - **Attack_Step_Technique $^{0..*}$**
        - **Attack_Step_Technique_Description**
        - **Leveraged_Attack_Patterns**
        - **Relevant_Attack_Surface_Elements**
        - **Observables$^{0..*}$**
        - **Environments**
      - **Indicator$^{0..*}$ (ID, Type(Positive, Failure, Inconclusive))**
        - **Indicator_Description**
        - **Relevant_Attack_Surface_Elements**
        - **Environments**
      - **Outcome$^{0..*}$ (ID, Type(Success, Failure, Inconclusive))**
        - **Outcome_Description**
        - **Relevant_Attack_Surface_Elements**
        - **Observables$^{0..*}$**
        - **Environments**
      - **Security Control$^{0..*}$ (ID, Type(Detective, Corrective, Preventative))**
        - **Security_Control_Description**
        - **Relevant_Attack_Surface_Elements**
        - **Observables$^{0..*}$**
        - **Environments**
      - **Observables$^{0..*}$**

Individual CAPEC Dictionary Definition (Release 1.2)

Blind SQL Injection

Attack Pattern ID	7
Pattern Abstraction: Detailed	
Typical Severity	High

Description

Summary

Blind SQL Injection results from an insufficient mitigation for SQL Injection. Although suppressing database error messages are considered best practice, the suppression alone is not sufficient to prevent SQL Injection. Blind SQL Injection is a form of SQL Injection that overcomes the lack of error messages. Without the error messages that facilitate SQL Injection, the attacker constructs input strings that probe the target through simple Boolean SQL expressions. The attacker can determine if the syntax and structure of the injection was successful based on whether the query was executed or not. Applied iteratively, the attacker determines how and where the target is vulnerable to SQL Injection.

In order to achieve this using Blind SQL Injection, an attacker:

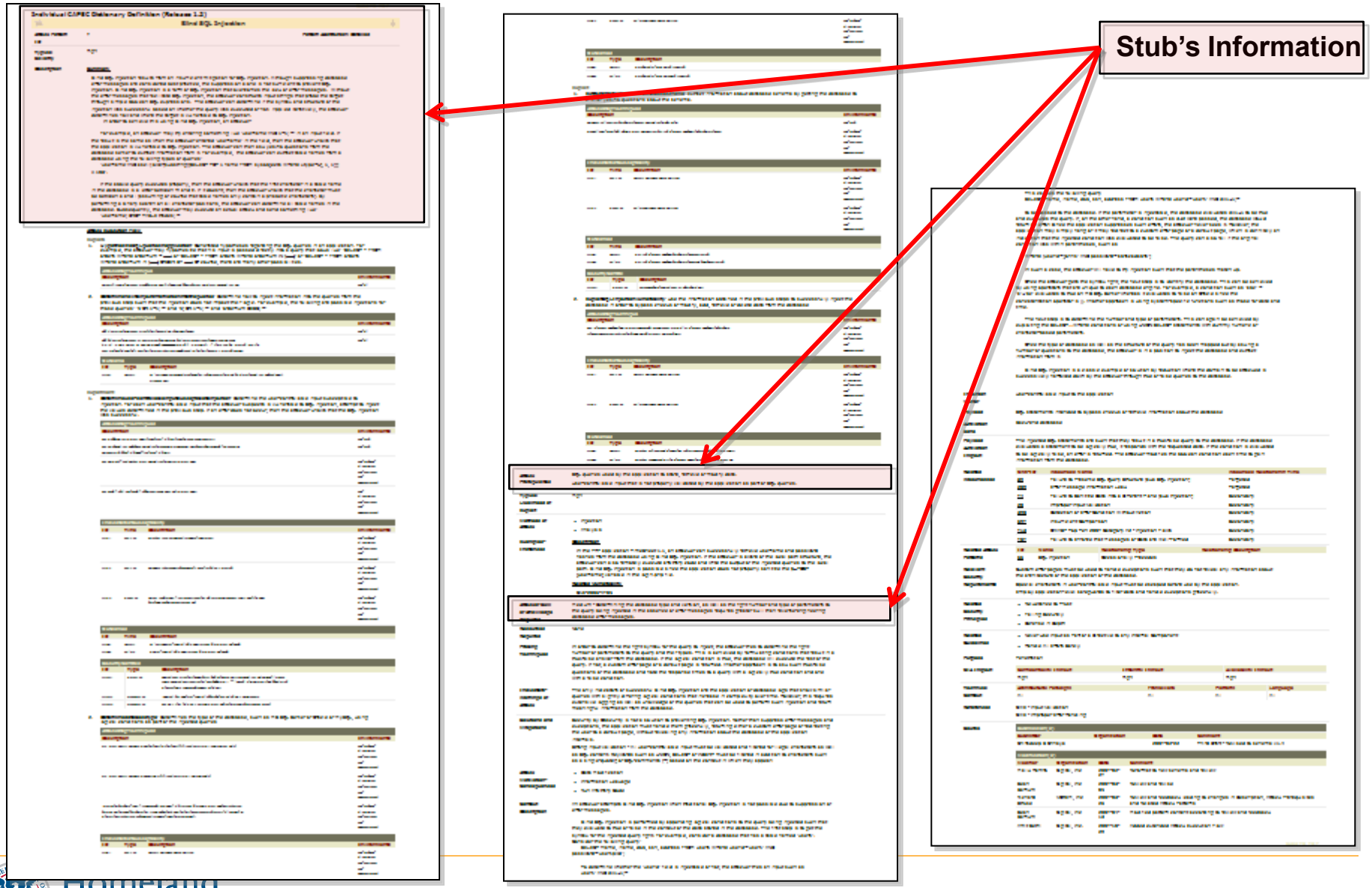
For example, an attacker may try entering something like "username' AND 1=1; --" in an input field. If the result is the same as when the attacker entered "username" in the field, then the attacker knows that the application is vulnerable to SQL Injection. The attacker can then ask yes/no questions from the database server to extract information from it. For example, the attacker can extract table names from a database using the following types of queries:

"username' AND ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype='U'), 1, 1))) > 108".

If the above query executes properly, then the attacker knows that the first character in a table name in the database is a letter between m and z. If it doesn't, then the attacker knows that the character must be between a and l (assuming of course that table names only contain alphabetic characters). By performing a binary search on all character positions, the attacker can determine all table names in the database. Subsequently, the attacker may execute an actual attack and send something like:

"username'; DROP TABLE trades; --

Complete CAPEC Entry Information

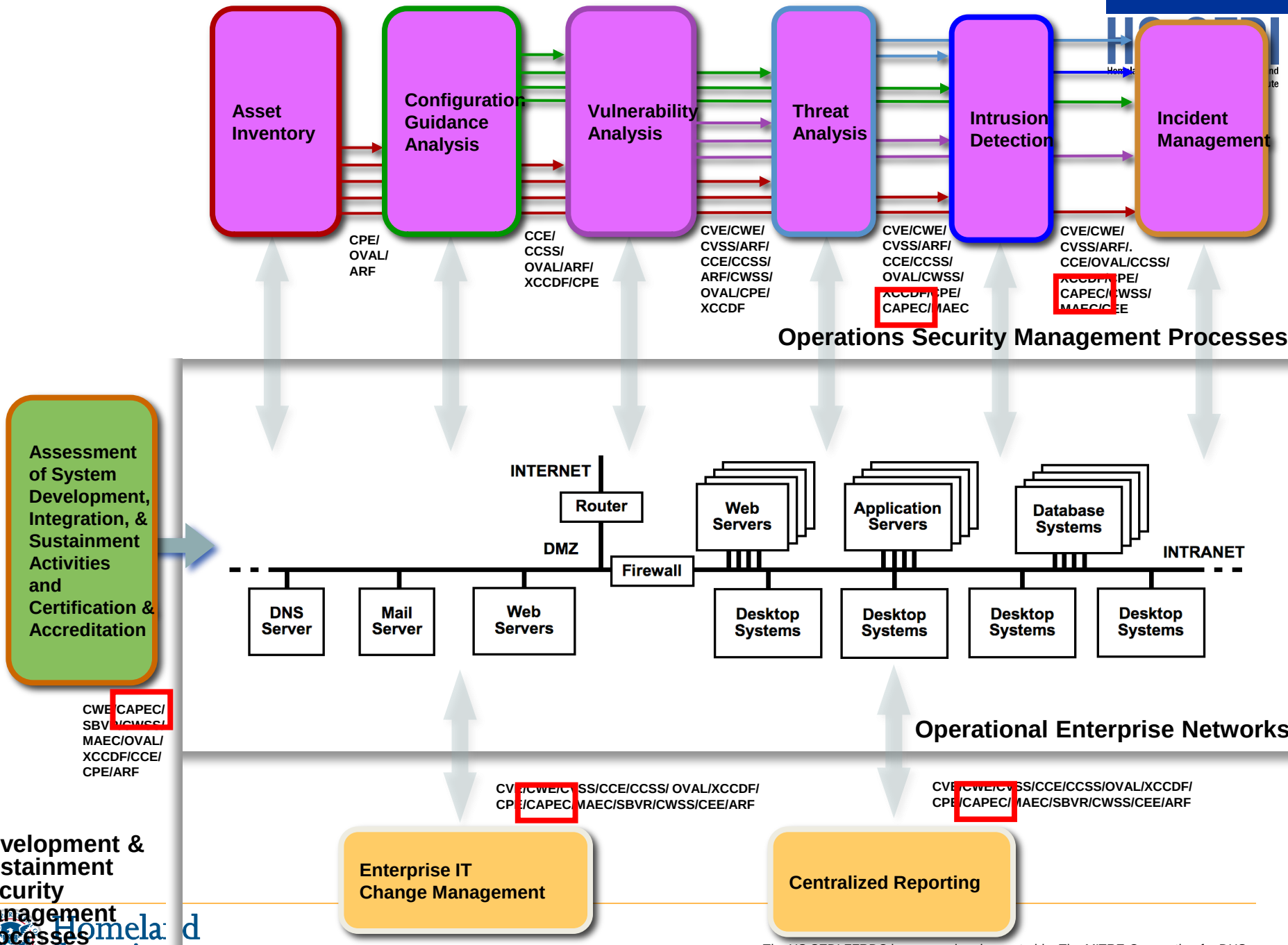


A Few Key Use Cases for CAPEC in Support of SwA

- Help developers understand weaknesses in their real-world context (how they will be attacked)
- Objectively identify specific attacks under which software must demonstrate resistance, tolerance and resilience for a given level of assurance
- Indirectly scope which weaknesses are relevant for a given threat environment
- Identify relevant mitigations that should be applied as part of policy, requirements, A&D, implementation, test, deployment and operations
- Identify and characterize patterns of attacks for security test case generation
- Identify and characterize threat TTPs for red teaming
- Identify relevant issues for automated tool selection
- Identify and characterize issues for automated tool results analysis

CAPEC and Security Measurement

- Measuring stick for evaluating and comparing penetration testing tools and application defense tools
  - Similar to CWE value to secure code analysis tools and CVE value to vulnerability scanners
- Measuring stick for attack resistance claims of assurance cases
  - It comes down to the proxies for measuring security (vulnerabilities, weaknesses and attack resistance)
- Characterizing the nature of software attack
  - Formalization of attack patterns to enable:
    - Recognition and mapping of attack instances from the operations realm
    - Refined ability to measure attack resistance in terms of resistance to individual sub-elements of attack with observables
    - Automated generation of penetration attack cases
    - Defining & mapping attack simulations in penetration testing tools
  - Alignment with malware characterization (MAEC)


Asset Inventory
Configuration Guidance Analysis
Vulnerability Analysis
Threat Analysis
Intrusion Detection
Incident Management
CPE/ OVAL/ ARF
CCE/ CCSS/ OVAL/ARF/ XCCDF/CPE
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ ARF/CWSS/ OVAL/CPE/ XCCDF
CVE/CWE/ CVSS/ARF/ CCE/CCSS/ OVAL/CWSS/
CAPEC/
CVE/CWE/ CVSS/ARF/. CCE/OVAL/CCSS/
CAPEC/
Operations Security Management Processes
Assessment of System Development, Integration, & Sustainment Activities and Certification & Accreditation
INTERNET
Router
DMZ
Firewall
Web Servers
Application Servers
Database Systems
INTRANET
DNS Server
Mail Server
Web Servers
Desktop Systems
Desktop Systems
Desktop Systems
Desktop Systems
MAEC/OVAL/ XCCDF/CCE/ CPE/ARF
Operational Enterprise Networks
OVAL/XCCDF/
MAEC/SBVR/CWSS/CEE/ARF
OVAL/XCCDF/
MAEC/SBVR/CWSS/CEE/ARF
Enterprise IT Change Management
Centralized Reporting
Development & Sustainment Security Management Processes
Enterprise IT Asset Management

CAPEC Status

HS SEDI
Homeland Security Systems Engineering and Development Institute

Where is CAPEC today?

- V1.4
  - Massive schema changes
    - Including addition of Observables structure
  - Some new content
  - Added initial set of network attack patterns
- V1.5
  - Added ~25 new network attack patterns
  - Added enhanced material to ~35 patterns
  - New View added for WASC Threat Taxonomy 2.0
  - Added ~65 mappings to CWE and several within CAPEC
- V1.6
  - Added 7 new application framework attack patterns as well as 68 new attack patterns in three new attack pattern categories: Physical Security Attacks, Social Engineering Attacks & Supply Chain Attacks
  - Added ~35 mappings to CWE and several within CAPEC

Currently 386 patterns, stubs, named attacks; 68 categories and 6 views

CAPEC Current Content (15 Major Categories)

HS SEDI
Homeland Security Systems Engineering and Development Institute

1000 - Mechanism of Attack

- **Data Leakage Attacks - (118)**
- **Resource Depletion - (119)**
- **Injection (Injecting Control Plane content through the Data Plane) - (152)**
- **Spoofing - (156)**
- **Time and State Attacks - (172)**
- **Abuse of Functionality - (210)**
- **Exploitation of Authentication - (225)**
- **Probabilistic Techniques - (223)**
- **Exploitation of Privilege/Trust - (232)**
- **Data Structure Attacks - (255)**
- **Resource Manipulation - (262)**
- **Physical Security Attacks (436)**
- **Network Reconnaissance - *(286)***
- **Social Engineering Attacks (403)**
- **Supply Chain Attacks (437)**

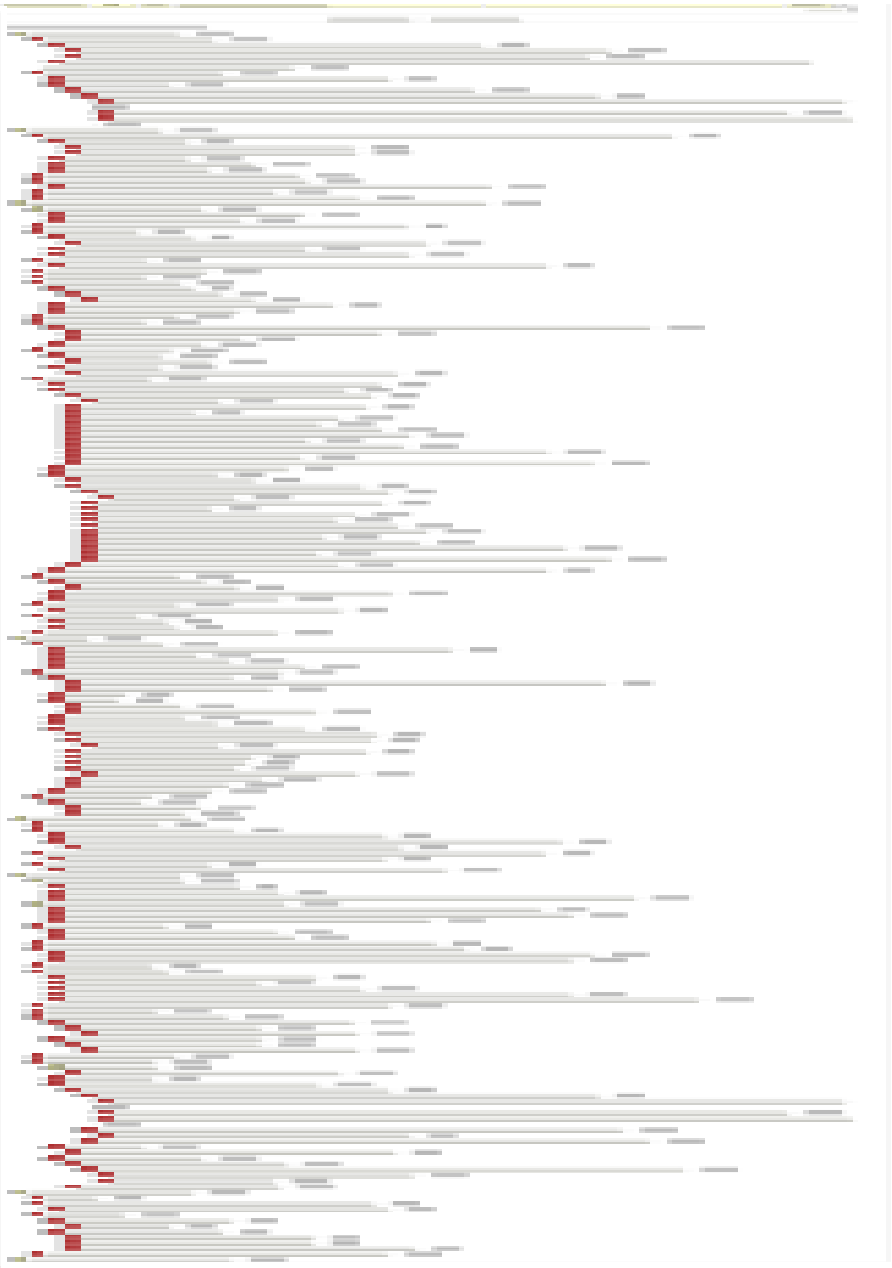
CAPEC Current Content (Which Expand to…)

HS SEDI
Homeland Security Systems Engineering and Development Institute

- 1000 - Mechanism of Attack
  - Data Leakage Attacks - (118)
    - Data Excavation Attacks - (116)
    - Data Interception Attacks - (117)
  - Resource Depletion - (119)
    - Violating Implicit Assumptions Regarding XML Content (aka XML Denial of Service (XDoS)) - (82)
    - Resource Depletion through Flooding - (125)
    - Resource Depletion through Allocation - (130)
    - Resource Depletion through Leak - (131)
    - Denial of Service through Resource Depletion - (227)
  - Injection (Injecting Control Plane content through the Data Plane) - (152)
    - Remote Code Inclusion - (253)
    - Analog In-band Switching Signals (aka Blue Boxing) - (5)
    - SQL Injection - (66)
    - Email Injection - (134)
    - Format String Injection - (135)
    - LDAP Injection - (136)
    - Parameter Injection - (137)
    - Reflection Injection - (138)
    - Code Inclusion - (175)
    - Resource Injection - (240)
    - Script Injection - (242)
    - Command Injection - (248)
    - Character Injection - (249)
    - XML Injection - (250)
    - DTD Injection in a SOAP Message - (254)
  - Spoofing - (156)
    - Content Spoofing - (148)
    - Identity Spoofing (Impersonation) - (151)
    - Action Spoofing - (173)
  - Time and State Attacks - (172)
    - Forced Deadlock - (25)
    - Leveraging Race Conditions - (26)
    - Leveraging Time-of-Check and Time-of-Use (TOCTOU) Race Conditions - (29)
    - Manipulating User State - (74)
  - Abuse of Functionality - (210)
    - Functionality Misuse - (212)
    - Abuse of Communication Channels - (216)
    - Forceful Browsing - (87)
    - Passing Local Filenames to Functions That Expect a URL - (48)
    - Probing an Application Through Targeting its Error Reporting - (54)
    - WSDL Scanning - (95)
    - API Abuse/Misuse - (113)
    - Try All Common Application Switches and Options - (133)
    - Cache Poisoning - (141)
    - Software Integrity Attacks - (184)
    - Directory Traversal - (213)
    - Analytic Attacks - (281)
  - Probabilistic Techniques - (223)
    - Fuzzing - (28)
    - Manipulating Opaque Client-based Data Tokens - (39)
    - Brute Force - (112)
    - Screen Temporary Files for Sensitive Information - (155)
  - Exploitation of Authentication - (225)
    - Exploitation of Session Variables, Resource IDs and other Trusted Credentials - (21)
    - Authentication Abuse - (114)
    - Authentication Bypass - (115)
  - Exploitation of Privilege/Trust - (232)
    - Privilege Escalation - (233)
    - Exploiting Trust in Client (aka Make the Client Invisible) - (22)
    - Hijacking a Privileged Thread of Execution - (30)
    - Subvert Code-signing Facilities - (68)
    - Target Programs with Elevated Privileges - (69)
    - Exploitation of Authorization - (122)
    - Hijacking a privileged process - (234)
  - Data Structure Attacks - (255)
    - Accessing/Intercepting/Modifying HTTP Cookies - (31)
    - Buffer Attacks - (123)
    - Attack through Shared Data - (124)
    - Integer Attacks - (128)
    - Pointer Attack - (129)
  - Resource Manipulation - (262)
    - Accessing/Intercepting/Modifying HTTP Cookies - (31)
    - Input Data Manipulation - (153)
    - Resource Location Attacks - (154)
    - Infrastructure Manipulation - (161)
    - File Manipulation - (165)
    - Variable Manipulation - (171)
    - Configuration/Environment manipulation - (176)
    - Abuse of transaction data strutcture - (257)
    - Registry Manipulation - (269)
    - Schema Poisoning - (271)
    - Protocol Manipulation - (272)
  - Network Reconnaissance - (286)
    - ICMP Echo Request Ping - (285)
    - TCP SYN Scan - (287)
    - ICMP Echo Request Ping - (288)
    - Infrastructure-based footprinting - (289)
    - Enumerate Mail Exchange (MX) Records - (290)
    - DNS Zone Transfers - (291)
    - Host Discovery - (292)
    - Traceroute Route Enumeration - (293)
    - ICMP Address Mask Request - (294)
    - ICMP Timestamp Request - (295)
    - ICMP Information Request - (296)
    - TCP ACK Ping - (297)
    - UDP Ping - (298)
    - TCP SYN Ping - (299)
    - Port Scanning - (300)
    - TCP Connect Scan - (301)
    - TCP FIN scan - (302)
    - TCP Xmas Scan - (303)
    - TCP Null Scan - (304)
    - TCP ACK Scan - (305)
    - TCP Window Scan - (306)
    - TCP RPC Scan - (307)
    - UDP Scan - (308)

CAPEC Current Content (386 Attacks…)

HS SEDI
Homeland Security Systems Engineering and Development Institute



Current Maturation Paths

HS SEDI
Homeland Security Systems Engineering and Development Institute

- Extend coverage of CAPEC
- Improve quality of CAPEC
- Expand the scope of CAPEC
- Bridge secure development with secure operations
- Improve integration with other standards (MAEC, CEE, etc.)
- Expand use of CAPEC

Industry Uptake

Manually review code after security education

Manual code review, especially review of high-risk code, such as code that faces the Internet or parses data from the Internet, is critical, but only if the people performing the code review know what to look for and how to fix any code vulnerabilities they find. The best way to help understand classes of security bugs and remedies is education, which should minimally include the following areas:

- C and C++ vulnerabilities and remedies, most notably buffer overruns and integer arithmetic issues.
- Web-specific vulnerabilities and remedies, such as cross-site scripting (XSS).
- Database-specific vulnerabilities and remedies, such as SQL injection.
- Common cryptographic errors and remedies.

Many vulnerabilities are programming language (C, C++ etc) or domain-specific (web, database) and others can be categorized by vulnerability type, such as injection (XSS and SQL Injection) or cryptographic (poor random number generation and weak secret storage) so specific training in these areas is advised.

Resources

- A Process for Performing Security Code Reviews, Michael Howard, IEEE Security & Privacy July/August 2006.
- .NET Framework Security — Code Review; http://msdn.microsoft.com/en-us/library/aa302437.aspx
- *Common Weakness Enumeration*, MITRE; http://cwe.mitre.org/
- Security Code Reviews; http://www.codesecurely.org/Wiki/view.aspx/Security_Code_Reviews
- Security Code Review — Use Visual Studio Bookmarks To Capture Security Findings; http://blogs.msdn.com/alikl/archive/2008/01/24/security-code-review-use-visual-studio-bookmarks-to-capture-security-findings.aspx
- Security Code Review Guidelines, Adam Shostack; http://www.verber.com/mark/cs/security/code-review.html
- OSWASP Top Ten; http://www.owasp.org/index.php/OWASP_Top_Ten_Project

Testing

Testing activities validate the secure implementation of a product, which re
the likelihood of security bugs being released and discovered by customers a
malicious users. The majority of SAFECode members have adopted the follo
software security testing practices in their software development lifecycle. The
is not to "test in security," but rather to validate the robustness and secur
the software products prior to making the product available to customers. T
testing methods do find security bugs, especially for products that may not
undergone critical secure development process changes.

Fuzz testing

Fuzz testing is a reliability and security testing technique that relies on bu
intentionally malformed data and then having the software under test consume the
malformed data to see how it responds. The science of fuzz testing is somewhat
new but it is maturing rapidly. There is a small market for fuzz testing tools today,
but in many cases software developers must build bespoke fuzz testers to suit specialized file and network data formats. Fuzz testing is an effective testing technique because it uncovers weaknesses in data handling code.

Resources

- Fuzz Testing of Application Reliability, University of Wisconsin; http://pages.cs.wisc.edu/~bart/fuzz/fuzz.html
- Automated Whitebox Fuzz Testing, Michael Levin, Patrice Godefroid and Dave Molnar, Microsoft Research; ftp://ftp.research.microsoft.com/pub/tr/TR-2007-58.pdf
- IANewsletter Spring 2007 "Look out! It's the fuzz!" Matt Warnock; http://iac.dtic.mil/iatac/download/Vol10_No1.pdf
- *Fuzzing: Brute Force Vulnerability Discovery*. Sutton, Greene & Amini, Addison-Wesley.
- *Open Source Security Testing Methodology Manual*. ISECOM.
- *Common Attack Pattern Enumeration and Classification*, MITRE; http://capec.mitre.org/

SAFECode
Software Assurance Forum for Excellence in Code
Driving Security and Integrity

Fundamental Practices for Secure Software Development

A Guide to the Most Effective Secure Development Practices in Use Today

OCTOBER 8, 2008

LEAD WRITER Michael Howard, Microsoft Corp.

CONTRIBUTORS
Gunter Bitz, SAP AG
Jerry Cochran, Microsoft Corp.
Matt Coles, EMC Corporation
Danny Dhillon, EMC Corporation
Chris Fagan, Microsoft Corp.
Cassio Goldschmidt, Symantec Corp.
Wesley Higaki, Symantec Corp.
Steve Lipner, Microsoft Corp.
Brad Minnis, Juniper Networks, Inc.
Hardik Parekh, EMC Corporation
Dan Reddy, EMC Corporation
Alexandr Seleznyov, Nokia
Reeny Sondhi, EMC Corporation
Janne Uusilehto, Nokia
Antti Vähä-Sipilä, Nokia

CWE
CAPEC

Linkage with Fundamental Changes in Enterprise Security Initiatives

Twenty Critical Controls for Effective Cyber Def
Guidelines

What the 20 CSC Critics say...

20 Critical Security Controls - Version 2.0

- 20 Critical Security Controls - Introduction (Version 2.0)
- Critical Control 1: Inventory of Authorized and Unauthorized
- Critical Control 2: Inventory of Authorized and Unauthorized
- Critical Control 3: Secure Configurations for Hardware and Sc
  Servers
- Critical Control 4: Secure Configurations for Network Devices
- Critical Control 5: Boundary Defense
- Critical Control 6: Maintenance, Monitoring, and Analysis of
- Critical Control 7: Application Software Security
- Critical Control 8: Controlled Use of Administrative Privilege
- Critical Control 9: Controlled Access Based on Need to Know
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro
- Critical Contro

CAG: Critical Control 7: Application Software Security

<< previous control | Consensus Audit Guidelines | next control >>

How do attackers exploit the lack of this control?

Attacks against vulnerabilities in web-based and other application software have been a top priority for criminal organizations in recent years. Application software that does not properly check the size of user input, fails to sanitize user input by filtering out unneeded but potentially malicious character sequences, or does not initialize and clear variables properly could be vulnerable to remote compromise. Attackers can inject specific exploits, including buffer overflows, SQL injection attacks, and cross-site scripting code to gain control over vulnerable machines. In one attack in 2008, more than 1 million web servers were exploited and turned into infection engines for visitors to those sites using SQL injection. During that attack, trusted websites from state governments and other organizations compromised by attackers were used to infect hundreds of thousands of

CWE and CAPEC included in Control 7 of the "Twenty Critical Controls for Effective Cyber Defense: Consensus Audit Guidelines"

Procedures and tools for implementing t

Source code testing tools, web application security scanning tools, and object code testing tools have proven useful in securing application software, along with manual application security penetration testing by testers who have extensive programming knowledge as well as application penetration testing expertise. The Common Weakness Enumeration (CWE) initiative is utilized by many such tools to identify the weaknesses that they find. Organizations can also use CWE to determine which types of weaknesses they are most interested in addressing and removing. A broad community effort to identify the "Top 25 Most Dangerous Programming Errors" is also available as a minimum set of important issues to investigate and address during the application development process. When evaluating the effectiveness of testing for these weaknesses, the Common Attack Pattern Enumeration and Classification (CAPEC) can be used to organize and record the breadth of the testing for the CWEs as well as a way for testers to think like attackers in their development of test cases.

Homela
Security

SC7
WG3

ISO/IEC JTC 1/SC 27 **Nxxxx**

ISO/IEC JTC 1/SC 27/WG x **Nxxxxx**

REPLACES: N

ISO/IEC JTC 1/SC 27

Information technology - Security techniques

Secretariat: DIN, Germany

DOC TYPE:	NB NWI Proposal for a technical report (TR)
TITLE:	**National Body New Work Item Proposal on "Secure software development and evaluation under ISO/IEC 15408 and ISO/IEC 18405"**
SOURCE:	INCITS/CS1, National Body of (US)
DATE:	2009-09-30
PROJECT:	**15408 and 18405**
STATUS:	This document is circulated for consideration at the forthcoming meeting of SC 27/WG 3 to be held in Redmond (WA, USA) on 2nd – 6th November 2009.
ACTION ID:	ACT
DUE DATE:	
DISTRIBUTION:	P-, O- and L-Members W. Fumy, SC 27 Chairman M. De Soete, SC 27 Vice-Chair E. J. Humphreys, K. Naemura, M. Bañón, M.-C. Kang, K. Rannenberg, WG-Conveners
MEDIUM:	Livelink-server
NO. OF PAGES:	xx

Secretariat ISO/IEC JTC 1/SC 27 –
DIN Deutsches Institut für Normung e. V., Burggrafenstr. 6, 10772 Berlin, Germany
Telephone: + 49 30 2601-2652; Facsimile: + 49 30 2601-1723; E-mail: krystyna.passia@din.de;
HTTP://www.jtc1sc27.din.de/en

Common Criteria v4 CCDB

- TOE to leverage CAPEC & CWE
- Also investigating how to leverage ISO/IEC 15026

NIAP Evaluation Scheme

- Above plus
- Also investigating how to leverage SCAP

New Work Item Proposal

NP submitting

PROPOSAL FOR A NEW WORK ITEM

Date of presentation of proposal: YYYY-MM-DD	Proposer: ISO/IEC JTC 1 SC27
Secretariat: National Body	**ISO/IEC JTC 1 N** XXXX ISO/IEC JTC 1/SC 27 N

A proposal for a new work item shall be submitted to the secretariat of the ISO/IEC joint technical committee concerned with a copy to the ISO Central Secretariat.

Presentation of the proposal

Title Secure software development and evaluation under ISO/IEC 15408 and ISO/IEC 18405

Scope

In the case where a target of evaluation (TOE) being evaluated, under ISO/IEC 15408 and ISO/IEC 18405, includes specific software portions, the TOE developer may optionally present the developer's technical rationale for mitigating software common attack patterns and related weaknesses as described in the latest revision of the Common Attack Pattern Enumeration and Classification (CAPEC) available from http://capec.mitre.org/. The developer's technical rationale is expected to include a range of mitigation techniques, from architectural properties to design features, coding techniques, use of tools or other means.

This Technical Report (TR) provides guidance for the developer and the evaluator on how to use the CAPEC as a technical reference point during the TOE development life cycle and in an evaluation of the TOE secure software under ISO/IEC 15408 and 18045, by addressing:

a) A refinement of the IS 15408 Attack Potential calculation table for software, taking into account the entries contained in the CAPEC and their characterization.

b) How the information for mitigating software common attack patterns and related weaknesses is used in an IS 15408 evaluation, in particular providing guidance on how to determine which attack patterns and weaknesses are applicable to the TOE, taking into consideration of

1. the TOE technology;
2. the TOE security problem definition;
3. the interfaces the TOE exports that can be used by potential attackers;
4. the Attack Potential that the TOE needs to provide resistance for.

c) How the technical rationale provided by the developer for mitigating software common attack patterns and related weaknesses is used in the evaluation of the TOE design and the development of test cases.

d) How the CAPEC and related Common Weakness Enumeration (CWE) taxonomies are used by the evaluator, who needs to consider all the applicable attack patterns and be able to exploit specific related software weaknesses while performing the subsequent vulnerability analysis (AVA_VAN) activities on the TOE.

e) How incomplete entries from the CAPEC are resolved during an IS 15408 evaluation.

f) How the evaluator's attack and weakness analysis of the TOE incorporates other attacks and weaknesses not yet documented in the CAPEC.

The TR also investigates specific elements from the ISO /IEC 15026 (and its revision) are applicable to the guidelines being developed in the TR within the context of IS 15408 and 18405.

ISO/IEC JTC 1/SC 27/WG 3, NWP
"Secure software development and evaluation under ISO/IEC 15408 and ISO/IEC 18045"

HS SEDI
Homeland Security Systems Engineering and Development Institute

- The way how the CAPEC and related CWE taxonomies are to be used by the developer, which needs to consider and provide sufficient and effective mitigation to all applicable attacks and weaknesses.

- The way how the CAPEC and related CWE taxonomies are to be used by the evaluator, which needs to consider all the applicable attack patterns and be able to exploit all the related software weaknesses while performing the subsequent AVA_VAN activities.

- How incomplete entries from the CAPEC are to be addressed during an evaluation.

- How to incorporate to the evaluation attacks and weaknesses not included in the CAPEC.

CAPEC Future Plans

HS SEDI
Homeland Security Systems Engineering and Development Institute

- V1.7 (within the next month or two)
  - Will flesh out ~30-40 stub patterns to full patterns
  - Will include existing content that has been refined for quality & consistency
  - Will incorporate initial use of the Observables sub-schema
- Strategic focus for the near to mid-term will be on utilizing CAPEC as a bridge between secure development and secure operations
- Continue expanding and refining content
- Continue expanding outreach and supporting CAPEC use
- Establish initial compatibility program

Questions?

sbarnum@mitre.org

Homeland Security