

Tolerating Change in a Secure Environment: A Visual Perspective



Shawn Bohner
Virginia Tech

May 15, 2007



Problem

Common Criteria Evaluation Dilemma

- **Common Criteria Security Evaluations (CCSE)**
Demand exceeding supply of Evaluators
 - Labor intensive CCSE process
 - Effort in Weeks and Calendar time in Months
 - National Information Assurance Acquisition Policy (NSTISSP #11) July 2002 mandate for security related software evaluation
 - Limited number of Testing Labs
 - And then there are all the software updates...
- **How can this situation be alleviated?**
 - Relax policy & allow lesser/non-evaluated systems
 - Increase supply of Evaluators
 - Increase the productivity of Evaluators ⚠

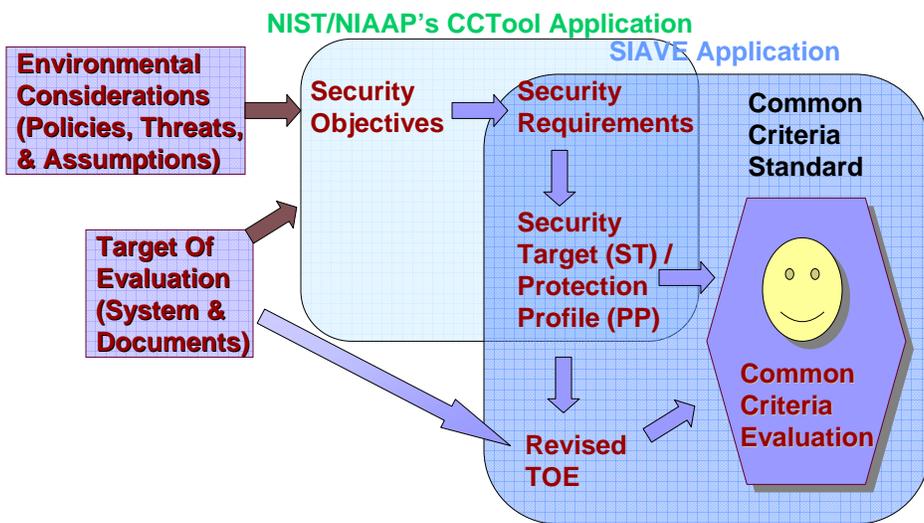


Quicken and Clarify CCSE

- **Improve Efficiency of CCSE Process through Better Navigation**
 - Reduce time in navigating the documentation (shorten the conceptual distances)
 - Reduce effort and time by identifying failing evaluations early
 - Reduce time for key time consuming activities
- **Improve Effectiveness of CCSE Process through Better Visibility**
 - Increase confidence of evaluations
 - Better decisions



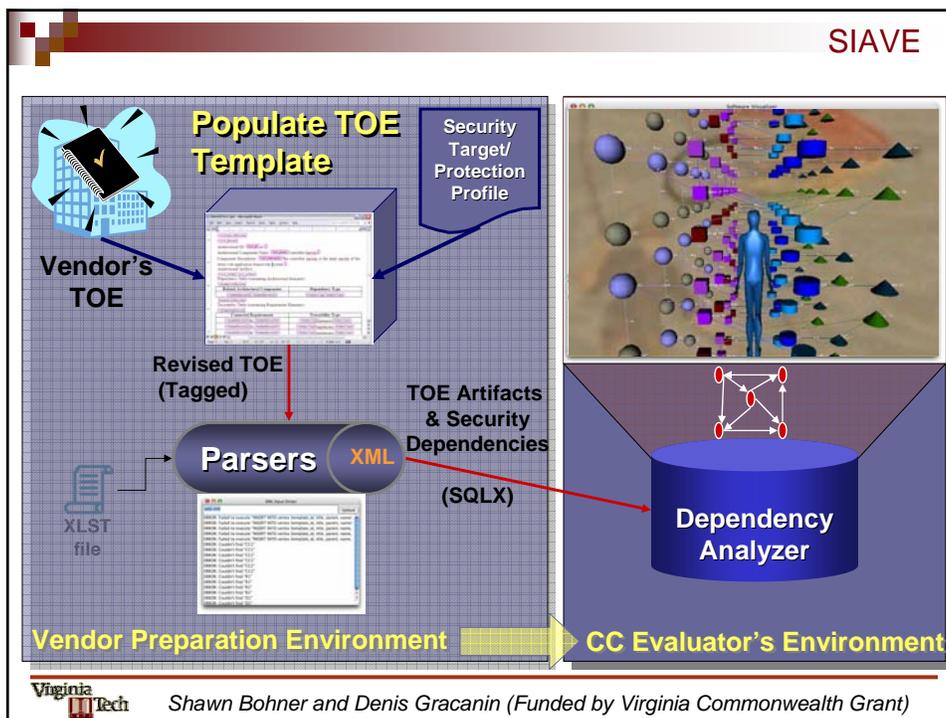
CCSE via Security Impact Analysis Virtual Environment



Shawn Bohner and Denis Gracanin and Funded by Virginia Commonwealth Grant

The SIAVE Research Vision

- Vendor Uses CCTool to Generate ST/PP
- ST/PP Used to Generate TOE Template in Vendor's Documentation Environment
- TOE Template Populated and Updated to form the Revised TOE
- Revised TOE Transformed into Software Life Cycle Objects that Populate the Database along with Dependency Relationships
- CC Evaluator Analyzes and Navigates Security Dependency Database in an Immersive Virtual Environment



Shawn Bohner and Denis Gracanin (Funded by Virginia Commonwealth Grant)

Technical Approach

- **Employ Complementary Technologies**
 - Software Impact Analysis (dependency based)
 - Software Visualization / Virtual Environments
- **Two Phase Approach– Evaluator then Vendor**
- **Phase 1: Automation for Evaluator’s Tasks**
 - Security Impacts Model to Analyze Relevant Dependencies
 - Visual Environment for Evaluators
- **Phase 2: Automate TOE capture for Vendors**
 - Build on CCTool to derive TOE templates
 - Start with common Vendor Documentation Tools
 - Templates & Parsers for TOE Capture
 - ST/PP derived TOE Template Generation
 - Capture & Revise TOE in Vendor friendly tools
 - MS Word to XML translation & DBMS population



TOE Template

SIAVEDTSv3.doc - Microsoft Word

File Edit View Insert Format Tools Table Window Help

ArchitecturalElements

Arch_Element

Architectural ID: Arch_ID(A1)

Architectural Component Name: Arch_Name(Controller Servlet)

Component Description: Arch_Description(The controller servlet is the main servlet of the struts web application framework system)

Architectural Artifact:

Arch_Artifact(Arch_Artifact)

Dependency Table (containing Architectural elements):

RelatedArchElements

| Related Architectural Components | Dependency Type |
|--|--------------------------------|
| RelatedElementID()RelatedElementID() | RelationType()RelationType() |

RelatedArchElements

Traceability Table (containing Requirement Elements):

RelatedReqElements

| Connected Requirements | Traceability Type |
|---|---|
| RelatedElementID(R2)RelatedElementID() | RelationType(Interacts)RelationType() |
| RelatedElementID(R4)RelatedElementID() | RelationType(Implements)RelationType() |
| RelatedElementID(R7)RelatedElementID() | RelationType(Implements)RelationType() |

Page 9 Sec 1 9/17 At 3.9" Ln 12 Col 35 REC TRK EXT OVR English (U.S.)

Visualization and Navigation

Software Visualizer

Name: Integrity Requirements [Design]

Title: R6

Description:
The system should be designed to maintain the integrity of all communication that occur between the system and intended external system

Strength Threshold:

Transitive Limit:

Basic Architecture

SIAVE Prototype

Evaluation Assurance Levels

- **EAL1– Functionally Tested:** Basic assurance of security by analyzing functional specifications and guidance.
- **EAL2– Structurally Tested:** Moderate level of assurance by EAL1 plus high-level design and independent testing of the security functions for vulnerability assessment.
- **EAL3– Methodically Tested and Checked:** Provides moderate level of assurance by including EAL2 plus evidence of sound development practices.
- **EAL4– Methodically Designed, Tested and Reviewed:** Moderate/high level of assurance - highest level economically feasible to retrofit an existing product line.
- **EAL5– Semiformally Designed and Tested:** Provides security engineering based upon rigorous commercial development practices to ensure resistance to attackers.
- **EAL6– Semiformally Verified Design and Tested:** High assurance through security engineering techniques in a rigorous development environment to reduce risks.
- **EAL7– Formally Verified Design and Tested:** Highest assurance level - requires formal design verification.



Status and Next Steps

- Completed two Phases of prototype of Evaluator's Visual Environment
- Populated SIAVE with Initial Test TOE
- Refining VE used to Analyze and Navigate TOE artifacts during Evaluation
- **Next frontier** is to introduce Formalism
- Moving into EAL 5-7 with formal specifications
 - Build on Lamsweerde's constructive approach to the modeling, specification, and analysis of application-specific security requirements
 - Consider Specifying Systems in B or VDM++
- Engaging Testing Lab to use live TOE and explore SBIR possibilities



