Tools for Information Security Assurance Arguments

Joon S. Park, Bruce Montrose, and Judith N. Froscher
Center for High Assurance Computer Systems
Naval Research Laboratory
{jpark, montrose, froscher}@itd.nrl.navy.mil

Abstract

To design a system that can be trusted or assess security properties in a system, the related assurance arguments[1] need to be developed and described effectively in an understandable way. To meet this pressing need, we have developed a prototype tool, VNRM (Visual Network Rating Methodology), to help users develop a map to assurance arguments and document it with related descriptions in a common environment. This map depicts the claim trees for the assurance arguments related to the enterprise security objective. VNRM supports ECM (Enterprise Certification Methodology) for deriving and organizing the related assurance arguments effectively and uses CAML (Composite Assurance Mapping Language) for describing the assurance arguments in the map. After the successful development of VNRM, we have started to develop a more robust tool, SANE (Security Assurance Navigation and Environment), providing more features, reusability of assurance arguments, and access control to CAML maps.

1. Introduction

The emergence of new information technology has changed life styles and the world economy. Information Assurance (IA) problems have become more important as the world uses more information systems, since the security and survivability of information systems directly or indirectly affect organizations, which rely on information technology.

Hence, it is obvious that we need effective and efficient methods for system design and assessment, providing causalities, relationships, vulnerabilities[2], threats[3], system-level viewpoints, and objectives of an entire enterprise. Designers (developers) and assessors (evaluators) need to clearly understand the problems at hand and describe them using a common environment and tools. Particularly, when we design or assess a large enterprise where many components are integrated with and dependent on each other, it is difficult to extract the necessary assurance arguments for components and describe their relationships and dependencies. To design a trusted system or assess security properties in a system, the related assurance arguments need to be developed and described efficiently in an understandable way by means of an easy-to-use language based on a sound methodology so that decision makers can make informed decisions. This can reduce lifecycle cost through early assurance discussions between designers and assessors, and reduce the need for expensive backtracking.

A variety of assurance techniques and methodologies are used to provide the confidence that a system or component satisfies its specification. As computing capabilities have evolved from closed timesharing environments to widely distributed computing resources that provide worldwide connectivity, security countermeasures have also migrated to a combination of protection, detection, response, recovery, and survivability mechanisms. Traditional assurance approaches (e.g., formal methods) do not scale for comprehensive enterprise assurance and have seldom been used to reason about detection and response, recovery, or survivability. Assurance techniques that are affordable, expressive, and effective are needed for enterprise assurance arguments.

In this paper, we introduce tools to satisfy the above requirements: VNRM (Visual Network Rating Methodology) and SANE (Security Assurance Navigation and Environment). SANE is a successor to VNRM and a standalone tool (does not require external programs as VNRM does), providing assurance argument reusability and other features. VNRM and SANE help users to derive and organize the related assurance arguments effectively based on our methodology, ECM (Enterprise Certification Methodology), and build a map of assurance arguments in our language, CAML (Composite Assurance Mapping Language). This map depicts the claim trees for the

[1] In this paper, we use the terms *information security assurance arguments*, *assurance arguments* and *security arguments* interchangeably.
[2] A vulnerability refers to a weakness of a system that can be exploited to undermine the availability, integrity, confidentiality, and/or authenticity of the information resources and/or the system itself.
[3] A threat refers to a circumstance, thing, person, or event - which may occur independent of the system - with the potential to violate the system security or assets.

assurance arguments related to the enterprise security objective, providing causalities, relationships, vulnerabilities, threats, and other system and environment related issues. Our tools also help users to document the CAML maps with related descriptions in a common environment.

This paper is organized as follows. In Section 2, we briefly explain the existing technologies that we applied to and extended for our work. In Section 3, we introduce an information security assurance argument map, including our underlying methodology (ECM), language (CAML) to build the map, and an example of the map. Section 4 describes VNRM. Section 5 describes SANE that we have been extending on to provide more services. In Section 6, we compare our work to the existing assurance technologies, including Systems Security Engineering Capability Maturity Model (SSE-CMM) and Common Criteria (CC). Finally, Section 7 concludes this paper.

2. Related Work

Systems engineers use fault trees [RVH81] to reason about the safety of composite safety critical systems. Fault trees are a graphic representation of a combination of events that can cause an undesired event to occur. They are developed for analyzing and assessing system dependability. In this approach, all the components and their failure probabilities are represented, which consequently cause the failure of the whole system. An event at level l_i is reduced to the combination of lower levels, through some logic ports. Researchers at the University of York and Defense Research Agency (DRA) applied the fault tree approach to the Goal Structuring Notation (GSN, [WKC97]), providing the breakdown of safety requirements to arguments based upon available evidence, to describe the safety arguments and safety assessments. They developed a PC based tool, Safety Argument Manager (SAM, [WMKF96]), which supports GSN.

Researchers at the University of Virginia (UVA) developed the Methodically Organized Argument Tree (MOAT, [KW96]). MOAT provides hierarchically organized arguments in a tree for each security property that is to be analyzed. Each MOAT encapsulates an argument that the system exhibits some desired security property. The MOAT representation looks similar to fault trees. However, the MOAT methodology permits the integration of existing security assurance techniques, such as formal methods, into a risk[4]-driven process model. It orders the construction of assurance arguments using a high-risk-first heuristic, and explores higher-risk areas more fully and precisely than lower-risk areas.

The National Security Agency (NSA) developed a matrix based methodology, Network Rating Methodology (NRM, [BGB97]), an extension of their previous work, Network Rating Model [LBB96], for assessing and evaluating network security, either in operation or in development, based upon a defined set of characteristics. NRM notation is based on a matrix representation. Each row represents an area of security concerns, such as confidentiality, integrity, availability, and authenticity. Each column represents a security service, such as personnel, operational procedure, technology, and physical environment. Each cell is filled with corresponding claims or evidence for the related area of security concern and service. The matrix is multi-dimensional depending on the granularity required by the system owner. In other words, lists of evidence in the next level of the matrix are given to support the first-level claims.

Researchers at the Naval Research Laboratory introduced the assurance strategy [PFL93, FNH94] in 1993. The assurance strategy concerns not only the components (hardware and software) of a system but also the environment in which the components operate. They identify assumptions and assertions that reflect information security requirements for systems and use those concepts to document the trade-off decision. In this approach, assertions are predicates that are enforced by the system, while assumptions are predicates that are enforced in the system's environment. The assurance strategy initially documents the set of assumptions and assertions derived from the requirements. It is elaborated and refined throughout the development, yielding the assurance argument, delivered with the system, which provides the primary technical basis for the certification decision. If assumptions for any discipline are identified that do not correspond to assertions for some other entity, then these assumptions represent vulnerabilities in using the system.

3. Information Security Assurance Argument Map

An information security assurance argument map depicts the claim trees for the assurance arguments related to the enterprise security objective, providing causalities, relationships, vulnerabilities, threats, and other system and environment related issues. In this section, we introduce the underlying methodology and language that we use in our tools to derive the assurance arguments comprehensively and organize them efficiently.

[4] A risk is a cost estimate based on the probability of a successful attack and the value of the vulnerable asset. For instance, when the probability of successful attack is high and the value of the vulnerable asset is high, the risk is high, and vice versa.

3.1. Underlying Methodology

To provide a good methodology for users to produce an efficient and comprehensive description of information security assurance arguments, we organize the arguments in four different disciplines in a map based on the NRM approach (described in Section 2): physical, personnel, technical, and operational. Physical Security involves the strength of physical mechanisms and structures used to protect and house the technology, such as strength of locks or safes. Operational Security involves the effectiveness of manual procedures, policies, and guidelines for handling and protecting information. The more conventional view of assurance comes from Technological Security, which involves security about combinations of hardware, software, and communications. Finally, Personnel Security involves assurance about people, their trustworthiness and capabilities through some processes, such as personnel background investigation, training, and evaluation. We do not propose to "hard code" the precise partitioning of the four disciplines in the argument maps for every application. Different systems or applications may have a different partitioning of their arguments for improved understandability or a different level of assurance. However, we believe the four different disciplines provide comprehensive categories for most applications or systems.

We extend and apply our assurance strategy (described in Section 2) to the four security disciplines in an assurance argument map to elaborate the high-level enterprise security objectives. We call this the Enterprise Certification Methodology (ECM). Based on this methodology, the map can be used to develop a rigorous specification of the security posture of the entire enterprise. As a result, an assurance argument map permits tracking security vulnerabilities by identifying assumptions made in one argument tree that are not matched by validating claims made in another argument tree in the map. In other words, a statement can be an assumption in one discipline while it can be a claim in another discipline. For instance, the statement "Users know authentication procedures" is an assumption under the technical discipline, while it is a claim under the personnel discipline. Such assumptions become *dependencies* of the argument. Assumptions that are not so linked become vulnerabilities that need to be considered when assessing residual risk. These vulnerabilities must be assessed when deciding whether the residual risk is tolerable in the operational environment. The map developer also needs to ensure the integrity of the assumption validation mapping itself. If a portion of the map for one application is reused for another application, the reader may notice the lack of any claims to ensure consistent application of the assumptions. Conscientious application of the above approach helps to uncover such gaps, identifying security vulnerabilities that were not previously considered.

3.2. A Language for Information Assurance Arguments

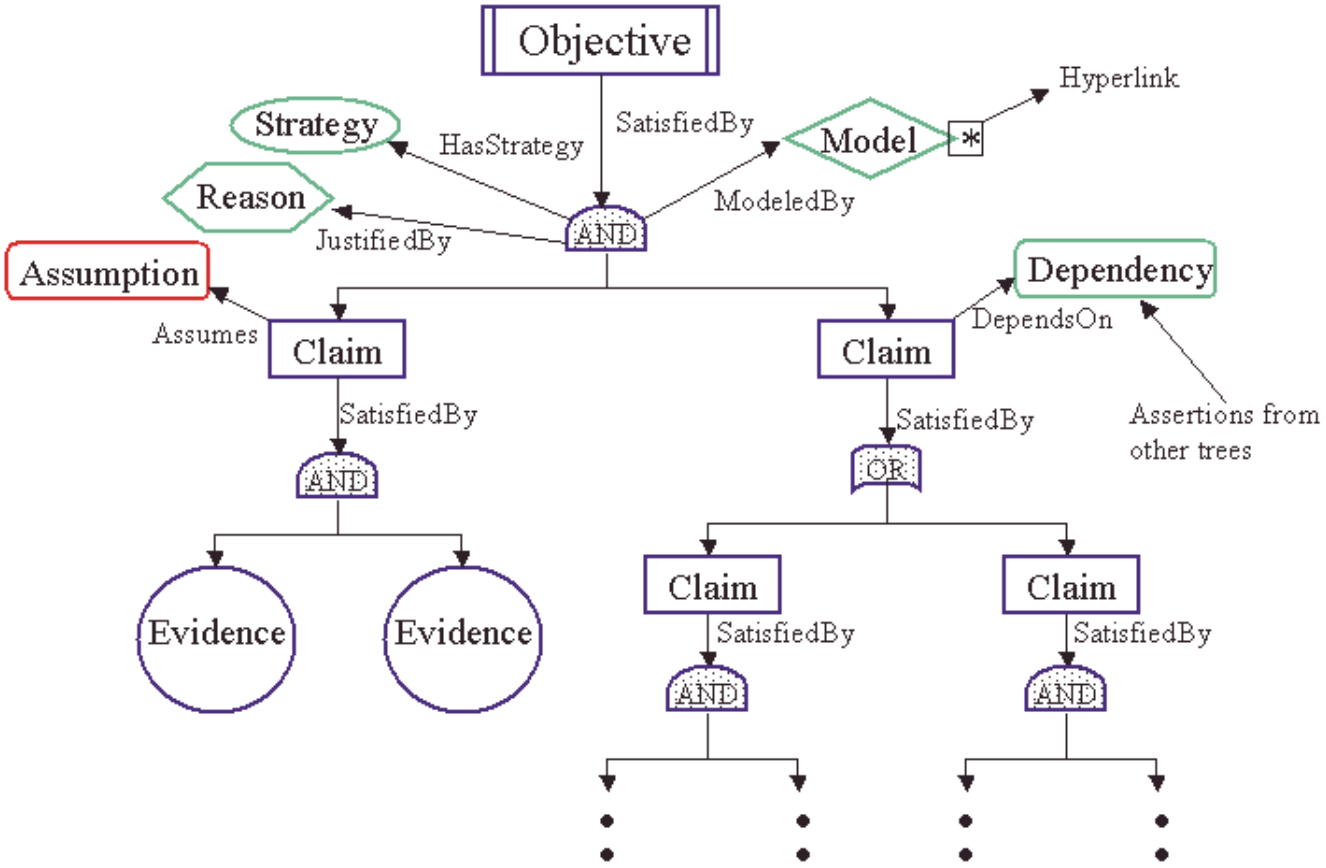
We have developed CAML (Composite Assurance Mapping Language) by merging and extending several existing technologies (described in Section 2) to describe information security assurance arguments effectively in a well-organized format. In this section, we briefly describe the rules and components of CAML. The detailed description and usage of CAML is available in [MMS00]. Figure 1 shows an example of CAML structure with its primitives and definitions.

Distinct graphical primitives in different shapes represent key components of the argument map. A textual summary of each component is shown inside each shape. The spine of an argument map hierarchically refines security *claims* about the system into sub-claims that, eventually, are linked with the *evidence* that a claim is satisfied. The flesh of an argument map describes supporting information about the refinement such as the general *strategy*, *assumptions* and *dependencies*, justifying *reasons*, and contextual *models*. Spine refinement may proceed using either *AND*-decomposition or *OR*-decomposition. By the AND-decomposition, all sub-claims or evidence must hold for the decomposed claim to hold. By the OR-decomposition, one of the sub-claims or evidences must hold for the decomposed claim to hold.

Not all CAML components are needed for every assurance map. Map developers use their discretion for choosing the necessary components to convey their argument satisfactorily. When a flesh component is connected to an AND/OR connector, it means this flesh component applies to all the arguments below the AND/OR connector. When a flesh component is connected to a spine (claim or evidence) directly, it means the flesh component applies to the particular spine. To provide more detailed descriptions of the map, the shapes can be hyperlinked. For instance, architectural diagrams can be hyperlinked to model shapes, and analytic proofs and tests can be hyperlinked to evidence shapes.

CAML provides a medium through which to develop a concise roadmap of the evidence that an enterprise conforms to required security properties. Resolving the competing factors involved with secure enterprise design typically requires depending on diverse assurances that are interrelated in complex ways. Users coherently and consistently map out the assurance evidence and its associated reasoning using CAML maps. The language helps to tame the complex inter-relationships of the diverse sources from which that evidence originates, and to

illuminate the process of balancing mission, security, and cost. Such a common language is critical as a communication medium for risk analysts, developers and evaluators. It will allow them to come to a consistent understanding and agreement on assurance requirements and how an enterprise satisfies those requirements. Well-constructed CAML maps contain rationale that promotes confidence in their own consistency and completeness. Additionally, getting early evaluator feedback in the assurance mapping process helps to ensure that the final enterprise design will be approved for operation with minimal changes.



Definitions

- Objective: A statement expressing a security requirement of the countermeasure, system, network or enterprise that is the reference of an argument.
- Claim: Statements that associate subjects with their attributes or properties.
- Assumption: A claim that is accepted without justification.
- Dependency: An assumption in one part of an argument that is validated by a claim in another part of the argument.
- Evidence: Data on which a judgment or conclusion about an assurance claim may be based.
- Hyperlink: A link from one component of an argument map to other components of the map or external components to provide for the detail or clarification to the argument.
- Strategy: The approach taken for refining a claim into sub-claims or into evidences supporting the claim.
- Reason: A set of statements that ties together a set of sub-claims or evidences to establish a claim.
- Model: The architectural context on which a claim is based.

[Figure 1] An Example of CAML Structure with Its Primitives and Definitions

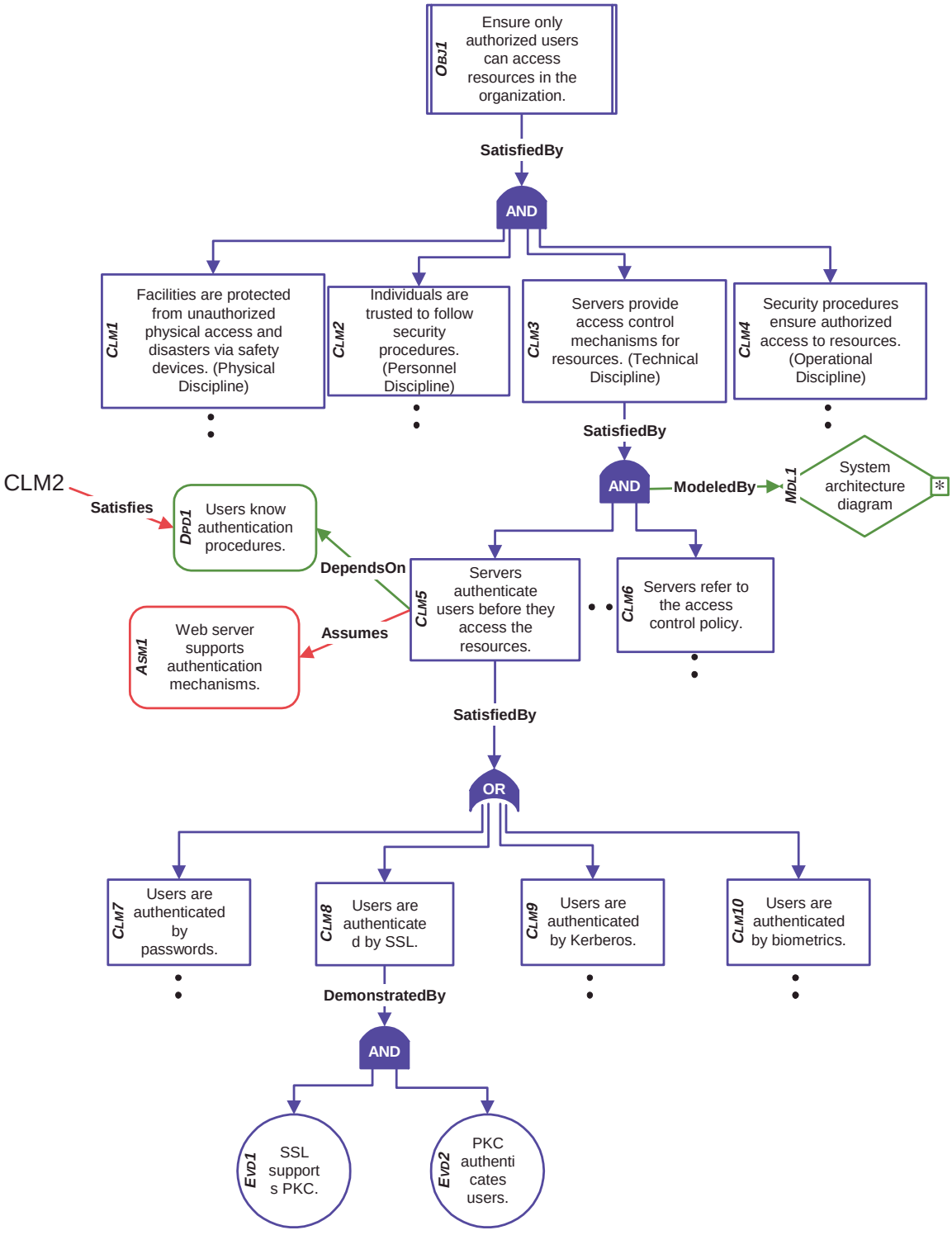


Figure 2. An Example of a CAML Map Developed Using VNRM

3.3 An Example of a CAML Map

In this section, we show an example of a CAML map (denoted in Figure 2) that we developed using VNRM (described in Section 4) based on ECM (described in Section 3.1) to illustrate the approach. Because of space constraints, we captured only part of the actual work in this paper. The black dots around the claim boxes represent the omitted parts in the figure. In the original VNRM environment, distinct graphical primitives are denoted in different colors as well as in different shapes. Note that a different person may develop a different assurance map from others for the same system.

The purpose of Figure 2 is to show the arguments related to the objective, "Ensure only authorized users can access resources in the organization." As we introduced in Section 3, we partition the related security arguments into the four disciplines: physical, personnel, technical, and operational security disciplines. Each discipline has a high level claim (CLM1 through CLM4 in the map), which is satisfied by sub-claims and related arguments. Those claims are connected by an AND-connector because all of them must hold for the decomposed objective to be satisfied. The decompositions for physical, personnel, and operational disciplines are omitted in this particular example. The claim for the physical security discipline (CLM1) would be decomposed with arguments related to physical entry to the facility, storage cabinets, safety devices, etc. The claim for the personnel security discipline (CLM2) would be decomposed with arguments related to users, administrators, guards, etc. The claim for the operational security discipline (CLM4) would be decomposed with arguments related to organizational policies, guidelines, etc.

In this example, we decompose the claim for the technical security discipline (CLM3). A system architecture diagram is provided by a Model shape (MDL1) that is hyperlinked (denoted by * symbol in MDL1) to external information (not shown in this paper). Technically, the servers authenticate users before users access the resources (CLM5), where we assume that users know the authentication procedures and the Web server supports authentication mechanisms. The former assumption is denoted in a Dependency shape (DSD1) because it has a verifying link[5] (CLM4) to a different partition (personnel discipline), while the latter assumption is denoted in an Assumption shape (ASM1), which still shows vulnerabilities, because it does not have verifying links (described in Section 3). Additionally, the servers refer to the access control policy of the organization to make an access control decision (CLM6). Since CLM5 through CLM6 are required to hold for CLM3 to be true, they are connected by an AND-connector. Continuing the decomposition, CLM5 is decomposed into four possible authentication techniques (CLM7 through CLM10) by means of an OR-connector. This means that one of those authentication techniques must be provided. Based on a given situation, we may add more authentication techniques under the OR-connector or remove some of them. Finally, the claim CLM8 (Users are authenticated by SSL) is decomposed into two evidences via an AND-connector. Those evidences demonstrate why we can trust that SSL can authenticate users. Similarly, the other claims can be decomposed into evidence.

[5] If the claim for the personnel security discipline would be decomposed more in this example, this assumption would be linked to more refined claims.

4. Visual Network Rating Methodology (VNRM)

We have developed a prototype toolset, VNRM (Visual Network Rating Methodology), to help users draw a graphical assurance argument map in CAML (described in Section 3.2) based on ECM (described in Section 3.1) that evaluates whether the target system adequately supports security services. The VNRM User's manual [MS00] is helpful in getting one started on using the tool. Additional information, including a packaged demonstration, is available at the VNRM website [VNRM00].

The VNRM toolset was built to be easy to use, maintain, and extend. Its design supports ease-of-use by providing graphical user interfaces that are familiar to a large portion of the potential user community. The design uses standard, widely accepted software components that support both the need for familiar interfaces and the need to maintain compatibility of VNRM with cutting edge technology. As well-supported software components evolve, VNRM can evolve in like manner with a minimal amount of effort. Finally, extensibility is important to the evolving VNRM design so that when users identify additional, value-added functions, the toolset can be extended easily with minimal changes to the existing implementation. The design supports extensibility primarily through the use of client-server and modular design techniques.

Figure 3 depicts the VNRM tool architecture. Solid lines represent invocation and message passing between VNRM components and dotted lines represent interactions between human beings and VNRM components. The VNRM tool consists of three major components: VNRM Explorer, VNRM Tool Library, and VNRM Database (VNDB). The VNRM Explorer, an interface that has the familiar look and feel of the Microsoft Windows Explorer, provides a user-friendly front-end to VNDB for tools in the VNRM Tool Library. It provides a utility to help manage, build, and review CAML assurance argument maps. The VNDB,

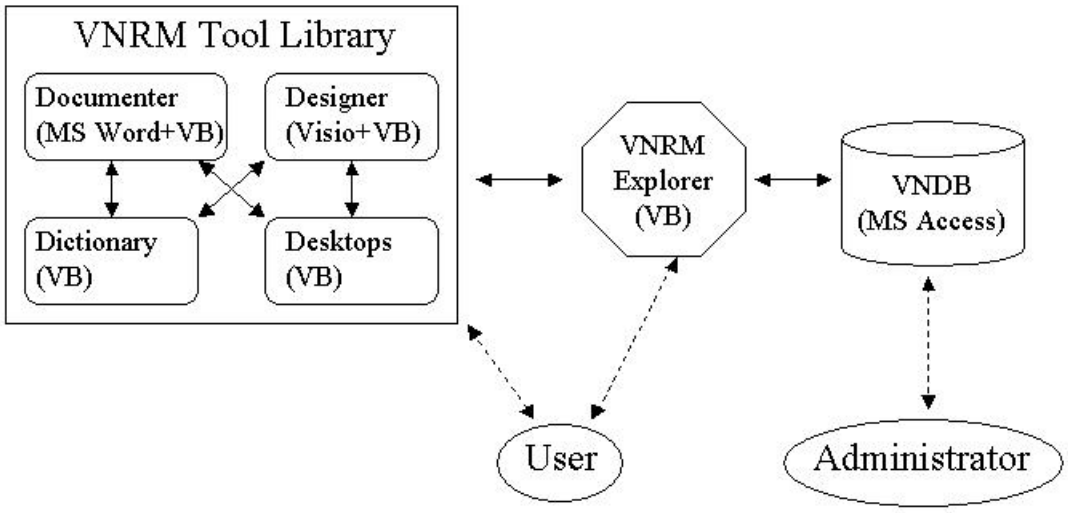


Figure 3. VNRM Tool Architecture

which is implemented in Microsoft Access, stores the artifacts of an assurance argument, including the CAML map and its documentation, on a project-by-project basis. Users manage and access VNRM projects through the VNRM Explorer. Tools in the library can update the VNDB only through the VNRM Explorer so as to preserve the consistency of the VNDB data and the tools' views of those data. The VNRM Explorer notifies any tools that have a need-to-know when data is updated.

Four tools that support creating and documenting CAML argument maps currently reside in the VNRM Tool Library. VNRM Designer uses the Visio extensible drawing package to create, analyze, and hyperlink CAML maps. These maps can be integrated (as OLE links) into textual documents using the VNRM Documenter, which is implemented using Microsoft Word. Both the Visio and Word environments were extended using Visual Basic (VB) to support VNRM-specific function. The VNRM Dictionary permits defining a standard terminology for consistent application across or within VNRM projects. Terms so defined are highlighted in the textual parts of the map and its documentation. Finally, VNRM Desktops provide a virtual desktop (which provides collections of pages that are developed, analyzed, and/or presented as a group) function to associate different segments of an assurance argument map for simultaneous elaboration or examination.

VNRM helps manage the complexity of composition of diverse enterprise assurance arguments by mapping out the assurance evidence, tracing meaningful threads of reasoning, and highlighting significant results. Taking an enterprise view of assurance gives us many places from which to derive confidence that information security is not compromised. It clearly shows how each piece of evidence in the map contributes to the overall argument. Furthermore, it supports traversing the hyperlinked argument and analyzing its security weaknesses. Documenting this rationale is important for understanding why key decisions were made and the impact of modifying the enterprise design has on information assurance.

Another purpose of this tool is to help users identify residual vulnerabilities and experiment with moving them around not only among the system components, but around the whole enterprise. This helps improve security, by keeping the vulnerability away from the threat agent[6], and permits the development of affordable solutions to information security. By making tradeoffs between responsibilities of different security disciplines, we can exchange costs and risks as appropriate for the system under construction.

VNRM can be used for a risk analysis process, an engineering process, and a security evaluation process to develop a secure enterprise. Early risk analysis involves identifying both natural and man-made threats to enterprise assets, and the corresponding consequence to the enterprise if those assets are compromised. This analysis results in a set of protection needs that provide a starting point for assurance mapping analysis. We translate these protection needs into a set of high-level claims to which the enterprise must conform by considering the role that the enterprise plays in meeting the protection needs. A primary goal of an assurance map is to show how and why these claims are satisfied. Assumptions on which this argument is based indicate potential enterprise vulnerabilities, to the extent

[6] The person, organization, or circumstance that implements a threat.

that if not properly validated they indicate a hole in the enterprise assurance argument. The strength of the reasoning and evidence on which the argument is based indicates the confidence that we have identified existing vulnerabilities in the map. Such vulnerability analysis is a critical component of the risk analysis process. Of course, the engineering process must consider other factors in addition to information security during enterprise design. Security protection must be balanced within the overall mission of the enterprise, such as with the associated developmental and operational costs. The way that the user allocates security responsibilities to different disciplines emphasizes certain factors over others.

To prove the feasibility of our work, we released VNRM to evaluate real applications. The DARPA IA Program's Assurance Working Group (AWG) has used VNRM to characterize the strength and effectiveness of two different security technologies: the Object-Oriented Domain-Type Enforcement (OO-DTE, [STM99]) and the Advanced Research Guard for Experimentation (ARGuE, [Eps99]). These efforts have largely been successful, as seen by the growing appreciation of the techniques on which VNRM is based. VNRM is currently being applied to the integration of DARPA IA technologies to implement a Virtual Private Network (VPN) in an enterprise context.

5. Security Assurance Navigation and Environment (SANE)

VNRM was developed using Visual Basic, and is dependent on external programs, such as Visio, MS Access, and MS Word. This requires a specific environment in order to use VNRM. Therefore, to provide higher portability and compatibility, we currently are developing a successor to VNRM, SANE (Security Assurance Navigation and Environment), purely in JAVA. It will provide all the drawing and documenting services without requiring external programs. It will also present new features to designers and assessors, associated with the reusability of assurance arguments, access control to CAML maps, and argument patterns.

Figure 4 shows how SANE helps users build CAML maps in conjunction with the Information Security Assurance Repository (ISAR) on the Web. ISAR contains common assurance objectives, arguments, problems, solutions, patterns, and some other information, which can help users build their CAML maps in SANE. When a user is developing a CAML map, SANE allows the user to browse the information in ISAR via HTTP. If she finds some useful information (e.g., problems and solutions), she can apply it to her map. If not, she is welcome to contribute her work to SANE for building more knowledge in it. The collaboration between SANE and ISAR increases the reusability of previous work by others, saving time, work, and money.

Generally, SANE users are allowed to access the whole portion of CAML maps. In some cases, however, parts of a CAML map should be available only to authorized users. For instance, if a set of arguments includes very sensitive information about the target system, such as threats and vulnerabilities of the key components in the system, only authorized users should be allowed to read the argument set. Probably, a system assessor should be able to access the whole map, but an external consultant should not access the confidential argument set. SANE can satisfy this requirement based on users' roles. When a user logs in to SANE, she is authenticated and assigned to a role (e.g., Assessor, Consultant, etc.). In Figure 4, if the user has the Assessor role, she is allowed to access the whole CAML map, but if she has the Consultant role, she is not allowed to access the confidential information, depicted "Hidden" in the figure.

We implement this access control mechanism using a popular technology, XML (Extensible Markup Language). When a SANE user develops a CAML map, she specifies the required roles (if any) for some parts of the map and saves it in a file. At that time, SANE converts the visual CAML map to XML, including the required roles for some parts of the map. Later, when a user tries to access the map, SANE compares the required roles for some parts in the XML file and the user's current role. If the user has the required role (or has a role that subsumes the required role), SANE shows the corresponding part of the map to the user. However, if she does not have the required role, SANE shows the map to the user, hiding the unauthorized parts.

Without reading a whole portion of the CAML map, it may be hard to make a correct decision. Therefore, we claim that a limited number of users (e.g., final decision makers or senior administrators) should be able to access the whole CAML map and make a final decision, reflecting other SANE users' (whose access privileges may be limited to the same map) comments on the partial maps they are authorized to see.

Our future work will specify reusable assurance argument patterns; abstractions from a concrete recurring solution that solves a problem in a certain context. We will investigate ways to define a notation and theory for specifying and composing argument patterns and will provide them to users through ISAR. Currently, we catalog the sets of assurance arguments from different components and try to extract patterns from them. These patterns will be populated in ISAR with other reusable information for assembling different combinations of mechanisms and their composite assurance arguments. Users will be able to construct and share argument patterns and instantiate and compose them into full-fledged assurance arguments for composite systems.

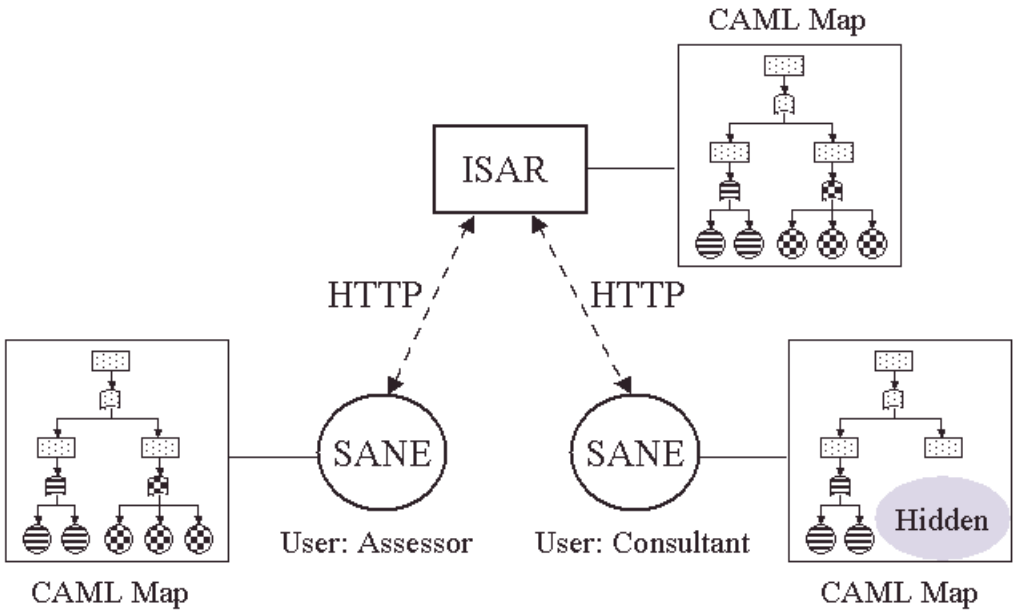


Figure 4. Information Sharing and Hiding by SANE and ISAR

6. Discussion

In this section, we compare our work to the existing information assurance approaches, including formal methods, Common Criteria (CC), and the Systems Security Engineering Capability Maturity Model (SSE-CMM).

Formal methods are extremely useful for proving that high assurance components satisfy their security requirements [KHA99]. However, using formal methods for enterprise assurance is far more challenging for several reasons. First, assurance components are integrated at different levels of abstraction in the enterprise. Secondly, enterprise security depends not only on technical security countermeasures but also on physical protection, administrative procedures, and on user trustworthiness. Using formal methods for specifying these types of countermeasures would prove to be quite difficult. Finally, all of these countermeasures at different levels of abstraction and from different disciplines must be combined to provide the security posture of the whole enterprise. Combining assurance for all countermeasures and specifying and deriving confidence about interdependencies within the enterprise would confound most attempts to formally model enterprise security.

The International Systems Security Engineering Association (ISSEA) has been promoting the use and adoption of the Systems Security Engineering Capability Maturity Model (SSE-CMM, [SSE00]) to measure an organization's capability to provide security products, services, or operations. The SSE-CMM describes an organization's security engineering process and categorizes security practices into base practices and generic practices. The base practices, grouped into process areas, collectively define specific aspects of security engineering. The generic practices, grouped by capability level, represent activities that should be performed as part of doing base practices, indicating process management and institutionalization capability. An SSE-CMM appraisal determines the capability levels for each of the process areas. An organization's capability to perform a particular activity is checked by combining the base practices and generic practices together. The SSE-CMM focuses on how to measure high-level characteristics of an organization's security engineering, but does not prescribe how to evaluate security critical components in the system. On the contrary, our tools (VNRM and SANE) provide a representation scheme for reasoning about enterprise security objectives as well as about security critical components.

The International Organization for Standardization (ISO) has developed a set of international standard security evaluation criteria, Common Criteria (CC, [CC99]), for IT systems and products. It opens the way to worldwide mutual recognition of evaluation results. The CC provides distinct categories for functional requirements (which

define desired security behavior) and assurance requirements (which are the basis for gaining confidence that the claimed security measures are correct). The security properties of the Target of Evaluation (TOE) are captured in the Protection Profile (PP) and Security Target (ST). A PP is intended to be reusable and allows prospective customers and developers to state standardized sets of implementation-independent security requirements and objectives to meet their needs in a category of products or systems. An ST is used by developers to identify the specific security requirements that their specific identified TOE satisfies. An ST is also used by evaluators as the basis for evaluation to determine what the security requirements are and to what extent they are satisfied for the TOE. The CC defines seven Evaluation Assurance Levels (EALs) for ranking the criteria. Evaluators use the Common Evaluation Methodology (CEM, [CEM99]) to evaluate a TOE by the CC. The usefulness of CC depends on the quality of PP and ST. However, the process for writing a good PP and ST has not been thoroughly described. The CC provides only limited guidance on how to write PPs and STs. Furthermore, it is not easy to understanding the CC document in one reading. Consumers, developers and assessors may have a lot of questions about CC evaluations because of the lack of concise information about using the CC.

It is always possible for our tools to integrate with the existing technologies, including the aforementioned approaches. For instance, when a user (of VNRM or SANE) needs to apply some formal proofs, classes in CC, or practices in SSE-CMM to her CAML map, she can easily provide the information via hyperlinks. Our tools can be used to describe the security arguments not only for IT products and systems but also components in the same language (CAML) based on the same methodology (ECM).

7. Conclusions

We have developed VNRM (Visual Network Rating Methodology) to help users develop a map of assurance arguments and document it with related descriptions in a common environment. This map depicts the claim trees for the assurance arguments related to the enterprise security objective. VNRM supports ECM (Enterprise Certification Methodology) for deriving and organizing the related assurance arguments effectively and uses CAML (Composite Assurance Mapping Language) for describing the assurance arguments in the map. After a successful development of a prototype tool, VNRM, we have been building a more robust tool, SANE (Security Assurance Navigation and Environment), providing more features, reusability of assurance arguments, and access control to CAML maps.

8. References

[BGB97] R. Bailey, B. George. C. Bowers, et al. *The Network Rating Methodology: a Framework for Assessing Network Security*. http://chacs.nrl.navy.mil/projects/VisualNRM/nrm_3rd_draft.html, National Security Agency, 1997.

[CC99] *Common Criteria for Information Technology Security Evaluation*. http://www.commoncriteria.org/cc/cc.html, CC Version 2.1, August 1999.

[CEM99] *Common Evaluation Methodology*. http://www.commoncriteria.org/cem/cem.html, CEM Version 1.0, August 1999.

[Eps99] J. Epstein. *Architecture and Concepts of the ARGuE Guard*. Proceedings of 15th Annual Computer Security Applications Conference (ACSAC), Phoenix, Arizona, December 1999.

[FNH94] J. Freeman, R. Neely, and M. Heckard. *A Valid Security Policy Modeling Approach*. Proceedings of 10th Annual Computer Security Applications Conference (ACSAC), Orlando, Florida, December 1994.

[KAH99] J. Kirby, M. Archer, and C. Heitmeyer. *Applying Formal Methods to an Information Security Devices: An Experience Report*. Proceedings of 4th IEEE International Symposium on High Assurance Systems Engineering (HASE), Washington, DC, November 1999.

[KW96] W. Wulf, C. Wang, and D. Kienzle. *A New Model of Security for Distributed Systems*. Proceedings of the New Paradigms in Security Workshop, Lake Arrowhead, California, 1996.

[LBB96] O. Lambros, R. Bailey, C. Bowers, et al. *Network Rating Model: An Approach for Assessing Network Security*, http://www.radium.ncsc.mil/nrm/rev961031.html, National Security Agency, 1996.

[MMS00] A. Moore, Bruce Montrose, and Beth Strohmayer. *A Tool for Mapping Enterprise Security Assurance*. Technical Report 5540-051a:apm, Naval Research Laboratory, September 2000.

[MS00] A. Moore and B. Strohmayer. *Visual NRM User's Mannual*. Technical Report NRL/FR/5540--00-9950, Naval Research Laboratory, May 2000.

[PFL93] C.N. Payne, J.N Froscher, and C.E. Landwehr. *Toward a Comprehensive INFOSEC Certification Methodology*. Proceedings of 16th National Computer Security Conference (NCSC), pages 165-172, Baltimore, MD, September 1993.

[RVH81] N. Roberts, W. Vesely, and D. Haasl. *Fault Tree Handbook*. NUREG-0492, Office of Nuclear Regulatory Research, 1981.

[SSE00] *Systems Security Engineering Capability Maturity Model (SSE-CMM)*. http://www.sse-cmm.org/model.htm, SSE-CMM Model Description Document, Version 2.0, April 1999.

[STM99] D. Sterne, G. Tally, C. McDonell, et al. *Scalable Access Control for Distributed Object Systems*. Proceedings of 8th USENIX Security Symposium, Washington, D.C., August 1999.

[VNRM00] *VNRM (Visual Network Rating Methodology): Tools for Mapping Assurance Arguments*. http://chacs.nrl.navy.mil/projects/VisualNRM/, Naval Research Laboratory, 2000.

[WKC97] S. Wilson, P. Kirkham, and M. Cassano. *SAM 4 User Manual*. University of York, 1997.

[WMKF96] S. Wilson, J. McDermid, P. Kirkham, and P. Fenelon. *The Safety Argument Manager: An Integrated Approach to the Engineering and Safety Assessment of Computer Based Systems*. Proceedings of IEEE Symposium and Workshop on Engineering of Computer-Based Systems, page(s): 198 –205, 1996.