Security Requirements of Components: A Mapping to the Common Criteria

Wes J. Lloyd, Sudipto Ghosh, Indrajit Ray
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523
{wlloyd, ghosh, indrajit}@cs.colostate.edu

Abstract

Components provide the building blocks for developing and delivering software systems in less time and with richer functionality than systems built using traditional software development practices. However, the reliance on using pre-built components may complicate the assessment of the overall level of security provided by the system. Assessment of system security will require security properties of individual components to be considered as well as the composite level of security provided by the component based system. This paper provides an analysis of how the Common Criteria, an internationally recognized standard for security requirements definition and security assessment, can be applied to aid in the specification and evaluation of security for components and component-based systems. By considering the software architecture of the component based system and the mapping of security requirements across the individual components of the system, insight is sought to harness the Common Criteria to assist with the identification and specification of security requirements and the assessment of security for component-based systems.

1. Introduction

The development of software systems in today's world of object-oriented, modularized software is increasingly relying upon the integration and assembly of pre-built off-the-shelf software components in order to fulfill the overall system's functional requirements. By using components to implement various functions of a software system, it is expected that these systems can be developed and delivered faster while exhibiting higher quality [7]. The use of components in building software systems is a methodology known as component-based software development (CBSD). CBSD processes include familiar development activities such as requirements definition, and design specification. In addition, CBSD processes include additional activities to evaluate, test, and select the most appropriate components to meet system requirements [1,15,11]. If software developers are to rely on the use of preexisting components to provide the implementation of vital portions of a system's functionality, then software engineers need to dedicate effort towards the evaluation, testing, and selection of components in order to ensure that the best components are selected. In this paper we consider the use of software components to implement the security requirements for a software system.

Depending on the software system's requirements software engineers will consider various issues in the evaluation and selection of components. Security issues for software systems should be established prior to the start of software development in order for them to be considered for the remainder of the software development lifecycle [8]. This paper considers how the Common Criteria, an internationally recognized standard for security requirements definition and security assessment, can be used to specify the security requirements of component-based systems.

Myers states that written and measurable objectives (requirements) are required in order to perform system testing [13]. It is widely agreed that requirements must be defined and quantifiable for testing to be effective. In addition to conformance testing of security requirements, which focuses on the correct operation of individual requirements, security testing at the system level is desired in order to assess the overall level of security of the system [4]. In addition to providing guidance for security requirements identification and specification, the Common Criteria provide seven evaluation assurance levels (EALs), which can be used to assess the overall level of security of the system. Based on the level of security assurance required, security testing can be performed appropriately in order to validate a system's compliance with a specified EAL [2].

In this paper we consider how to use the Common Criteria to identify the security requirements of components, and component based systems. This identification is accomplished by using the Common

Criteria as a guide to authoring the software system's specification document. Once security requirements are identified using the Common Criteria, various testing methods can be used to assure compliance at the desired EAL. In addition to simply verifying conformance to security requirements, security testing techniques such as penetration testing [12] and vulnerability testing using fault injection [5] are approaches to security testing which could be considered when establishing a system's EAL.

Understanding how the Common Criteria security requirements map to different types of software components should help the software designer first in formally identifying and specifying the security requirements for the system, and second in constructing the software design. If the designer identifies specific security requirements for a system being built, then the component type mapping can provide insight on which types of components could be used to provide the security implementation. The Common Criteria can then be used to provide the security assessment using the formal specification of system security requirements generated in the requirements analysis phase of system development. In addition to assisting with the security specification, design and assessment, the generation of a Common Criteria mapping to component types may help to identify the most common security requirements of concern to software components and component based systems. By identifying these common security requirements, research efforts can be directed on developing specific component security tests that test these requirements. This component mapping may also be helpful in identifying where testing efforts should be concentrated when conducting a security assessment of a component based system. Considering this research some questions to consider include: What types of components will require more rigorous testing to ensure the overall security of the software system? What is the impact of a component based system's architecture with respect to the difficulties of assessing the system's composite security level? Is the difficulty of security assessment affected by how the security requirements are implemented across different types of components? What are the most common security requirements of software components?

2. Background

Component selection involves a certain amount of risk due to the inability to foresee complications and problems once components are put to use. Time available for testing and evaluation of the components is often limited due to project budgets and deadlines. Some risks associated with selecting a poor component for implementation of security functions include:

- Component does not meet basic functional security requirements.

Effect: Upon delivery the software system does not provide required security functions.

- Component in its current state does not meet future security requirements making it difficult for the system to be extended to support future needs.

Effect: The component must be replaced or updated in order for the software system to support future security requirements.

- A complex component providing security functions decreases system maintainability, and may even result in errors in the original system construction.

Effect: The complex component is difficult to understand, leading to additional maintenance activities associated with using it, which results in higher software maintenance costs.

- Component decreases the overall level of system security because of security flaws leading to confidentiality violations and data integrity problems.

Effect: Use of the component exposes security holes that can be exploited causing loss of system availability, data integrity, and data confidentiality.

The risks identified above must be considered when evaluating components that will provide the implementation of system security functions. Non-functional security requirements such as understandability, adaptability and maintainability are common concerns of software developers attempting to make component selection decisions. Other issues of concern include: Will components uphold the integrity and confidentiality of system data they are exposed to? Do the software components provide vulnerable mechanisms that can allow intruders access to system data? Consideration of these risks should help improve component selection decisions.

The assessment of component security requires more than the common penetrate and patch techniques for assessing security. Penetrate and patch techniques involve the assessment of security by exercising well-known vulnerabilities of a system in the attempt to bypass security. Once a given vulnerability is discovered within a system, a fix is created and the system is subsequently patched [6]. Several problems exist with this approach including: allowing hackers the opportunity to exploit systems that have not been thoroughly tested, continual maintenance requirements of installing frequent patches, and the potential that a patch itself may expose new security holes.

Recent research has considered the need to understand security properties of components used in component-based systems. Khan and Han have developed a security characterization framework, which allows the security attributes of components to be defined as a compositional security contract. Using Khan's security contracts developers can ensure that one component's required security properties conform to the ensured security properties of another component. The ensured and required security properties of component interactions are defined [9]. In a similar but unrelated work Sewell and Vitek provide a small programming language known as box-p calculus [14]. The language is used to express the composition of components while supporting the enforcement of security policies.

Software organizations developing software requiring high levels of security are pressured today to produce highly secure systems with limited time and budget constraints [3]. The U.S. Government has been pressured to move towards using component-based development in order to meet cost, quality, and schedule constraints. A standardization of security requirements for software systems has been identified called the Common Criteria for Information Technology Security Evaluation. The standard has been drafted by the Common Criteria project sponsoring organizations, which includes seven organizations across six countries in North America and Europe. [8]

There are two major portions of the Common Criteria: Security Functional Requirements, and Security Assurance Requirements. Security Functional Requirements are functional requirements that can be identified and used to evaluate a Target of Evaluation (TOE). A TOE is the software system of concern that is being evaluated. Security Assurance Requirements define the scope, depth, and rigor of the security evaluation activities in order to assure a certain level of security. Security assurance evaluates all aspects of a system including: configuration management, delivery, operation, development, documentation, testing, vulnerability, and life cycle support.

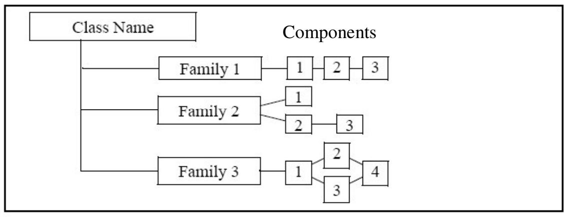

Figure 1 - Sample class decomposition diagram

The Common Criteria are broken into Classes, Families, Components, and finally Component elements (see Figure 1) [8]. A class is a grouping of families with a focus on a common set of security attributes. A family is a grouping of components that share security objectives. A component, with respect to the Common Criteria, represents a set of individual related security requirements. Component elements are specific individual security requirements. An example of a component element is: "The software system's security functions shall protect the stored audit records from unauthorized deletion." The label "FAU_STG.1.1" is used to identify the requirement where "FAU" represents the class, "STG" the family, "1" the component within the family, and finally "1" the component element which states the unique security requirement.

In [10] Khan considers how the security requirements of one particular CC security class, user data protection, apply to a component used in a medical software system. Kahn states that all of the requirements in the Common Criteria may not be applicable directly to software components due to the distributed nature of components and also the complexity of the component composition. In this paper the applicability of security requirements based on the type of component is considered and also the mapping of requirements based on the system's component architecture.

Voelter has defined a component classification with two categories of components: logical, and technical [16]. Logical components are further broken down into the subcategories of Domain, Data, and User components. In this paper these classifications are mapped to the classes of security requirements of the Common Criteria.

Logical components are:

- Domain: Provide business logic often referred to as middleware (the controller, in the model, view, controller aggregate design pattern)

Example: A domain component will perform work on data within a system. A mortgage component will provide a set of financial functions for computing mortgage information. In addition a mortgage component may interact with data components containing financial data.

- Data: Provide data access services including validation, conversion (the model in the model, view controller aggregate design pattern)

Example: A data component provides access to data. A data component must ensure data integrity, confidentiality, and accessibility. Domain components will interact with domain and user components

- User: Provide user interface, access to domain and data components (the view in the model, view, controller aggregate design pattern)

Example: A user component provides user interface functionality interconnecting domain and data components. A text box graphical widget provides the ability to display and modify the contents of a data component. A text box may also provide an interface to invoke business functions of a domain component.

Technical components act as containers or frameworks that provide a runtime environment for the component. They handle cross cutting technical concerns such as transactions, security, failover, and load balancing. What the specific technical concerns are, is determined based on the application domain. Using container components, the separation of technical concerns can be handled centrally.

3. Component Mapping

There are 11 different security classes in the Common Criteria. These classes include families of related security requirements. The (11) classes are: Security Audit (FAU), Communication (FCO), Cryptographic Support (FCS), User Data Protection (FDP), Identification and Authentication (FIA), Security Management (FMT), Privacy (FPR), Protection of the system security functions (FPT), Resource Utilization (FRU), System Access (FTA), and Trusted path/channels (FTP). Table 1 shows the relative sizes of the different security classes in the common criteria. Classes vary in size considerably. The User Data Protection (FDP) is a very large class that addresses the numerous requirements concerning the protection of user data. The Resource Utilization (FRU) class, a much smaller class, addresses security issues associated with the availability of required resources. It is likely that components, which implement requirements from the larger security classes, will require more testing effort.

CC Security Class	Number Of CC Requirements
(FAU) - Security Audit	27
(FCO) – Communication	12
(FCS) – Cryptographic Support	5
(FDP) - User Data Protection	67
(FIA) – Identification and Authentication	20
(FMT) – Security Management	19
(FPR) – Privacy	20
(FPT) – Protection of the System's Security Functions	50
(FRU) Resource Utilization	9
(FTA) System Access	15
(FTP) Trusted Path/Channels	6

Table 1 - Number of Security Requirements for Common Criteria Classes

In order to consider a mapping of the Common Criteria to Voelter's component types the role of the software architecture should be considered. Based on the software architecture the mapping of common criteria security requirements to components will vary. Figure 2 depicts a simplified view of a component-based architecture. This architecture includes components from each of the four component types identified by Voelter. The data components provide access to a database, or backend data store. These data components provide wrappers to allow access to the data. The user interface is implemented using a set of user components. Domain components perform the work of the system by interacting between data and user components. Cross cutting security requirements that are of consequence to all components in the architecture can be implemented in the component framework, which consists of a set of technical components. The precise distribution of the implementation of security requirements for a component-based system is not definite. Many

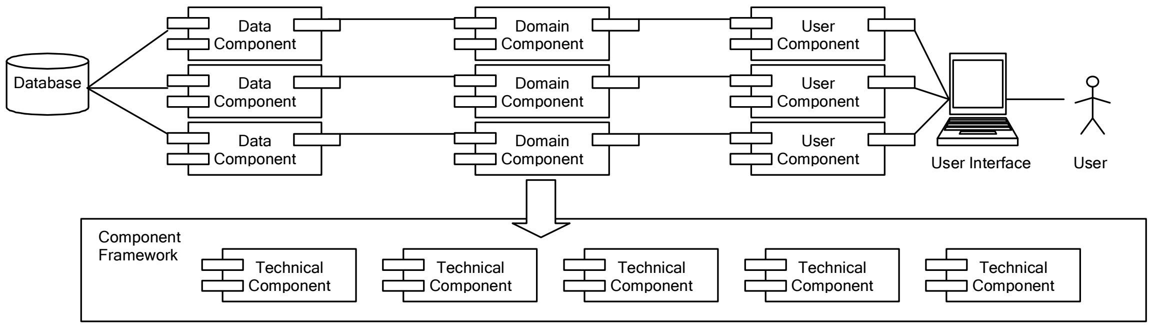

Figure 2 - Component Based Architecture

mappings are possible. In one scenario security requirements could be implemented in various components across a 3-tier architecture. In another scenario security requirements could be implemented using technical components in a common component framework. The framework provides a central location for implementing shared security functions. For a distributed system this central component framework may provide required functions for the distributed components to interact. This central framework is likely to implement various security functions. It is not reasonable to precisely define where all software architectures should implement security requirements. It is possible to identify the common locations where particular security requirements are likely to be implemented.

The Security Audit class (FAU) considers requirements for providing security audit facilities. Auditing consists of: generating logs of system activity and events, logging configuration functions, and providing access to the logs. In addition the audit logs can require automated analysis to detect potential breaches of security. Any component that provides access to system resource(s) may need to support security-auditing capabilities. Data components provide a wrapper interface to system data. Data components that provide operations to read/write data could include audit functions. Domain components that perform system tasks could include security audit functions to log when a particular operation is performed and who requested the operation(s) to be performed. The security auditing features of a component-based system (CBS) could also be implemented solely using technical components. This approach considers auditing as a crosscutting requirement that pertains to the entire system. In this case auditing functions could be implemented centrally within a technical component. Other components would need to access the auditing functions provided by the technical components through some interface.

Communication requirements are defined in the Communication Class (FCO). Communication security requirements deal primarily with assuring the identity of parties participating in data exchange. Non-repudiation of the message originator and receiver are primary concerns. Within component based systems, any component that communicates with another component either in a distributed system or in a centralized system will need to be concerned with the repudiation of identities. Domain components that perform the primary work of the system are most likely to communicate with external components. These domain components typically provide external interfaces enabling system functions to be invoked externally. It is important that only those who are properly identified and authorized have access to system functions. For example: Consider a component based e-commerce system. One particular component is responsible for accepting payment information to authorize purchases. In this scenario the component should be capable of verifying the identity of the parties involved in the commerce exchange. Any domain components providing this type of functionality should be concerned with non-repudiation requirements. Technical components typically provide internal functions and services for the components making up the component-based system. For a distributed system, technical components in a common framework may provide basic communication functions allowing distributed components to interact. The technical components, which compose this framework, may need to consider non-repudiation requirements in order to validate the authenticity of parties involved in communication. Finally a data component, which directly provides network access to a backend data store, may need to be concerned with the repudiation of parties making data read/write requests.

The Cryptographic Support class (FCS) defines security requirements for cryptography to ensure information confidentiality. Two families of requirements consider cryptographic key management and the operation of encryption algorithms. Any component, which uses cryptographic techniques to encrypt data, will need to consider FCS requirements. Since cryptography deals with confidentiality of data, data components are most likely to require cryptographic capabilities. Data components provide an interface to read/write data. These components could include the necessary encryption and decryption functions to ensure confidentiality. Domain components that enable system events such as system log-on and user authentication may require cryptography for exchanging confidential information. Domain components may provide secure encrypted communication mechanisms to external systems. Whenever a domain component is to provide a facility for confidential data exchange across a network, cryptographic functions should be considered. Distributed systems using a component framework to enable interactions among distributed components may need encryption capabilities if confidentiality of data being sent across a network is of concern. Encryption is likely to be a security requirement that cross cuts multiple components in the system. Several data components may need encryption services and it would be inefficient if each data component defined its own encryption functions. A solution that encourages software reuse is to implement encryption services in a common library of functions in a common framework of technical components.

The User Data Protection (FDP) class is the largest class in the Common Criteria consisting of 67 unique security requirements. Since FDP is such a large class elements of it can be mapped to all types of components. The FDP class is concerned with security functions to enforce integrity of user data. This class is broken down into four primary groups: policy related requirements, specific forms of user data protection, user data management/storage, and inter-system communication.

Most FDP security requirements are of concern to data components since they provide the primary interface to the system data. However FDP requirements are not solely implemented within the data components themselves. Access control and information flow policies are likely to be required for the information exchange functions provided by these data components. However the implementation of these security policies is a common security requirement for the entire system. Consequently information flow and access control policies are likely to be implemented using technical components in a central component framework.

Data components may implement specific forms of user data protection to ensure integrity, or they may use security functionality provided in a component framework. Data management functions including support for offline storage, exporting data, importing data, and protection of data are security functions data components are likely to implement.

Domain components are likely to implement communication with other software systems. This communication can result in sending and receiving significant amounts of user data across a network. In this case domain components are likely to be concerned with data integrity. These domain components may implement specific forms of user data protection or call upon functions in a central component framework to provide security. User components need to ensure the integrity of information provided by users to the system. Data collected from user components should not be corrupted by a malicious party.

Requirements related to establishing and verifying user identify are considered in the Identification and Authentication class (FIA). Logging into a system typically requires that a user first be identified and then authenticated based on a password response. FIA requirements are likely to be implemented in domain components. The login process is an activity the system must perform, and domain components typically enable the actions within a system. For distributed systems, components communicating across the network must identify and authenticate parties involved in the communication to prevent confidentiality and integrity violations. Such distributed systems may use technical components in a central component framework to provide identification and authentication functions.

Requirements to manage a system's security data are defined in a class known as Security Management (FMT). Management of security data such as access control lists, capability lists, and security configuration settings are responsibilities considered in this class. Components implementing security functions will require access to this security data. Technical components in a central framework are likely to provide a good location for implementing security data management. Security functions needing access to security data are likely to be implemented across many components in the system. By providing central access to security data its integrity and confidentiality is more easily ensured. Using a centralized approach to managing security data is likely to improve the overall security system's maintainability and understandability from a developer's point of view. An alternative to having central management of security data is to scatter security data across components requiring access to it. Although this approach is possible it is less ideal because it complicates the management of security data by distributing implementation sparsely across the components in the system.

Security requirements related to user identity protection and prevention of improper use of user identify (identify theft) are considered in the Privacy class (FPR). Since domain components are likely to handle user identification and authentication requirements, they could be good candidates to implement user identity protection and theft requirements. The user is identified through interaction with domain components. Steps should be taken within these domain components to enable anonymity (no identity exposure), pseudonymity (no identity exposure but user held accountable), unlinkability (unable to determine identity of repeating system usage), and unobservability requirements (unable to monitor system use). Implementation of privacy security requirements could be handled centrally by technical components interacting with the domain components that provide the user identification and authentication services.

The FPT class considers the protection of the application's overall security system. The FPT class considers the overall management of the security functions of the software system. The FPT is a large class in the Common Criteria consisting of 50 unique requirements. There are 16 families of requirements in the FPT class. The majority of the requirements in FPT apply to system-wide security functions. These system security functions are easily implemented in technical components as part of a central component framework. Many security requirements central to a system as a whole are found in these families:

Abstract Machine Test, Availability of Security Data, Confidentiality of Security Data, Integrity of Security Data, Internal System Security Data Transfer, State Synchrony Protocol, Time Stamps, Inter-TSF Data Consistency, Internal Security Data Replication Consistency, Reference Mediation, Domain Separation, and Self Security Tests. Security requirements from the families: Trusted Recovery of the System on Restart, and Replay Detection, could apply to Domain components. Domain components are responsible for executing system actions. A startup action could be executed by a domain component. Replay detection could be required of domain components, which are providing external communication interfaces.

The Resource Utilization class (FRU) considers availability of required resources for supporting the security functions of a system. This includes requirements to ensure the availability of resources during all system conditions, including system failure. In a component-based system the allocation and management of shared resources is a function that cross cuts many components. FRU requirements are most easily implemented using technical components in a central component framework. FRU requirements could be implemented in a domain component, when the domain component is the only component providing access to a resource. (E.g. the domain components act as a wrapper to a system resource) If multiple components access a shared resource then a mediator, in the form of a domain component, is needed to manage access. Finally data components providing security data may need to be fault tolerant in the event of a failure of the data store.

The System Access class (FTA) considers security requirements related to establishing a user's session. A domain component could provide session tracking requirements, however all aspects of session management would need to be supported by a single domain component. This implementation is unlikely since the implementations of session tracking requirements are not easily split across multiple domain components. Imagine finding a set of commercial components that handles all aspects of session management. It is unlikely to find a component that manages session privileges for users, and another that enables session creation. Technical components as part of a central component framework could more easily implement session tracking features. The component framework should already have access to system security data; therefore it should be easier to implement session management capabilities using technical components in the component framework.

The trusted path/channels class (FTP) defines requirements for having a trusted communication path between users and the system's security functions, and also between the system and other trusted software systems. Communication activities are likely to be provided by domain components. A domain component can provide the required network functions for the establishment of external connections to the component-based system. For distributed systems communication functions could be implemented in a central component framework. In this case the implementation of a trusted path would take place using technical components in the component framework.

4. Discussion

To summarize the component mapping in the previous section a 3-point ordinal scale is used to express the likelihood of a given Common Criteria security class's mapping to Voelter's component types. The ordinal scale used is described in Table 2.

Rank	Description
Empty (0)	No Mapping: Security requirements of the CC class are unlikely to be implemented in this type of component.
1	Weak Mapping: One or more, but not all of the security requirements of the CC class could be implemented in this component type.
2	Possible Mapping: More than one and possibly all of the security requirements of the CC class could be implemented by multiple components of this type.
3	Strong Mapping: All of the security requirements of this CC class could be entirely implemented by a single component or a fixed set of components of this type.

Table 2 - Ordinal Scale Describing Component Mappings

An empty rank (0) indicates that the specified component type does not implement security requirements from the CC class. A weak mapping (1) implies that one or more, but not all of the security requirements from the CC class could be implemented by components receiving this ranking. A possible mapping (2) indicates that multiple components of the specified type could fully implement all of the security requirements of the CC class. When new functions and components are added to the system, additional implementation of security may be required. A strong mapping (3) indicates that a single component or a fixed set of components can implement all of the security requirements of the CC class. If new functions and components are added to the system the implementation of security functions can entirely rely on preexisting components. Table 3 provides a ranking of the mappings of security requirements to

Function	Domain components	Data Components	User Components	Technical Components	Avg mapping to component types
(FAU) – Security Audit	2	1		3	1.5
(FCO) – Communication	2	1		3	1.5
(FCS) – Cryptographic Support	1	2		3	1.5
(FDP) - User Data Protection	2	2	1	2	1.75
(FIA) – Identification and Authentication	2			3	1.25
(FMT) – Security Management	1	1		3	1.25
(FPR) – Privacy	3			1	1
(FPT) – Protection of the Security System	1			3	1
(FRU) Resource Utilization / Availability	2	1		3	1.5
(FTA) System Access	1			3	1
(FTP) Trusted Path/Channels	2			3	1.25
Totals	19	8	1	30	-

Table 3 - Common Criteria Security Classes mapping to component types

various component types using the ordinal scale of Table 2.

Table 3 summarizes the mapping between Voelter's component types and the Common Criteria security classes. From the totals provided by summing the columns it seems that user components rarely implement security functions (total=1), whereas data components are somewhat more involved with the implementation of system security requirements (total=8). Domain components could implement nearly all aspects of security of the Common Criteria (total=19) if necessary. Technical components can be used to implement a common framework to provide security functions across the application (total=30). Such a common framework is especially important for distributed systems, which have additional communication requirements not present for centralized systems. In a distributed system, components may need to interact with a component framework in order to communicate and interact with the system as a whole.

Domain components typically handle action related security requirements. Tasks such as establishing a user's session, or requesting access to an audit log, are action-oriented tasks, which the user may request. Domain components typically provide the realization of system actions in a component-based system; therefore we can expect to see some security requirements being implemented in domain components.

Data components provide access data stored in backend data stores such as relational databases, file systems, etc. Data components are primarily concerned with the integrity and confidentiality of data as it flows into and out of data components. Data components may need to implement special user data protection schemes and employ access control policies to ensure data integrity. Encryption may be required to ensure confidentiality of the data when transmitted over insecure networks.

User components provide the application interface to the user. Common user components include graphical widgets often referred to as controls. For example: A calendar component is a user component, which allow for the selection of dates by displaying a month-based calendar. Sophisticated data grid components are user components that can display data in a variety of numerical formats. For user components one significant security concern is that of user data protection. When data is provided to/from user components, the system must ensure that the integrity and confidentiality of the data is maintained.

The final column shows the "average mapping to different component types". A larger number suggests that the security requirements from the CC security class could potentiality be implemented in several different types of components, whereas a small number suggests that the security requirements of the CC class are likely to be confined to fewer types of components. For example privacy requirements deal with assuring that the user's identity remains anonymous. This requirement could be implemented by a domain component, because the domain component provides the user login and authentication capabilities, or within a trusted component which is part of the component framework that handles many crosscutting requirements. In contrast the security auditing could be implemented in most of the different component types. Auditing could occur in a domain component for recording

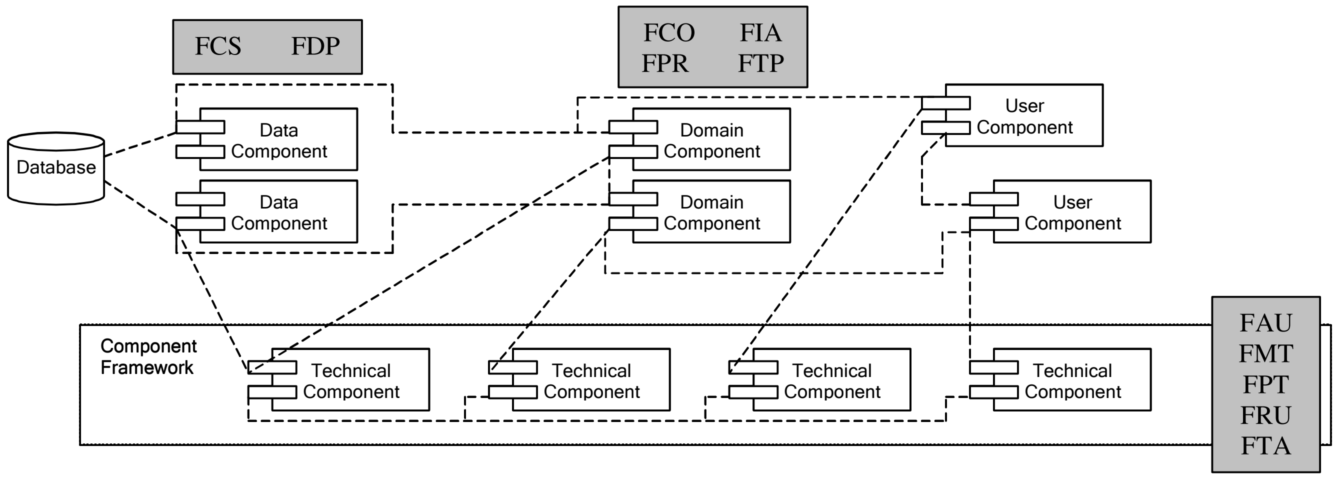

Figure 3 - Component Based System Architecture

results of user driven actions, in a data component for recording access and changes to data, or in a technical component as part of a central component framework.

Figure 3 shows a possible architecture of a component-based system. In this system various components interact with the component framework, which consists of technical components. There are also various interactions between data, domain, and user components. The interactions between components do not conform to any strict rules. This architecture represents an ad-hoc arrangement of security functions throughout the components of the system. In the absence of a common component framework the mapping of security requirements to components is likely to be more scattered. Security requirements will likely be implemented across many of the components in the system, especially considering that off-the-shelf domain and data components may include various security features. The suggested mappings of CC classes are shown next to the relevant component types in Figure 3. Even with several CC security classes handled by the component framework, data and domain components may still need to be concerned with the implementation of various security functions of the system.

After the requirements definition phase for a component-based system, by considering the mapping presented in section 3, insight to the mapping of security requirements to the components and component architecture of the system is possible. This mapping of security requirements to components can later help to identify where testing efforts should be focused. Security assessment is likely to be simpler if a component framework provides the implementation of common security requirements. Rigorous unit testing of this framework could lead to better detection of security flaws in the system. By centralizing the implementation of security functions the overall security assessment for the system may require less effort.

The common criteria define the TSF as the software system whose security system is being evaluated. Using the Common Criteria a concise view of the TSF is a precondition for conducting a security assessment of the system. For the evaluation of a component-based software system using the Common Criteria we can propose that by using the providing mapping it may be easier to derive the security specification required for the security evaluation.

In addition to the complications for security assessment, a component-based system having security requirements implemented sparsely across many components of the system will likely also suffer from poor software maintainability. Developers will have difficulty understanding the software design, and extending it to include new security capabilities. Diagnosing the cause of a security fault is further complicated due to the distributed implementation of security functions across the system's components.

5. Conclusion and Future Work

In evaluating the Common Criteria applicability to assessing security of components and component-based systems the importance of the system's software architecture is identified. Depending on the architecture chosen, many security requirements could be implemented by technical components in a central framework or the security requirements could be scattered more sparsely across various components of a component based system. User components that provide an interface for an application have only a minor role in implementing security requirements for a component-based system. Data and Domain components may both implement various security requirements with respect to their functional application in the system. Ultimately the level of system security provided by a component based system is affected by the security properties provided by the software components, as well as their composite interactions achieved through integration in the system architecture.

This paper has suggested how security requirements are mapped across software components in a component based system. Analyzing component-based systems to see if the security requirements are implemented where suggested could further enhance this mapping. What are the pitfalls of using the Common Criteria to assess the security provided by a component based system? How does the application domain affect the mapping of security requirements to components? How can the mapping help to focus security-testing efforts? Future work is needed to improve component based software development so that it is a viable development methodology for building systems with significant security concerns.

6. References

[1] Bergner, K., Rausch, A., Sihling, M., Vilbig, A., Componentware – Methodology and Process, in Proceedings of 1999 International Workshop on Component Based Software Engineering held in conjunction with ICSE99, Los Angeles, CA, USA, pp. 194-203, 1999.

[2] Caplan, K., Sanders, J.L., Building an International Security Standard, in IT Professional, vol.1, no.2, March-April 1999, pp. 29-34.

[3] Devanbu, P. and Stubblebine, S. Software Engineering for Security: a Roadmap. In The Future of Software Engineering. Special volume of the proceedings of the 22nd International Conference on Software Engineering - ICSE 2000, June 2000.

[4] Dima, A., Wack, J., Wakid, S., Raising the Bar on Software Security Testing, in IT Professional, vol. 1, no. 3, May-June 1999, pp. 27-32..

[5] Du, W., Mathur, A.P., Testing for Software Vulnerability Using Environment Perturbation, in Proceedings for 2000 International Conference on Dependable Systems and Networks (DSN 2000), pp. 603-612, 2000.

[6] Ghosh, A. K., McGraw, G. An Approach for Certifying Security in Software Components. In proceedings of the 21st National Information Systems Security Conference, pp. 42-48, 1998.

[7] Goulao, M., Abreu, F.B., The Quest for Software Components Quality, in Proceedings for 2002 Computer Software and Applications Conference, (COMPSAC '02), pp. 313-318, 2002.

[8] ISO/IEC-15408 (1999) Common Criteria for Information Technology Security Evaluation, v 2.0, Nat'l Inst. Standards and Technology, Washington, DC, June 1999, http://csrc.nist.gov/cc

[9] Khan, K.M. Han, J., Composing security-aware software, IEEE Software, vol.19, no.1, Jan.-Feb. 2002, pp.34-41.

[10] Khan, K.M., Han, J., Zheng, Y., Characterizing User Data Protection of Software Components. In Proceedings of the 2000 Australian Software Engineering Conference, Gold Coast, Queensland, Australia, April 2000.

[11] Kotonya, G., Onyino, W., Hutchinson, J., Sawyer, P., Canal, J., COTS Component-Based System Development: Processes and Problems, appears in Business Component-Based Software Engineering, Kluwer Academic Publishers, pp. 228-245, 2003.

[12] McDermott, J.P., Attack Net Penetration Testing, in Proceedings of 2000 workshop on New Security Paradigms, pp. 15-21, 2001.

[13] Myers, G., Software reliability: principles and practices: New York: John Wiley & Sons, 1976.

[14] Sewell, P., Vitek, J., Secure Composition of Insecure Components. In proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW-12), Mordano, Italy, 1999.

[15] Syperski, C., Gruntz, D., Murer, S., Component Software: Beyond Object-Oriented Programming Second Edition, Addison-Wesley / ACM Press, 2002.

[16] Völter, M., A Taxonomy for Components, in Journal of Object Technology, vol. 2, no. 4, July-August 2003, pp. 119-125., http://www.jot.fm/issues/issue_2003_07/article3