01 Jul 2005

Efficiently Managing Security Concerns in Component Based System Design

Ammar Masood

Sahra Sedigh
Missouri University of Science and Technology, sedighs@mst.edu

Arif Ghafoor

Recommended Citation

A. Masood et al., "Efficiently Managing Security Concerns in Component Based System Design," *Proceedings of the 29th Annual IEEE Computer Software and Applications Conference (2005, Edinburgh, Scotland)*, vol. 1, pp. 164-169, Institute of Electrical and Electronics Engineers (IEEE), Jul 2005.
The definitive version is available at https://doi.org/10.1109/COMPSAC.2005.71

Efficiently Managing Security Concerns in Component Based System Design*

Ammar Masood[1], Sahra Sedigh-Ali[2], Arif Ghafoor[1]
[1]School of Electrical & Computer Engineering, Purdue University, West Lafayette, IN
[2] School of Electrical & Computer Engineering, University of Missouri-Rolla, MO
ammar@ecn.purdue.edu, sedighs@umr.edu,ghafoor@ecn.purdue.edu

Abstract

Component-based software development (CBSD) offers many advantages like reduced product time to market, reduced complexity and cost etc. Despite these advantages its wide scale utilization in developing security critical systems is currently hampered because of lack of suitable design techniques to efficiently manage the complete system security concerns in the development process. The use of Commercial of the Shelf (COTS) components can introduce various security and reliability risks in the system. In this paper we propose a methodology for efficient management of all the system security concerns involved in the design of component based systems. Our methodology is based on formally representing the system security specifications and component capabilities. We identify the metrics for correlating both and suggest extensions to a previously proposed software development process, for selection of suitable components and integration mechanisms. The proposed solution ensures due treatment of all the security concerns for the complete system in the acquisition efforts.

1. Introduction

COTS components are defined as components that are bought from a third party vendor and integrated into a system [6]. Because of the savings in development resources, offered by CBSD and the fast pace at which current software development is proceeding, large-scale component reuse has resulted. Advantages like reduced product time to market, reduced complexity and cost etc, offered by CBSD have resulted into increased COTS component acquisition. Despite these advantages quality and risk concerns currently limit the application of COTS based systems to non-critical applications [10, 11]. The COTS components usage can also result into various other problems like increasing the system vulnerability to risks arising form third-party development, such as vendor longevity and intellectual property procurement [7].

An optimized software development process for CBSD has been proposed in [7]. The presented approach uses software metrics for guiding acquisition and integration efforts by using multi objective optimization. Multi objective optimization is used to select highest quality components that will yield the highest quality within the given cost. Reliability, complexity and cost are the component metrics used for system level optimization. The security considerations for the complete system are not considered in this approach. Given the benefits associated with COTS based design the major hurdle in their wide scale usage especially in critical applications is lack of system design techniques that could ensure complete fulfillment of all the system security specifications.

In this paper we propose a methodology for efficient management of all the system security concerns involved in the design of component based systems. Our methodology formally represents the system security specifications and component capabilities. We identify the metrics for correlating both and propose extensions to the software development process presented in [7], for selection of suitable components and integration mechanisms. The black box testing of selected components is then performed to assign *confidence levels* (ranging from 0 to 1) to each component based on its verification results. The confidence level assignment helps the system designers in more accurate assessment of risks associated with selected configurations. The proposed solution thus ensures due treatment of all the security concerns for the complete system in the acquisition efforts.

The organization of the rest of the paper is as follows. We discuss the related work in component based system design in section two. The outline of proposed methodology is presented in section three. The discussion about integration of proposed methodology with the approach presented in [7] is given in section four

*This work was supported by NSF award IIS 0209111

followed by conclusion and discussion about avenues for future research.

2. Related Area

The problem of integrating COTS components into systems with high dependability requirements has been considered in [4]. The untrustworthy COTS components are treated as a potential source of faults and protective wrappers are added to them to generate *idealized fault-tolerant COTS components*. This mitigates the possible problems due to architectural mismatch between the components and the rest of the system. The *sandboxing* approach presented in [11] is similar to the concept of *restriction* (section 3.4) presented in this paper; however our methodology also guides the system designer through all the phases of system design including component *sandboxing*. Error containment wrappers have been used in [5] to increase the robustness of the components in dealing with errors coming from their environment. The proposed approach adopts a *gray-box* perspective of the component i.e. considering that the component structure is known but *non-modifiable*. Our approach is however more general as we consider the components to be black boxes (which is mostly the case with COTS components) for which only known information is their advertised interfaces and capabilities.

An architecture for integrating COTS components has been presented in [9] along with some informal rules to facilitate the integration; whereas our approach guides the integration process through a formal methodology. A reference model for the assembly of components based systems has been proposed in [3] to account the key activities in CBSD process. The focus has been on the technical aspects of the integration of COTS based system. Our approach complements this work by providing formal methodology to efficiently manage all the security concerns in the integration process. The importance of assessing the risks associated with selection of COTS components has been highlighted in [2]. The step 5 in our methodology (section 3.2) is a step forward in this direction by formally guiding the system developer through assignment of confidence levels to the components.

3. Proposed Methodology

3.1 Assumptions

Here we discuss few assumptions that we make while developing the methodology. We assume that for all the components the system designer has the complete knowledge of their functional specifications and security capabilities. Where security capabilities reflect the security controls implemented in the component by the vendor and advertised by them. We assume that for each functional requirement of the system a family of components (FOC) is available that includes all the components capable of meeting that particular functional requirement. The FOC_i will be defined as set of components in i^{th} FOC i.e. $FOC_i = \{C_{i1}, C_{i2}, \cdots C_{im}\}$ where C_{il} represents the l^{th} component in that FOC. The components within a FOC might have different security capabilities. Moreover following simplifying assumptions, as also done in [7] are made.

1. The families of components have non-overlapping functionality. Thus, exactly one component is chosen from each family to design the system.
2. The components are connected serially, i.e., each component is interfaced to at most two other components.

These assumptions do not lead to a significant loss of generality as also pointed out in [7]. Further we assume that system designer is able to breakdown the complete system security specifications into components level security requirements/specifications in a consistent manner. Such an approach for assuring security constraints in architectural decomposition has been presented in [8]. This decomposition is used for matching the security capabilities of components in each FOC with the security specifications for that FOC.

3.2 Methodology Outline

Following is the outline of the complete methodology followed in guiding the CBSD process; same is also represented in figure 1.

1. Mathematical formulation of the system security specifications and components security capabilities.
2. Identification of metrics for correlating FOC security specifications with components capabilities.
3. Specification of system security objectives.
4. Selection of component configurations to ensure that overall system functional objectives are optimized while guaranteeing that the system security objectives are achieved. A reasonable fraction of total no of possible configurations is selected.
5. Verification of selected components security capabilities through black box testing and assigning confidence levels to them based on their testing results.
6. The assigned confidence levels are compared against predetermined threshold for each FOC. This comparison might result into rejection of all selected configurations; therefore the component configuration selection process of step 4 may need to be again followed with the components confidence level as the new

parameter used in selection process. The final selected configuration will be assigned a confidence level based on the component confidence levels.

3.3 Mathematical Formulation

The formalization of the system security specifications and components capabilities is carried out as per following details.

1. $S = \{S_1, S_2, ...S_n\}$ is the set of system security specifications, such that S_i corresponds to system specifications for i^{th} FOC where n is the total number of FOC's.

2. $S_i = \{SS_1, SS_2, ...SS_k\}$ is a set of sub system specifications for each FOC.

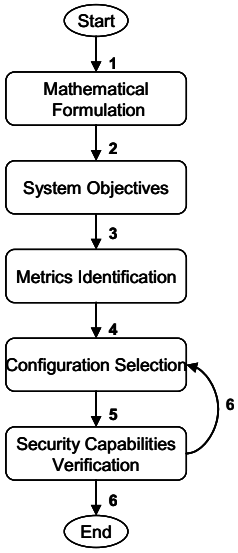

Figure 1: Methodology outline

3. $SS_i = (Sp, Attrib, P)$ is a tuple where Sp represents the security policy corresponding to each sub system specification and $Attrib$ specifies the attribute for that security policy and P is a set of parameters $P = \{Pa_1, Pa_2, ...Pa_j\}$. In case if the security policy Sp is unconstrained then P is set to *none*. In case of common Sp of different SS_i, the SS_i's will be partially ordered based on $Attrib$ and P. For example if $SS_1 = \{AC, RBAC, none\}$ and $SS_2 = \{AC, TRBAC, none\}$ then $SS_1 \prec SS_2$ is true. Here AC represents access control and RBAC and TRBAC represents role based access control and temporal role based access control respectively. The condition $SS_1 \prec SS_2$ is true because TRBAC extends RBAC with temporal constraints and thus subsumes RBAC. The individual members of a particular SS_i can be referenced as $Sp(SS_i)$, $Attrib(SS_i)$ and $P(SS_i)$.

4. $Pa_i = (name, value)$ is a tuple where *name* and *value* represents the details for a specific parameter corresponding to a specific sub system specification.

5. $Sc_{il} = \{SS_1, SS_2, ...SS_r\}$ is a set of security capabilities for l^{th} component of i^{th} FOC where SS_i's are same as defined earlier. Individual security capabilities of component will be referred by $Sec_j(Sc_{il})$ such that $Sec_j(Sc_{il}) = SS_j \quad of\ component\ C_{il}$.

3.3.1 Example 1. Consider a system in which components are to be selected from two FOC's and we assume two components in each FOC. The system security specifications for the FOC's are following.

System Specifications	FOC 1	FOC 2
Cryptographic support	DES	RSA
Authentication Mechanism	SSL/TLS	SSL/TLS
Access Control	RBAC	GTRBAC capable of handling *location* and *system_load* context

Further on the components capabilities are.

Component Capabilities	$Comp_{11}$	$Comp_{12}$	$Comp_{21}$	$Comp_{22}$
Cryptographic support	RSA	DES	RSA	DES
Authentication Mechanism	Kerberos	SSL/TLS	SSL/TLS	Kerberos
Access Control	MLS	TRBAC	GTRBAC	RBAC

The abovementioned terms are explained here. Two major cryptographic schemes in use today are *symmetric key* and public key based schemes. The Data Encryption Standard (DES) and Rivest Shamir Adleman (RSA) are examples of symmetric key and public key systems, respectively. Secure Socket Layer/ Transport Layer Security (SSL/TLS) and Kerberos both provide authentication support through cryptographic mechanisms but use different technologies. Multi Level Security (MLS) model of access control is based on assigning security classifications to objects and permitting user access to these based on users security clearances. The General TRBAC (GTRBAC) access control model extends TRBAC by including temporal and contextual constraints.

In the above example $Comp_{il}$ represents l^{th} component of i^{th} FOC. The mathematical formulation for this example would be:

$S = \{S_1, S_2\}$ Where

$S_1 = \{(Crypto, DES, none), (Auth, SSL/TLS, none)$
$(AC, RBAC, none)\}$

$S_2 = \{(Crypto, RSA, none), (Auth, SSL/TLS, none)$
$(AC, GTRBAC, [(location, true)(system\_load, true)])\}$

$Sc_{11} = \{(Crypto, RSA, none), (Auth, Kerberos, none)$
$(AC, MLS, none)\}$

$Sc_{12} = \{(Crypto, DES, none), (Auth, SSL/TLS, none)$
$(AC, TRBAC, none)\}$

$Sc_{21} = \{(Crypto, RSA, none), (Auth, SSL/TLS, none)$
$(AC, GTRBAC, none)\}$

$Sc_{22} = \{(Crypto, DES, none), (Auth, Kerberos, none)$
$(AC, RBAC, none)\}$

3.4 Identification of Metrics

Suitable selection of metrics for comparison of FOC security specifications with the components capabilities is important for guiding the configurations selection process (step 4). We have borrowed the concepts of *functional deficit* and *functional excess* used in [6] to define *capability excess* and *specification deficit.* Before defining these terms we first define the *difference (-)* operator.

Definition 1:- *The difference of a system specification S_i and a component capability Sc_{is} is denoted by $S_i - Sc_{is}$ and defined by:*

$$S_i - Sc_{is} = \{s : s \in S_i \quad and \quad s \notin Sc_{is}\}$$

Definition 2:- *The specification deficit between a system specification S_i and component capability Sc_{is} is denoted by $\alpha(S_i, Sc_{is})$ and defined by:*

$$\alpha(S_i, Sc_{is}) = S_i - (S_i \cap Sc_{is})$$

The *specification deficit* between a system specification S_i and COTS component capability Sc_{is} reflects the no of features in the specification of i^{th} FOC that are left unfulfilled by the capabilities of s^{th} component in that FOC. If S_i and Sc_{is} are equal then this would be assigned *none* value.

Definition 3:- *The capability excess of component capability Sc_{is} with respect to a system specification S_i is denoted by $\beta(S_i, Sc_{is})$ and defined by:*

$$\beta(S_i, Sc_{is}) = Sc_{is} - (S_i \cap Sc_{is})$$

The *capability excess* between a COTS component capability Sc_{is} and system specification S_i reflects the no of features in the capabilities of s^{th} component in i^{th} FOC that are irrelevant to the specification of that FOC. If S_i and Sc_{is} are equal then this would be assigned *none* value.

As the values of both *capability excess* and *specification deficit* could be either none or set of SS_j, thus they would be reflexive, transitive and antisymmetric for $\prec$ relation (because of partial ordering in SS_j) . Hence they will also have partial ordering. We consider

$\alpha(S_i, Sc_{is}) \prec \beta(S_i, Sc_{is})$ iff

$\forall SS_j \in \alpha(S_i, Sc_{is}), SS_j \prec SS_r \ \forall SS_r \in \beta(S_i, Sc_{is})$

where $Sp(SS_j) = Sp(SS_r)$

Definition 4:- *The deficit cover for specification deficit $\alpha(S_i, Sc_{is})$ and capability excess $\beta(S_i, Sc_{is})$ is denoted by $\phi(S_i, Sc_{is})$ and defined by:*

$$\phi(S_i, Sc_{is}) = True \quad \Leftrightarrow \alpha(S_i, Sc_{is}) \prec \beta(S_i, Sc_{is})$$
$$else\ False$$

The "*True*" value of *deficit cover* thus denotes that all the deficient features in the specification of i^{th} FOC are capable of being fulfilled through the features in the capabilities of s^{th} component in that FOC.

For the security specifications and capabilities given in example 1 the value of *specification deficit* and *capability excess* are:-

$\alpha(S_1, Sc_{11}) = S_1$, $\alpha(S_1, Sc_{12}) = \{AC, RBAC, none\}$

$\alpha(S_2, Sc_{21}) = \{AC, GTRBAC, [(location, true)$
$(system\_load, true)]\}$, $\alpha(S_2, Sc_{22}) = S_2$

$\beta(S_1, Sc_{11}) = Sc_{11}$, $\beta(S_1, Sc_{12}) = \{(AC, TRBAC, none)\}$

$\beta(S_2, Sc_{21}) = \{(AC, GTRBAC, none)\}$, $\beta(S_2, Sc_{22}) = Sc_{22}$

Moreover

$\alpha(S_1, Sc_{12}) \prec \beta(S_1, Sc_{12}) \Rightarrow \phi(S_1, Sc_{12}) = True$ and

$\alpha(S_2, Sc_{21}) \prec \beta(S_2, Sc_{21}) \Rightarrow \phi(S_2, Sc_{21}) = True$

3.5 Security Objectives

The security objectives for the system would be to ensure that the system security specifications for each FOC are met by the security capabilities of the selected components (there could be more than one component selected from each FOC in the multiple configurations selected in step 4) in that FOC. Hence if we consider C_{i*} being the component finally selected in i^{th} FOC, in some configuration, then this objective can be stated as:-

$(\alpha(S_i, Sc_{ic_{i*}}) == none) \vee \phi(S_i, Sc_{ic_{i*}}) \quad \forall i \ , \quad 1 \le i \le n$

This means that with respect to system security considerations a selected component is only acceptable either if its capabilities fully match the system specifications for that FOC (in that case $\alpha(S_i, Sc_{ic_{i*}}) == none$) or its *specification deficit* is fully covered by its *capability excess* i.e. $\phi(S_i, Sc_{ic_{i*}})$ is true. The second condition implies that selected component has capabilities that are in excess of the system specification and therefore the integration module would also have to curtail its additional capabilities. We define this process as *restriction.*

4. Security Concerns Integration

4.1 Components Selection Strategy

As already explained in the previous section the selected components in each FOC should be able to satisfy the system security specifications for that FOC thus initial *pruning* of components would be carried out before the selection with respect to functional objectives proceed.

Definition 5:- The *process of selection of components that satisfy system security specification is called pruning and is formally defined as:-*
Pruning:

$$FOC_i \rightarrow FOC_i' \quad \forall i, 1 \le i \le n \quad st \quad \forall C_s \in FOC_i'$$
$$(\alpha(S_i, Sc_{ic_s}) == none) \vee \phi(S_i, Sc_{ic_s})$$

Thus after *pruning* only those components are left in each FOC for which either there is no *specification deficit* or the *deficit cover* is true. The functional objectives would now guide the selection process. In case if *deficit cover* is true this implies that the component capabilities in excess of system specifications need to be curtailed by *restriction.* Thus additional cost of *restriction* code needs to be considered in the component selection process. If the objective is to select most reliable configuration with minimum possible complexity within a total system cost constraint then the metrics of interest could be reliability, complexity and cost of the components and integration modules. The component selection process will then proceed as a multi objective optimization problem as per procedure used in [7] by including the additional cost, reliability and complexity of *restriction* module (in case if components with lesser capabilities are also required to be considered for selection then additional cost will be due to integration mechanisms providing the deficit capabilities). The above multi objective optimization procedure would return a set of suitable configurations which will be filtered to select t number of best configurations from them, where t is derived as follows:

$$t = \frac{Total \quad no \quad of \quad suitable \quad configurations}{Threshold \quad parameter \ (\lambda)}$$

The threshold parameter λ will be selected by system developer based on previous results and empirical observations. Note that all the selected t configurations meet the system security specifications. The reason for not considering all the suitable configurations is the huge cost of black box testing procedure (performed in step 5) if all the suitable configurations are selected.

4.2 Verification of Component Capabilities

In this step the security capabilities of the components used in configurations selected in previous step will be verified by black box testing. Black box testing is actually functional testing of the software product as the internal structure of the product is not considered while testing rather its external behavior model, which is in turn guided by its specifications, guides the testing process [1]. The purpose of this process is to verify the match between the advertised component capabilities and their implementation. We suggest this capability verification for components security capabilities because it is important to ensure that the components are actually able to provide security functionality as per their advertised capabilities. Failure of a component to provide the strength of security controls as advertised can result into catastrophic failures of the complete system.

The outcome of this process will be assignment of *confidence levels* (ranging from 0 to 1) to each component based on its verification results. Higher levels reflect more trust on the implementation. Assignment of these levels definitely depends on the extent of testing carried out and component performance. For different system designs their may be dissimilar stress on the security policies implemented in the component (where these policies are also definitely reflected in component security capabilities), therefore a formal framework is required to express the overall confidence level as weighted average of confidence level of individual capabilities. We define $\gamma(Sec_j(Sc_{il}))$ as confidence level for the security capability $Sec_j(Sc_{il})$ of component C_{il} and overall confidence level is defined as follows.

Definition 6:- *The component security confidence level is denoted by* $\gamma(C_{il})$ *and is defined as:-*

$$\gamma(C_{il}) = \frac{\sum_j \gamma(Sec_j(Sc_{il})) \times \sigma_j}{\sum_j 1}$$ *Where* σ_j *is the assigned weightage for the component security capability* $Sec_j(Sc_{il})$

In case if all the security capabilities are equally important then σ_j would be 1 for all. As for a particular selected configuration only one component would be selected from a specific *FOC* therefore component *confidence level* can also be interpreted as *FOC confidence level* and thus $\gamma(FOC_i) = \gamma(C_{il})$.

4.3 Finalizing System Design

The computed component *confidence levels* will be used for further refinement of selected configurations. *Threshold parameters* $\nu_i \; 1 \le i \le n, \quad 0 \le \nu_i \le 1$ will be used for rejecting all those configurations for which the following condition does not hold true.

$$\gamma(FOC_i) \ge \nu_i \qquad 1 \le i \le n$$

Thus all the configurations with *confidence level* for any FOC below a specific threshold will be rejected. Therefore it might happen that all the selected configurations are rejected, in that case the system designer can either relax the *threshold parameters* or select the next *t* best configurations from the results obtained after step 4(component selection strategy). The new selected configurations will again go through step 5 to finalize selection of best configuration.

If after the above scrutinizing process the number of available selected configurations is greater than or equal to 2, then a modified component selection strategy (modification to step 4) will be used for final configuration selection. This component selection process can again proceed as a multi objective optimization problem by including the component confidence levels as another metric and solving the resulting configuration problem. The confidence level of the finally selected configuration will be a vector consisting of confidence levels of each FOC.

5. Conclusion

Despite the many advantages offered by CBSD, its wide scale utilization in developing security critical systems is currently hampered because of lack of suitable techniques to efficiently manage the complete system security concerns in the development process. The methodology proposed in this paper allows efficient management of system security concerns during all the stages of CBSD process. The proposed methodology formally guides the system designer in the selection of suitable configurations while ensuring due consideration is given to all the system requirements.

In the future we plan to conduct case studies to formally evaluate the efficacy of our proposed methodology. We also plan to study the robustness of our methodology in terms of variation in the component confidence levels that can be tolerated by the system safely. An important avenue of future research is to extend the proposed methodology to accommodate dynamic composition of component based systems. We expect this study to be very valuable in guiding the wide scale adoption of CBSD techniques.

References

[1] B. Beizer. "Black Box Testing". *John Wiley & Sons, Inc*. 1995

[2] L. C. Briand. "COTS Evaluation and Selection". *Proceedings of the IEEE International Conference on Software Maintenance*. 16-20 Nov. 1998

[3] A.W. Brown, K.C. Wallnau. "Engineering of component-based systems". *Proceedings of the 2nd IEEE International Conference on Engineering of Complex Computer Systems (ICECCS '96).*Oct 21-25, 1996.

[4] P. A. de C. Guerra and C. Rubira and A. Romanovsky and R. de Lemos. "Integrating COTS Software Components into Dependable Software Architectures". *Proceedings of the 6th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC-03).* IEEE Computer Society Press, May 2003.

[5] A. Jhumka, M. Hiller, and N. Suri. "An Approach to Specify and Test Component-Based Dependable".*7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02)* .October 23 - 25, 2002

[6] L. L. Jilani and A. Mili. "Estimating COTS integration: An analytical approach". *Proc. of the 5th Maghrebian Conf. on Software Eng. and Artificial Intelligence*, Dec. 1998.

[7] Sahra Sedigh-Ali. "Metrics-Guided Models and Methods for Cost and Quality Management of Component-Based Software". *PhD thesis*. Dept of Electrical and Computer Engineering, Purdue University West Lafayette. 2003.

[8] Yi Deng, J. Wang, J.J.P Tsai and K. Beznosov. "An approach for modeling and analysis of security system architectures". *IEEE Transactions on Knowledge and Data Engineering*, 15(5): 1099 - 1119, Sept.-Oct. 2003

[9] M.R. Vigder and J. Dean. "An architectural approach to building systems from COTS software components". *Proceedings of the conference of the Centre for Advanced Studies on Collaborative research (CASCON'97).* 1997

[10] J. M. Voas. "Certifying off-the-shelf software components". *IEEE Computer,* 31(6):53–59, Jun. 1998.

[11] Q. Zhong and N. Edwards. "Security Control for COTS Components". *IEEE Computer*, 31(6):67-73, Jun. 1998.