Security Assurance in Component Based Software Systems

A principle is an accepted or professed rule or conduct,.....
A fundamental doctrine, right rules of conduct.
J Sein

Security for a component based systems is a necessary prerequisite. Since most of its application has been involved in business logic, security mechanisms should be included in order to ensure message confidentiality and integrity.

One of the key problems when designing component based software is the integration of security aspects into the development process. A developer will always have to consider security aspects to a certain extent.

Nowadays, one of the main challenges facing software community is increasing security threats against them. The security issue is very significant in development of security-critical applications, such as banking, military and ecommerce systems. Security functions like encryption, digital signature and authentication are often needed.
Component-Based Software Engineering (CBSE) is so widely recognized that its principles are naturally applied to business application. In this scope, we discuss security concerned with component-based application.

Security improvement has become the primary approach to improve software quality, reliability, and customer satisfaction. Component developer pays little attention for security requirement in component-based software system.

Security assurance in component-based software systems covers the whole lifecycle of systems, i.e., from the acquisition, design, development, testing, deployment, and maintenance. It involves assessment and evaluation of the system's security requirements and specification. Specific mechanisms are necessary to address systems security aspects. Security requirement can be classified into functional and non-functional category.

Component-based software paradigm is a new trend for commercial application, but actually, it is difficult to estimate how the use of software components impacts on security. The major problem in the area of security is in its expression and is measured in different ways. It is hard to compare and trace it. We still do not know how to measure and assess security, how to benchmark security of components or systems and how to prove security to third parties.

6.1 Scope and Goal

The scope of this study is to provide a framework of security requirement, security characteristic to investigate fundamental issues related to security and trustworthiness of Component-based software development. An additional goal is to focus, the
certification of security properties of component based systems and its consistent components. This is to provide a base for QoS and Quality Assurance in CBSE. Measuring the security of a component based software systems is a difficult problem. The present work argues that the security properties of a composite system can be viewed either from the end-user’s or the software integrator’s point of view.

Building a (sufficiently) secured system is a challenging task. Besides the web based application complexity of security is a standalone application to be dealt with as distributed execution. The main focus of this study is on describing security characteristic of composite system. In this study we also propose a scheme that could be used as a guideline to specify the security requirement of new component as well as for the component based system [Zhong 1998]

For access control mechanism based on user identity we propose cryptography techniques digital signature.

6.2 Security Assurance Paradigm

After an Information Technology boom in the mid 90’s with small companies and independent software developers, the information technology industries are re-discovering the need for distributed systems. Software reuse catalyzes improvements in quality by incorporating components whose reliability has already been established. Use of large distributed applications play an increasing central role in today’s environment. The openness of these systems have given rise to questions of security in Component based software paradigm [Gousios 2005]. Security properties of software components can be classified into two groups:-
The issues that arise in the context of component-based security can be viewed in three different perspective angles: (i) the component developer's perspective, (ii) Integrator's perspective, and (iii) the systems user's perspective. These three have different security requirements and visibility of the component. The developer sees the component as a white box but he has no knowledge in which application it is used. Therefore, the developer has no idea about security level requirements. Integrator is unaware of the component code and internal details. He or she treats it as a black box and mainly interested in integrating components to build the system. The component user is much interested in the functionality of the system. He is typically unaware of the component's internal structure and treats it as a black box. Consequently, security requirement is increasing bottom to top and complexity is vice-versa i.e., top to bottom. It means developers, integrator, and users of a component-based system have different needs, expectations, and are concerned with different problems.
From this figure we know that user of component based application is more concerned in ultimate security objective, whereas integrator is interested in security requirement and ensured these requirement. Component is constructed for generic application therefore scope of developer is wide as compared to others.

6.3 Security Assurance Terminologies

6.3.1 Security

Several difficult technical problem remains to be explored and resolved in this field, many resulting from component heterogeneity and the lack of behavioral information about deployed components. In practice, software composers are almost forced to compose systems with components for which they have partial or no knowledge about their underlying security properties.
Measuring the security of a component based software system is a difficult problem. One of the major questions that organizations often face is about the security of the systems. To answer such a question is quite difficult while we don’t have security measurements with absolute certainty; we usually rely on measurement of security characteristic in assessing security [Dixit 2007a].

In component based software system security cannot be defined as a binary logic. It means systems are neither secure nor in secure. Security is often described in relative context. A system is more, or less secure than something else. A component may be proved secure in one application in a particular operating environment, but the same component may not be considered secure at all in a completely different application.

In the assembled systems many of the components may be acquired from the third parties, these are black box components. The security properties of each component will be part of and make impact on the target composite system security.

6.3.2 Certification Process of Component

The goal of certification process is to verify claim or specification provided by the component developers. The certifier is an independent organization which is not related with developers, integrators and users.

\[
\begin{align*}
\text{Developer} & \quad \text{Certifier} \\
\leftrightarrow & \\
\text{Component} \quad \text{verification} \quad \text{Certification} \\
+ & \\
\text{Specification}
\end{align*}
\]

**Figure 6.2 Certification Process**
Due to black box nature of component, sometime the issue relate of security measurement have been neglected. To keep this view in mind we propose a model in which some important factors have been considered like atomic component functionality, interface and protocol between components.

6.4 Security Measurement of Component Based Software System

A key property of software component technology is predictability, which means that the properties of a component based software systems can be deduced from the properties of the individual components.

(1) In CBSE Security is an emergent property, so it is insufficient for a component to be secure. For the whole system to be secure, all relevant components must collaborate to ensure the security of the system?

(2) Component based software systems are comprised of a number of components assembled into a unified, meaningful, single module. In this way they resemble an entire system. They have well defined inputs, outputs and functions.

Suppose that the development of composite system C divides the inputs space of C into a finite no. of sub domains C₁, C₂, C₃, \ldots, Cₙ

Let interface between two component i and j is \( I_{ij} \)

Assembled System \( C_a = \{ C_i : 1 \leq i \leq n \} \)

Property P can be Defined as \( P(C_a) = F\{ P(C_1), P(C_2), \ldots, P(C_n) \} \)

Then Security Requirement will be defined as

\( S(C_a) = S\{ F(I_{i1} C_1), F(I_{i2} C_2), \ldots, F(I_{in} C_n) \} \)
6.4.1 Security Policy in CBSS

The issues that arise in the context of component-based systems can be viewed from three perspectives: (i) the component level perspective, (ii) compositional level perspective and (iii) the systems level perspective [Dixit2007d].

6.4.1.1 Atomic Level Security:

Atomic components are the elementary units in component based software systems. Security characteristic of any individual component is directly proportionate to its functionality.

Let there is component $C_i$ which shows functionality $f_i$ then security characteristic is represented by

$$S_i \propto f_i(C_i)$$

6.4.1.2 Compositional Level Security

In the compositional level two components join together to achieve required functionality. Security in the compositional level depend on individual component, interface between them and protocol used for communication.

If there are two component $C_i$ and $C_j$, for communication between these components require interface $I_{ij}$ and some rule of communication known as a protocol $P_{ij}$. In this case security characteristic can be represented as

$$S_{ij} \propto F_{ij}(C_i, C_j, I_{ij}, P_{ij})$$
6.4.1.3 Systems Level Security

Security is an emergent property, so it is insufficient for a component to be secure. The whole system is to be secured, all the components must be combined in a manner to ensure the security of the system. The system level is security mainly concerned with users and integrators of component based systems. In this case user are interested in ultimate security characteristic and integrator with required and ensured security goals.

![Diagram of security levels]

**Figure 6.3 Security Levels** Component Based Software

Putting the Pieces Together In Reference of Component Based Software

Engineering 167
For atomic level component security requirement, we will consider standalone component. It means that security characteristic of this component depends on its functionality. Compositional level security depends on both components functionality as well as interface between them. All the principles involved in compositional level are easily incorporated in systems level but wide scope of system security has made it more complex.

System Level security is quite complex and these are fundamental aspects for critical application. (i) Identification, (ii) Authentication, (iii) Authorization, (iv) Privileges, (v) Security Policies, and (vi) confidentiality

Component based development is an important emerging topic in software engineering, promising long sought benefits like increased reuse and reduced time-to-market. However, there are at present many obstacles to be overcome for success of component based development. Certification of software component security characteristic is a great challenge for research community. A lot of research efforts have been devoted to the analysis and design methods for component based systems. However, only few address the security and certification issue in component based software. Experiments have showed that many undesired result are generated during the integration of third party components. Component Integrator experience difficulties with integrating third party component into systems, this is particularly the case for ‘binary’ components, with no accompanying source code. Today, many applications are re-engineered to use component based technologies. Much attention has been devoted recently to security issues and it is apparent that a high level of security is a fundamental prerequisite for component based software systems.
The secure component based application is widely recognized as a crucial problem in component based software engineering and it has major impact on the overall quality of component based systems. Security in components is going to be an essential feature because components are available as third-party products and these have to be assembled with other components. Thus the main requirement lies in providing security to a component with respect to other component or a group of components. So, there is a need for two levels of security, one at component level and other at compositional level. Security will be given to components in terms of their properties. So, there are two security properties namely Functional and Non-Functional Security (NFS).

The Functional properties are those which provide inter-component security. They will provide security to one component from other components. For giving this type of security, the external security features can be added as functions to that component. These external features can be named of a component as it distinguishes this component from others and encryption technology the component uses as it also distinguishes its communications to specify components with its digital signature on it. Functional Security properties protect the enclosing system from being assembled with unauthorized components. These properties are particularly significant in dynamic assembly scenario where target component may be located in a remote server whose identity can be a questionable one.

Non-Functional Security properties provide intra component security i.e. security within the component. Intra-component security provides security to data within the component. These security mechanisms are inbuilt. Non-Functional
Security properties are attached with various aspects of the component functionality in different layers of implementation, each representing a specific level of abstraction to achieve certain security objective. A component may employ certain Non-Functional Security properties to guard its sensitive data and functionality from being violated by other unauthorized entities. The Non-Functional Security properties embedded with the component's functionality may have substantial impact on the entire security mechanism of the composed system. If there is a weak Non-Functional Security property in a sensitive operation, the rest of the strong SFs may not help much to protect the component.

Signature mechanism is common policy framework to authentication, this concept also elaborate in digital word by using digital signature. Security access policy can be implemented by signature using cryptography principle. The application of component based software systems is increasing day by day for commercial purposes i.e, Web based truncation, Banking (ATM). The concept of digital signature is used as a method to validate the authenticity of customers.

6.5 Digital Signature in Component Based Software

The purpose of security policy is to prevent unauthorized access. It can be classified in two groups

(i) Authentication: Who are you?

(ii) Authorization: Are you permitted?

The main motivation in the both cases is that only authenticated persons are permitted to carry out authorized actions. To solve these types of problems we propose digital

6.5.1 Cryptosystem

Cryptography is mathematical concepts in which text is transfer into some intermediate form known as cipher text. Main motivation behind this mechanism is to prevent unauthorized access of the data.

A cryptosystem can be defined as a five-tuple \((P, C, K, D_R, E_R)\), where the

- \(P\) is a finite set of plain texts.
- \(C\) is a finite set of cipher texts
- \(K\) is a finite set of keys
- \(D_R\) is a finite set of decryption rules
- \(E_R\) is a finite set of encryption rules

For each \((E_R)^k: P \rightarrow C\),

\[\text{Cond. } D_R \quad (E_R(X)) = X\]

This cryptography principle can be easily incorporate in digital signature technique.

6.5.2 Digital Signature in Component based systems

A digital signature in component based system can be defined as a five-tuple

\[(C, S, K, S_a, V_a)\]

*Putting the Pieces Together In Reference of Component Based Software Engineering* 171
$C$, is a finite set of component

$S$, is a finite set of signature

$K$ is a finite set of keys

$S_a$ is a finite set of signature algorithm

$V_a$ is a finite set of verification algorithm

Any key $k \in K$, signing algorithm $\text{sig} \in S_a$, and corresponding verification algorithm $\text{ver} \in V_a$

$\text{sig}^k : C, \rightarrow S, \text{ and } \text{ver}^k : C, \ast S \{\text{true, false}\}$

It can be easily implemented by using XML Modeling. XML is a widely recognize general purpose markup language for representing.

6.5.3 Digital Signature Schema Implementation Using XML Modeling

`<? xml version = “1.0”?>`

`<? xml version = “1.0” encoding= “ ISO -10646-ucs-2”?>`

`<! Signature Modeling-->`

`<title xml:lang= “en-us”> the cryptography based signature modeling </title>`

`<signature id=“XXXX” xmlns= http://www.w3.org/..../../xmladsig#>`

`<SignedInfo>`
Chapter 6

<CanonicalizationMethod
Algorithm="http://www.w3.org//.../REC-xml-c14n-20010315"/>

<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>

<Principal>DSS</Principal>

<Reference
URI="http://www.w3.org/TR/2000/REC-xhtml11-20000126/">

<Transforms>

<Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>

</Transforms>

<Interface Signature="avcsdf" id="Gobal"/>

<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<DigestValue>************=/DigestValue

</Reference>

</SignedInfo>

<SignatureValue>*******=...

</SignatureValue>

<KeyInfo>

Putting the Pieces Together In Reference of Component Based Software Engineering 173
A number of individual components in any component-based bank application work as independent. The service includes keeping customer information, account type, privileges, etc. Digital signature is a good solution for banking truncation security. We are applying the proposed schema to the banking sector case study. When any person interested to access bank data first he or she transmits account ID, signature.

For access-based security, the Digital Signature Component is used to verify against the identity of the customer; for specific operations, the operation access component is used to check against individual customer permission.

**Digital Signature Application in Banking Sector**

Public Component DSPermission (

String name ,

int CustomerID ,

String role ,

bool is Authenticated)
Public Component Access {

    String AuthenticationType(get;)

    Bool IsAuthenticated(get;)

    String Name (get;)

}

Public Component Interest {

    String AccountType(get;)

    Bool IsAuthenticated(get;)

    String Name (get;)

    Int RateIn(get;)

}

Public Component Print {

    String AuthenticationType(get;)

    Bool IsAuthenticated(get;)

    String Name (get;)

}

Trusted Customer

Using System;

Using System.Security.Dss;

Component AuthenticatedCustomer
```csharp
{static void Main(string [] avgs)
{
    call Component DSPermission()

call Component Access ()

try{
    ComponentAction.Doneedful()
}
catch (SecurityException sc)

{ console.writeLine(" SecurityException: " +sc.MSG)
}
}
```

**Figure 6.4** Digital Signature Applications in Security in Banking Sector

In this diagram we consider four component i.e,(i) access right, (ii)interest calculation, (iii)print and(iv) operation (action, event). Any person interested to access his/her account first of all he/she enter account id with some password.
According to the command they provide other component activate. Every component there are interface present to communicate action carried out by customer.

Black-box nature of component, user of component based system concerned with trustworthiness in their applications. Component certification is used to improve trust in sense of quality assurance. Certification is becoming increasingly more important as component based software applications are used for complex system design and decision making problems.

Certification is a very complex process, involves the measurement and evaluation of all the qualitative and quantitative elements of the system.

6.6 Intra Component Security Certification

The traditional certification model in Component Based Software is based on controlling access to abstract resources. However, such a model is inappropriate for real world commercial applications where the security requirements are more concerned with integrity and availability. Furthermore, the security model requires components to be made security aware at build time and this makes it difficult if not impossible to adapt third party components to meet additional commercial security requirement. Measuring the security of a Component Based Software is a difficult task.

Security cannot be defined in binary term. Systems are neither secure nor insecure. Security is often described in relative terms. A component is more, or less, secure than something else. This is caused by the difficulty of specifying how secure a certain system or component is?
Third-party software certification should attest that the software product satisfies the required confidence level according to certification standards such as ISO/IEC 9126, ISO/IEC14598 or ISO/IEC25051.

The main aim of certification is to make components reliable. We can divide certification into three categories. They are Framework certification, component certification and system certification. If there is any fault in that mechanism then the components assigned to that framework also works badly. So, framework's certification is necessary. Component certification deals with mainly internal functionality of component. This certification will deal whether the component is doing required task or not. These components and framework constitutes Component Based System. So, a certification is needed to know how the securities of individual components and frameworks are affecting whole system.

Because of importance of certification, our works mainly focus on how to certify components through their functional and non-functional properties. This leads to studying different certification models and different certification processes. Although, there were different certification models, their main concentration is on certifying quality of the components. Quality itself includes security, availability, reliability, performance, efficiency, etc. Because of the importance of security in component based software engineering, it needs good focus. Though, there were also models on security, but few. So, our work is restricted to studying security aspects of component through their non-functional properties that means intra-component security certification. If we look into that procedure, it is based on evaluation scheme which
compares the system specific security requirements of the enclosing application with component specific security rating.

An evaluation scheme is there which gives component security in terms of rank, based on this comparison. This method is based on CC (Common Criteria) evaluation. Analyzing component’s security properties and comparing with system specific properties is the main objective of this method. But it has the drawback that it only tests the functionality of components as its main security measure[14]. Our work is also similar to this method, which gives an algorithmic way of component certification. Our model can be used as a component itself to work as an automatic component certifier. The main aspect of proposed model is to upgradation of a pre-issued certificate based on some properties. As components are composed dynamically, a separate mechanism is needed to dynamically update the existing certified components. This will save a lot of time which otherwise should be wasted by doing certification from the scrap.

6.6.1 Certification Algorithm

Main steps:

1) Retrieve Component from reusable Repository.

2) Identifying Security Requirements as Security Goals.


4) Prepare Security Metric in X.Y format, here X denote total security characteristics and Y denote particular security goals

6) Evaluation of all Sub-goals.

7) Verification of these Goals with component specification.

8) Evaluation of user's requirement using those sub-goals.

9) Certify Component according their security characteristic.

The purpose of introducing the Certification Driver is mainly to verify the security characteristic of individual components and after this verification it provides a certification of individual component with detail of number of security sub-goals satisfied through it [Dixit2009*2].

6.7.2 Algorithm: Certification Driver

certification_driver( SecurityGoal : in component, Component : in Repository, SecurityMeasure : out Percentage)

CallRetrieve (User_Requirement, SecurityGoals) /*Collect user requirement with security characteristics */

for( i=1; i<=n ;i++)

findComponent( requirement ,specification) /* match requirement with component specification */

GeneraterTest(testData , securityTemplate) // testing process

SecurityMetric (X,Y) /* X -> No. Security properties and Y-> particular security characteristic */

User Security Requirement: Required Security Goals (SG1, SG2, SG3,…… SGn)
Input: Component C, Security Goals SGi[N], Security Metric X.Y format

Output: Security characteristic evaluated Component with parentage goals

Requirements: Test Case Database, Security Template.

Component Function: COMPONENT (name, specification)

RetrieveComponent(Ci, Si)

Certification Driver Function: CERTDRIVER (component name)

for component Ci, security Goals SGi[N], and Security Metric , Xj. Yi

Step 1: CertificationDRIVER take input component from reusable repository, Security Goals, Security Metric.

CertificationDRIVER (Ci, SGi[N], Xj. Yi)

Step 2: If Y -> 0, then go to step 5.

means we are not interested in any particular security characteristic

Step 3: for 0 to N-1

Evaluate Security Goals SGi[N]

3.1: call Security Metric SGi[N]

3.2: Identify the sub goals based on user requirement Sub[n]

3.3: Read sub goals into SUB []

3.4: From SUB [0] to SUB [n]

3.5: call TestCase (SUB [j])

3.6: If there is failure of a sub-goal, increment rank of that test case.
3.7: Otherwise return the percentage value and store it in SUB [i]

3.8: Go to Step 3.4

3.5: Sum up all the percentages in SUB []

3.6: Return that percentage value to SG [i]

Step 4: Go to step 3

Step 5: Call SGi[N] and if available sum up all values in SG [] otherwise continue.

Step 6: Attribute this percentage to COMPONENT

Step 7: Make ‘Y’=0 in Security Metric.

Step 8: Check if there is any other component ready for certification

Step 9: If ready go to step1 otherwise Exit Certification

6.7.3 Security Metric

This concept plays a pivotal role in evaluation of component security certification. We follow some syntax while defining security metric of a particular component. Let us consider “Security Metric X.Y” is a notation of a component, then ‘X’ denotes the number of security properties that component have and ‘Y’ denotes which properties of the components are going to evaluate. Here ‘X’ is an integer and ‘Y’ is a string of integers. ‘Y’ can be of [{x, z}; b] where {x, z} means there are three security characteristic from x, y and z and ‘b’ properties to be checked by third-party certifier. To get a clear understanding, let us consider the following example notation, let security metric 52. 32

compare with security metric format X:, Y,
here i = 2 ; X = 5 ; and Y = 3 means 2\textsuperscript{nd} component have 5 security properties and we are interested to evaluate 3\textsuperscript{rd} security characteristic of this component.

for this implementation we need to have an order among security properties as we are mentioning them as numbers in version code and matching should be perfect. For this we can implement an alphabetic technique for mentioning these security properties or developer should issue a report specifying how he mentioned security properties. The conditions that can be retrieved in Security Metric through X and Y are as follows. Here, ‘X’ denotes the number of security goals prescribed in Security Metric of a component and ‘Y’ denotes the number of security goals which needs again certification. The pre-condition on ‘X’ was that it cannot be zero at any time, only ‘Y’ can vary.

Given X, Y! = 0 and X = Y, then the component is a new component

Given X! = 0, Y = 0 and X = Y, then that component is already certified one.

Given X, Y! = 0 and X > Y, then some security goals needs to be re-certified.

Given X, Y! = 0 and X < Y, not possible condition.

CASE STUDY

We discussed in previous sections how to do certification of component through component’s Security Metric. To get a clear understanding of the algorithm, take an example, a component by name WebTrunc and apply this algorithm. As our model is based on Common Criteria, this example is also based on some specifications given in Common Criteria. The component WebTrunc, tries to store all the events generated by the user. This component can be useful in many domains. The same
component can be used in banking systems to retrieve the previous illegal operations of any user or it can be used in inter-networking of systems where any user tries to hack other computers.

The main objective of this component is to secure other components through information security. WebTrunc can be given with specification along its Security Metric to Certification DRIVER. The User Requirement should contain the required security goals of WebTrunc. Here Security Metric can be given depending on the state of the component. If that component is a new one, then it’s Security Metric should be represented as Version 5.0, assuming that there are five intra-security properties. Suppose it is already certified once and changes made to that component effect only some security goals of that component, then these can be represented as Version 5.2, if only two security goals should be again certified. The security template shown in the above diagram is stored in a database and it will be connected to Certification DRIVER. When Certification DRIVER calls the component WebTrunc, the WebTrunc sends its required goals as Security Goals and Security Metric in X.Y format. Here, the version was given as 5.5 which mean that either it's a new component or a component modified such that all its security goals might be affected. By reading one security goal at a time, Certification DRIVER stores them in SG[]. For these every three goals, Certification DRIVER calls Security Template, to find out its sub-goals. The division into sub-goals mainly depends on the process which would be followed to achieve required level of security these security goals. First, consider the Event Selection security goal, the main objective is to selecting events individually, that means atomicity should be maintained while selecting
events. In the same way, the Event Time goal can also be divided into techniques used to get correct measuring of time at any situation, the Event storage goal can be divided into the efficient memory techniques for maintaining a list of operations what had happened and what effects they got on other events. These storage details can be maintained by some check points.

![Diagram showing the flow of atomicity checks, selection, time, and storage components.](image)

**Figure 6.5 Security Goal Templates**

In this figure, we observed that all the security sub-goals are connected to a Test Case Data Base. This is maintained as a hierarchical data base in which test cases are arranged in descending order of their rank. For maintaining hierarchical test data base, we use XML. The hierarchy in test cases is dependent on the domain of the application and its sub-goals required. In our example, the first hierarchy will be domain and name of the component, second will be security goal of the component.
and third will be sub-goals of the component. Thus, by specifying in this order it will be very easy for retrieving test cases whenever required. The test cases depend on the nature of the component. For white-box components, code-coverage technique will be good and for black-box components, fault-injection technique test cases will yield good result. Like the same way we did distribute percentages to different security goals, we will distribute percentages to each and every sub-goal. At the end, all these percentages which got success while testing are summed up and produced as whole component's security percentage.
Chapter 6

Component input
User

Retrieve 'Y' value

Certification of the component is over.

No

Y = Y - 1

Yes

Security Goal
Call Security Metric

If there is any match

Find Sub-Goal SUB [ ]

Attribute a percentage to component

Calculate percentage security goals of component

Retrieve the percentages stored

A security evaluated component

Figure 6.6 Flow Chart Certification DRIVER

Putting the Pieces Together In Reference of Component Based Software
Engineering 187
This part of research can be concluded by pointing out this view that concepts of security for component based software system is a burning issue for the research community. It describes a new approach for certify intra-component security in component based software systems. This is a brief overview of security requirement by its user, integrator and developers. Digital Signature concept in component based software is also introduced to assure the security in critical application.