A Security Evaluation Framework Based on STRIDE Model for Software in Networks

Xi Chen, Yun Liu, Jin Yi

Abstract

Software in networks, which is a special kind of applications in service-oriented computing and ultra-large-scale systems, is a complex software system deploying on network environment. Requirements of networked software pose many security problems owing to the dynamic topology structure and users' uncertainty. How to evaluate the degree of software security in networks is a challenging problem. In this paper, we present a framework for flexible assessing software to determine how well it can satisfy intended security requirements. On the basis of analyzing the threats which software in network facing, a security evaluation method based on STRIDE model is proposed. According to its own features of networked software and threat classification method of STRIDE model, we design a SN-Security Evaluation Model, in which the dependability-based, vulnerability-based and risk-based approaches are incorporated for the software security estimation. It provides a valuable way to help users to create the threat modeling and evaluating the safety degree for software security. A case study is conducted to verify the framework proposed in the paper.

Keywords: Software Security; STRIDE; Software Dependability; Risk Evaluation

1. Introduction

People find that software is not always trustworthy. Sometimes the behaviors, the results or the performance don't fully meet original expectations. Web applications are often vulnerable for attackers easily access to the application's underlying database. [1] Therefore, the trustworthiness of the software referred to the security and reliability is attracted high attention by researchers. The increased emphasis on system security has brought new researchers from different backgrounds to the field, bringing different perspectives and different skillsets. All of this is for the good since new viewpoints can lead to new insights. [2]

A common characteristic of these factors is that they are dynamic in nature. Such factors include new vulnerabilities and threats, the network policy structure and traffic. Concerning the security assessment, generally, current methods can be classified into three categories, the qualitative, the quantitative and the combination of these two kinds. [3]

In this paper, we propose a novel and practical framework for estimating software security. The evaluation platform is based on STRIDE model, and helps to generate quantitative assessment on security of active software in network.

Aiming at decreasing errors in evaluation, our platform assesses software in three approaches: dependability-based, vulnerability-based and risk-based. With the case study of Intellective Subway application system in a certain enterprise, the experimental results show that this model has a certain degree of generality and reference value.

Our approach yields three main contributions toward efforts to advance the evaluation of security in software systems:

1. We uniquely apply the combination of dependability-based, vulnerability-based and risk-based to estimating security of software in network.

2. It presents a model for the quantitative estimation of the security level of a software, which enables users to comparatively assess the relative security of different software and approaches in a defensible manner.
3. The implementation of the platform provides flexibility, which implies two aspects: (1) There is a defined standard interface to receive testing data with different forms, and that enlarge the evaluating scale; (2) According to different targets, it needs to set parameters and choose testing approaches, which provide the scalability of evaluation platform.

The paper is organised as follows: Section 2 discusses related work. Section 3 introduces a brief overview of STRIDE model, while section 4 provides an estimation model of software security. Section 5 describes the proposed evaluation framework for details. Section 6 presents a practical case study that illustrates the use of our approach to assess security of software in network. Section 7 contains our conclusions and a discussion of future work.

2. Related work

2.1. Dependability-based evaluation

The work by Alkussayer et al. [2] surveyed existing model-based techniques for evaluating system dependability, and summarize how they are now being extended to evaluate system security. It found that many techniques from dependability evaluation can be applied in the security domain, but that significant challenges remain, largely due to fundamental differences between the accidental nature of the faults commonly assumed in dependability evaluation, and the intentional, human nature of cyber attacks. Yu et al. [4] constructed a three-state nonhomogenous Markov model for software safety assessment. The two most important metrics for safety assessment, Steady State Safety and MTTUF, are estimated using the three-state Markov model.

2.2. Vulnerability-based evaluation

Liu et al. [5] proposed VRSS for qualitative rating and quantitative scoring vulnerabilities, which can combine respective advantages of all kinds of vulnerability rating systems. Houmb et al. [6] presented a risk estimation model that makes use of one such data source, the Common Vulnerability Scoring System (CVSS). The CVSS Risk Level Estimation Model estimates a security risk level from vulnerability information as a combination of frequency and impact estimates derived from the CVSS.

2.3. Risk-based evaluation

Saripalli et al. [7] used the definition of risk as a combination of the probability of a security threat event and its severity, measured as its Impact. Our framework complements this work by introducing risk-based evaluation to cohesively encapsulate the architectural security knowledge of the system. Ni et al. [8] described software implementation for on-line risk-based security assessment which computes indices based on probabilistic risk for use by operators in the control room to assess system security levels. Chan et al. [9] constructed a quantitative Bayesian index model for the assessment of enterprises’ Information Security risk. Zhang et al. [3] proposed the assessment method based on attack tree and Bayesian Network to obtain risk with an intuitive manner. Ahmed et al. [10] provided a security metric framework that quantified objectively the most significant security risk factors, which cover both the service aspect and the network aspect of risk toward a system. Risk avoidance based on life cycle management theory can be not only used in software projects, but also in other industries. The method can forecast related risk completely and in a timely fashion [11].

3. STRIDE Model

In information security, a threat represents a potential violation of the security of a system with some negative impact, whereas vulnerability is an actual security flaw which makes a system susceptible to an attack. An attack is an exploitation of a vulnerability to realize a threat. Threat modeling for security evaluation can help identify the threat events, their attack surface and the entry or access points on the software in the context of each threat; analyze the threats and associated risks; and developing mitigating strategies.
The commonly used method of determining the threat is to classify the threat and determine its composition elements, and STRIDE is a typical threat classification model for security evaluation. It can classify the threat according to different sources of the system threat, and it is the English acronym of the following six threat types:

- **Spoofing**: Illegal use of another user's user name and password, authentication information, etc.;
- **Tampering**: Maliciously modify data;
- **Repudiation**: Users refuse to engage in activities, and there is no way to prove that he refuse the agreement;
- **Information Disclosure**: Information is exposed to people who are not allowed to access;
- **Denial of Service**: Refuse to serve the legitimate users;
- **Elevation of Privilege**: Unprivileged users gain access privileges;

Common threats on software in networks can be documented in threat event catalogs, as shown in Lists 1. List 1 is drawn based on top 19 security threats listed for web applications [12] and Web 2.0 systems [13]. Each item here would correspond to a threat event $e$, used in the software security estimation analysis.

**List 1. Threat events compromising Internet security**

1. **Cross site Scripting (XSS)**: script executes in victims browser to hijack user sessions, deface web sites, and introduce worms etc.
2. **Injection Flaws**: user data sent to the web application is not properly validated, which can manipulate a query on the server.
3. **Malicious File Execution**: PHP, XML or any other framework which accept a file from the user is vulnerable to this attack, as the file can contain a malicious script.
4. **Insecure Direct Object Reference**: direct reference to any internal implementation object such as file, database record, key etc can be exploited.
5. **Cross-site Request Forgery**: a logged on user’s pre-authenticated data of a web site can be exploited by attacker’s application when he visits his site.
6. **Information Leakage by Improper Error Handling**: attack on the applications to sniff information on system resources, working and configuration.
7. **Broken Authentication and Session Management**: attack on account credentials and session tokens which are not protected.
8. **Insecure Cryptographic Storage**: web applications which do not use cryptographic functions to protect the data are exploited.
9. **Insecure Communication**: failing to encrypt the network traffic leads to this attack.
10. **Failure to restrict URL Access**: prevention of access by not displaying the urls to unauthorized users can be exploited by direct access of urls.
11. **XML poisoning**: XML traffic between the server and the browser is poisoned.
12. **Malicious AJAX Code Execution**: malicious AJAX code in the web application silently executes attacker’s intent.
13. **RSS/Atom Injection**: RSS feeds are injected with literal Java Script that can generate attacks on client’s browser.
14. **WSDL Scanning and Enumeration**: WSDL (Web Services Definition Language) file is attacked.
15. **Client side validation in AJAX routines**: same as 2 in the above section.
16. **Web Services Routing Issues**: unencrypted SOAP messages in WS Routing leads to this attack.
17. **Parameter Manipulation with SOAP**: attacker manipulates the variables in the SOAP messages.
18. **XPATH Injection in the SOAP Messages**: XPATH statements which take user input are manipulated.
19. **RIA Thick Client Binary Manipulation**: RIA components such as the Flash, Active X Controls and applets downloaded as binary components are decompiled to obtain the code. Applying patches to these binaries can bypass security.
4. SN-Security evaluation model

4.1. Model design

According to its own features of software in networks and threat classification method of STRIDE model, this paper designed $SN – SEM$ (Software in Networks-Security Evaluation Model).

Definition 1: $SN – SEM$ is an evaluation model for the capacity property of SN-Security, which can be denoted by the following elements:

$$SN – SEM (A, P, F, S)$$

Where $A$ is the property set of Software-Security attributes; $P_i$ is the weighting factor; $F$ is the set of trust functions of a variety of software security; and $S$, the set of a variety of evaluation strategies.

Definition 2: the set of security attributes, which imply anti-threat ability property of software

1) The attribute of anti-counterfeiting $A_c$: The commonly used method is identity validation, and the accuracy of validation determines the strength of anti-counterfeiting capability of software in the network.

2) The attribute of anti-tampering $A_t$: Message integrity validation method is generally used, thus the attribute to ensure the message integrity determines the attribute of anti-tampering of software in the network.

3) The attribute of anti-denying $A_d$: Signature is generally used, thus the attribute to ensure the authenticity of the signature determine the attribute of anti-denying of software in the network.

4) The attribute of anti-leaking information $A_l$: The method of information encryption and securing communication channels is commonly used, thus the attribute of ensuring the safety of communication channel capacity and the strength of information encryption determines the attribute of anti-leaking information of software in the network.

5) The attribute of anti-refusing service $A_f$: refers to the attribute of keeping the system away from the critical condition. There are two types of refusing service attacks: one is to overload the system (the most common is the DDOS); the second is to make the system run into the error. These two methods are essentially the same, and both of them make the system enter the critical condition. So the ability of keeping the system away from the critical condition determines the ability of anti-refusing service of software in the network.

6) The attribute of anti-escalating privileges $A_e$: A reasonable distribution strategy of privileges and effective audit mechanism is the effective means to reduce of privilege and escalate attacks, so the reasonability of distribution strategy of privileges and the effectiveness of audit mechanisms determines the attribute of anti-escalating privileges of software in the network.

We defined six security attributes of software in networks according to STRIDE Model. For completeness, we have categorized the STRIDE threat events, shown in list.1, to map to one or more of the 6 Security attributes. $SN – SEM$ methodology would work with any such framework, by assigning relative weights of importance to each SA category, as will be shown later.

Definition 3: Weighting factor $P_i$ refers to the weighting influence coefficient that the 6 parameter values of security attributes of software in the network accounts in the security assessment model.

Definition 4: the strategy set of security evaluation $S$ is the corresponding security strategy set when evaluating the degree of security aiming at security attributes of software security.
4.2. Dos evaluation

To reasonably evaluate the security of software and calculate the security value is a reliable prerequisite and foundation of determining whether the software is safe or not. We can evaluate and measure the security of software by using dependability-based, vulnerability-based and risk-based estimation of software security.

Definition 5: Degree of security, dos in short, is a measurement value to measure the anti-attack ability of software and a function to the degree of security the software can reach as well as the quantitative basis of classifying the security relationship. Because the degree of security is a continuous cumulative, synthetic evaluation information on the property of security strategy and security attribute which all-directionally act on the target entity all round. Therefore in the following part, we mainly use the accumulated comprehensive evaluation index to evaluate the degree of security.

Contribution degree of software security property: scan the security of software by combining the security attribute of software on a regular basis and thus indirectly measure the contribution degree of security. We assume that, if the more times the security attribute property of a software is scanned and the more inclusive area it covers, we think the higher of the software’s security degree will be; otherwise, if the vulnerability is rarely scanned or the scanning range is not wide, then we believe the lower of the software’s security degree will be, which is defined as follows:

\[ \text{dos}_i = \sum_{k \in A} f(A_k) \times p_k \]

Where \( A \) represents the property set of security attribute, \( f(A_k) \) represents the influencing factor of the security attribute property which acts on \( A \), and \( p_k \) is the corresponding weighting factor.

5. The Security evaluation framework

The inspiration for our evaluation framework comes from recognition of the critical need for assessing the security of a software system. The proposed technique strengthens the estimating accuracy of a software security by incorporating three distinct approaches seamlessly into a cohesive framework. Degree of risk evaluation software

The initial component of the estimation is a dependability-based approach. A dependability-based approach is an effective way of ensuring design quality and addressing architectural concerns. The key objectives of dependability-based evaluation are to evaluate MTTUF (Mean Time To Unsafe Failure) and (Steady-state safety). And it could calculate MTTUF of next round.

Secondly, the incorporation of vulnerability-based estimation improves the quality of security components in the architecture. Scoring software vulnerabilities have proven effective for dealing with security problems in a software. Therefore, the proposed framework incorporates vulnerability-based estimation as one core component.

The third factor is the incorporation of a risk analysis model. Babar et al. [14] surveyed the state of practice in evaluating software architecture and concluded that 88% of the survey participants conducted security review with the goal of identifying potential risks. The proposed evaluation framework encompasses three main aspects. A detailed explanation of these approaches follows.

5.1. Dependability-based estimation

At any time safety-critical software could be in the three states: the operational state, the fail-safe state, and the fail-unsafe state. A software is in the operational state when it is operating correctly, and the probability of the software staying in the operational state at \( t \) is denoted as \( P_O(t) \); a software is in
the fail-safe state when it has ceased to perform its functions but in a safe manner, and the probability of the software staying in the fail-safe state at $t$ is denoted as $P_{FS}(t)$; a software is in the fail-unsafe state when it has failed and the failures have not been handled in a manner that guarantees the safe operation of the software, and the probability of the software staying in the fail-unsafe state at $t$ is denoted as $P_{FU}(t)$. At any time $t$, software obeys the following equation, since at any time $t$, the software can be in one and only one of the three states. These three states are mutually exclusive states, and at $t = 0$, the software always starts at the operational state.

$$P_O(t) + P_{FS}(t) + P_{FU}(t) = 1$$  \hspace{1cm} (4)$$

![Figure 1: The three-state Markov model](image)

Given the existence of a fault, a system could be recovered or not be recovered. “a system recovers” means that the fault falls within the risk containment region of the system and we call the fault a safe fault. If a system is not able to recover, it means that the fault falls beyond the risk containment region of the system and we call the fault an unsafe fault. Safe faults cause a system to go to the fail-safe state, a non-operational state that will not cause a mishap. Unsafe faults cause a system to go to the fail-unsafe state, a non-operational state that will cause a mishap [4].

Definition 6: $N$ is a random variable which follows the geometric distribution. In the independent Bernoulli trials, the system has probability $C$ to arrive at the fail-safe state and has probability $1 - C$ to arrive at the fail-unsafe state when a failure occurs, according to the definition of the coverage. According to [4],

$$C = \frac{N_{s,s}}{N_S}$$  \hspace{1cm} (5)$$

Where $N_{s,s}$ is the number of safe faults, and $N_S$ is the number of faults. Both of them are numbers that have limited values.

According to Bound Theory[4], we could be able to get:

$$MTTSF \geq \frac{e \cdot t}{N_{s,s}}$$  \hspace{1cm} (6)$$

Using Wald’s equality, we are able to prove:

$$MTTU_U = \frac{MTTSF}{1 - C}$$  \hspace{1cm} (7)$$

5.2. Vulnerability-based evaluation
Vulnerabilities are extremely important for network security. IT management must identify and assess vulnerabilities across many disparate hardware and software platforms to prioritize these vulnerabilities and remediate those that pose the greatest risk.[5] Our solution of vulnerability in software is to scan vulnerabilities which is classified into six types, according to security attributes defined before. Then, scoring the vulnerabilities with base score produced by CVSS. According to common sense, if the software in the past are safe and reliable, then the software has a relative high degree of security. Based on this assumption, $V_A$, the vulnerability score of a security attribute defined as follows:

$$V_A = \frac{1}{n} \sum_{i=1}^{n} V_i$$  \hspace{1cm} (8)

As mentioned before, $n$ threats form a security attribute $A$. Then, vulnerabilities to the application integrated over the six security attributes is a weighted average:

$$V = \sum_{A_i} W_A V_A$$  \hspace{1cm} (9)

where $W_A$ is the relative weight assigned to an attribute $A$, representing the importance of each security attribute’s contribution degree in the calculate the degree of security.

5.3. Risk-based evaluation

Our Risk-based evaluation is based on QUIRC, which is proposed in[7]. And according to STRIDE model, we defined six attributes of software in the network, as mentioned before. Then, get the average value of all the risk factor of attack technology that may be used in a threat, and get the rounding number. The greater this value is, the greater threat posed to the system. After calculating the risks of each threat, decide which threat shall be given priority to mitigation according to the size of risk.

The scheme proposes a definition for risk as a product of the Probability ($P_c$) of a security compromise, which is combination of the probability, or frequency, of a security threat event and the magnitude of its consequence. i.e. a threat event, $c$, $P_c$ typically is a fraction less than 1, whereas $I_c$ may be assigned a value on a numerical scale.

$$R_c = P_c I_c$$  \hspace{1cm} (10)

The overall platform security risk for the given application under a given attribute $R_A$ would be average over the cumulative, weighted sum of $n$ threats which map to that six attributes:

$$R_A = \frac{1}{n} \sum_{i=1}^{n} P_c I_c$$  \hspace{1cm} (11)

Then, Net Security Risk ($R$) to the application integrated over the six security attributes is a weighted average:

$$R = \sum_{A_i} W_A R_A$$  \hspace{1cm} (12)

where $W_A$ is the relative weight assigned to an attribute $A$, as mentioned before.

In summary, the normalized calculation formula of dos is as follows:

$$dos = m_R R + m_{MTTUF} MTTUF + m_V V$$  \hspace{1cm} (13)

Where $R$ is the normalized risk of software $R$, similarly, as $MTTUF$, $V$. And $m_R$, $m_{MTTUF}$ and $m_V$ are the weighting values checking and validating the system. In order to quantize the software security, we should analyze these threats according to the specific circumstances, calculate the various values, and then calculate the dos.

6. A Practical case study
For illustration purposes, we use Intellective Subway (IS for short) case study. IS is a management software which is applied in subway networks. This context represents a general form of most of today's online web systems from the customer's perspective. Our scheme has identified nineteen specific threats (called vulnerabilities), shown in List.1. These threats convey a simpler form of threat profiling for this case study. Next, we will walk through the framework to evaluate the security of IS.

With the reference to the previous definition of $SN – SEM$ model, carry out the threat modeling and dos evaluation of the SOA application system according to the following steps:

Step 1, Put the test data, which is transformed to defined form, into the evaluation platform. That helps to process data from different testing tools, and make the evaluation platform with scalability.

Step 2, Choose any kinds of appropriate methods between dependability-based, vulnerability-based and risk-based. Then set the weight coefficients according to options and the test object. Figure.2 and figure.3 show dependability-based and risk-based evaluation user interfaces respectively.

Step 3, Calculate the security of software with different three methods according to the options.

Step 4, Evaluate the dos quantized value arrive with the contribution degree weighting factor of security property to get the reference value of software security evaluation of IS system, and then put forward reference for the system administrators to find the weak links in security defense and optimize the design of system security.

![Software Security Comprehension Evaluate]

**Figure.2** Dependability-based evaluation user interface
7. Conclusions

This paper combines dependability-based and vulnerability-based with risk-based analysis to create a framework for software security evaluation. We have carried out applied research based on the evaluation method of STRIDE model in risk assessment, and carried out the threat classification and dos evaluation of software provided by SN – SEM for the target system. We also present a standard interface for receiving testing data with different forms that provides the scalability of evaluation platform. Furthermore, we illustrate the applicability of our framework utilizing a common case study that clearly demonstrates the benefits of evaluating the security of software architecture during the design phase. However, as with all new approaches to improve software security, further validation is required. In future work, we continue to work towards the full formalization and realization of the framework.

8. Acknowledgments

This work is partly supported by National Natural Science Foundation of China under Grant No. 61071076, National High-tech Research And Development Plans(863 Program) under Grant No. 2011AA010104-2 and The Academic Discipline and Postgraduate Education Project of Beijing Municipal Commission of Education.

9. References


