SECURITY RISK ANALYSIS IN WEB APPLICATION DESIGN

Rattikorn Hewett, Phongphun Kijsanayothin
Department of Computer Science, Texas Tech University

Meinhard Peters
Department of Industrial Engineering Technology, The University of Johannesburg

Keywords: Risk assessment, computer security, software design, web-based systems.

Abstract: Web-based information systems play increasingly important roles in providing functions and business services for many organizations. Because of their ubiquitous natures dealing with a huge and diverse population of users, web applications must be tolerant to errors, adverse interactions and malicious attacks. The ability to quickly estimate security risks early in the system development life cycle can be beneficial in making various decisions. This is particularly crucial for large and complex web applications that are asset-critical and evolve rapidly through long life cycles. This paper presents a systematic approach for the automated assessment of security risks, at the design stage, of web-based information systems. The approach combines risk concepts in reliability engineering with heuristics using characteristics of software and hardware deployment design to estimate security risks of the system to be developed. It provides a simple early estimate of security risks that can help locate high-risk software components. We discuss limitations of the approach and give an illustration in an industrial engineering and business-to-business domain using a case study of a web-based material requirements planning system for a manufacturing enterprise.

1 INTRODUCTION

The scope and complexity of web applications have grown significantly from small-scale information dissemination to large-scale sophisticated systems that drive services, collaboration and business on a global scale. Unlike conventional software systems, web-based systems grow and change rapidly. They also tend to be living systems having long life cycles with no specific releases and thus, require continuous development and maintenance (Ginige and Murugesan, 2001). Designing web applications involves not only technical but also social and political factors of the system to be built. Poorly developed web applications that continue to expand could cause many serious consequences. Because of their ubiquitous natures that deal with a huge and diverse population of users, web applications must be tolerant to errors, adverse interactions and malicious attacks. It is desirable to have mechanism that can quickly estimate the security risks of such systems at an early stage of their life cycles.

Various techniques and guidelines to security risk analysis are available (Landoll, 2006; Stoneburner et al., 2002) and activities to increase security assurance of software are integrated throughout the software development life cycle (e.g., security requirements, code analysis, security and penetration testing). However, most security risk assessments of software systems are done at a system level after the systems have been implemented.

Software vulnerabilities can occur in codes (e.g., buffer overflow, SQL injection) and designs (e.g., internal controls that attackers can compromise). They can even exist in components that are not involved with security procedures. As many soft ware-intensive infrastructure systems increas-ingly utilize COTS (Commercial Off-The-Shelf) components for rapid application development, ability to analyze security risks at a software component level becomes essential.

As web applications are implemented on standard platforms, they inherit similar security weaknesses that are published and available publicly (Bugtrag,

Hewett R., Kijsanayothin P. and Peters M. (2007).
SECURITY RISK ANALYSIS IN WEB APPLICATION DESIGN.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Internet Technology*, pages 28-35
DOI: 10.5220/0001266700280035

2006). These known vulnerability databases are used widely as parts of basic principles for guidelines, standards, and vulnerability scanning tools such as Nessus (Nessus, 2006). Currently, like other information systems, most web-based systems rely on standard checklists, firewall rules and scanners for system security risk testing. One drawback of these tools is their lack of ability to assess vulnerabilities specific to custom web applications (Walt, 2002).

Most traditional risk analysis techniques are difficult to apply directly to modern software design, as they do not necessarily address potential vulnerabilities and threats at a component or environment level (Verdon and McGraw, 2004). Particularly, web-based information systems are designed on multi-tier architectures spanning over multiple boundaries of trust. A vulnerability of a component depends on both its platform (e.g., J2EE on Tomcat/Apache/Linux, C# on a Windows .NET server) and environment (e.g., exposed network such as LAN versus secure DMZ). To adapt risk methodology to web application design, security risk analysis should consider risk factors from different design levels including software components, hardware platforms and network environments.

Risk analysis aims to provide some assessment of risk to serve as a basis for decision-making to prevent, mitigate, control and manage risks of loss. This paper presents a systematic approach for automated security risk assessment of a web application system from its component-based designs. By combining risk concepts in reliability engineering with heuristics using characteristics of software design and hardware platforms, the security risks of a proposed system can be estimated. In contrast with rigorous analysis of precise security risks during the operational phase, our approach provides a simple early estimate of security risks that can help locate design components that are at risk. Thus, mitigating strategies (e.g., allocating resources for rigorous testing, or redesigning the high-risk component) can follow.

Section 2 describes related work. Section 3 defines relevant security terminologies and risk concepts and describes the proposed approach. Section 4 illustrates our approach to a web application in industrial engineering. Conclusion and future work are discussed in Section 5.

2 RELATED WORK

Much research in web information systems is concerned with techniques and methodologies for engineering web-based systems effectively by adapting principles from software engineering to web application design and development (Barna et al., 2006; Qiang et al., 2001; Ginige and Murugesan, 2001).

Risk methodologies have been employed to assess safety and performance associated with software systems in various application domains including industrial engineering, business and space science (Shahrokhi and Bernard, 2004; Cortellessa et al., 2005; Yacoub et al., 1999). Risk methodology including FMEA (Failure Mode and Effect Analysis) has been applied to gain understanding about usability of web information systems (Zhang et al., 2001). Unlike our approach, none of these work addresses risks in security contexts. Identifying likelihoods in risk methodology in these web information systems are mostly subjective rather than analytical and objective.

Our work is most similar to a methodology and framework proposed by Yacoub et al. (Yacoub et al., 1999). However, instead of assessing the reliability of software components, we assess the security risks associated with software. Thus, their risk models contain different attributes than ours. Reliability risks are estimated from the complexity of software components whereas security risks are based on characteristics of software design as well as hardware platforms to be deployed.

3 SECURITY RISK ANALYSIS

In this section, we define basic risk analysis concepts and relevant security elements in Section 3.1. Section 3.2 describes the proposed methodology.

3.1 Preliminaries

Risk is defined as a multiplication of the probability of occurrence of an undesired event and its consequences (ISO, 2002). *Risk analysis* is a process that aims to assess, manage and communicate risk information to assist decisionmaking. *Risk assessment* involves identification of adverse situations (or *hazards* or *threats*) and quantifying their risks by determining all *likelihoods* of their causes and the *severity* of their impacts. The risk measures obtained can assist in cost benefit study for resource planning to prevent, eliminate or mitigate these undesired situations. These are activities of risk management. Many techniques for risk assessment exist (e.g., FMEA, HAZOP (Hazard and Operability), FTA (Fault Tree Analysis) (Haimes, 2004)), but they have been mainly applied to hardware rather than software artifacts. Risks associated with software (or *software risks*) are risks that account for undesired consequences of software behaviour, in the system in which the software is used.

Security and *safety* are two related perspectives of value judgements of acceptable risks. Security is concerned with the protection of ownerships of *assets* and privacy, while safety is more concerned with the protection of human lives and environments, factors that cannot be evaluated easily in monetary terms. Examples also include some assets in web applications - e.g., trust and a company's reputation. Unlike safety, sources of security threats include malicious (e.g., deliberate attack, code injection) as well as non-malicious actions (e.g., natural disaster, human error, software system failure) that may lead to a security breach.

Security attacks, whether intentionally targeted or not, can originate from internal and external threats depending on the degree of *exposure* of assets and how vulnerable the system is to attack. A system's *vulnerability* refers to its weakness (i.e., specific attribute/environment) that makes a system more prone to attack or makes an attack more likely to succeed. Exploited vulnerabilities can compromise security.

In finance, risks are estimated in terms of expected loss per year, ALE (Annualized Loss Expectancy), which is a product of the annual rate of occurrence and single loss expectancy. The former reflects probability, and the latter for consequences over a period of time. In security risk analysis, risks are estimated from adapted ALE formulae. Basic probability factors are related to exposure degrees, vulnerabilities and threats, whereas basic consequence factors are measured by asset values. Thus, for a given threat/adverse action, its security risk can be roughly estimated by:

$$Risk = threat \times exposure \times vulnerability \times severity,$$

where *threat* actually measures the threat likelihood. The first three contributes to probability measures, whereas the last measures its consequence. In this paper we omit threat, which can be easily integrated into our analysis at a later stage of the design. While the ALE concept works well in general financial and security risks, it may not be applicable for cases like terrorist attacks, which are rare or may never occur.

3.2 Methodology

Quantitative risk assessment methodologies, such as the ALE-based approach, provide useful mechanisms to facilitate risk ranking for decision-making in risk management. However, they are often criticised for producing results that are too general (statistic-like) to be useful for specific decision-making, and producing results that are not credible enough as they rely heavily on subjective value assignments.

The authors' proposed methodology given below has two objectives: (1) to estimate security risks of a web-based software system from its early design configuration, and (2) to estimate probability measures more objectively. The latter is achieved by using heuristics based on relevant design attributes (more details later). Because web applications are typically designed with multiple architectural layers and trust boundaries, our risk methodology must incorporate relevant attributes from multiple design levels (e.g., component, network and hardware platforms). Figure 1 shows basic steps in the methodology.

Inputs: Software design, hardware deployment design, likelihood of each scenario. Let s_k be the likelihood of scenario k.

Outputs: Security risks of a system to be built and security risks of each component.

1) **Determine interaction rates in each scenario**: For each use case, for each scenario k, for any two components i and j, compute $I_{ijk} = n_{ijk}/N_{ik}$, where I_{ijk} is an interaction rate from i to j in k, n_{ijk} is the number of interactions from i to j in k, and N_{ik} is the number of all interactions out of i in k.
2) **Determine transition rates in all scenarios**: Construct a dependency graph of components where component transitions are based on interaction rates from all scenarios. Let t_{ij} be a transition rate from components i to j. Then $t_{ij} = \sum_k s_k I_{ijk}$. Normalize t_{ij} so that a total sum of transition rates from each component is one.
3) **Estimate component vulnerability**: By using hardware deployment, determine V_i, an estimated *vulnerability* ratio of component i. Note that a total sum of vulnerability ratios over all system components is one.
4) **Combine likelihood estimates**: Determine probability of security breach of each component by identifying all access paths to the component. For each path, estimate the likelihood by using the transition rates (t_{ij}'s) and vulnerability ratios (V_i's) of components along the path. Combine the likelihoods obtained for each alternative path to the component.
5) **Severity Analysis**: Determine severity of each adverse action in each component. Categorize the severity and select the worst case in each component.
6) **Compute security risks**: for each component and the overall system using results obtained from 4) and 5).

Figure 1: Proposed risk methodology.

The methodology requires two inputs: a software design and a hardware deployment design. We assume that the software design is expressed in terms of an extended activity or workflow diagram, where the software components of the web applications and use case actors are represented as well as activities in use cases (Barna et al., 2006). The software design includes all use cases, each of which may have one or more scenarios.

The hardware deployment design provides information about hardware platforms including host or server, their attributes (e.g., services, operating environments), and network attributes (e.g. IP addresses for network configuration, and LAN, WAN or DMZ for connection). Security scanning tools use this information to identify vulnerabilities of each host in the system. The hardware deployment plan also indicates software components designated to use, or run on, corresponding hosts. Examples of a workflow diagram and a hardware deployment diagram are shown in Figure 3 and 4, respectively.

As shown in Figure 1, the first two steps use software design characteristics as heuristic measures of probability factors of risk. Step 1) computes *interaction rates* among software components for each scenario based on the design workflow. For simplicity, we assume that each outgoing interaction from the same component occurs equally likely (when insights about interaction likelihoods are available, appropriate weights can be applied). Step 2) constructs a graph whose nodes represent software components. *Transition rates* among components are computed by taking scenario likelihoods into account for combining interaction rates from each scenario. In security risk contexts, scenario likelihoods are heuristic measures of probabilities of potential threats (e.g., when scenarios involve adverse actions), whereas interaction rates are heuristics that partially measure degrees of component exposure.

Step 3) uses characteristics of hardware deployment design as a heuristic measure of an additional probability factor of security risk. We first determine vulnerability sources of each host. This can be done by a table lookup procedure on publicly available vulnerability databases for vulnerability sources based on relevant host and network attributes (e.g., IP addresses, services). Examples of vulnerability sources include Apache Chunked-Code software on web servers, operating environments Windows XP SP2 and JRE 1.4.2 for running JAVA programs on application servers, and Oracle and TNS Listener software for database servers. Such vulnerabilities and their sources can be easily identified by scanning tools when hardware platforms to be deployed are already in operation.

Next step is to measure vulnerabilities of each software component. Each component may require one or more hosts. For example, a component that runs on one application server may also need to query data on another database server. From the deployment design, we can determine all hosts required by each component. Let H_i be a set of hosts required by software component i and v_h be a number of vulnerabilities of host h. For component i, a number of vulnerabilities, v_i and its normalization to a *vulnerability ratio*, V_i are estimated as:

$$v_i = \sum_{h \in H_i} v_h \text{ and } V_i = v_i / \sum_i v_i \ .$$

Here vulnerability ratios of software components are heuristic measures of likelihood factors of security risks. The more vulnerable software component is, the more likely that the system security can be compromised.

Step 4) combines all probability factors obtained in Steps 2) and 3) to estimate the likelihood of each component being attacked (or security breach). Here we interpret this likelihood as a chance of a component under investigation to be the first being attacked in the system. Thus, all components and links along the path to reach the component are secured (unexposed to attack and unexploited vulnerability). The likelihood of a node being secured is estimated by subtracting its exposure degree, estimated by its weaknesses (i.e., vulnerability ratio), from one.

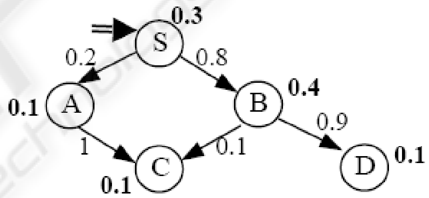

Figure 2: Likelihoods of security breach in components.

We elaborate this step by a small example of a dependency graph shown in Figure 2, where nodes S, A, B, C and D represent components of a software design. A double-lined arrow indicates that S is an entry point to software system execution. Each component and transition is annotated with a corresponding vulnerability ratio (in bold) and a transition rate, respectively. As shown in Figure 2, the likelihood of S to breach security is determined by the vulnerability ratio of S itself. The chance of A being attacked (via $S \rightarrow A$) is estimated by its chance to be reached without attacks (i.e., S being secured and S transits to A) and its exposure factor (weaknesses). The former is estimated by the likelihoods of S being secured and S transiting to A, whereas the latter is estimated by A's vulnerability ratio. Thus, the chance of A being attacked is estimated by $(1-0.3) \times 0.2 \times 0.1 = 0.014$. Similarly, the likelihood estimate of security breach in B is 0.224. The chance of C being attacked is estimated by the sum of the chances of C being attacked via $S \rightarrow A \rightarrow C$ (i.e., $(1-0.3) \times 0.2 \times (1-0.1) \times 1 \times 0.1 = 0.0126$) and via $S \rightarrow B \rightarrow C$ (i.e.,

$(1 - 0.3) \times 0.8 \times (1 - 0.4) \times 0.1 \times 0.1 = 0.00336)$. This gives 0.01596.

In general, the estimated probability along an access path is analogous to propagation of probabilistic effects in Bayesian network by using Bayes Rule (Pearl, 1997). The transition rate plays a similar role to a conditional probability of a destination being attacked given a source being reached. For simplicity, we do not explicitly represent access privilege. We assume that any activity in the same component can be equally accessed. However, not every component deployed by the same host can be accessed equally. We also assume that network connections between hosts do not have vulnerabilities. Such vulnerabilities can be easily incorporated in our proposed approach. Note that since all probability factors (transition rate, vulnerability ratio) are no greater than one, the longer the path to reach a component is, the harder it is to attack (i.e., its chance of being attacked decreases). For a path that contains cycles, we consider the maximum likelihood that can be obtained. Thus, the heuristic function we use for estimate likelihoods conforms to intuition in security contexts.

In Figure 1, Step 5) estimates consequences of each adverse action in each software component using approaches similar to HAZOP or FMEA. In this paper we focus on probability estimation. Thus, we apply a typical severity analysis technique, categorize and quantify component consequences according to security standards (ISO, 2002). By using the maximum severity obtained in each component, we can compute an estimate of security risks of each component of the system being design. For a risk estimate of an overall system, we estimate the likelihood of a system by multiplying the likelihood of a system to safely terminate by a vulnerability ratio of overall system components, and the system severity. We estimate the system severity by taking a majority of severity categories of components in the overall system. Section 4 illustrates our methodology in a case study of web application design.

4 A CASE STUDY

This section presents an illustration in an industrial engineering and business-to-business (B2B) domain with a case study, adapted from (Qiang et al., 2001).

4.1 Web Application Design

A web-based material requirements planning (MRP) system takes customer's product order through the B2B web front and performs material requirements planning. The goal of the system is to respond quickly and effectively to changing requirements to minimize delay in production and product delivery. The system automatically updates orders and predicts the daily planned orders, which may be overwritten by a manager. The daily planned orders are sent to a job shop that simulates real-time scheduling and production. At the same time, the system determines materials required for the daily planned orders and sends orders of raw materials to appropriate suppliers. Job shop simulator reports on finished parts/products to the MRP system, where the manager may adjust some initial parameters (e.g., lead time, lot size) for scheduling production plan to meet customer order requirements. A supplier can view material order status and acknowledge the receipt of the order.

The web application design consists of seven software components: MRP, MPS (Master Production Schedule), BOM (Bill of Material), JOB (Job shop simulator), CI (Customer Interface), MI (Manager Interface), and SI (Supplier Interface). MRP validates logins, records product orders and determines customer's billing. MPS schedules appropriate daily production plan from a given product demand (e.g., from customer order) and other input parameters. The details of MPS are beyond the scope of this paper. BOM determines parts required for each product to be produced and sends order for raw materials to suppliers. JOB is a job shop component that simulates real-time scheduling and production in a manufacturing plant. CI, MI and SI facilitate web interfaces for different users. Customers and suppliers are required to login via a web front, while a manager can do remote login to MRP.

To focus our illustration on risk methodology, we simplify a B2B process by assuming that all payment between a customer and a manufacturer, or a manufacturer and a supplier are off-line transactions. We represent the design in an extended workflow diagram as described in Section 3.2. To make it easy to understand our analysis approach, we present the workflow in three use cases, each of which has one scenario. The three use cases are *customer orders products*, *manager adjusts demands/requirements*, and *supplier acknowledges material order*. We assume that the likelihood of occurrence of each scenario is 0.4, 0.2, and 0.4, respectively.

Figure 3 shows a workflow diagram of a process to handle customer orders for products. We assume customers have established business relationships with the manufacturing enterprise and therefore MRP is restricted for use with these customers only. Thus, customers and suppliers are required to login via a web front. The workflow starts at a dark circle and ends

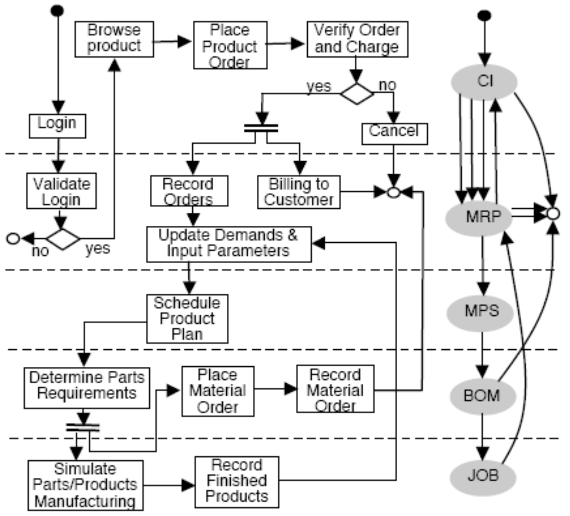

Figure 3: Workflow diagram of a customer use case.

at a white circle. It represents both sequential (by directed link) and concurrent (indicated by double bar) activities among different components. For example, after determining parts required for products production, BOM sends results to job shop to simulate production of product orders and concurrently it places order of raw materials to appropriate suppliers. The workflow diagram is extended to include software components responsible for functions/executions of these activities. The components are shown in oval shapes on the right of the figure. The arrows between components indicate the interactions among them.

Second part of the design is a hardware deployment plan. Figure 4 shows the design of hardware platforms required. Like many web applications, the MRP system has a three-tier architecture. The presentation layer requires a web server, W. The business logic layer contains three application servers: $A1-A3$. Finally, a data layer employs four database servers: $D1-D4$, each facilitates data retrieval from different databases as summarized at the bottom of the figure. The servers/hosts are connected via LAN with a firewall between the web server, W and the LAN access to the three application servers. The deployment plan annotates each server with software components that rely on its services as appeared in a square box on top of each server. For example, as shown in Figure 4, a manager can login remotely to MRP via MI, both of which run on application server $A1$. MRP and MI allow him to review production status and adjust product demands. Since MRP is also responsible for login validation, it needs to retrieve password data. Thus, unlike other application servers, $A1$ needs to be connected with $D1$. Next section describes the analysis for assessing security risks associated with each of these software components from a given design.

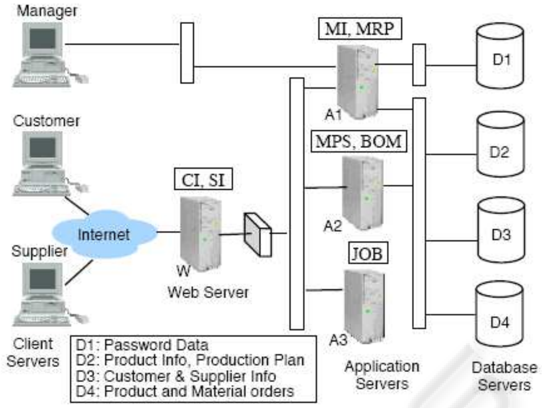

Figure 4: Hardware architecture and deployment plan.

4.2 Estimating Security Risks

The first step for the proposed risk methodology is to calculate interaction rates from each scenario of the web application design. As shown in Figure 3, MRP has one interaction to MPS but has a total of four interactions from it (two from MRP to a terminal state, one to CI and one to MPS). Thus, an interaction rate from MRP to MPS is 1/4 as shown in table M_1 of Figure 5. M_1 represents interaction rates among components obtained from a "customer orders product" scenario, where T is a terminal state in the workflow. Similarly, we can obtain tables M_2 and M_3 representing interaction rates in the manager and supplier scenario, respectively.

M_1	*CI*	*MRP*	*MPS*	*BOM*	*JOB*	*T*
CI	0	3/4	0	0	0	1/4
MRP	1/4	0	1/4	0	0	1/2
MPS	0	0	0	1	0	0
BOM	0	0	0	0	1/2	1/2
JOB	0	1	0	0	0	0

M_2	*MI*	*MRP*	*MPS*	*BOM*	*JOB*	*T*
MI	0	2/3	0	0	0	1/3
MRP	1/3	0	1/3	0	0	1/3
MPS	0	0	0	1	0	0
BOM	0	0	0	0	1/2	1/2
JOB	0	1	0	0	0	0

M_3	*SI*	*MRP*	*BOM*	*T*
SI	0	1/2	1/2	0
MRP	1/2	0	0	1/2
BOM	0	0	0	1

Figure 5: Tables of interaction rates in each scenario.

Step 2) constructs a dependency graph of software components. For each pair of components, we compute a transition rate using the corresponding interaction rates obtained from each scenario (M_1, M_2 and

M_3) and a given likelihood of each scenario, as described in Figure 1.

In Step 3), determine vulnerabilities of hardware platforms from a given deployment plan shown in Figure 4 by applying a scanning tool (if the required host and network configurations of the system are already in operation), or (otherwise) a table lookup procedure to public vulnerability databases. Table 1 shows vulnerability sources and their counts on our application deployment plan. As described earlier, the table lookup for vulnerability sources are based on software and its environments required on each host to provide their services.

Table 1: Vulnerabilities from hardware deployment plan.

Host	Vulnerability Sources	Count
W	Apache-Chunk, PHP 4.2	2
*A*1	Jboss, JRE 1.4.2, Windows, Tomcat-3.2.1	4
*A*2	Jboss, JRE 1.4.2, Windows	3
*A*3	Telnetd	1
*D*1	Oracle, TNS Listener	2
*D*2	Oracle, TNS Listener	2
*D*3	Oracle, TNS Listener	2
*D*4	Oracle, TNS Listener	2

Table 2: Component demands and vulnerability ratios.

	W	*A*1	*A*2	*A*3	*D*1	*D*2	*D*3	*D*4	#*Vuls*	V_i
CI	1					1			4	0.09
SI	1						1	1	6	0.14
MI		1				1		1	8	0.2
MRP		1			1	1	1	1	10	0.23
MPS			1			1	1		7	0.16
BOM			1				1	1	7	0.16
JOB				1					1	0.02

Next, to find a vulnerability ratio, V_i of component i, we determine host services required by the component. For example, the customer interface, CI requires the web server W for login and ordering products, and also requires query retrievals of product information from the database $D2$ for browsing. Similar results can be obtained for other components as summarized in Table 2. The second to last column shows the number of vulnerabilities associated with each component. This is estimated by summing a total number of vulnerabilities in each of the hosts required in the component. The last column shows vulnerability ratios with a total ratio of one for overall system vulnerability.

Figure 6 shows the resulting graph obtained from Step 2) with annotated transition rates and vulnerability ratios (in bold), heuristic estimates of vulnerability of the design components. Here we assume that at the starting point S (a solid circle), the system has no vulnerability, whereas the terminating point T (a white circle) has a vulnerability ratio of the whole system (i.e., one).

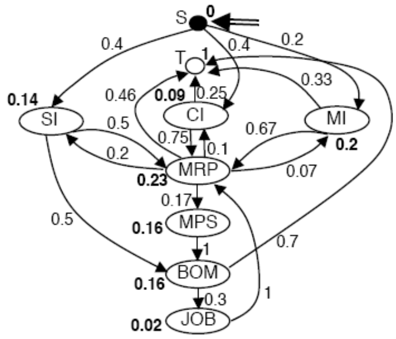

Figure 6: A dependency graph of software components.

Step 4 determines likelihoods of attacks to (or security breach in) each component using the approach as described in the example of Figure 2. For example, the chance of CI being attacked (via $S \rightarrow CI$) is $1 \times 0.4 \times 0.09 = 0.036$. Note that the path to CI from MRP (via $S \rightarrow CI \rightarrow MRP \rightarrow CI$) is ignored since it produces a lower likelihood and thus, not the worst case as desired. The chance of MRP being attacked is computed as a sum of likelihoods of MRP being reached from CI, MI, SI, and JOB, i.e., $0.06 + 0.02 + 0.04 + 0.01 = 0.14$. A summary of component security breach likelihoods is given in the first row of Table 4. In Step 5), we estimate consequences of adverse actions in each component by categorizing consequences into five categories: CT (catastrophic: 0.95), CR (critical: 0.75), MG (marginal: 0.5), and MN (minor: 0.25) (ISO, 2002). The estimated severity for each component is shown in Table 3. By selecting the maximum severity obtained in each component to be the component severity, and applying standard quantification to it, we can obtain risk of each component as shown in Table 4. Here MRP is the most high-risk component with 34.31% risk, whereas MI is the lowest with 2.64% risk. The risk of overall system is computed by multiplying the likelihood

Table 3: Severity analysis of the web application design.

SW units	Malicious actions	Consequences	Severity
CI	change product order	order unverified	MN
	increase access level	increase vulnerabilities	**CR**
MRP	change product order	incorrect charge, plan	MG
	change demands	over/under stock	MG
	change charges	lose profits	CT
	access customer info	lose financial info	**CT**
	access product info	lose product info	MG
MPS	change schedule	incorrect plans	**CR**
	access product info	lose product info	MG
	access supplier info	lose supplier info	MG
BOM	change parts	incorrect product plan	**CR**
	increase access level	increase vulnerabilities	MG
	access product info	lose product info	MG
	access supplier info	lose supplier info	MG
JOB	change finished time	late delivery, lose cust.	MG
	access product info	lose product info	**MG**
MI	change prod. status	wrong update	MN
	increase access level	increase vulnerabilities	**MN**
SI	increase access level	increase vulnerabilities	**MG**

Table 4: Risk analysis results.

	CI	*SI*	*MI*	*MRP*	*MPS*	*BOM*	*JOB*	*T*
Likelihoods	0.04	0.06	0.04	0.14	0.07	0.08	0.13	0.665
Severity	0.75	0.5	0.25	0.95	0.75	0.75	0.5	0.75
Risk	0.03	0.03	0.01	0.13	0.05	0.06	0.07	0.499
%Risk	7.13	7.39	2.64	34.31	14.51	16.77	17.25	

of reaching the terminating point, T with its vulnerability and a severity of the overall system. Here we use, CR, which is a severity category of a majority of all components. The result in Table 4 shows that the security risk of overall system is about 50%, which can be used as a baseline to compare with alternatives. These risks give relative measures for assisting decisionmakings on security options from different designs.

5 CONCLUDING REMARKS

This paper attempts to provide a general framework for security risk analysis of web application software design. We present a systematic methodology for automated preliminary risk assessment. Our focus here is not to estimate precise risk measures but to provide a quick and early rough estimate of security risks by means of heuristics based on characteristics of software design and hardware platforms to help locate high-risk components for further rigorous testing. Our risk model is far from complete partly due to inherent limitations in the design phase. Although, we illustrate our approach to web applications, it is general to apply to other softwareintensive systems and to extend the framework and the methodology to include additional security elements in the model.

The approach has some limitations. Besides obvious requirements on knowledge about system usage scenarios and component-based design, the approach is limited by insufficient data available in the early phases of software life cycle to provide precise estimation of the system security. However, the risk analysis methodology suggested in this paper can be used to estimate security risks at an early stage, and in a systematic way. Finally, the approach does not consider failure (to satisfy security criteria) dependencies between components. Thus, risks of a component connected to attacked components are determined in the same way as those that are not attacked. We plan to extend this framework to address this issue by providing mechanisms that take user roles, access privileges and vulnerability exploits into account. Additional future work includes refinement of explicit severity analysis and incorporation of attack scenarios in the risk models.

REFERENCES

Barna, P., Frasincar, F., and Houben, G.-J. (2006). A workflow-driven design of web information systems. In *ICWE '06: Proceedings of the 6th international conference on Web engineering*, pages 321–328, New York, NY, USA. ACM Press.

Bugtrag (2006). Retrieved October 11, 2006, from http://www.securityfocus.com/archive/1.

Cortellessa, V., Appukkutty, K., Guedem, A. R., and Elnaggar, R. (2005). Model-based performance risk analysis. *IEEE Trans. Softw. Eng.*, 31(1):3–20.

Ginige, A. and Murugesan, S. (2001). Web engineering: an introduction. *Multimedia, IEEE*, 8(1):14–18.

Haimes, Y. Y. (2004). *Risk Modeling, Assessment, and Management*. John Wiley & Son, 2nd edition.

ISO (2002). Risk management vocabulary guidelines for use in standards. ISO Copyright Office, Geneva.

Landoll, D. J., editor (2006). *The Security Risk Assessment Handbook*. Auerbach Publications.

Nessus (2006). Nessus vulnerability scanner. Retrieved October 11, 2006, from http://www.nessus.org/.

Pearl, J. (1997). Graphical models for probabilistic and causal reasoning. In *The Computer Science and Engineering Handbook*, pages 697–714.

Qiang, L., Khong, T. C., San, W. Y., Jianguo, W., and Choy, C. (2001). A web-based material requirements planning integrated application. In *EDOC '01: Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*, page 14, Washington, DC, USA. IEEE Computer Society.

Shahrokhi, M. and Bernard, A. (2004). Risk assessment/prevention in industrial design processes. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, pages 2592–2598.

Stoneburner, G., Goguen, A., and Feringa, A. (2002). Risk management guide for information technology systems. Technical Report 800-30, Computer Security Division, Information Technology Laboratory, NIST.

Verdon, D. and McGraw, G. (2004). Risk analysis in software design. *IEEE Security and Privacy*, 2(4):79–84.

Walt, C. v. d. (2002). Assessing internet security risk, part four: Custom web applications. Retrieved from http://www.securitydocs.com/library/729.

Yacoub, S. M., Cukic, B., and Ammar, H. H. (1999). Scenario-based reliability analysis of component-based software. In *ISSRE '99: Proceedings of the 10th International Symposium on Software Reliability Engineering*, page 22, Washington, DC, USA. IEEE Computer Society.

Zhang, Y., Zhu, H., Greenwood, S., and Huo, Q. (2001). Quality modelling for web-based information systems. In *FTDCS '01: Proceedings of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems*, page 41, Washington, DC, USA. IEEE Computer Society.