Ontology-based Security Assessment for Software Products

Ju An Wang, Minzhe Guo, Hao Wang, Min Xia, & Linfeng Zhou
Southern Polytechnic State University
1100 South Marietta Parkway
Marietta, GA 30060
(01) 678-915-3718
jwang@spsu.edu

ABSTRACT
This paper proposes an ontology-based approach to analyzing and assessing the security posture for software products. It provides measurements of trust for a software product based on its security requirements and evidence of assurance, which are retrieved from an ontology built for vulnerability management. Our approach differentiates with the previous work in the following aspects: (1) It is a holistic approach emphasizing that the system assurance cannot be determined or explained by its component assurance alone. Instead, the software system as a whole determines its assurance level. (2) Our approach is based on widely accepted standards such as CVSS, CVE, CWE, CPE, and CAPEC. Our ontology integrated these standards seamlessly thus provides a solid foundation for security assessment. (3) Automated tools have been built to support our approach, delivering the environmental scores for software products.

Categories and Subject Descriptors
C.2.0 [Computer-Communication Networks]: General [Security and protection]; K.6.5 [Management of Computing and Information Systems]: Security and Protection

General Terms
Ontology, Security, Vulnerability Analysis and Management

Keywords
Software products, Security metrics, Environmental score, Ontology.

1. INTRODUCTION
A significant amount of research has been done in the area of security content automation and security assessment [1-4, 7, 8, 12-15]. There is a moving trend of applying semantic technologies to information security. Andreas Ekelhart et al. [5] developed an ontology to organize and capture the meaning of structured knowledge about threats, safeguards, assets and their relationship. Gernot Goluch et al. [6] applied semantic technologies in risk assessment and high-level business process management. Recently, a lot of effort has been put on building security automation tools [8]. The Information Security Automation Program (ISAP) is a U.S. government multi-agency initiative to enable automation and standardization of technical security operations. The CVSSWizard [2, 3] is an enhanced automated scoring tool to evaluate the severity of individual vulnerability over CVSS (common vulnerability scoring system) [1]. However, the algorithm provided in [1] could not be applied directly to a software product or a computing system. System assessment and evaluation is a process in which the evidence for assurance is gathered and analyzed against criteria for functionality and assurance. This process can result in a measure of trust that indicates how well a system meets particular criteria [7]. As [8] points out, organizations should measure the security, relative to known security related software flaws and misconfigurations, of all operating units using standard impact scores that can be customized to each particular environment. The resulting measurements should be aggregated by adopting SCAP (security content automation protocol) standard protocols, standard impact scores for vulnerabilities, and customizing those scores to particular environment. The problems an organization may face include complexity, difficulty of analyzing oceans of data, lack of tool support, overwhelmed by segmented methods without an integrated environment.
An ontology is a specification of concepts and their relationship. Ontology represents knowledge in a formal and structured form. Therefore, ontology provides a better tool for communication, reusability, and organization of knowledge. Moreover, ontology not only provides a tool for communication, but also a foundation for high-level reasoning and decision-making. In particular, ontology provides the potential of formal logic inference based on well-defined data and knowledge bases. In the area of vulnerability management, ontology may capture the relationships between collected security data and use the explicit knowledge of vulnerabilities deduce the implicit and inherent knowledge. As a matter of fact, we have built an ontology for vulnerability management, called OVM, which includes facts and rules, concepts, concept taxonomies, relationships, properties, axioms and constraints for security vulnerabilities. OVM can extract information about software products, vulnerabilities, attackers, security metrics, countermeasures, and other relevant information, to help make high-level decisions about vulnerability mitigation. With the OVM, we can describe the pattern of external threats and internal vulnerabilities formally and precisely. Based on the pre-loaded data and rules in the OVM, users can make inference and make high-level decisions in a more effective way. With the seamless integration of common vulnerabilities and their related concepts such as attacks and countermeasures, OVM provides a promising pathway to making ISAP successful.
This paper proposes an ontology-based approach to analyzing and assessing the security posture for software products. It provides measurements of trust for a software product based on its security requirements and evidence of assurance, which are retrieved from an ontology built for vulnerability management.

Our approach differentiates with the previous work in the following aspects: (1) It is a holistic approach emphasizing that the system assurance cannot be determined or explained by its component assurance alone. Instead, the software system as a whole determines its assurance level. (2) Our approach is based on widely accepted standards such as CVSS [1], CVE [4, 12], CWE [14], CPE [13], and CAPEC [12. 13]. Our ontology integrated these standards seamlessly thus provides a solid foundation for security assessment. (3) Automated tools have been built to support our approach, delivering the environmental scores for software products.

The rest of the paper is organized as follows: Section 2 discusses our methodology for evaluating a software product or a software system through all the vulnerabilities contained in the software. Section 3 presents our algorithm and formulas to calculate security metrics for software products. Section 4 gives a few examples, and in Section 5, we draw conclusions and discuss further development.
2. EVALUATING SOFTWARE TRUSTWORTHINESS

The NVD (National Vulnerability Database) [4] provides standardized information regarding existing vulnerabilities for most of the software products available today. The CVSS (Common Vulnerability Scoring System) [1] is a tool to quantify the severity and risk of an individual vulnerability to an information asset in a computing environment. However, a software product usually contains multiple vulnerabilities, and CVSS could not be used directly to assess the security level or trustworthiness of a software product due to its design goal targeting individual vulnerabilities only. On the other hand, any software is running within a computing environment. Therefore, we must calculate the environmental severity scores for vulnerabilities in a software product in order to assess the security posture of the software product. In the following, we present our approach to obtaining software product security level through environmental scores of vulnerabilities contained in the software product. To get the right information for the software product and its vulnerabilities from our vulnerability ontology called OVM (Ontology for Vulnerability Management) developed for vulnerability analysis and management. OVM provides the same vulnerability information as NVD and it also contains the predefined relationship of software products, such as categorizing software products according to their functionalities. The vulnerability ontology and security metrics in an IT environment will help IT administrators to measure the severity of the target software product in their IT environment. Moreover, users can query the OVM to infer the similar products and collect the related vulnerability information for each product. We will calculate the environmental security scores for each product according to the environmental security metrics.

The key question this paper is going to answer is: Given a software product or system, what is the trustworthiness of it? If we have a number of similar products, which one is the best product in terms of security? There are three steps to obtain the desired answer as discussed below.

(i) Data Collection for Software Environmental Metrics
To evaluate a software product in a specific IT environment, we need environmental data that reflect where and how the software is going to run in the environment. For example, we need software name, version, percentage of target software distribution in a computing environment, type of network connection, authentication level, reporting confidence, impact on confidentiality, integrity, and availability if the software is exploited by an attacker, and so on. All of the information could be collected from our vulnerability ontology.

(ii) Ontology Query and Inference
When the necessary information has been prepared by the user, the environmental information will be pulled from the ontology OVM using appropriate queries. Because all the information provided by the user is based on the target working environment, the vulnerability information returned from the ontology corresponds to the vulnerabilities that are most likely to attack the target software in this environment. The most important information is what are the existing vulnerabilities associated with the software product.

(iii) Calculate the Software Security Score
Based on the given environmental information in (1) and the results of all the vulnerability information associated with the software products in the environment from (2), we can obtain a quantitative score for the software based on the CVSS environmental metrics. However, as a software product may contain multiple vulnerabilities, we have to design a formula to do this, which will be discussed in detail in Section 4. When the overall scores for the target software and its similar products come out, the OVM will compare them and give a suggestion indicating the most secure product if it exists.

Figure 1 shows the flow diagram of the assessment process.

3. SOFTWARE SECURITY ASSESSMENT ALGORITHM

Rigorous measurement of software security provides substantial help in the evaluation and improvement of software products. However, little agreement exists about the meaning of software security and how to define software security. As a matter of fact, there may have many vulnerabilities existing in one software product. If we choose to use CVSS scoring algorithm to evaluate the software security, there will have many CVSS scores for those individual vulnerabilities without a unified score for the whole software product. It is difficult to use individual CVSS scores for vulnerabilities within a software product to evaluate the trustworthiness of the software product without a holistic approach combining all those vulnerability scores together. Therefore, we propose a new algorithm capable of generating an overall score based on the CVSS environmental metrics of all the vulnerabilities inside the software.

Given a software product for assessment, all its related vulnerabilities and their CVSS base scores form a vector of metrics. The vector components are the data used to calculate the base scores, and they can be obtained from the ontology OVM to serve as the input to our software security assessment algorithm. The algorithm consists of the following steps:

Step 1: Group the vulnerabilities into cases.

The CVSS base score metrics contains six vectors: Access Vector (AV), Access Complexity (AC), Authentication (AU), Confidentiality Impact (CI), Integrity Impact (II) and Availability Impact (AI). Among them, the values of AV, AC, AU are closely related to the software product’s running environment. The vulnerabilities with the same combination values of AV, AC, AU can be regarded as the vulnerabilities that take effect in the same context, or in the same environmental “case” of the product’s running environment. The vulnerabilities in a software product tend to be more active in some environmental cases, namely, some combination of AV, AC and AU values. For example, with the analysis of the vulnerabilities of various web browser products, we found that the values of vector AV in the majority of them are set to ‘NETWORK’, while the values of AU are set to ‘NONE’, just leaving the value of AC in change. This might indicate that most of vulnerabilities of web browsers take effect when the environmental case is network accessible with no authentication required. Therefore, for most of the web browser products, their vulnerabilities can be grouped into three cases, as the AC can only have three different values. So it is important to differentiate the environment impact of vulnerabilities in different environmental cases.

Suppose the software product is \( p \), and the number of vulnerabilities retrieved from the ontology OVM is \( k \). Then the first step of the algorithm is to group those \( k \) vulnerabilities into \( n \) cases according to the value combinations of AV, AC and AU in the CVSS base score metrics of each vulnerability. Suppose case \( i \) contains \( k_i \) vulnerabilities, then we have the formula (1) as follows:

\[
k = \sum_{i=1}^{n} k_i \tag{1}
\]

Step 2 Compute the environment score for each case with regard to the given product and the given environment

The second step of the algorithm is to calculate the environment score for each case. For case \( i \), there are \( k_i \) vulnerabilities retrieved from the OVM:

EnvironmentalScore\(_i\) = (AdjustedTemporal\(_i\) + 10 * AdjustedTemporal\(_i\))  
* CollateralDamagePotential * TargetDistribution

EnvironmentalScore\(_1\) : The CVSS environmental score for each case of the software product.

AdjustedTemporal\(_i\) = AdjustedBaseScore * AverageTemporalExploitability\(_i\) * AverageRemediationLevel * AverageReportConfidence

AdjustedTemporal\(_1\) : The CVSS AdjustedTemporalScore for each case of the software product.

AverageTemporalExploitability\(_i\) = \( \frac{\sum_{j=1}^{k_i} \text{TemporalExploitability}_j}{k_i} \)

AverageTemporalExploitability : The average of TemporalExploitability for each case of the software product.
The average of Availability Impact for each case of the software product.

\[
\text{AdjustedBaseScore}_i = \frac{\sum_{j=1}^{n} \text{AdjustedBaseScore}_j}{n} 
\]

Confidentiality Requirements, Integrity Requirements, and Availability Requirements need to be input by users. After getting the CVSS Adjusted Base Score and those metrics collected from users, we are able to calculate the CVSS environmental score for each case. Finally, we can use weighted averages to get the final CVSS environmental score for the software product.

4. EXAMPLES

This section provides a case study of using the vulnerability ontology to evaluate a number of software products in terms of their security. We are especially interested in answering the question: For a list of similar software products, which is the most secure one for the given IT environment? Below we describe the process of assessing web browsers using our vulnerability ontology OVM.

Let us start with the target software product Mozilla Firefox 3. The user queries OVM to find out the values for Mozilla Firefox 3 the following data: platform and environment, TD (target distribution), CDP (collateral damage potential), Temporal Metrics, CR, IR, and AR. By querying OVM, the user can find out the similar products such as Internet Explorer, and the latest versions for each similar product, such as Internet Explorer 7. The ontology will discover the related vulnerabilities for the target product and each similar product. Finally, the algorithm discussed in the previous section will be applied to transform the retrieved data into the software security metrics, and then calculate the score for each similar product. Let us first calculate the security metrics for Mozilla Firefox 3.

Mozilla Firefox 3

To analyze the result effectively, let us assume that we have acquired the following data from the user:

- TemporalExploitability=1
- RemediationLevel=1
- ReportConfidence=1
- TargetDistribution=1
- CollateralDamagePotential=0

1. Retrieve from OVM the possible combinations of Access Vector (AV), Access Complexity (AC), and Authentication (Au) values as defined in CVSS for Mozilla Firefox 3.

We find there are three combinations of AV, AC, Au for Mozilla Firefox 3 through the ontology, so there are three cases.

Case 1: AV="Network", AC="Low", Au="None"
Case 2: AV="Network", AC="Medium", Au="None"
Case 3: AV="Network", AC="High", Au="None"

2. Find the vulnerabilities of each case through ontology OVM.

2.1 The vulnerabilities for case 1 and 2 are shown in the following table:

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>AV</th>
<th>AC</th>
<th>Au</th>
<th>CVE ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2008-2785</td>
<td>N</td>
<td>C</td>
<td>N</td>
<td>CVE-2008-2785</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2008-2934</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2008-2934</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2008-3837</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2008-3837</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2008-4065</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2008-4065</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2008-4067</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2008-4067</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2008-4582</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2008-4582</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2008-5508</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2008-5508</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2008-5512</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2008-5512</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2008-5513</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2008-5513</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2009-0253</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2009-0253</td>
<td>Network access to sensitive data</td>
</tr>
<tr>
<td>CVE-2009-0652</td>
<td>C</td>
<td>C</td>
<td>N</td>
<td>CVE-2009-0652</td>
<td>Network access to sensitive data</td>
</tr>
</tbody>
</table>
2. The vulnerabilities for case 3 are shown in the following table:

<table>
<thead>
<tr>
<th>Case</th>
<th>AV</th>
<th>AC</th>
<th>Au</th>
<th>TemporalExploitability</th>
<th>RemediationLevel</th>
<th>TargetDistribution</th>
<th>CollateralDamagePotential</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

3. 3.1. Case 1
We find there are three combinations of AV, AC, Au in Microsoft Internet Complexity (AC), Authentication (Au) related to Microsoft Internet

1. Find the possible combinations of Access Vector (AV), Access
Complexity (AC), Authentication (Au) related to Microsoft Internet

3.2. Case 2

3.3. Case 3

3. Get the average of Confidentiality Impact, Integrity Impact, and
Availability Impact for each case of Mozilla Firefox 3

3.1. Case 1

<table>
<thead>
<tr>
<th>Mozilla Firefox 3</th>
<th>Average of Confidentiality Impact</th>
<th>Average of Integrity Impact</th>
<th>Average of Availability Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.186</td>
<td>0.232</td>
<td>0.186</td>
</tr>
</tbody>
</table>

3.2. Case 2

<table>
<thead>
<tr>
<th>Mozilla Firefox 3</th>
<th>Average of Confidentiality Impact</th>
<th>Average of Integrity Impact</th>
<th>Average of Availability Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2475</td>
<td>0.3175</td>
<td>0.3175</td>
</tr>
</tbody>
</table>

3.3. Case 3

<table>
<thead>
<tr>
<th>Mozilla Firefox 3</th>
<th>Average of Confidentiality Impact</th>
<th>Average of Integrity Impact</th>
<th>Average of Availability Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2475</td>
<td>0.3175</td>
<td>0.3175</td>
</tr>
</tbody>
</table>

4. Get the Adjusted Impact for each case of Mozilla Firefox 3.
The Adjusted Impact for case 1, case 2, and case 3 are, respectively, 7.2454, 5.3924, and 4.5835.

5. Get the Exploitability for each case of Microsoft Internet Explorer 7.
The Exploitability for case 1, case 2, and case 3 are, respectively, 9.9968, 7.2454, and 5.3924.

6. Get the Environmental Score for each case of Microsoft Internet Explorer 7.

6.1. The Environmental Score for case 1 = 8.0573

7. Get the Final Environmental Score for Microsoft Internet Explorer 7.
The Final Environmental Score = 8.0573.

5. CONCLUSION AND DISCUSSION
There are different approaches to define the security metrics for individual vulnerabilities. But few result existed discussing the security metrics for a software product or system. This paper proposes an ontology-based approach to analyzing and assessing the security metrics for software products. The measurement of assurance for a software product is based on its security requirements and evidence of assurance, which are retrieved from an ontology built for vulnerability management. Our approach differentiates with the previous work in that it is a holistic approach emphasizing that the system assurance cannot be determined or explained by its component assurance alone. Instead, the software system as a whole determines its assurance level. Our approach is based on widely accepted standards such as CVSS, CVE, CWE, CPE, and CAPEC. Our ontology integrated these standards seamlessly thus provides a solid foundation for security assessment. We have built some automated tools to support our approach, delivering the environmental scores for software products.

Future work and development in this field include building more user friendly front-end tools for our ontology OVM. For instance, a natural language based user interface to interact with the vulnerability ontology.

6. REFERENCES


