Computer Security Training Recommender for Developers

Muhammad Nadeem, Edward B. Allen, Byron J. Williams
Department of Computer Science and Engineering
Mississippi State University
Mississippi State, MS, USA
{mn338, edward.allen, byron.williams}@msstate.edu

ABSTRACT
Vulnerable code may cause security breaches in software systems resulting in loss of confidential data and financial losses for the organizations. Software developers must be given proper training to write secure code. Conventional training methods do not take the code written by the developers over time into account, which makes these training sessions less effective. We propose a Computer Security Training Recommender to help identify focused and narrow areas in which developers need training. The proposed recommender system leverages the power of static analysis techniques to suggest the most appropriate training topics for different software developers; moreover it utilizes public vulnerability repositories to suggest community accepted solutions to different security problems. This paper presents an architecture of the proposed recommender system and a proof-of-concept case study. We found that vulnerabilities, flagged in source code by static analysis tools, can be mapped to relevant articles in a vulnerability repository. Hence, the mitigation strategies given in such articles may be used as a resource to train individual software developers. Preliminary empirical evaluation shows that the proposed system is promising.

Categories and Subject Descriptors
D.4.6 [**Security and Protection**]: Information flow

General Terms
Security, Experimentation

Keywords
Recommender system, security vulnerabilities, training, CWE

1. INTRODUCTION
The proposed recommender system is based on the static analysis of code written by software developers over time. Static analysis tools e.g., FindBugs report suspected security and other vulnerabilities present in source code. These results may be used as the basis for recommending the most appropriate training topics to the software developers who contributed to writing those modules, hence improving their skill in terms of software security.

We utilize the valuable knowledge in vulnerability repositories such as Common Weakness Enumeration, CWE [2], which is contributed by software security experts across the globe, and is available for public use for free.

RecSys 2014 Poster Proceedings, October 6–10, 2014, Foster City, Silicon Valley, USA.

2. PROPOSED ARCHITECTURE
The proposed architecture of the recommender system, which is an extension to our previous work [1], is shown in Figure 1 below.

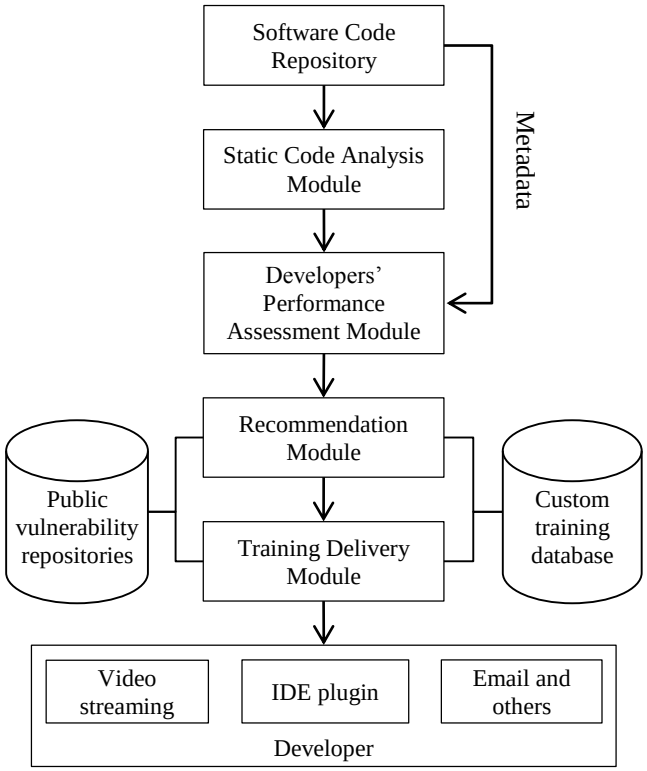

Figure 1. Architecture of the proposed system

The Software Code Repository contains code and metadata which is analyzed by the recommender system.

The Static Code Analysis Module contains static analysis tool(s) which scan the given code repository to find vulnerabilities.

The Developers' Performance Assessment Module creates a profile for each developer containing the description of vulnerabilities induced in the code over time.

A Public vulnerability repository contains information on vulnerabilities, potential mitigations, and consequences etc. Examples are NVD [3] and CWE [2].

A Custom training database refers to a database of custom designed training modules on various topics.

The Recommendation Module calculates the similarity scores between vulnerability descriptions and the articles from vulnerability repositories.

The Training Delivery Module delivers the training modules to the developers.

Table 1. Relevance of CWE articles for flagged security vulnerabilities

Vulnerability types in Security Category	Recommended CWE Articles with the titles		TFIDF Score	Relevance
Cross site scripting vulnerability	CWE 079	Improper Neutralization of Input During Web Page Generation	1.0470	High
	CWE 644	Improper Neutralization of HTTP Headers for Scripting Syntax	0.4749	Medium
Hardcoded constant database password	CWE 522	Insufficiently Protected Credentials	0.7739	High
	CWE 259	Use of Hard-coded Password	0.7359	High
	CWE 256	Plaintext Storage of a Password	0.6858	High
	CWE 640	Weak Password Recovery Mechanism for Forgotten Password	0.6342	Medium
	CWE 798	Use of Hard-coded Credentials	0.5803	High
	CWE 261	Weak Cryptography for Passwords	0.5737	Medium
	CWE 620	Unverified Password Change	0.5231	Low
	CWE 309	Use of Password System for Primary Authentication	0.5142	Low
	CWE 187	Partial Comparison	0.5003	Low
	CWE 260	Password in Configuration File	0.4536	Medium
	CWE 258	Empty Password in Configuration File	0.3816	Medium
	CWE 263	Password Aging with Long Expiration	0.3794	Low
HTTP Response splitting vulnerability	CWE 113	Improper Neutralization of CRLF Sequences in HTTP Headers	1.5627	High
	CWE 650	Trusting HTTP Permission Methods on the Server Side	0.6569	High
	CWE 644	Improper Neutralization of HTTP Headers for Scripting Syntax	0.5360	High
Non-constant SQL string passed to execute method	CWE 089	Improper Neutralization of Special Elements used in SQL stmt.	0.4768	High
	CWE 484	Omitted Break Statement in Switch	0.3839	Low

3. EVALUATION

We conducted a proof of concept case study to evaluate the feasibility of mapping vulnerabilities to articles in CWE repository.

Table 2. Summary of static code analysis

Category	Flagged vulnerability types	Total Occurrences
Bad practice	25	195
Correctness	19	64
Dodgy code	20	226
Experimental	2	31
Internationalization	1	105
Malicious code vulnerability	8	323
Multithreaded correctness	1	1
Performance	15	220
Security	4	14
Total	95	1179

An open source system, Tolven 2.0, having 418K lines of code in 2957 Java modules, was the target source code. It has been used in other studies [5] as well. For static code analysis, we used the open source tool FindBugs 2.0.3. Every category listed in Table 2 has a number of flagged vulnerability types, e.g., there are 4 types of security vulnerabilities with 14 occurrences in total. A short description of each type in security category is given in Table 3.

Table 3. Flagged Security vulnerability types

Vulnerability types in Security Category	Occurrences
Cross site scripting vulnerability	11
Hardcoded constant database password	1
HTTP Response splitting vulnerability	1
Non constant SQL string passed to execute method	1
Total	14

We used Vector Space Model (VSM) with term frequency-inverse document frequency (TFIDF) [4] weights to calculate the similarity between the vulnerability descriptions and CWE articles. Out of 95 flagged vulnerability types listed in Table 2, 81 were successfully mapped to the CWE articles. Due to space limitation, we only list the CWE articles mapped for each vulnerability in "security" category in Table 1. TFIDF score and manually calculated relevance by subject expert are also shown for each CWE article.

4. CONCLUSION AND FUTURE WORK

The proof of concept case study demonstrated the practical feasibility of automatically finding relevant security articles based on source code vulnerabilities. Though, the current implementation uses only the CWE articles, however other vulnerability databases e.g., National Vulnerability Database, NVD, host useful data related to security checklists, security related software flaws, misconfigurations, and impact metrics. By utilizing the NVD database along with CWE articles, the scope of our recommender system may expand. Another short term goal is to use more than one static code analysis tools so that false positive detections may be reduced by comparing the outputs of multiple tools.

5. REFERENCES

[1] Muneer, S., Nadeem, M., and Allen, E. B., 2014. Recommending Training Topics for Developers Based on Static Analysis of Security Vulnerabilities, In *Proceedings of the 52nd ACM Southeast Conference*, (Kennesaw, GA, Mar. 28-29, 2014). ACMSE'14. ACM New York, NY.

[2] CWE, Common Weakness Enumeration, A Community-Developed Dictionary of Software Weakness Types, http://cwe.mitre.org, accessed Apr. 30, 2014.

[3] NVD, National Vulnerability Database Version 2.2, https://nvd.nist.gov, accessed Apr 25, 2014.

[4] Binkley, D., & Lawrie, D. 2010. Information retrieval applications in software development. *Encyclopedia of Software Engineering*, 231-242.

[5] Austin, A. and Williams, L. 2011. One Technique is Not Enough: A Comparison of Vulnerability Discovery Techniques, In *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement* (Banff, Canada, September 22-23, 2011). ESEM'11. CPS, Los Alamitos, CA, 97-106.