ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining

Diagne Papa Ousmane Thiaw
Research Scholar,
Dongua University, Shanghai, China.

Jianguo Zheng
Professor,
Dongua University, Shanghai, China.

ABSTRACT:
Although a large research effort has been going on for more than a decade, the security of web applications continues to be a challenging problem. An important part of that problem derives from vulnerable source code, often written in unsafe languages like PHP. Source code static analysis tools are a solution to find vulnerabilities, but they tend to generate false positives and require considerable effort for programmers to manually fix the code. We explore the use of a combination of methods to discover vulnerabilities in source code with less false positives. We combine taint analysis, which finds candidate vulnerabilities, with data mining, in order to predict the existence of false positives. This approach brings together two approaches that are apparently orthogonal: humans coding the knowledge about vulnerabilities (for taint analysis) versus automatically obtaining that knowledge (with machine learning, for data mining). Given this enhanced form of detection, we propose doing automatic code correction by inserting fixes in the source code. Our approach was implemented in the WAP tool and an experimental evaluation was performed with a large set of PHP applications. Our tool found 388 vulnerabilities in 1.4 million lines of code. Its accuracy and precision were approximately 5% better than PhpMinerII's and 45% better than Pixy's.

INTRODUCTION:
Since its appearance in the early 1990s, the Web evolved from a platform to access text and other media to a framework for running complex web applications. These applications appear in many forms, from small home-made to large-scale commercial services (e.g., Google Docs, Twitter, Facebook). However, web applications have been plagued with security problems. For example, a recent report indicates an increase of web attacks of around 33% in 2012 [34]. Arguably, a reason for the insecurity of web applications is that many programmers lack appropriate knowledge about secure coding, so they leave applications with flaws. However, the mechanisms for web application security fall in two extremes.

On one hand, there are techniques that put the programmer aside, e.g., web application firewalls and other runtime protections, On the other hand, there are techniques that discover vulnerabilities but put the burden of removing them on the programmer, e.g., black-box testing and static analysis. The paper explores an approach for automatically protecting web applications while keeping the programmer in the loop. The approach consists in analyzing the web application source code searching for input validation vulnerabilities and inserting fixes in the same code to correct these flaws. The programmer is kept in the loop by being allowed to understand where the vulnerabilities were found and how they were corrected. This contributes directly for the security of web applications by removing vulnerabilities, and indirectly by letting the programmers learn from their mistakes. This last aspect is enabled by inserting fixes that follow common security coding practices, so programmers can learn these practices by seeing the vulnerabilities and how they were removed.

We explore the use of a novel combination of methods to detect this type of vulnerabilities: static analysis and data mining. Static analysis is an effective mechanisms to find vulnerabilities in source code, but tends to report many false positives (non-vulnerabilities) due to its undecidability. This problem is particularly difficult with languages such as PHP that are weakly typed and not formally specified. Therefore, we complement a form of static analysis, taint analysis, with the use of data mining to predict the existence of false positives. This solution combines two apparently opposite approaches: humans coding the knowledge about vulnerabilities (for taint analysis) versus automatically obtaining that knowledge (with supervised machine learning supporting datamining). To predict the existence of false positives we introduce the novel idea of assessing if the vulnerabilities detected are false positives using data mining. To do this assessment, we measure attributes of the code that we observed to be associated with the presence of false positives, and use a combination of the three top-ranking classifiers to flag every vulnerability as false positive or not.

We explore the use of several classifiers: ID3, C4.5/J48, Random Forest, Random Tree, K-NN, Naïve Bayes, Bayes Net, MLP, SVM, and Logistic Regression. Moreover, for every vulnerability classified as false positive, we use an induction rule classifier to show which attributes are associated with it. We explore the JRip, PART, Prism and Ridor induction rule classifiers for this goal. Classifiers are automatically configured using machine learning based on labeled vulnerability data.Ensuring that the code correction is done correctly requires assessing that the vulnerabilities are removed and that the correct behavior of the application is not modified by the fixes.

We propose using program mutation and regression testing to confirm, respectively, that the fixes do the function to what they are programmed to (blocking malicious inputs) and that the application remains working as expected (with benign inputs). Notice that we do not claim that our approach is able to correct any vulnerability, or to detect it, only the input validation vulnerabilities it is programmed to deal with. The paper also describes the design of the Web Application Protection (WAP) tool that implements our approach. WAP analyzes and removes input validation vulnerabilities from code1 written in PHP 5, which according to a recent reportis used by more than 77% of the web applications. WAP covers a considerable number of classes of vulnerabilities: SQL injection (SQLI), cross-site scripting (XSS), remote file inclusion, local file inclusion, directory traversal/path traversal,source code disclosure, PHP code injection, and OS command injection.

EXISTING SYSTEM:

» In the existing system, the system begins by giving a survey of web application attacks and vulnerabilities, also approaches to improve the web application security using intrusion detection systems and scanners based on machine learning and artificial intelligence.

» When it comes to vulnerability, it is also an attack which exploits this vulnerability; therefore the existing system presents web intrusion detection system based on detection of web vulnerabilities. Experimental results have been acquired from HTTP simulations in our network and from responses of HTTP requests sent to a bunch of websites and applications to test the efficiency of our intrusion detection system. This efficiency can be noticed from a High detection rate which is greater than 90%.

PROPOSED SYSTEM:

* In the proposed system, the system explores the use of a novel combination of methods to detect this type of vulnerabilities: static analysis and data mining. Static analysis is an effective mechanism to find vulnerabilities in source code, but tends to report many false positives (non-vulnerabilities) due to its undesirability. This problem is particularly difficult with languages such as PHP that are weakly typed and not formally specified.

* Therefore, the system complements a form of static analysis, taint analysis, with the use of data mining to predict the existence of false positives. This solution combines two apparently opposite approaches: humans coding the knowledge about vulnerabilities (for taint analysis) versus automatically obtaining that knowledge (with supervised machine learning supporting data mining).

IMPLEMENTATION:

• Admin

In this module, admin has to login with valid username and password. After login successful he can do some operations such as view all user, their details and authorize them , view all owners, their details and authorize them, view all attackers details(like ip address and host name), view all sql injection vulnerabilities and block them(those who are execute wrong query), view all file access vulnerabilities of users(those who are used wrong secret key) , view all blocked data owners , view unblock requests and unblock them, view all secret key requests and generate, view all users download history, view SQL Injection Vulnerabilities in chart, View number of remote vulnerabilities in chart.

• User

In this module, there are n numbers of users are present. User should register before doing some operations. After registration successful he can login by using valid user name and password. Login successful he will do some operations like view profile details, request secret key and view response, search documents and download by entering secret key.

• Data Owner

In this module, there are n numbers of owners are present. Owner should register before doing some operations. After registration successful he can login by using valid user name and password. Login successful he will do some operations like view profile details, Upload

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

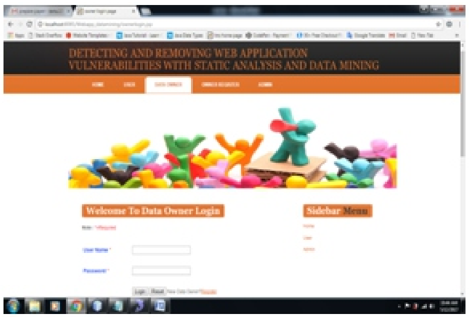
documents and generate digital sign, view uploaded documents, verify his documents and recover if it is attacked, view all on his documents like(downloads and attacked details), Execute SQL queries if query is incomplete the it is SQL Injection Vulnerabilities.

- **Attacker**

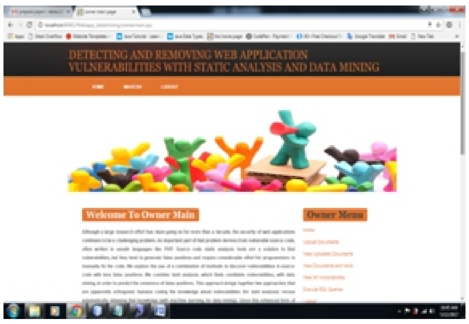
Attacker searches the documents and edit document by changing content.

SCREEN SHOTS:

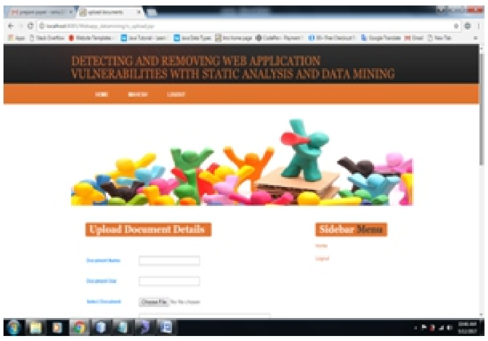
Data Owner Login:


DETECTING AND REMOVING WEB APPLICATION VULNERABILITIES WITH STATIC ANALYSIS AND DATA MINING
Welcome To Data Owner Login
Sidebar Menu

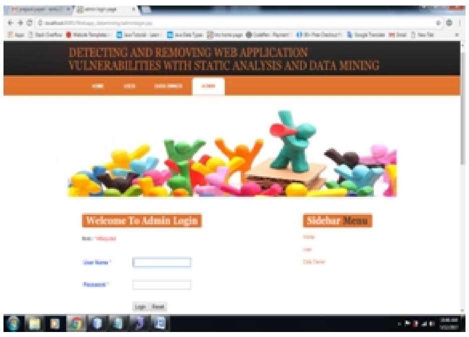
Data Owner Home:


DETECTING AND REMOVING WEB APPLICATION VULNERABILITIES WITH STATIC ANALYSIS AND DATA MINING
Welcome To Owner Main
Owner Menu

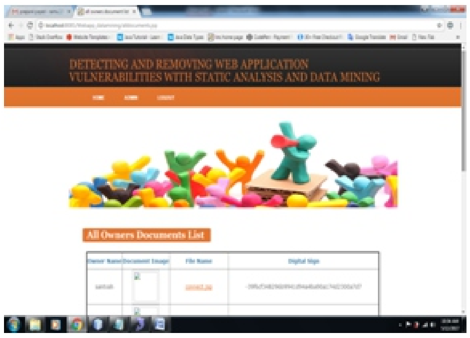
Upload Document:


DETECTING AND REMOVING WEB APPLICATION VULNERABILITIES WITH STATIC ANALYSIS AND DATA MINING
Upload Document Details
Sidebar Menu
Choose File

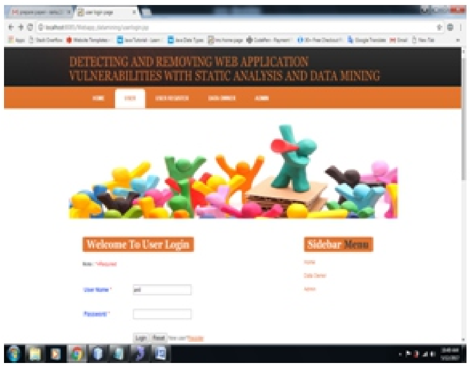
Admin Login:


DETECTING AND REMOVING WEB APPLICATION VULNERABILITIES WITH STATIC ANALYSIS AND DATA MINING
Welcome To Admin Login
Sidebar Menu
User Name
Password
Login
Reset

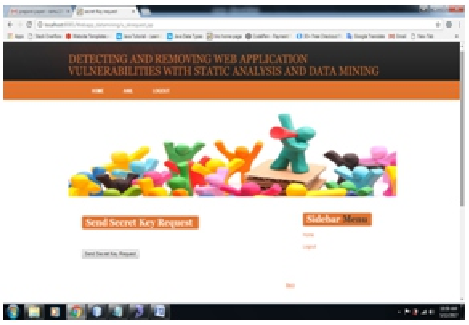
View Documents:


DETECTING AND REMOVING WEB APPLICATION VULNERABILITIES WITH STATIC ANALYSIS AND DATA MINING
All Owners Documents List
Owner Name
Document Image
File Name
Digital Sign

User Login:


DETECTING AND REMOVING WEB APPLICATION VULNERABILITIES WITH STATIC ANALYSIS AND DATA MINING
Welcome To User Login
Sidebar Menu
User Name
Password
Login
Reset

Send Secret Key Request:


DETECTING AND REMOVING WEB APPLICATION VULNERABILITIES WITH STATIC ANALYSIS AND DATA MINING
Send Secret Key Request
Sidebar Menu
Send Secret Key Request

CONCLUSION:

The paper presents an approach for finding and correcting vulnerabilities in web applications and a tool that implements the approach for PHP programs and input validation vulnerabilities. The approach and the tool search for vulnerabilities using a combination of two techniques: static source code analysis and data mining. Data mining is used to identify false positives using a top three of machine learning classifiers and to justify their presence using an induction rule classifier. All classifiers were selected after a thorough comparison of several alternatives. It is important to note that this combination of detection techniques cannot provide entirely correct results. The static analysis problem is not decidable and the resort to data mining cannot circumvent this undecidability, only provide probabilistic results. The tool corrects the code by inserting fixes, i.e., sanitization and validation functions. Testing is used to verify if the fixes actually remove the vulnerabilities and do not compromise the (correct) behavior of the applications. The tool was experimented with synthetic code with vulnerabilities inserted on purpose and with a considerable number of open source PHP applications. It was also compared with two source code analysis tools, Pixy and PhpMinerII. This evaluation suggests that the tool can detect and correct the vulnerabilities of the classes it is programmed to handle. It was able to find 388 vulnerabilities in 1.4 million lines of code. Its accuracy and precision were approximately 5% better than PhpMinerII's and 45% better than Pixy's.

REFERENCES:

[1] WAP tool website. http://awap.sourceforge.net/.

[2] J. Antunes, N. F. Neves, M. Correia, P. Verissimo, and R. Neves.Vulnerability removal with attack injection. IEEE Transactions on Software Engineering, 36(3):357–370, 2010.

[3] E. Arisholm, L. C. Briand, and E. B. Johannessen. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. Journal of Systems and Software, 83(1):2–17, 2010.

[4] R. Banabic and G. Candea. Fast black-box testing of system recovery code. In Proceedings of the 7th ACM European Conference on Computer Systems, pages 281–294, 2012.

[5] L. C. Briand, J. Wüst, J. W. Daly, and D. Victor Porter. Exploring the relationships between design measures and software quality in objectoriented systems. Journal of Systems and Software, 51(3):245–273, 2000.

[6] G. T. Buehrer, B. W. Weide, and P. Sivilotti. Using parse tree validation to prevent SQL injection attacks. In Proceedings of the 5th International Workshop on Software Engineering and Middleware, pages 106–113, Sept. 2005.

[7] N. L. de Poel. Automated security review of PHP web applications with static code analysis. Master's thesis, State University of Groningen, May 2010.

[8] R. A. DeMillo, R. J. Lipton, and F. G. Sayward. Hints on test data selection: Help for the practicing programmer. Computer, 11(4):34–41, Apr 1978.

[9] J. Demšar. Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research, 7:1–30, Dec 2006.

[10] D. Evans and D. Larochelle. Improving security using extensible lightweight static analysis. IEEE Software, pages 42–51, Jan/Feb 2002.

[11] W. Halfond and A. Orso. AMNESIA: analysis and monitoring for neutralizing SQL-injection attacks. In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, pages 174–183, Nov. 2005.

[12] W. Halfond, A. Orso, and P. Manolios. WASP: protecting web applications using positive tainting and syntax-aware evaluation. IEEE Transactions on Software Engineering, 34(1):65–81, 2008.

[13] J. C. Huang. Software Error Detection through Testing and Analysis.John Wiley and Sons, Inc., 2009.

[14] Y.-W. Huang, S.-K.Huang, T.-P.Lin, and C.-H. Tsai. Web application security assessment by fault injection and behavior monitoring. In Proceedings of the 12th International Conference on World Wide Web, pages 148–159, 2003.

[15] Y.-W. Huang, F. Yu, C. Hang, C.-H.Tsai, D.-T.Lee, and S.-Y. Kuo.Securing web application code by static analysis and runtime protection. In Proceedings of the 13th International World Wide Web Conference, pages 40–52, 2004.