Industrial Engineering Journal
ISSN: 0970-2555
Volume : 50, Issue 7, No. 1, July 2021

WEB APPLICATION RECOGNITION AND STATISTICAL ANALYSIS TO REDUCE VULNERABILITIES IN DATA MINING

Modem. Jeevan Kumar, Dept of Computer Science and Engineering, Sree Venkateswara College Of Engineering, Nellore (Dt), Andhra Pradesh, India.
k. Srikanth Reddy, Dept of Computer Science and Engineering, Sree Venkateswara College Of Engineering, Nellore (Dt), Andhra Pradesh, India.
P. Mohan, Dept of Computer Science and Engineering, Sree Venkateswara College Of Engineering, Nellore (Dt), Andhra Pradesh, India.
P.Sravan Kumar Reddy, [4]Dept of Electronics and Communication Engineering, Sree Venkateswara College Of Engineering, Nellore (Dt), Andhra Pradesh, India.

ABSTRACT
Despite extensive research on the subject for more than ten years, web application security is still a problem. Vulnerable source code, which is frequently written in dangerous languages like PHP, is a major contributor to this issue. While static analysis techniques can help find vulnerabilities in the original code, they frequently produce false positives and demand a lot of manual work from programmers to fix the code. We investigate how to employ several techniques to find source code errors while generating fewer false positives. To anticipate the likelihood of false positives, we integrate data mining with taint analysis, which identifies potential vulnerabilities. Humans have programmed their understanding of vulnerabilities (for taint analysis), and they have combined that knowledge with what appears to be an orthogonal method of mechanically gathering that information (using machine learning, for data mining). This method combines these two ostensibly irreconcilable tactics. In light of this better way of detection, we advise implementing fixes in the source code to execute automatic code rectification. We completed an experimental evaluation with a substantial number of PHP applications and included our strategy into the WAP tool. In 1.4 million lines of code, 388 vulnerabilities were discovered by our tool. It performed about 5.2% and 46% better than accuracy and precision.
Keywords: CNN, RCNN, SSD, dataset, weapon detection.

1. INTRODUCTION
Agriculture is India's main source of welfare. Rain is essential to the success of agriculture. Additionally, it enhances water resources. Previous rainfall data allows farmers to better manage their crops, which benefits the economy of the country. Precipitation forecasting aids in flood prevention, protecting lives and property. Due to fluctuations in timing and amount of precipitation, meteorologists find it difficult to anticipate rainfall. One of the most challenging difficulties for academics from a variety

of fields, such as meteorological data mining, environmental machine learning, functional hydrology, and numerical forecasting, is developing a prediction model for precise rainfall. A common question in these challenges is how to leverage future predictions and infer past projections.. The main process in rainfall is often made up of a number of smaller processes. Precipitation forecasts made by its global system are not always accurate. Climate forecasting stands out among all the benefits and services provided by the meteorological department for all countries on the earth. The work is particularly challenging since precise figures must be used and all signals must be indicated without confidence. The need for precision precipitation forecasting has long been acknowledged by hydrological science since early warning of impending natural disasters can help prevent harm and loss of life. The integration of several models and the modular model theory have recently drawn more attention as solutions to this problem. There are several different techniques for predicting rainfall in India. In India, there are essentially two methods for predicting rainfall. Regression, Artificial Neural Networks (ANN), decision tree algorithms, fuzzy logic, and group data processing methods are the most widely used computing approaches for weather forecasting. Following informational norms and linkages while acquiring illusory and sometimes expensive knowledge is the main objective. Artificial neural networks are one potential area within this large topic. The scientific community, corporations, governments, and risk management organisations have all raised concern about how challenging it is to predict when it will rain. The amount of rainfall is a climatic factor that affects a number of human endeavours, such as forestry, power production, tourism, and agricultural output [1]. Because it is most frequently associated with unfavourable natural occurrences like landslides, flooding, mass movements, and avalanches, rainfall prediction is essential in this regard. These events have had a long-lasting effect on society [2]. By employing a suitable method for rainfall prediction, it is therefore possible to implement preventative and mitigation measures for these natural phenomena.

To remove this ambiguity, we developed precise forecasts using a variety of machine learning models and methods. A full overview of the machine learning life cycle, from data preprocessing to model deployment and evaluation, is the goal of these papers. The feature transformation, categorical feature encoding, feature scaling, and feature selection procedures are part of the data preprocessing process. To serve as an evaluation, models like Logistic Regression, Decision Trees, K Nearest Neighbour, Rule-based, and Ensembles were used.

2. LITERATURE SURVEY

W. Halfond, A. Orso,andP. Manolios, Today, many software programmes include a web-based component that allows users to access them online and exposes them to

various web-based attacks.SQL injection is one of these risks, and it has become increasingly common and harmful. It can grant attackers full access to the databases that support Web applications.In comparison to the majority of existing methodologies, the novel highly automated methodology presented in this study has both conceptual and practical advantages. The novel idea of positive tainting and the concept of syntax-aware evaluation serve as the method's conceptual pillars. In terms of practise, our method is precise and efficient, needs little work for deployment, and often only has a little performance overhead. We put our methods into practise with the Web application SQL-injection preventer (WASP) tool, which we utilised to conduct an empirical evaluation on a variety of Web applications that we put through a wide range of attacks and authorised accesses. All of the otherwise successful attacks could be stopped by WASP, and no false positives were produced**.**

C. V. Berghe and Pietraszek Application-level security is seriously threatened by injection vulnerabilities. SQL injection, cross-site scripting, and shell injection vulnerabilities are a few of the most prevalent varieties. Existing defences against injection attacks, or assaults that take advantage of these flaws, rely mainly on the application developers and are consequently prone to error.

X. Wang, C. Pan, P. Liu, and S.Zhu We recommend SigFree, an application-layer approach that blocks code-injection buffer overflow attack messages targeted at various Internet services, such as Web services, online and without the need for a signature. SigFree prevents attacks by spotting the presence of code since it was developed in response to the observation that normal client requests almost never include executables whereas buffer overflow assaults frequently do. SigFree uses a new data-flow analysis method called code abstraction that is general, quick, and challenging for exploit code to dodge in contrast to the previous code detection algorithms.

J.Antunes,N.F.Neves,M.Correia,P. Verissimo Higher degrees of dependability are required when networked computer systems are relied upon more frequently. This is even more important because systems' security is continually being jeopardised by emerging threats and attack vectors. This work offers an attack injection mechanism for the automatic detection of vulnerabilities in software components in order to address this issue. The suggested methodology, used in AJECT, adopts a strategy used by security researchers and hackers to identify holes in network-connected systems. AJECT automatically generates a huge number of assaults using established test case generating techniques and a definition of the server's communication protocol.. Then, as it launches these attacks over the network, it keeps an eye on the target system's server's operation and the client responses. When an unusual behaviour is seen, it is possible that a vulnerability existed and was exploited by a specific attack (or set of assaults). The attack can then be used to duplicate the anomaly and aid in fixing the mistake. Several attack injection campaigns were run with 16 publicly accessible POP and IMAP servers to

evaluate the efficacy of this strategy. The findings demonstrate that AJECT might be successfully leveraged to find vulnerabilities, even on well-known servers evaluated over time.

3. EXISTING SYSTEM

There is a substantial body of relevant literature, thus in order to save space, we only briefly examine the key subjects by highlighting a few exemplary works. Static analysis software automates the auditing of source, binary, and intermediate code. The unspoiled qualifier identifies if a function or parameter either gives reliable data (such as a sanitization function) or whether a function parameter needs reliable data (such as mysql_query) in order to annotate source code. These qualifiers are used by taint analysis programmes like CQUAL and Splint (both for C code). It is referred to as being contaminated when a function or parameter produces unreliable results (this includes functions that read user input, for example).

4. PROPOSED SYSTEM

This study looks into a strategy for automatically securing web applications with the programmer involved. The process involves looking for input validation flaws in the source code of the web application, then introducing fixes to fix these issues in the same code. By letting the programmer know where the vulnerabilities were found and how they were fixed, the programmer is kept up to date. By eliminating vulnerabilities, this method directly increases the security of online applications, and it indirectly increases security by allowing programmers to learn from their mistakes. This final feature is made possible by including solutions that adhere to best practises for security coding, allowing programmers to learn these techniques by observing the vulnerabilities and how they were fixed. We investigate the application of a unique approach combination—combining static analysis with data mining—to find this kind of vulnerability. Static analysis is a useful tool for identifying vulnerabilities in source code, but because of its unfavourable nature, it frequently reports false positives (vulnerabilities that don't exist). The taint analyzer is a static analysis tool that works with an AST made for PHP 5 in our example by a lexer and a parser. All symbols (variables, functions) are unaltered at the start of the analysis unless they constitute an entry point. Each cell in the tainted symbol table (TST) created by the tree walkers contains a programme statement from which we want to gather information. Each cell has an AST subtree as well as some data. For example, the TST cell in the sentence $x = $b + $c comprises the subtree of the AST that symbolises the dependence of $x on $b and $c.

Predicting False Positives

It is well known that Turing's halting problem and the static analysis problem are

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 50, Issue 7, No. 1, July 2021

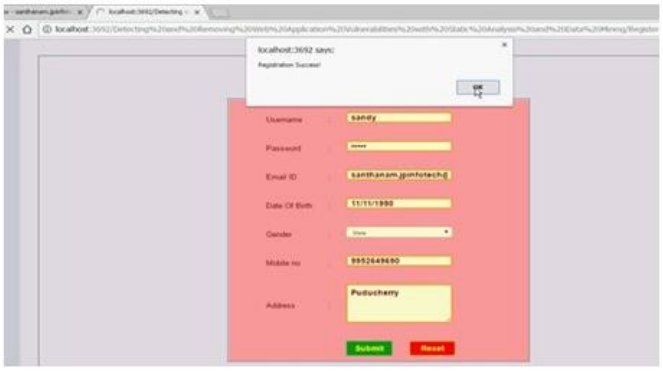
inseparable for non-trivial languages. In fact, the solution to this issue is as simple as doing a partial analysis of specific language structures, making static analysis tools ineffective. This problem might occur, for instance, while using our approach to manipulate strings. It is unclear what should be done with the state of a malformed string when it is put through operations, like those that return a substring or concatenate it with another string. If either operation will untaint the string, we cannot be certain. We decided to taint the string because there could be erroneous positives but no false negatives.

Code Correction

Our solution involves automatically conducting code correction after the taint analyzer and the data mining component have found the vulnerabilities. Information on the vulnerability is provided by the taint analyzer, including its class (for instance, SQLI) and the impacted code segment. A repair is a call to a function that cleans up or verifies data before it is sent to a sensitive sink. Sanitization entails altering the data to remove any potentially harmful Meta characters or metadata. Validation entails examining the data and deciding whether to run the sensitive sink or not in light of the verification.

5. RESULTS

Our patches were created to prevent changing the programmes' intended (proper) behaviour. We have not yet observed any instances in which a WAP-fixed application started to operate wrongly or in which the repairs themselves performed poorly. To boost the certainty of this observation, however, we suggest applying software testing approaches to a programme to see, for example, if the programme has mistakes in general or if changes to the programme have produced errors. By examining if these test scenarios result in unexpected or wrong behaviour or outputs, this verification is carried out. For each of these two verifications, we employ one of the following two software testing methodologies like retesting and program mutation.



Industrial Engineering Journal
ISSN: 0970-2555
Volume : 50, Issue 7, No. 1, July 2021

Fig.1.Login page

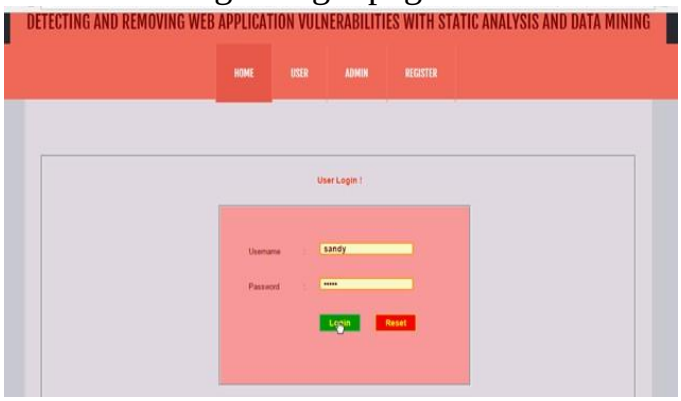


Fig.2.Home page

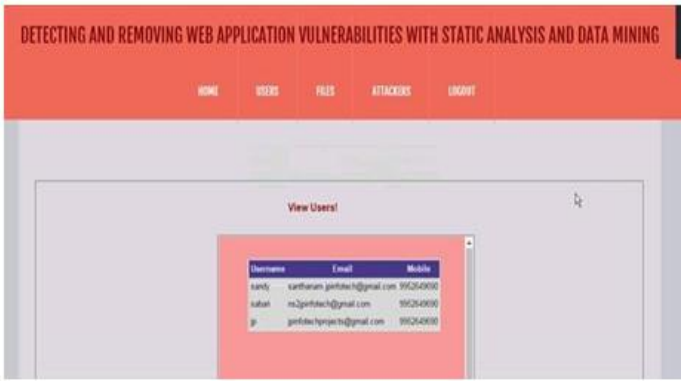


Fig.3.User page

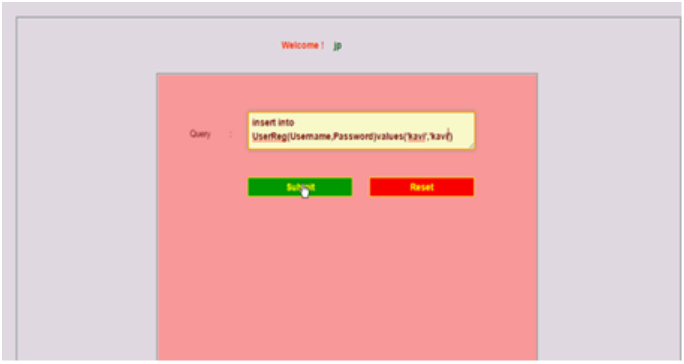


Fig.4.Login code page

CONCLUSION

This paper describes a technique for identifying and fixing faults in online applications, as well as a tool that applies the technique to fix input validation bugs and PHP programming problems. Static source code analysis and data mining are used by both the technique and the software to look for vulnerabilities. Once the top 3 machine

learning classifiers have identified false positives, an induction rule classifier is used to confirm their existence. All classifiers were chosen after carefully examining all of the potential outcomes. It's crucial to remember that this combination of detection methods doesn't always produce trustworthy results.Data mining can only generate results with a high degree of probability when used to address the static analysis problem because the problem cannot be separated. By adding fixes, such as sanitization and validation functions, the tool fixes the code. Testing is done to ensure that the fixes work as intended and do not affect the applications' (proper) behaviour. Both synthetic code and open source PHP programmes with intended flaws were used to test the tool on a huge number of different occasions. It was also compared to two source code analysis programmes, Pixy and PhpMiner II. The programme is able to find and patch vulnerabilities in the classes it is intended to handle, according to this review. It was able to identify 388 bugs in 1.4 million lines of code. It outperformed PhpMinerII and Pixy in terms of accuracy and precision by roughly 5% and 45%, respectively.

REFERENCES

1. Symantec,Internetthreatreport.2012trends,vol. 18, Apr. 2013.
2. W.Halfond,A.Orso,andP.Manolios,“WASP:protectingwebapplicationsusingpositive tainting and syntax aware evaluation,”IEEE Trans. Softw. Eng., vol. 34, no. 1, pp. 65–81, 2008.
3. T. Pietraszek and C. V. Berghe, “Defendingagainstinjectionattacksthroughcontext-sensitive string evaluation,” in Proc. 8th Int.Conf. Recent Advances in Intrusion Detection,2005, pp. 124–145.
4. X.Wang,C.Pan,P.Liu,andS.Zhu,“SigFree:Asignature-freebufferoverflowattack blocker,” in Proc. 15th USENIX SecuritySymp.,Aug. 2006, pp. 225–240.
5. J.Antunes,N.F.Neves,M.Correia,P.Verissimo,andR.Neves,“Vulnerabilityremoval withattackinjection,”IEEETrans.Softw. Eng., vol. 36, no.3, pp. 357–370, 2010.
6. R. Banabic and G. Candea, “Fast black-boxtesting of system recovery code,” in Proc. 7thACM Eur. Conf. Computer Systems, 2012, pp.281–294.
7. Y.-W.Huangetal.,“Webapplicationsecurityassessmentbyfaultinjectionandbehavior monitoring,” in Proc. 12th Int. Conf.WorldWide Web, 2003,pp. 148–159.
8. Y.-W.Huangetal.,“Securingwebapplication code by static analysis and runtimeprotection,”inProc.13thInt.Conf.WorldWideWeb, 2004, pp. 40–52.
9. N. Jovanovic, C.Kruegel,andE.Kirda,“Precisealiasanalysisforstaticdetectionof webapplicationvulnerabilities,”inProc.2006WorkshopProgrammingLanguagesand Analysisfor Security,Jun. 2006,pp. 27–36.
10. U. Shankar, K. Talwar, J. S. Foster, and D.Wagner,“Detectingformatstringvulnerabilities with type qualifiers,” inProc.10th USENIX Security Symp., Aug. 2001, vol.10,pp. 16–16.