DETECTION AND CLASSIFICATION OF INPUT VALIDATION ATTACKS USING MACHINE LEARNING - MIXED-METHOD ANALYTICS

*P. Thangamariappan** [1], *S. Mithuna* [2]

ABSTRACT

Because of the widespread and extensive deployment of susceptible web applications, application security is still a challenging field. This maintains the vulnerability of easily accessible entry points that can compromise entire apps. The primary cause of this problem is the widespread lack of strong validation procedures on the client and server sides. Input Validation Attacks (IVA)—which include risks such as Cross-Site Scripting (XSS), SQL injection (SQLi), path traversal, and command injection (CMDi)—occur when inputs are not sufficiently sanitized. Security is a top priority in the application area, particularly as web application security is an essential interface for communicating with computer systems and the vast Internet. Given these difficulties, the goal is to improve web application security by implementing hybrid analysis-based static and dynamic deterministic pushdown automata that are enhanced by an intelligent framework. This will allow for the proactive detection of vulnerabilities and the differentiation of normal from abnormal requests. Even while the methods available today are successful in detecting web attacks, there are situations where a combination of approaches is required. As such, the research explores hybrid approaches, enhanced by a machine learning framework. Next, we assess if an unsupervised method for input validation attacks based on a quick mining tool is feasible. The results of our study demonstrate a significant improvement in accuracy when compared to baselines. We highlight the improved accuracy of our technique with a 3% improvement over DEKANT, 6% over WAP, 11% over PhpMinerII, 54% over Pixy, and 21% over RIPS. Moreover, its precision is 20% higher than static taint analysis.

Keywords: Deterministic Pushdown Automata, Input Validation Attacks, multi-classification, ML classifiers.

I. INTRODUCTION

The use of online applications has increased dramatically in the modern digital age. Contemporary technology has been effortlessly incorporated into a wide range of online activities, including social networking, email communication, banking, and shopping. Notably, more than 105 billion applications were downloaded in 2018. Estimates indicate that between 2018 and 2021, downloads would rise by 25%. However, online applications are vulnerable to vulnerabilities due to the ubiquity of bad design, insufficient input/output validation, incorrect implementations, and weak business logic. In this situation, hackers actively look for these weaknesses and make efforts to take advantage of them [13].

Adapting Research Projects and Threat Strategies: Attackers use a variety of tactics to conceal their threats and avoid discovery. However, significant research efforts are devoted to locating and addressing these weaknesses. Web application vulnerabilities are a serious problem because they act as security gaps that allow hackers to access private user data and other application data without authorization [16]. Two major groups of vulnerabilities are highlighted in the current landscape of online application threats: SQL injection (SQLi) and Cross-Site Scripting (XSS).

Reports like the Acunetix Web Application Vulnerability Report 2019 show that SQL Injection and Cross-Site Scripting (XSS) are among the top five security threats. This makes it very clear that fixing these vulnerabilities is important. These vulnerabilities are a result of poor design decisions made during the application development and deployment phases. Two of the top 10 threats connected to web applications, according to the Open Web Application Security Project (OWASP), are SQL injection and cross-site

Department of Computer Science and Engineering[1],
Karpagam Academy of Higher Education Coimbatore, India[1]
thangamariappan.ponnusamy@kahedu.edu.in[1]
Department of Computer Science and Engineering[2],
K S Rangasamy College of Technology, Tiruchengode, Tamilnadu, India[2]
smithuna0399@gmail.com[2]

* Corresponding Author

scripting. Specifically, online application programming languages frequently foster an environment susceptible to input manipulation, which allows malicious code to be directly inserted into SQL queries. Alternative methods are required since developers are still concerned about the lack of system notifications during compiler and runtime validation. Research Scope and Proposed Methodology

New Approaches and Contributions: This work presents a novel methodology that actively detects and inhibits prospective input validation assaults in addition to preemptively identifying input validation vulnerabilities, which contrasts with the prevalent prevention-oriented methodologies. The main idea is based on a hybrid approach that combines dynamic and static Deterministic Pushdown Automata (DPDA) methods. Static analysis is a quick and easy way to examine application files without having to run them; dynamic analysis goes further and examines how these files behave. The detection efficiency is increased while reducing false positives by combining the thoroughness of dynamic analysis, the speed of static analysis, and machine learning approaches.

II. OVERVIEW OF THE ATTACK ON INPUT VALIDATION

Between November 2017 and March 2019, SQLI attacks accounted for 65% of all web attacks, according to US-based cloud service company Akamai. Significant events in April 2019 highlighted how dangerous Cross-Site Scripting (XSS) may be. A WordPress plugin injection event led to the introduction of malicious JavaScript code, which made XSS exploitation easier. Users were consequently taken to strange adverts, dangerous websites, and updates that could have been damaging.

A. SQL Injection

SQL injection is a serious security flaw in online applications that arises when malicious code is introduced with user input to cause the program to generate unexpected SQL statements and unapproved access to data on the back end [9].

Figure 1: Input to sample search form

1.Cross Site Scripting (XSS)

One input validation exploit that allows a hacker to secretly install malicious script in a target user's browser and collect sensitive data is called cross-site scripting [10]

III. RELATED WORK

Web applications are widely used in the modern internet world, and given the volume of research, security risks also occur as a result of the Internet's widespread use. Table 1 lists recent research studies that have been conducted to comprehend the current state of circumstances. A straightforward technique to identify SQLIA was described in [5] Inyong Lee, Soonki Jeong. It involves extracting the values from SQL queries in dynamic analysis and comparing them to predefined values. If they match, it indicates that the request is malicious SQLI; if not, the query is run at the web applications' backend [8]. Table 1 provides an overview of this project.

Table 1: Comparison of related works

Ref.	Year	Techniques Proposed	SQLI	XSS	Merits	Demerits
[5]	2012	Attribute Removal Method	NO	YES	highly useful and simple to use since SQL attribute-based detection	It causes an Integrity issue.
[8]	2013	Classification and Analysis	NO	YES	doesn't need changing the source code	Avoid only SQLI attacks
[7]	2015	Hybrid Analysis with Machine Learning	YES	YES	scalable while detecting online attacks	Reduced precision in the identification of vulnerabilities
[6]	2016	Static Detection	YES	YES	Learn about online assaults quickly.	Web assaults can only be detected using PHP.
[3]	2016	Static Analysis and Data Mining	YES	YES	Web vulnerability detection and automatic code correction	Poor grammar construction
[4]	2016	Static Analysis	YES	YES	Use NLP and static analysis to find web attacks.	needs instruments
[2]	2017	Server Side Code Modification	YES	NO	practical and simple because runtime prevention-based query	Manual assistance is needed for application testing.
[12]	2018	TT-XSS	NO	YES	Help the developer create safe online applications.	Structure of defense against DOM-XSS.
[1]	2019	Django Checker	NO	YES	Effectively predicted were context-sensitive cross-site scripting (XSS) assaults.	Methods that yield data are collected from various instruments, such as scanners
[20]	2020	MERLIN	YES	YES	able to detect security holes in several languages	A configuration file must be included for every programming language.
[21]	2023	Static taint analysis	YES	YES	Better Capability Identification	challenges in precisely assessing the data flow and pinpointing weak points.

Inyong Lee and Soonki Jeong [5] have clarified a simple method for identifying SQL Injection Attacks (SQLIA). The method extracts attribute values from SQL queries and compares the extracted value to predefined values during dynamic analysis. If a match is found, the query is classified as a malicious SQLI request; if not, web backend execution is initiated. utilization [8]. A system that divides users into categories based on ethics and unethical behavior was proposed by Nithya V. and S. Lakshmana Pandean [7]. Through the comparison of static and dynamic studies, this method finds SQLIA assaults. In the method, a dynamically created query at runtime is compared with the query's Deterministic Finite Automata (DFA) structure at compile time. An attacker's malicious code that modifies the static structure is recognized as an abnormal query and prevents the attacker from accessing the application's backend.

A method for anticipating web attacks using a machine learning predictor was presented by Lwin Khin Shar, Lionel C, et al. [7]. Instead of using trained data, this entails using malicious scripted data for both static and dynamic analysis. Both supervised and semi-supervised learning techniques are used in the development of the predictive models [6]. Data flow analysis, which includes sanitization and validation methods to uncover vulnerabilities, including inter- and intra-procedural analysis, was the focus of Johannes Dahse's study. Moreover, PHP injection vulnerabilities are predicted with the extension of data flow analysis programs. To stop web application attacks, Ibéria Medeiros et al. [3] suggested an automated approach that combines data mining and static analysis methods.This method uses static analysis, Natural Language Processing (NLP), and Hidden Markov Models (HMM) to identify web application assaults.

The task of creating an intelligent intrusion detection system (IDS) was met with difficulty by M. R. Gauthama Raman and Kannan K. [24]. They investigated using Artificial Neural Network (ANN) models to improve the performance of IDS. In their study, they proposed a new method for IDS classification called the Helly property of Hypergraph and Arithmetic Residue-based Probabilistic Neural Network (HG AR-PNN). In order to detect potentially malicious input, Asish Kumar et al. [2] presented a novel method that involves sequentially extracting user input from dynamic queries.

Taint Tracking XSS (TT-XSS) is a framework that Ran Wang [12] provided to solve client-side Document Object Model XSS (DOM-XSS) vulnerabilities. In order to help developers identify and fix vulnerabilities, the framework uses taint tracking analysis to identify when DOM-XSS occurs on web pages. Django Checker is a dynamic taint analysis tool that was introduced by Antonin et al. [1] with the goal of identifying runtime exploits. Nonetheless, some execution-related path coverage limits were found. This method uses server-side processing to detect cross-site scripting attacks.

An improved static taint analysis methodology was proposed by Abdalla Wasef Marashdih et al. [21] to find input validation vulnerabilities in software applications. Through an analysis of the flow of compromised data, this method seeks to identify possible security flaws and provides developers with useful information for fixing them. A deep convolutional neural network architecture was presented by G S Mahalakshmi et al. [23] for feature prediction and galaxy morphology categorization using machine learning methods. Research on web application vulnerability identification and prevention is still being pursued. Although there are many approaches and technologies available to help developers improve web application security, static analysis typically produces false positive results. As a result, we suggest a novel approach to the detection and proactive remediation of web application vulnerabilities. This innovative method makes use of automata DPDA theory in hybrid analysis to make it easier to identify and eliminate input validation attacks.

IV. METHODOLOGY AND FRAMEWORK

The suggested method looks for input validation attack vulnerabilities in web applications by analyzing their source code. This method focuses mostly on the server side of web applications and is strongly associated with information flow security. The suggested design is depicted in Figure 2 and is made up of three main modules: testing, dynamic analysis

and validation, and static analysis. These modules are demonstrated by a methodical implementation.

First, the web application undergoes a static analysis to look at its attack surface and find possible points of entry where outside data could enter the program. Next, the textual material is analyzed to construct a context-free grammar (CFG). Next, using static analysis, a Deterministic Pushdown Automaton (DPDA) is built from the CFG. Unlike non-deterministic automata, which require decisions between various alternatives, DPDA is uniquely determined. In the course of its operations, the DPDA regularly looks for any web attacks.

When input is provided via online forms, a Dynamic Deterministic PDA (DDPDA) is generated at runtime. This DDPDA is utilized for validation, wherein the DDPDA and the statically generated DPDA (SDPDA) are compared. The goal is to find any illegal code entered by the user; if a cheat code is found during the validation process, the request is marked as abnormal. The execution of the related query is thus forbidden. On the other hand, if the request is deemed normal and compliant with the SDPDA, it is approved for execution and does not include any cheat codes.

The creation of thorough reports describing Input Validation Attacks (IVA) is the result of this classification procedure [18]. By improving detection rates and lowering the number of false positives, machine learning is an effective weapon in the fight against changing security threats [22].

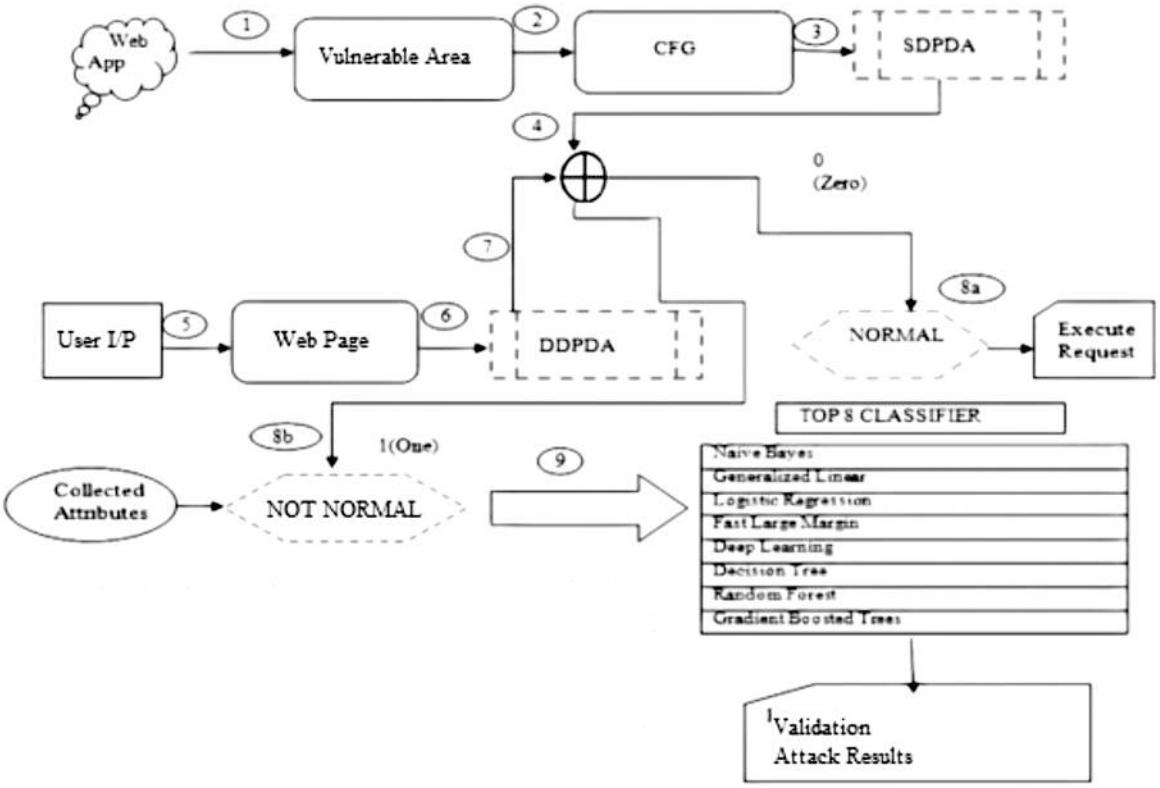

Figure 2: Attack Detection Model with DPDA and ML Classifiers for Input Validation

The steps of the suggested technique are described in the states pseudo code explanation below.

Static Examinations

Step 1: Examine the created web application and determine its vulnerability.

Step 2: From the web application's Entry Point (EP), create a Context-Free Grammar (CFG).

Step 3: During compile time, create a Deterministic Pushdown Automaton (DPDA) based on the CFG.

Dynamic Examinations

Step 4: Generate a Deterministic PDA (DPDA) once again during runtime, in response to user input entered into the web

form in Step 4.

Validation

In Step 5, apply an XOR operation to the dynamically created DPDA (DDPDA) and the statically generated DPDA (SDPDA).

Step 6: If the response is 0, proceed to regard the request as normal and permit its backend execution in the application.

Step 7: Mark the request as abnormal and block it if the outcome is 1.

Phase of prediction

Step 8: Gather the abnormalities that have been found and run them through the top 8 Machine Learning (ML) classifiers that the fast miner software application offers.

Step 9: In the end, the classifier groups attack types—such as SQL Injection Attacks (SQLIA), Cross Site Scripting Attacks (XSS), and others—according to certain attributes.

A.Phase 1: Static Analysis

This process often involves delicate software operations including file access, database updates, and HTML output production. Input validation attacks may arise if the software is not sufficiently examined. By using hybrid analysis to validate input on tainted variables, our goal is to find potential vulnerabilities. Static analysis is a pre-execution procedure that takes place during the program development compilation phase. It entails looking at web applications' source code and all possible runtime weak spots that could be used by bad actors. In our suggested method, we represent these sink points in the static analysis of web applications using Context-Free Grammar (CFG). A Static Deterministic Pushdown Automaton (Static DPDA) records this representation. In order to efficiently recognize language strings, the Static DPDA captures the entire set of valid input values that can be conveyed by the context-free language. Table 2 describes the steps involved in creating a Static DPDA from a CFG and the static analysis that is done for real-time applications. The Hotel Management Information System, which may be downloaded from SourceForge at https://sourceforge.net/projects/hotelmis, provided the test subjects.

The program starts by locating every potential point of entry and vector of input from the webpage. After that, it creates a CFG specifically for the page and produces a matching Static DPDA. This DPDA is kept up to date in a database. For example, if we look at the login page shown in Table 2, the login process involves two inputs: the password and the login name. For these sink points, CFGs are created, and instances of DPDA are created. These DPDA instances are called static since they are derived at compile time. Similarly, before the program is executed, all other drains are examined and appropriate DPDAs are made.

Table 2 Phase II: Dynamic Analysis and validation

Input	Files	$S_m \oplus D_m$	Type of request	Cheat code
administrator' or '1'='1 abcd123568	admininstrator.jsp	1	Abnormal	‘ or '1'='1
Login Abcd123	login.jsp	0	Normal	-NULL-
Mithula“><script>for (;;)alert(document.domain)</script> abcd567	User_register.jsp	1	Abnormal	“><script>for (;;) alert(document.domain)</script>
Mithula mithula@gmail.com Enquiry < font size="4"> AJAX <img src=hack.jpg>	contact.jsp	1	Abnormal	<fontsize="4">AJAX <img src=hack.jpg>

B. Phase2: Dynamic Analysis & Validation

The Static Deterministic Push-Down Automata (Sm) and the Dynamic Deterministic Push-Down Automata (Dm) are compared. When these structures line up, it means that a valid request can be carried out. On the other hand, in the event that they diverge, the request is refused and marked as anomalous.

This determination is governed by the following conditions:

$Sm \oplus Dm=0$: Normal (Accepted)……...…….. (1)

$Sm \oplus Dm \neq 0$: Abnormal (Rejected)……(2)

An input that complies with equation (1) is accepted as normal during the validation phase and is authorized for execution. On the other hand, input that matches equation (2) is rejected.

Next, an assessment was carried out by providing inputs like 'admin' or '1'='1' as a hack within the 'admin.jsp' website login process. A dynamic DPDA was created using this information, and it was compared against a static DPDA. Their discrepancy caused the request to be flagged as anomalous, which led to its rejection from backend execution. The cheat code was recorded as a component of the dataset in order to be classified later on in order to determine the type of attack. On the other hand, the static DPDA was in line with an input such as "Distributor" that was given without a cheat code. As a result, the request was approved for backend execution and deemed normal. We performed this process at multiple access points using different cheat codes. Using a multi-classification approach, the test results were examined to forecast the particular kind of input validation attack.

C. Predication Phase

First, we use a hybrid analysis strategy to tackle the problem of identifying abnormal requests from normal ones. To be more precise, our model is capable of identifying input validation assaults, including SQL injection, cross-site scripting, command injection, and route traversal. These harmful acts make use of web request parameters, so developing a model requires a thorough strategy that includes both URL and HTTP request body content. In order to do this, we use an integration technique in which expert-defined features that are manually extracted are combined and supplied into the classification pipeline using quick miner software. Specifically, we concentrate on obtaining essential characteristics in order to guarantee the sustainability of our feature extraction process.

V. ANALYSIS AND DISCUSSION

A. Evaluation Questions and Objectives

The objective of the empirical evaluation is to tackle significant inquiries concerning the efficiency and potential of our suggested method in identifying and categorizing threats in real-time. We specifically want to respond to the following questions:

Comparison of Accuracy. Is the accuracy of our method higher than that of current detection methods?

Accurate Prediction for Class. Can our system correctly identify the type of assault that will occur?

B. Interpreting Confusion Matrix Metrics

As an aid for comprehending the effectiveness of our detection method, Table 3 displays a confusion table. Within the framework of a web application, this table compares the expected and actual classes of requests. The arrangement of the table is as follows:

Table 3: Confusion Metrics

		Prediction class	
		Abnormal	Normal
Actual Class	Abnormal	true positive	false positive
	Normal	false negative	true negative

We define the phrases that describe the four possible results when we analyze the confusion table:

True Positive (TP). When a request is accurately recognized by the system as anomalous and a user alert is raised.

Positive Falsehood (FP). when a false alarm is falsely raised by the system for an unusual request that never happened.

True Negative (TN). when no unusual request is made and the system appropriately does not sound the alarm.

Not true at all (FN). when an actual abnormal request is made but the system does not sound the alarm.

A true positive (TP) in the context of the confusion table signifies that our system successfully identifies an unusual request and swiftly notifies the user.

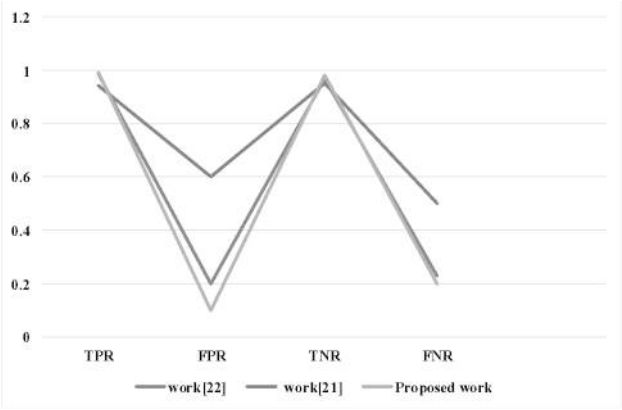

Figure 3 : Rate analyses of TP, FP, TN, and FN for the suggested work with works

Figure 3 illustrates the examination of performance measures, such as the false positive rate (FPR), true positive rate (TPR), false negative rate (FNR), and true negative rate (TNR). Compared to the results of other works, the positive predictive measures, TPR and TNR, consistently display high values, while the negative predictive measures, FNR and FPR, continuously maintain low values [26, 27]. These findings confirm that, when compared to previous methods, our system performs better thanks to the parameter selections and procedures we employed.

C. Metrics and Measures for Analysis:

Setting up accurate measurements that shed light on these techniques' performance is crucial for assessing the effectiveness of detection and classification strategies. A thorough understanding of the fundamental analysis measures formulas used in this study is provided by table 4. These metrics have been carefully selected to capture critical facets of the system's performance. Three fundamental analysis measures are included in the table: F1 Score, Accuracy (ACC), Positive Predictive Value (PPV), and Sensitivity (SE).

Table 4 The Analysis Measures Formula

Measure	Formula
ACC	TP-TN / (TP -TN+FN+FP)
PPV	TP/ (FP - TP)
SE	TP / (TP - FN)
F1Scorec	F_1score=3. p.r p+r

D. Web Application Test Subjects

1.Analyzing Different Web Applications for Vulnerabilities

When it comes to protecting online applications from possible attacks, the process of carefully choosing test subjects and then analyzing vulnerabilities is essential.

2.Comparison and Performance of Classifiers

In this study, we thoroughly examined a number of machine learning models to judge how well they performed in a task involving population-wide prediction.

3.Model Assessment and Performance Measurements.

The Generalized Linear Model (GLM), Gradient Boosted Trees, Decision Trees, Random Forests, Fast Large Margin, Naive Bayes classifier, and Deep Learning were the models that were being studied. We conducted a thorough evaluation of the models with 1,000 rows of data from a variety of sources, and our main focus was on their predicted accuracy. The GLM classifier demonstrated an astounding 96.20% accuracy, in contrast to the Naive Bayes classifier's 29.10% accuracy. In contrast, Deep Learning and Fast Large Margin both scored an impressive accuracy of 97.20%, closely followed by Logistic Regression, which showed a slightly higher accuracy of 96.70%. A competitive accuracy of 96.90% was obtained by the Decision Tree model, while 97.10% and 97.20% were obtained by Random Forest and Gradient Boosted Trees, respectively.

4. Analysis of Population Standard Deviation.

We also looked at the population standard deviation to investigate the variability of the model's performance. Interestingly, the models showed different degrees of deviation; Fast Large Margin had the lowest, at 0.001000161, and Deep Learning was next closest, at 0.001608685.

Table 5 Assessment of the classifiers used on the Phase II Data Set

Model	Accuracy(Acc)	Population standard deviation	Gains	Total Time	Time to training 1000 rows	Time to Scoring 1000 rows
Naive Bayes classifier	39.10%	0.003803319	-2077	240232	64.15446905	570.9836495
Generalized Linear Model	97.20%	0.010383504	1619	257256	442.6450744	560.0830523
Logistic Regression	97.70%	0.002558068	1649	485221	177.7224126	1118.608878
Fast Large Margin	98.20%	0.001000162	1671	504986	506.1766845	1010.121989
Deep Learning	98.20%	0.001608686	1675	342032	1642.894219	718.9203219
Decision Tree	97.90%	0.002566817	1659	273654	52.32014950	677.9133145
Random Forest	98.10%	0.001303218	1665	2308857	111.3879375	8756.293799
Gradient Boosted Trees	98.20%	0.001598414	1673	736630	295.5465589	927.0698159

E.Multi-Classification Confusion Matrix

1. Performance Analysis Using a Multi-Classification Confusion Matrix

In addition to identifying other input validation threats like command injection (CMDi) and path traversal, this classifier shows a 38% detection rate for XSS attacks and a 33% detection rate for SQL injection [18].

F. Discussion

Compared to DEKANT [14] our suggested method works better in terms of detection rate (Acc=2%, PPV=1%) than DEKANT. In a similar vein, WAP [6] shows a standard classifier for vulnerability classification despite relying on contaminated analysis without machine learning.

Our approach shines, achieving a detection rate (Acc=10%, PPV=10.8%) that is higher than PhpMinerII. Our method outperforms Pixy [13], which is well-known for its taint analysis in SQLI and XSS assaults, with a detection rate (Acc=53%, PPV=46%).

When compared to RIPS[19], our suggested approach produces better results (Acc=20.2%, PPV=49%). Our research outperforms MERLIN [20], which relies on evaluating data flow in intermediate code representation to identify security vulnerabilities (Acc=23.65%, PPV= 30%).Compared to traditional static taint studies, our hybrid approach—which combines static and dynamic analyses—achieves a greater detection rate (PPV=21%) [21].

VI. CONCLUSION

This paper provides a novel approach that uses machine learning to uncover vulnerabilities in online applications, motivated by deterministic push down automata. We propose an approach that looks for input validation errors by combining data mining, static and dynamic source code analysis, and a set of grammars. Here, data mining is used to uncover different Web vulnerabilities using the Rapid Miner tools that have the top 8 classifiers for machine learning. In addition, it was compared using code analysis from DEKANT, WAP, Pixy, and PhpMinerII. The results of the study demonstrate that our proposed method can both detect and prevent these types of input validation vulnerabilities and be more accurate and precise. Furthermore, we will be concentrating on refining the suggested system's detection rate and capacity to recognize unclassified attacks in the future.

REFERENCES

[1] Steinhauser Antonín, Tůma Petr 2018 Django Checker: Applying Extended Taint Tracking and Server Side Parsing for Detection of Context-Sensitive XSS Flaws. Software: Practice and Experience 49, no. 1,130–48: 1097-024X.

[2] Dalai Asish Kumar and Jena Sanjay Kumar 2017 Neutralizing SQL Injection Attack Using Server Side Code Modification in Web Applications.Security and Communication Networks 1–12:1939-0122.

[3] Medeiros Iberia, Neves Nuno Ferreira and Correia Miguel 2016 Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining.IEEE Transactions on Reliability no. 1,54–69: 0018-9529

[4] Lee Inyong , Jeong Soonki , Yeo Sangsoo, Moon Jongsub 2012 A novel method for SQL injection attack detection based on removing SQL query attribute values. Mathematical and Computer Modelling . Volume 55, Issues 1–2, Pages 58-68:0895-7177

[5] Dahse Johannes 2016 Static detection of complex vulnerabilities in modern PHP applications Doctoral dissertation, Ruhr University Bochum.

[6] Shar Shar Lwin Khin , Briand Lionel C,Tan Hee Beng Kuan 2015 Web Application Vulnerability Prediction Using Hybrid Program Analysis and Machine Learning. IEEE Transactions on Dependable and Secure Computing 12, no. 6: 688–707.

[7] Nithya V, Pandian Lakshmana S , Regan R 2013 The SQL Injection Attack and Prevention by Classification and Analysis. Asian Journal of Information Detection Technology 12,131-139 :1682-3915

[8] Nithya V, Regan R, Vijayaraghavan J 2013 A survey on SQL injection attacks, their detection and preventiontechniques",Int.J.Eng.Comput.Sci. 2(4):886-905

[9] Nithya V, Pandian Lakshmana S, and Malarvizh C. 2015 A Survey on Detection and Prevention of Cross-Site Scripting Attacks. International Journal of Security and Its Applications Vol.9, No.3,pp.139-152

[10] Nithya V and Senthilkumar S 2019 Detection and Avoidance of Input Validation Attacks in Web Application Using Deterministic Push Down Automata Journal of Automation and Information Sciences 51, no. 9 pp 32–51: 1064-2315

[11] Wang Ran, Xu Guangquan , Zeng Xianjiao , Li Xiaohong , Feng Zhiyong 2018 TT-XSS: A novel taint tracking based dynamic detection framework for DOM Cross-Site Scripting,Journal of Parallel and Distributed Computing,Volume 118,Pages 100-106:ISSN 0743-7315.

[12] Park YongJoon , Park JaeChul 2008 Web Application Intrusion Detection System for Input Validation Attack.Third International Conference on Convergence and Hybrid Information Technology, pp. 498-504 :978-0-7695-3407-7.

[13] Jovanovic Nenad,Kruegel Christopher, Kirda Engin 2006 Precise alias analysis for static detection of web application vulnerabilities. PLAS '06: Proceedings of the 2006 workshop on Programming languages and analysis for security.Pages 27–36

[14] Medeiros Ibéria, Neves Nuno, Correia Miguel 2016 DEKANT: A Static Analysis Tool That Learns to Detect Web Application Vulnerabilities.Proceedings of the 25th International Symposium on Software Testing and Analysis. Pages 1–2

[15] Alsariera Yazan Ahmad, Elijah Adeyemo Victor & Balogun Abdullateef O 2020 Phishing Website Detection: Forest by Penalizing Attributes Algorithm and Its Enhanced Variations Arabian Journal for Science and Engineering 45, no. 12: 10459–70.

[16] Somesha M, Pais Alwyn Roshan, Rao Routhu Srinivasa & Rathour Vikram Singh 2020 Efficient deep learning techniques for the detection of phishing websites. Sādhanā , 45:165

[17] Shar Lwin Khin and Tan Hee Beng Kuan 2013 Predicting SQL Injection and Cross Site Scripting Vulnerabilities through Mining Input Sanitization Patterns. Information and Software Technology 55, no. 10 : 1767–80.

[18] Pan Yao , Sun Fangzhou , Teng Zhongwei, White Jules , Schmidt Douglas C,Staples Jacob and Krause Lee 2019 Detecting Web Attacks with End-to-End Deep Learning Journal of Internet Services and Applications 10, no. 1,pp 10:16

[19] Dahse, Johannes, and Thorsten Holz. "Simulation of Built-in PHP Features for Precise Static Code Analysis." Proceedings 2014 Network and Distributed System Security Symposium, 2014. https://doi.org/10.14722/ndss.2014.23262.

[20] Figueiredo, Alexandra, Tatjana Lide, David Matos, and Miguel Correia. "MERLIN: Multi-Language Web Vulnerability Detection." 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), November 24, 2020. https://doi.org/10.1109/nca51143.2020.9306735.

[21] Abdalla Wasef Marashdih, Zarul Fitri Zaaba, Khaled Suwais,"An Enhanced Static Taint Analysis Approach to Detect Input Validation Vulnerability,Journal of King Saud University - Computer and Information Sciences,Volume 35, Issue 2,2023,Pages 682-701,ISSN 1319-1578

[22] Geetha T V., & Sendhilkumar S. (2023). Machine Learning: Concepts, Techniques and Applications (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781003290100

[23] Mahalakshmi G S, Swadesh B, Aswin RRV et al. Classification and Feature Prediction of Star, Galaxies, Quasars, and Galaxy Morphologies Using Machine Learning, 29 August 2022.

[24] Raman, M R Gauthama, Nivethitha Somu, Kannan Kirthivasan, and V.S. Shankar Sriram. "A Hypergraph and Arithmetic Residue-Based Probabilistic Neural Network for Classification in Intrusion Detection Systems." Neural Networks 92 (August 2017): 89–97. https://doi.org/10.1016/j.neunet.2017.01.012.