International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 1 Jan. 2023, pp: 357-365 www.ijaem.net ISSN: 2395-5252

Predicting Vulnerabilitiesin Websites based on a Security Model.

Parth Kasurde

Date of Submission: 01-01-2023 Date of Acceptance: 10-01-2023

ABSTRACT
One of the most urgent roads of correspondence between different specialist co-ops and clients are currently web applications. Web applications have become more essential, however their absence of safety has likewise become more dangerous. Web computer programmers need support in tracking down helpless code because of the restricted time and assets accessible. They would have the option to concentrate security reviewing endeavors on the off chance that there was a plausible method for foreseeing weak code.
In the proposed approach, input approval and sterilization code designs are portrayed as huge marks of web application weaknesses by cross breed (static + dynamic) program highlights. The accessibility of information with the weakness data marked for preparing is a requirement for the ongoing vulnerable expectation draws near. Past weakness information is every now and again inaccessible or in any event deficient for most of web applications. Accordingly, this approach might be used to address the two situations where named earlier information is very much accessible or not completely accessible. The web program is partitioned into more modest sinks, and info approval and disinfection ascribes are created utilizing dynamic and static program investigation.

I. INTRODUCTION

Many of our daily activities, including social media, email, banking, shopping, registrations, and so forth, depend heavily on web applications. Web application vulnerabilities may potentially have a bigger impact than flaws in other types of software because web software is also more accessible. The security of web applications is the sole responsibility of web developers. Unfortunately, they frequently lack the security training necessary to become familiar with cutting-edge web security methods and frequently have little time to follow up on newly discovered security concerns. They can use input validation and input sanitization, two secure coding techniques, to shield their systems from these widespread flaws.Data length, range, type, and sign are a few examples of necessary qualities that are commonly checked during input validation. In general, input sanitization removes unwanted characters from an input string by allowing only those that have been pre-defined and rejecting those that have special meanings for the interpreter being taken into account. It makes sense that if the developers didn't use these strategies properly or to a high enough degree, an application would be vulnerable.

It may be possible to anticipate web application vulnerabilities using the code attributes that describe the validation and sanitization code used in the programme.

Based on this premise, we provide a group of code traits known as input validation and sanitization (IVS) attributes, from which we develop precise, scalable, and fine-grained vulnerability predictors. Because it locates vulnerabilities at programme statement levels, the methodology is fine-grained. To extract IVS properties, we employ both static and dynamic programme analysis techniques. Static analysis can be used to evaluate a program's fundamental attributes.

Dynamic analysis, however, can concentrate on more precise code features that are an addition to the knowledge discovered by static analysis. Instead of exactly proving their correctness, we merely infer the potential types of input validation & sanitization code using dynamic analysis, and then we utilise machine learning to identify vulnerabilities based on these inferences.

As a result, we address the scalability problem that dynamic analysis frequently has.

In order to accurately and scalability predict vulnerabilities, our suggested IVS attributes reflect relevant properties of a implementations of input validation and input sanitization methods in web programmes.Additionally, our method may be applied in situations when there is a lack of training vulnerability data because it uses both supervised

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 1 Jan. 2023, pp: 357-365 www.ijaem.net ISSN: 2395-5252

learning & semi-supervised learning techniques to create vulnerability predictors from IVS properties.

CURRENT SYSTEM

The most pervasive and risky web-based application weaknesses these days that jeopardize the security and protection of the two clients and applications are SQL infusion (SQLI), cross-website prearranging (XSS), remote code execution (RCE), and record consideration (FI). Input approval and information sterilization are two secure coding systems that web engineers can use to protect their projects from these boundless defects. Information length, reach, type, and sign are a couple of instances of fundamental characteristics that are normally checked during input approval. By and large, input sterilization eliminates undesirable characters from an information string by permitting just those that have been pre-characterized and dismissing those that have unique implications for the translator being considered. It's a good idea that in the event that the designers didn't utilize these procedures appropriately or to a sufficiently high degree, an application would be helpless. Various web weakness discovery methods, including static pollutant investigation, dynamic impurity examination, demonstrating checking, emblematic and concolic testing, have been proposed to address these security issues.Approaches for static taint analysis are generally scalable but useless in practise due to large false positive rates.

While very accurate since they can generate actual attack values, dynamic taint analysis, model checking, symbolic, and concolic testing methodologies have scaling problems for large systems because of the path explosion problem. Scalable vulnerability prediction techniques are also available.

However, current prediction methods only discover vulnerabilities just at level of software modules, which is a coarser level of granularity.

SUGGESTIVE SYSTEM

They can use input validation and input sanitization, two secure coding techniques, to shield their systems from these widespread flaws. Data length, range, type, and sign are a few examples of necessary qualities that are commonly checked during input validation. In general, input sanitization removes unwanted characters from an input string by allowing only those that have been pre-defined and rejecting those that have special meanings for the interpreter being taken into account. It makes sense that if the developers didn't use these strategies properly or to a high enough degree, an application would be vulnerable. We postulate that the program's validation and sanitization code characteristics could be used to anticipate web application vulnerabilities.Based on this premise, we provide a group of code traits known as input validation and sanitization (IVS) attributes, from which we develop precise, scalable, and fine-grained vulnerability predictors. Because it locates vulnerabilities at programme statement levels, the methodology is fine-grained. To extract IVS properties, we employ both static and dynamic programme analysis techniques. Static analysis can be used to evaluate a program's fundamental attributes. Dynamic analysis, however, can concentrate on more precise code characteristics that are an addition to the knowledge discovered by static analysis. Instead of exactly proving their correctness, we used dynamic analysis simply to infer the potential types of input validation & sanitization code and then applied machine learning to these inferences to vulnerability prediction.As a result, we address the scalability problem that dynamic analysis frequently has.

In order to accurately and scalability predict vulnerabilities, our suggested IVS attributes reflect relevant properties of a implementations of input validation and input sanitization methods in web programmes. Additionally, we construct vulnerability predictors (Figure 1) from IVS features using supervised learning & semisupervised learning techniques so that our method can be applied in situations when there is a dearth of vulnerability data for training.

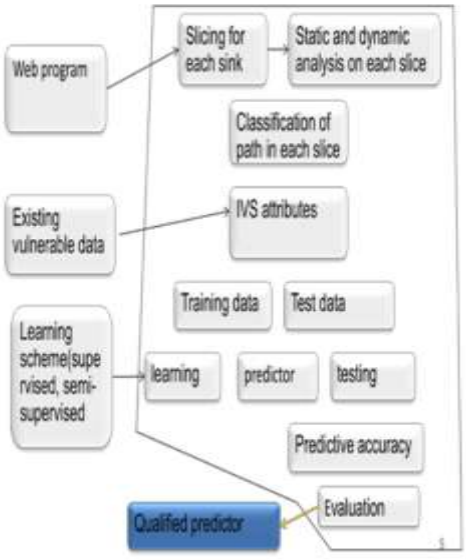

FIGURE 1: PROPOSED SYSTEM DIAGRAM

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 1 Jan. 2023, pp: 357-365 www.ijaem.net ISSN: 2395-5252

1. PROGRAM ANALYSIS, STATISTICAL AND DYNAMIC

To extract IVS properties, static and dynamic programme analysis approaches are also applied. Static analysis can be used to evaluate a program's fundamental attributes. Dynamic analysis, however, can concentrate on more precise code characteristics that are an addition to the knowledge discovered by static analysis. Instead of exactly proving their correctness, the dynamic analysis is just utilised to infer the potential types of input validation and sanitization code.

2. BACKWARD CUTTERSLICING

By dissecting the data and control flow of programmes, programme slicing is a technique for programme analysis and change. A slice is an executable programme that, given an imperative programme, must behave exactly like the specific subset of the behaviour of the original programme. The programme statements that are (perhaps) connected to the results of computations made at programme points and/or variables are referred to as programme slices.

3. ANALYSIS OF HYBRID PROGRAM

The review depends on the web application program's control stream diagram (CFG), program reliance chart (PDG), and framework reliance diagram (SDG).

One source code articulation is addressed by one hub in the charts. Subsequently, contingent upon the circumstance, we can utilize program proclamation and hub conversely.

A sink is a hub in a CFG that utilizes factors characterized from sources other than the info and is subsequently possibly open to enter control endeavors. Accordingly, we can expect shortcomings at the assertion level.

The hubs at which information from the external climate are gotten to are known as information hubs.

In the event that a variable is characterized from input hubs, it is polluted. For each sink and the arrangement of corrupted factors utilized, a regressive static program cut is processed as the most important phase in the strategy, as was recently referenced. As to the cutting measure, the regressive static cut comprises of any hubs (counting predicates) in the CFG that could impact the upsides of the subset of factors. As indicated by the methodology given by Ferrante et al., we initially make the PDG for the primary strategy for a web application program and afterward make PDGs for the techniques that are called from the principal strategy. The SDG is then fabricated.

SHARING OF Each and every SINK

Program cutting is a method for consequently separating programs by taking a gander at their information and control streams.

Cutting decreases a program to its most essential structure while keeping up with the capacity to make a given subset of conduct. For each sink and the assortment of spoiled factors used in the sink, a retrogressive static program cut is figured as the most vital phase in our procedure.

DYNAMIC AND STATISTICAL ANALYSIS OF EACH SLICE

The developers will employ acceptable input validation & sanitization techniques, but they might overlook some inputs for validation since they don't identify all the data that may be changed by outside users. As a result, it's critical to first determine all the input sources when performing security analysis.The reason for categorising the inputs into multiple sorts is that each class of inputs generates a distinct form of vulnerability, and it could be necessary to secure these various classes of inputs using various security defence strategies.

PATH CLASSIFICATION IN EACH SCLICE,

A backward static programme slice with regard to the sink statement and the variables utilised in the sinks is calculated for each sink. To extract each path's validation and sanitization effects on those variables, a hybrid (static and dynamic) analysis is used to study each slice's path. The path is subsequently categorised in accordance with the hybrid analysis's inferred input validation and sanitization effects.

IVS ATTACHMENTS

These characteristics describe several kinds of programme operations and functions that are frequently employed as input validation and sanitization steps to counter web application vulnerabilities. These characteristics are used to categorise functions and operations in accordance with their security-related characteristics. Attributes to be extracted using static analysis and dynamic analysis are called hybrid analysis-based attributes. Our categorization system includes input sources because the majority of frequent vulnerabilities result from incorrectly classifying inputs.

To put it another way, developers might employ acceptable input sanitization and validation techniques, but they might miss some inputs for validation if they don't recognise all the data that

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 1 Jan. 2023, pp: 357-365 www.ijaem.net ISSN: 2395-5252

could be changed by outside users. As a result, it's crucial in security analysis to first identify all.

This hybrid analysis-based classification is used for validation and sanitization techniques that employ both common and uncommon security functions. We classify them based on their security-related information if there are just standard security functions to be classified; otherwise, dynamic analysis is employed. Various built-in and/or custom language functions can be used to provide input validation and sanitization procedures. Since web application inputs are naturally strings, specific input validation and sanitization procedures are typically implemented using string replacement/matching functions or string manipulation techniques like escaping. A group of string functions that accept safe strings or reject unsafe strings often makes up a solid security function.These functions are undoubtedly crucial vulnerability indicators, but we still need to consider each validation and sanitization function's goal since various protection strategies are typically needed to stop various types of vulnerabilities. It is crucial to categorise the methods used in a programme path into different types so that our vulnerability predictors may learn this information and forecast upcoming flaws when combined with the related vulnerability data.

BUILDING A MODEL FOR PREDICTING VULNERABILITY

Building vulnerability predictors can make use of a variety of machine learning approaches. Regardless of the specific method employed, the objective is to discover and generalise sink-related data patterns that can be effectively used to forecast susceptibility for new sinks. It is crucial for a vulnerability analysis approach to be flexible as more advanced security assaults are found. Re-training with machine learning allows one to adjust to new vulnerability patterns.

A. REPRESENTATION OF DATA

A path in a slice of a sink serves as our unit of measurement, or an instance in machine learning terms, and we describe each path using IVS properties. Depending on the amount of programme operations or functions that are categorised, the attribute values may range from zero to an upper bound.

Each path would've been represented by a 33-dimensional attribute vector because 33 IVS properties are recommended.

II. PROCESSING OF DATA

The ratio of vulnerable sinks versus non-vulnerable ones is generally low in our datasets. This is an issue with data imbalance that is typical of many vulnerability datasets. The performance of machine learning classifiers can be significantly impacted by imbalanced data, as some of the data may not be learned by the classifier because of its lack of representation, leading to induction rules that tend to explain the majority of the class data and favour its predictive accuracy.We require a high level of prediction accuracy for this class since missing vulnerability is considerably more important than sending a false alarm because for our situation, the minority data set capture the instances that are "vulnerable." We employ a sampling technique known as adaptive synthetic oversampling to solve this issue. It reduces the bias caused by the class imbalance problem by generating synthetic, false data for the minority class instances to balance the (unbalanced) data.

Since it doesn't involve changing existing classifiers, it can easily be implemented as a data granularity pre-processing step.

DIRECTED LEARNING

Classification is an example of supervised learning because each training instance needs to have a class label assigned to it. In this work, we create Random Forest (RF) and logistic regression models from the suggested features. Statistical categorization models include LR. Based on one or more predictor qualities, it can be used to predict the result (class label) of a dependant attribute. Models are used to describe the probability of the potential outcomes of a given incident. The kinds of monotonic relationships that can be modelled between the properties of the predictor and the probability of vulnerability are adjustable in the context of logistic regression analysis. RF is an ensemble learning technique for categorization that uses a number of classifiers with tree structures.When a group of learners rather than a single learner produces the final forecast, the predictive accuracy is frequently considerably increased.

Each tree casts a vote (classification) based on the input sample, and the forest outputs the classification that received the majority of votes from the trees.

LEARNING THAT IS SEMI-SUERVISED

As individual classifiers are combined in ensemble learning, a sizable amount of labelled data is often needed for training. Relevant and tagged data that are available for learning may be

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 1 Jan. 2023, pp: 357-365 www.ijaem.net ISSN: 2395-5252

scarce in some industrial environments. For training, semi-supervised algorithms [39] combine a modest amount of labelled data with a considerably larger amount of unlabeled data. When there are very few labelled data, this technique that uses unlabeled data can enable ensemble learning.

There are various benefits to combining ensembles and semi-supervised learning. Unlabeled information is used to enhance labelled training samples and enable ensemble learning. With unlabeled material that was labelled by the ensemble of all other learners, each individual learner improved. There are a few distinct kinds of semi-supervised approaches that have been proposed in the literature, including EM-based, clustering-based, and disagreement-based learning. However, none of these methods have yet been investigated for vulnerability prediction. We investigate the application of an algorithm dubbed Co Forest, Co-trained Random Forest (CF), which applies semisupervised learning to RF based on these objectives. It is a semi-supervised, disagreement-based learner. In order to utilise unlabeled data, CF employs several, different learners, combines them (semi-supervised learning), and maintains a significant degree of disagreement between the learners.

ADVANCED PREDICTOR

With the use of input validation & sanitation attributes, machine learning algorithms, and other factors, a qualified web application vulnerability predictor can be created. We can create a web application predictor that is highly accurate, fine-grained, and scalable by using the aforementioned features.

A. DERIVATION OF IVS ATTRIBUTES in V IMPLEMENTATION

It very well might be feasible to expect web application weaknesses utilizing the code ascribes that depict the approval and sterilization code utilized in the program.

In view of this reason, we give a gathering of code qualities known as information approval and disinfection (IVS) credits, from which we create exact, versatile, and fine-grained weakness indicators. The strategy is granular in light of the fact that it spots defects at the program proclamation level. To remove IVS properties, we utilize both static and dynamic program investigation strategies. Static investigation can be utilized to assess a program's crucial traits.

Dynamic examination, nonetheless, can focus on more exact code qualities that are an expansion to the information found by static investigation. Rather than precisely demonstrating the legitimacy of info approval and sterilization codes, we just construe them through unique investigation.

Part B: Rundown of IVS ascribes,

1. Client-available info got from HTTP demand boundaries like HTTP Get
2. Record based input, like treats and XML,
3. Text-data set - Admittance to a data set for text-based input
4. Numeric-data set - Getting to a data set for numeric-based input
5. Meeting: Information that is gotten through a super durable information object, like a HTTP Meeting

Uninit - A uninitialized program variable

7. Un-pollute - Capability that profits information that has been foreordained or information that has not been adjusted by different clients.
8. Known-vuln-client - A custom capability that has in the past prompted security issues.
9. Known-vuln-sexually transmitted disease - An implicit language highlight that has in the past prompted security issues.
10. Spread - A capability or strategy that engenders a string's whole or halfway worth.
11. Numeric Change from a text to a numeric capability

DB-administrator

12. Sifting capability for inquiry administrators like (=)

DB-remark delimiter

13. Separating of question remark delimiters, including (-)
14. The DB-unique capability channels extra information base exceptional characters, for example, (x00) and (x1a), that are unmistakable from the ones recorded previously.
15. The (') and (") string delimiters are separated by the string-delimiter capability.

Lang-remark delimiter 16. Separating capability for programming language remark delimiters like (/)

17. The "Other-delimiter" capability channels other delimiters other than those recorded above, for example, (#).
18. The Content label capability channels client script labels that are dynamic, for example, (script>).
19. A HTML-label capability that channels out (div>) labels from static client scripts
20. Occasion controller capabilities like (onload =) that disallow the use of contributions as the upsides of client-side occasion overseers.

Separating invalid byte (%00) with the invalid byte capability

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 1 Jan. 2023, pp: 357-365 www.ijaem.net ISSN: 2395-5252

22. Spot Separating Capability Speck (.)
23. The dotdot-slice (../) grouping separating capability
24. The oblique punctuation line sifting capability ()
25. A cut (/) sifting capability
26. The newline (n) sifting capability
The colon (,) or semi-colon channel capability is number 27. (;)
28. The Other-extraordinary Capability sift through some other exceptional characters not recorded previously.
29. An encoding capability that changes the configuration of a string
30. The canonicalize capability lessens a string to its most fundamental, standard structure.
31. An index way or URL sifting capability
Limit-length is a capability or methodology that confines a string's length to a specific worth.

III. INTERFACE DEVELOPMENT

The creation of the interface is the system's final element.

3.1 Interface Design

The interface is constructed to allow interaction using Java and the net beans environment. The ontology model, rules, and application interface are integrated using the Jena API.

FIGURE 2: APPLICATION INTERFACE

3.2 Usefulness

As shown in Figure 2, the interface is made to be user-friendly. It has two basic components: the first section collects user input, and the second part displays data.Below is a discussion of the application interface's specific features.

The ontology model, the rules, and the SPARQL query are loaded to anticipate the attacks in the first section once the user selects the vulnerability from the list box and submits the query. To manage any errors, the alert message is displayed as seen in Figure 3.

FIGURE 3:APPLICATION INTERFACE WITH ALERT

As indicated in Figure 4, the second step involves predicting and categorising attacks based on user input. The necessary defences against the attack and safeguards against web application attack prevention are also shown.

FIGURE 4: PREDICT AND CLASSIFY ATTACKS

IV. EVALUATION

A web application that addressed the weaknesses and implemented defences against the anticipated assaults was created in order to test the proposed solution.

Less vulnerabilities were discovered when the web app was scanned. The accuracy and prediction rate of the information that was retrieved were

International Journal of Advances in Engineering and Management (IJAEM)
Volume 5, Issue 1 Jan. 2023, pp: 357-365 www.ijaem.net ISSN: 2395-5252

measured using the parameters precision, recall, and F-measure.
These are used in their computation:
Precision equals Right / (Right + Wrong)
Correct/(Correct + Missed) = Recall
Where,
Correct: Both the algorithm and a human are able to extract the expected number of records from the overall number of irrelevant records (information).
Wrong: The system, not a human, pulls the number of expected records from the total amount of irrelevant records (information).
Precision + Recall = F-Measure = (2*(Precision*Recall))
Figure 5 displays the suggested system's prediction rate plotted for each attack. The graphic shows that each attack has a high prediction rate, which aids attackers in understanding the implications of the web application's developing vulnerabilities.
The findings of the suggested system's prediction rate are summarised in Table 1 and compared to security ontology .
Table 1 makes it evident that the proposed system uses the inference process to forecast more attacks than the current system does.
Additionally, the attack classification is contrasted with the current framework. The comparison chart for the assault classification percentage is shown in Figure 6. attack-classification capacity of the current system is low.

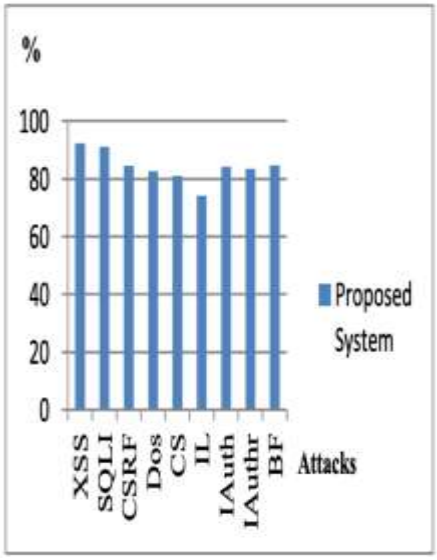

FIGURE 5: PREDICTION RATE OF PROPOSED SYSTEM

Table: 1. Comparison of Prediction rate

Web Application attacks	Proposed System	Existing System
Cross Site Scripting (XSS)	92.3	86.9
SQL Injection (SQLI)	91.05	87.09
Denial of services (Dos)	84.56	63.04
Cross Site Request Forgery (CSRF)	84.56	63.04
Content Spoofing (CS)	82.57	72.43
Information Leakage (IL)	74.09	70.08
Insufficient Authentication (IAuth)	84.23	66.89
Insufficient Authorization (IAuthr)	83.45	64.23
Brute Force (BF)	84.67	60.56

The experimental data demonstrates that our proposed system outperforms the current system in terms of prediction ability and attack categorization rate. When compared to previous systems, the system's average prediction rate is high and it successfully predicts web application attacks.

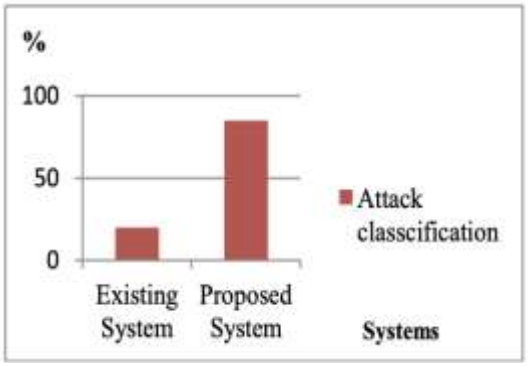

FIGURE 6: Comparison of Attack classifications

In addition, compared to the current system, our proposed system attack categorization rate is very high. This is because by assessing the vulnerabilities that may be exploited by the assaults and the threats that were deployed, our system can successfully forecast advanced attacks. In order to predict the majority of attacks, an inference method is employed to learn new information and rules.

V. CONCLUSION AND NEXT WORK

The survival indicates that e-commerce and information exchange are growing quickly, but security of personal information is also crucial. Hackers and crackers can access users' private information and take advantage of weaknesses in web applications. Web application developers and testers often create insecure applications because they lack sufficient awareness of threats, vulnerabilities, and attacks.

Although there are many technologies available to address this issue, none of them are fully safe and cannot be used to construct secure web applications. Web application threats can be predicted and categorised using the ontology-based approach.

The suggested system analyses threats and vulnerabilities that could be used in web application assaults.The suggested system is excellent at identifying threats and weaknesses that could be used in web application assaults.

Ontology models for threats, vulnerabilities, attacks, and rules are used to successfully and efficiently forecast sophisticated attacks. The list of attacks is generated by an inference engine using knowledge about web application vulnerabilities. The severity of the attacks on security objectives determines how the attacks are classed (Confidentiality, Integrity and availability).

The suggested solution also offers recommendations for attack mitigation and prevention. For developers and testers to handle assaults and create safe applications, this information is highly helpful. The proposed ontology model can be used in subsequent work to identify web application threats during testing.

BIBLIOGRAPHY

[1]. Khalid, M.N., Iqbal, M., Alam, M.T., Jain, V., Mirza, H., Rasheed, K.: Web unique method (WUM): an open source blackbox scanner for detecting web vulnerabilities. Int. J. Adv. Comput. Sci. Appl. (IJACSA) 8(12), 411–417 (2017)

[2]. Kaur, D., Kaur, P.: Empirical analysis of web attacks. Proc. Comput. Sci. 78, 298–306 (2016)

[3]. Alhassan, J.K., Misra, S., Umar, A., Maskeliūnas, R., Damaševičius, R., Adewumi, A.: A fuzzy classifier-based penetration testing for web applications. In: Rocha, Á., Guarda, T. (eds.) Information Theoretic Security. AISC, vol. 721, pp. 95–104. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73450-7_10

[4]. M. Vrancianu, L.A. Popa, Considerations regarding the security and protection of e-banking services consumers' interests, The Amfiteatru Economic Journal 12 (28) (2010) 388–403.

[5]. J. Kannan, P. Maniatis, B.G. Chun, Secure data preservers for web services, in: Proceedings of the 2nd USENIX Conference on Web Application Development, USENIX Association, 2011, pp. 3–3.

[6]. J. Undercoffer, J. Pinkston, A. Joshi and T. Finin, "A target-centric ontology for intrusion detection", In 18th International Joint Conference on Artificial Intelligence, pp. 9-15, March 2004.Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[7]. L.Daniel Costa, L. Matthew Collins, J.Samuel Perl, J.Michael Albrethsen, J.George Silowash, L. Derrick Spooner, An Ontology for Insider Threat Indicators Development and Applications, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA .

[8]. A. Herzog, N. Shahmehri, C. Duma, An ontology of information security, Techniques and Applications for Advanced Information Privacy and Security: Emerging Organizational, Ethical, and Human Issues (2009) 278–301.

[9]. J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by Lincoln laboratory, ACM Transactions on Information and System Security 3 (4) (2000) 262–294.

[10]. Carlos Blanco, Joaquín Lasheras, Eduardo FernándezMedina, Rafael Valencia-García and Ambrosio Toval, "Basis for an integrated security ontology according to a systematic review of existing proposals", Computer standards and Interfaces, Vol. 33, No. 67, pp. 372- 388, June 2011.

[11]. S. Fenz, G. Goluch, A. Ekelhart, B. Riedl, E. Weippl, Information security fortification by ontological mapping of the iso/iec 27001 standard, in: 13th Pacific Rim International Symposium on Dependable Computing, 2007, PRDC 2007, IEEE, 2007, pp. 381–388.

[12]. S.Parkin, A. Moorsel, and R. Coles, (2009). An information security ontology incorporating humanbehavioural implications. In Proceedings of 2nd International Conference on Security of Information and Networks, pp. 46–55.

[13]. Nadya ElBachir El Moussaid, Ahmed Toumanari, "Web Application Attacks Detection: A Survey and Classification", International Journal of Computer Applications, 2014,Vol 103, No.12.

[14]. F. Abdoli and M. Kahani, "Ontology-based Distributed Intrusion Detection System", In Proceedings of the 14th International CSI Computer Conference.

[15]. Golnaz Elahi, Eric Yu, and Nicola Zannone, "A Modeling Ontology for Integrating Vulnerabilities into Security Requirements Conceptual Foundations", Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 99-114, 2009.