Hybrid Optimization Driven Technique for Malicious Javascript Detection Based on Deep Learning Classifier

Scaria Alex, T Dhiliphan Rajkumar

Abstract—The growth of the web users and thecontents are increasing in a daily basis. In all these webpages the implementation of javascripts are a common factor. These scripts are used for the simplicity and achieve interaction with the user, but, also could be used to harm the end user by stealing information, redirecting to phishing pages and installing harmful softwares. This alarms an immediate look into the security concerns of the javascript. There exist many machine learning-based malicious script detection approaches, but majority of them follow a shallow discriminating models where manual definition of features are constructed with artificial rules. In this paper, a deep learning framework for detecting malicious JavaScript code is proposed combing the optimization power of Bird Swarm Algorithm. To extract high-level features from JavaScript code Stacked denoising auto-encoders are implemented and BSA is used to optimise the features and identify the malicious codes. The theoretical model [2] have an accuracy of 94% in identifying the malicious codes.

Keywords—Deep learning framework, javascript, Bird Swarm Algorithm, Stacked Denoising Auto-encoders

I. INTRODUCTION

As days passes by more and more users are using internet and the web itself is expanding with data. This alarms the increase in distributing malware among the users. Malware distributors Through the internet the malwares are distributed through various methodologies like: phishing to download sites, redirecting to unauthorized webpages, fake codec installation requests, malicious advertisements and spam messages on blogs, social network sites and other web pages. Most commonly the attacker uses malicious JavaScript codes during part of the attack,which includes cross-site scripting (XSS) and web-based malware distribution. JavaScript is a tool that may be used by the attacker to create a redirection for a user to a website hosting malicious software, to create a pop-up window recommending users to download a fake codec, to also detect which software versions the user is currently using and select a preferable method to exploit it. The initial infection method for any malware is to malicious the javascript. These malicious javascripts hides known exploits and save themselves from being detected by rule-based anti-malware software or anti-malware softwares based on regular expressions.

Revised Manuscript Received on December 16, 2019.

Dr. T Dhiliphan Rajkumar, Assistant Professor Department of Computer Science and Engineering, Kalasalingam University

Mr. Scaria Alex, Research Scholar, Department of Computer Science and Engineering, Kalasalingam University

The complexity related to each of these obfuscation techniques have been increased, raising the resources that are necessary to counter the attacks [9].The existing methodology for JavaScript security solution over XSS attacks is based on sand-boxing technique, which blocks the code to be performed on a restricted environment only. Within a browser the JavaScript programs are considered as untrusted software variables that have accessability to only a limited set of resources. The problem associated to the current solution is that the javascripts will be conformed to the sand-box policies, but still it will violate the security within the system. The sand-boxing mechanism embedded within the browsers prevent JavaScript code from being compromising the client's environment security , but, there exists a number of attacks that can be used to steal confidential information from the user which includes the cross site scripting attacks and pressurize users for providing highly sensitive information like passwords and online account details to unauthorized parties by the means of phishing attacks [10].

The JavaScript based malware attacks are of increase and have a better success rate in mass-scale exploitation. From the viewpoint of an attacker, the primary advantage is that the attacks could be carried out against an ordinary user visiting an ordinary web page. There are many techniques proposed to incur these attacks, but, in-browser implementation has been slow due to the performance overhead [11]. It is well known that JavaScript is the main vehicle for web-based attacks, enabling the delivery of sophisticated social engineering, drive-by malware downloads, cross-site scripting, and other attacks. It is therefore important to develop a system that could analyse the deep working of the javascript based malware attacks, thus enabling a better and robust defensive systems. However, while extensive previous work exists on JS code inspection [13] and web-based attack analysis [14], an important problem remains: to evade defense systems and security analysts, web-based attacks are often developed to be ephemeral and to deliver the actual attack code only if certain restrictive conditions are met by the potential victim environment [15]. Therefore there is a need for javascript based attack analysis tool that can capture in-browser activities and subsequently reconstruct the live security flaws while the end user browse the web. Malicious JavaScript is code that shows some kind of malicious unwanted behaviour, such as drive-by downloading, installation of other malware such as fake codec's, unwanted advertisements, or spam. The code is often hidden, making the basic code analysis and detecting the malware difficult. Malicious JavaScript code is often used as a first step for other malware attacks, tricking a user to install other kinds of malicious software, or to directly install and execute exploits [16].

II. LITERATURE REVIEW

Some approaches for malicious JavaScript detection use dynamical code analysis, such as client honeypot techniques [17], or statical analysis such as pattern matching [18]. Maintaining pattern-based systems can become a tedious task as new malicious scripts are published, creating a moving target, and using dynamical code analysis is typically computationally expensive. Some services, including Google Safe Browsing, maintain a black list of URLs with malicious content of some sort, and yet other approaches uses code signatures for detection. The black-list approach can provide a certain level of security, and is currently implemented in web browsers such as Firefox and Chrome [16].In [19], a malware detection framework was improvised by selecting application program interface (API) call statistics as malware features and by using the SVM as the classifier. In [20], malware behaviours has been classified by extracted features from the sequences of API calls and the k-nearest neighbor algorithm. In [21], a mining and machine learning approach to identify malicious JavaScript code was provided. In [22], a C4.5 decision tree algorithm was introduced that identified the unwanted scripts by analysing a set of features of traffic statistics, file system structure, and webpage contents. Even though these traditional mechanism of machine learning-based methods was able to predict the presence of an unknown malicious JavaScript code, the time taken to test the availability of malicious code is too expensive. In [23] The malicious codes are disassembled into opcodes by implementing N-gram algorithms that extracted features. The detection technology in the current world is moving faster from the traditional pattern-based matching to the newly generated machine-based learning enhancing more automatic and intelligent direction. The major requirement for detection results are not just encapsulated to the ability to accurately identify the known attacks but also to fight against potentially suspicious attacks.

A. Challenges

The attacker inserts malicious JavaScript code into the vulnerable web pages to expose the visitors onto severe network attacks like virus distribution, Trojan attack and confidential information extraction [1]. Malicious JavaScript code is hard to detect due to its hidden feature and complexity. Identifying such code have a considerable cost over the process. It is hence so because there are multiple pathways for an attacker to insert unwanted scripts. Thus JavaScript codes should be dynamically inspected and should identify different types of vulnerabilities within the browser'senvironment [2]. Javascript have the feature of collaborating with multiple programming languages. This interaction paves the way for the attacker to utilize the system [7]. The attacker could easily integrate javascript code to access camera, GPS, speaker, data transmission and other system information of an unsuspected user. The attackers hide there identity by disguising as a legitimate company by referencing them and includes highly sensitive malicious code within them [9].

II. PROPOSED METHODOLOGY

The primary intention of this paper is to design and implement a technique for malicious JavaScript detection. The overall procedure of the proposed technique involves: feature extraction and classification. At first, the input JavaScript codes will be subjected to the feature extraction phase in which the significant features will be extracted. Since each javascript code have its own unique structure and purpose, its not an easy task to define the features for a legitimate code. Even if a blacklist of the virus signatures are defined, a manual updation of the blacklist and matching the supplied code with the defined blacklist will be a hilarious task. Even for a heuristic detecting method, where the security experts set a block of rules to identify malicious from legitimate codes, there should be a frequent modification of the rules based on the newly identified malicious javascripts. Another approach called dynamic analysis verify the code within a controlled environment causing less deterioration to the end users. This approach have high level of security but it consumes a lot of time and cannot be represented as a real application because the malicious code will behave differently with different triggering conditions.

It is difficult to hence decide on the features to be chosen to detect the presence of malicious javascript codes due to having multiple links to other pages, data encapsulation, code reordering and rubbish strings insertion. To ease the process of feature extraction of the javascript code the deep learning technique is used which require least manual intervention. It is achieved by implementing the multiple-layer stacked denoising autoencoders (SdA) which will extract the features automatically. The learned features are then inputted to Bird Swarm Algorithm (BSA) to optimize the features extracted and help in identifying the malicious code from the legitimate ones. The advantage of the defined methodology is the learning based detection, where the deep- learning technique helps in identifying the key features of java script code which require zero intervention. The second advantage is the zero-day attack detection, which is achieved by extracting the intrinsic features of the attack. Thus the model can prevent even a previously unknown attack.

III. DETECTION METHODOLOGY

Using the deep learning technology the provided java script is scanned for vulnerabilities. The multi-layered stacked denoising auto-encoder(SdA) is used to extract the javascript features for analysis. The extracted features are then optimized using a logistic regression Bird Swarm Algorithm(BSA). The resulted feature helps in discovering the provided javascript code is malicious or not.

A. Deep learning

Deep learning is a sub- topic on Machine Learning. Its purpose is to learn high level representation of data with deep layer-wise method. Each layer first undergoes a pre-training with unsupervised data and then is fine tuned in a supervised mechanism.

Here deep learning is achieved using stacked denoising auto-encoder because its best in text classification [2].

B. Denoising auto-encoder (dA)

As input is fed to the system, an unmonitored pre-training is conducted on each layer. During this phase noise is added to the input so that the hidden layer will discover more authentic features instead of learning the mere identity. Thus the dA reconstructs the input from the corrupted noised version of the original data. The denoising auto-encoder thus have two functions: First it try to preserve the information about the input and second it tries to recover from the corrupted input stochastically. The half of the input value is set to zero by the stochastic corruption process.

C. Stacked denoising auto-encoders (SdA)

Denoising autoencoders can be repeated multiple times hierarchically by inputting the output of the denoising autoencoders' previous layer to the top layer. The unmonitored pre-training of js code vector is done one layer at a time method. Each layer of the denoising autoencoder is trained and the output will be generated by minimizing the error in reconstructing its input. Once the first n layers are trained with the noised input, we can train the n+1-th layer by inputting the code from the output of the nth layer.

Once all denoising autoencoders' layers are pre-trained, the output generated goes through a second stage of training called fine-tuning. The prediction error is considered to be reduced due the monitored fine-tuning. To achieve this, a logistic regression layer is first added on top of the received output code generated through the last layer. Then the entire network is trained as in multilayer perceptron. At this stage only the encoded part of the auto -encoder is only considered. This stage is done under supervision because the resultant class is used during the later training sessions.

D. Bird Swarm Algorithm (BSA)

The inputted JavaScript code is converted to binary feature vectors which will act as the input of the deep learning model. Every character in JavaScript codesegments is converted into an eight bit ASCII binary codes. All the JavaScript code segments are stored in the form of a binary file. This will generate over 20,000 feature dimentionalities. Inorder to reduce this high collection of input data to an adequate quantity that could decreases the processing cost, Bird Swarm Algorithm is proposed [26].

The birds occasionally have three basic behaviors: foraging, vigilance and fly in flocks. These are similar to the swarm behaviour of separation, alignment and cohesion. The birds fly in group in search of food. They identify the food source from the collective search of the group. One finds and others feed on it. While having food they raises their head for looking out predators. So they are always vigilant and its always better to have more in number than being alone to protect themselves from predators. They fly in a group and all try to be on the center as its the safest position. But the position is based on the bird with the highest reserve of food.

BSA is integrated as a solution to different types of problems on regression and classification of data. It has unique features that includes swarm algorithm, searching methods, population diversity, and local optima avoidance. The BSA is initiated by considering N number of birds in a X dimensional search space
[29]. The swarms fitness value is averaged to calculate the effect of change in surroundings as the birds move to the center.

In this context the birds will be the eight bit binary code extracted during the SdA, with its weights and biases fractions. The proposed method is initiated by specifying the SdA structure which includes the number of features, and the total number of biases and weights. Then, a random set of SdA networks is developed, that represents N birds. In the next stage the fitness value of individual bird is calculated based on a fitness function and the training dataset. In order to train dAs, the best global fitness, and best personal fitness of individual bird is updated first. Individual bird's vector value is updated then based on the bird's current status. The above defined stages are looped until the maximum number of iterations are reached. The resultant will be a collection of optimized features which can be used for evaluating the javascript. This results in faster generation of solutions with less time complexity.

E. Efficiency

According to [2], the SdA have produced an effective result of 94.82% accuracy with a True Positive Rate of 93.95% and False Positive Rate of 4.13%. The result was then compared to other machine learning algorithms like naive Bayes which scored only 93.32%, RBF SVM scored 91.67%, RIPPER scored 90.19%, and ADTree scored 85.08%. Among all the SdA outperformed. On the other hand BSA alone [29] have produced the highest classification rate when verifying with other algorithms like DE, GA, PSO, ACO, ES and ABC. Combining the features of BSA into SdA have increased the accuracy rate to 94% [29].

IV. CONCLUSION

This paper proposes a new method on deep learning methodology based on Bird Swarm Algorithm which will produce a significant improvement on detecting malicious JavaScript on webpages. The main reason for selecting BSA as an optimization algorithm was its algorithm, where different groups are generated for a cluster of particles and generates a high local optima avoidance. According to the theoretical concept BSA can be used to train SdA for generating a larger collection of datasets with different characteristics. According to author [2][29], the SdA was able to produce an accuracy of 94% with false positive rate to only 4% which is the best result compared to all other existing algorithms.

REFERENCES

1. Fang, Y., Huang, C., Liu, L. and Xue, M., "Research on Malicious JavaScript Detection Technology Based on LSTM," IEEE Access, vol.6, pp.59118-59125, 2018.
2. Wang, Y., Cai, W.D. and Wei, P.C., "A deep learning approach for detecting malicious JavaScript code," security and communication networks, vol.9, no.11, pp.1520-1534, 2016.

3. Hsu, F.H., Hwang, Y.L., Lee, C.H., Lin, C.J., Chang, K. and Huang, C.C., "A Cloud-based Protection approach against JavaScript-based attacks to browsers," Computers & Electrical Engineering, vol.68, pp.241-251, 2018.
4. Mao, J., Bian, J., Bai, G., Wang, R., Chen, Y., Xiao, Y. and Liang, Z., "Detecting Malicious Behaviors in JavaScript Applications," IEEE Access, vol.6, pp.12284-12294, 2018.
5. Mosaad, S., Abdelbaki, N. and Shosha, A.F., "A Postmortem Forensic Analysis for a JavaScript Based Attack," In Computer and Network Security Essentials, pp. 79-94, 2018.
6. Gupta, S. and Gupta, B.B., "A robust server-side javascript feature injection-based design for JSP web applications against XSS vulnerabilities," In Cyber Security, pp. 459-465, 2018.
7. Song, W., Huang, Q. and Huang, J., "Understanding JavaScript Vulnerabilities in Large Real-World Android Applications," IEEE Transactions on Dependable and Secure Computing, 2018.
8. Abdel Khalek, M. and Shosha, A., "Jsdes: An automated de-obfuscation system for malicious javascript," In proceedings of the International Conference on Availability, Reliability and Security, ACM, pp. 80, 2017.
9. Likarish, P., Jung, E. and Jo, I., "Obfuscated malicious javascript detection using classification techniques," In proceedings of International Conference on Malicious and Unwanted Software, pp. 47-54, 2009.
10. Hallaraker, O. and Vigna, G., "Detecting malicious javascript code in mozilla,"In proceedings of International Conference on Engineering of Complex Computer Systems, pp. 85-94, 2005.
11. Curtsinger, C., Livshits, B., Zorn, B. and Seifert, C., "Zozzle: Low-overhead mostly static JavaScript malware detection," In Proceedings of the usenix security symposium, pp. 3-3, 2011.
12. Li, B., Vadrevu, P., Lee, K.H. and Perdisci, R., "JSgraph: Enabling Reconstruction of Web Attacks via Efficient Tracking of Live In-Browser JavaScript Executions," In Annual Network and Distributed System Security, 2018.
13. C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert, "Zozzle: Fast and precise in-browser javascript malware detection," in proceedings of USENIX Conference on Security, pp. 3–3, 2011.
14. B. Burg, R. Bailey, A. J. Ko, and M. D. Ernst, "Interactive record/replay for web application debugging," in proceedings of ACM symposium on User interface software and technology, pp. 473–484, 2013.
15. C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifert, "Rozzle: De-cloaking internet malware," in proceedings of IEEE Symposium on Security and Privacy, 2012.
16. Mogren, O., "Malicious JavaScript detection using machine learning," learning, vol.10, no.11, pp.12, 2017.
17. Yaser Alosefer and Omer Rana, "Honeyware: a web-based low interaction client honeypot," In proceedings of Third International Conference on Software Testing, Verification, and Validation, IEEE, pp. 410–417, 2010.
18. YoungHan Choi, TaeGhyoon Kim, SeokJin Choi, and Cheolwon Lee, "Automatic detection for javascript obfuscation attacks in web pages through string pattern analysis," In proceedings of International Conference on Future Generation Information Technology, Springer, pp. 160–172, 2009.
19. Huda M, Abawajy J, Alazab M, Abdollalihian M, Islam R, Yearwood J, "Hybrids of support vector machine wrapper and filter based framework for malware detection," Future Generation Computer Systems, 2014.
20. Alazab M, "Profiling and classifying the behavior of malicious codes," Journal of Systems and Software, vol.100, pp.91–102, 2015.
21. AL-Taharwa IA, Lee H, Jeng AB, Wu K, Ho C, Chen S, "JSOD: JavaScript obfuscation detector," Security Comm. Networks, vol.8, pp.1092–1107, 2015.
22. Soska K, Christin N, "Automatically detecting vulnerable websites before they turn malicious," in USENIX Security, pp. 625–640, 2014.
23. Yuxin, D., Wei, D., Yibin, Z. and Chenglong, X., "Malicious code detection using opcode running tree representation," In proceedings of International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 616-621, 2014.
24. Canfora, G., Mercaldo, F. and Visaggio, C.A., "Malicious javascript detection by features extraction," e-Informatica Software Engineering Journal, vol.8, no.1, 2014.
25. Bansal, J.C., Sharma, H., Jadon, S.S. and Clerc, M., "Spider monkey optimization algorithm for numerical optimization," Memetic computing, vol.6, no.1, pp.31-47, 2014.
26. Meng, X.B., Gao, X.Z., Lu, L., Liu, Y. and Zhang, H., "A new bio-inspired optimisation algorithm: Bird Swarm Algorithm," Journal of Experimental & Theoretical Artificial Intelligence, vol.28, no.4, pp.673-687, 2016.
27. "Malicious JavaScript," https://www.cs.bham.ac.uk/research/projects/infotools/leakiest/examples/javascript.php, Accessed on March 2019.
28. Javascript Malware Collection," https://github.com/HynekPetrak/javascript-malware collection, Accessed on March 2019.
29. Ibrahim Aljarah, Hossam Faris, Seyedali Mirjalili, Nailah Al-Madi, Alaa Sheta and Majdi Mafarja, " Evolving neural networks using bird swarm algorithm for data classification and regression applications", Journal of Cluster Computing, Springer, Published on 15 Feb 2019

AUTHORS PROFILE

Dr. T Dhiliphan Rajkumar, Assistant Professor Department of Computer Science and Engineering, Kalasalingam University

Mr. Scaria Alex, Research Scholar, Department of Computer Science and Engineering, Kalasalingam University