International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Reuse Attack Prevention Through Randomization Traversal Algorithm with the Code Reduction Technique for Operating System Security

Prashant Johri[1], Madhavi Dhingra[2], Dilli Babu.M[3], Bipin Sule[4], Mr. Arvind Kumar Pandey[5], Dr Ankita Vitthal Karale[6]

Submitted: 02/11/2022 **Revised:** 10/01/2023 **Accepted:** 30/01/2023

Abstract: Computer security is considered the important end system for the complete network improvement in the host. Despite of the advancement in end-system security, the network is subjected to different malicious and network attacks in the larger network for the constant threat to data protection for data privacy and integrity. Security risk management comprises of two issues in the security of the larger network environment with the secured system environment. The code reuse attack is a severe threat in the computer network environment due to alteration in the complete network. This paper proposed a Preorder Randomization Traversal Algorithm (PreorderRTA) for the prevention of code reuse attacks. With the proposed PreorderRTA comprises randomized features for the generation of the keys in the computer network. The developed model concentrated on code reuse attack detection and prevention. The proposed PreorderRTA model achieves a higher detection rate for the rootkit, worms and Viruses in the system compared with the existing technique. The proposed PreorderRTA achieves the detection rate of 99.34% while the existing approaches achieves below 90%.

Keywords: *Computer Security, Randomization Traversal Algorithm, Preorder, Attack Reuse, detection rate.*

1. Introduction

The continuing growth of Internet-connected devices will drive malware authors to use either unpatched system or software vulnerabilities as a way to point a full-blown attack. All modern computer systems have a piece of most important system software, entitled Operating System (OS) or kernel which basically runs on top of the hardware that assigns the necessary system resources and supervises the execution of each application within the system [1]. The OS as a whole consists of the kernel and may comprise other relevant programs for providing necessary services for each incoming request. More importantly, the kernel which acts as a part of the OS is responsible for many functions such as system calls, manage memory, and interrupts, exceptions, etc, [2].

Analysis of malicious code attack risk enable defenders to model attack reasoning and scenarios about the relationship between dependencies between attack paths [3]. By generating comprehensive models about different attack scenarios, it is possible to design a specific quantitative measurement technique for attack risks coupled with a network settings. Then the outcome can be later used to improve the security configuration of the network. However, such a quantitative attack measurement model technique considers several technical design challenges [4]. First, the outcome of a quantitative measurement model must have clear semantics which could permit for the development of deterministic algorithm for generating the expected results. Second, the quantitative analysis of different security risks must be able to generate useful conclusions even in the absence of sensitive sampling security data. Finally, very large networks are usually highly dynamic in nature. Generating attack graphs are normally a well-known method which can offer the expected information about attack scenarios and its dependencies of a malicious executable [5]. Although there were many quantitative assessment models exist to depict attack scenarios of a specific network, this kind of static analysis suffers from several limitations to handle current state of art on security [6].

[1] *Professor, Department of Computer Application, Galgotias University, Greater Noida, Uttar Pradesh, India.*
prashant.johri@galgotiasuniversity.edu.in,
0000-0001-8771-5700
[2] *Computer science and engineering, Amity University Madhya Pradesh, Maharajpura Dang, Gwalior (MP)-474005, India.*
madhavi.dhingra@gmail.com
0000-0002-9883-7620
[3] *Assistant professor, Department of Information Technology. Panimalar Engineering College, Chennai, Tamil Nadu, India.*
deenshadilli@gmail.com
[4] *Professor, Department of Computer Engineering, Vishwakarma Institute of Information Technology, Pune, India.*
bipin.sule@vit.edu
0000-0003-1409-2156
[5] *Assistant Professor, School of Engg. &IT, ARKA JAIN University, Jamshedpur, Jharkhand, India.*
arvind.p@arkajainuniversity.ac.in
0000-0001-5294-0190
[6] *Associate professor, computer engineering, Sandip Institute of Technology and Research Centre, Nashik, India.*
ankita.karale9@gmail.com

An attack graph model offers only a partial interpretation about attack scenarios of a network. The existing methods lack of hard theoretical foundations. Existing design approaches to quantify network attack graphs fail to consider the dynamic nature of a networked environment [7]. For larger networks, quantitative analysis of attack graphs fall under NP-complete problem which is always non-trivial. Dynamic analysis based defense mechanisms have been developed to overcome these issues. Such approach works by utilizing the execution flow of legitimate applications to discover the presence of a malware [8]. However, attacks such as mimicry and shadow attack weaken the dynamic malware analyser. Most OS kernels often enforce only a limited access restriction on the application program permitted to carry out its execution. As a result, malicious software program which runs as a stand-alone process abuse system resources for its execution. Once installed on the victim computer, malicious executable can freely run to execute privileges associated with the current user account running the process [9]. This may lead to affect the entire network. Therefore, securing both user-mode and kernel-mode of an end-system is very important. Dynamic malware analysis based user-mode malware detection techniques and anti-malware detection tools have been studied and developed. As advanced malware incorporate rootkit techniques to evade detection, different algorithms have also been developed to optimize the malware detection system. But malware attacks that target kernel level compromise is an issue. Hence, this paper focused on the randomized model for the code reuse attack in the computer network security. The proposed model is stated as PreorderRTA for the computation of the constructed tree and pre order values.

2. Related Works

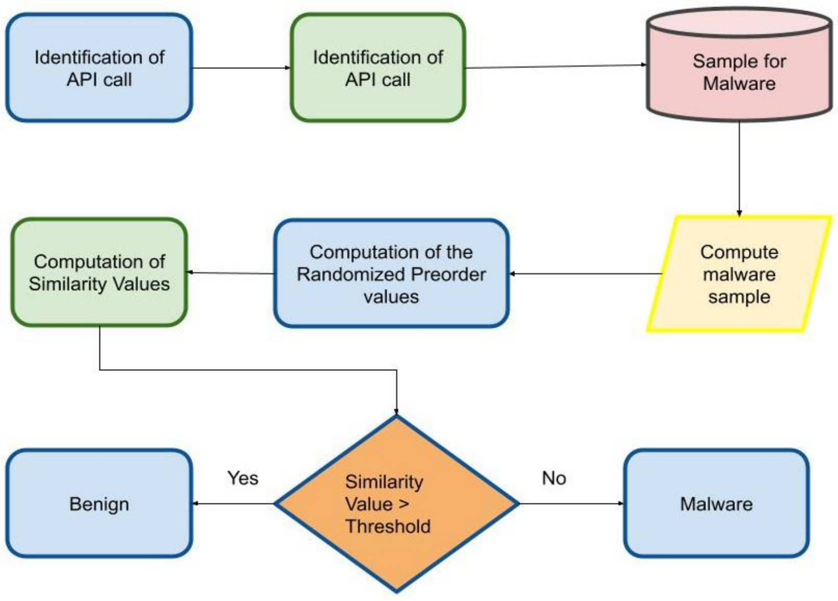
In [10] proposed a system named PAIDS (Proximity-Assisted IDS) with the goal of identifying the new and fast propagating worms. PAIDS has been trying to obtain enhanced performance by working collaboratively with existing anomaly-based IDS. Their approach assumes that during the worm-propagation starting phase, the infected victim hosts can be grouped based on IP address and DNS used. In [11] proposed an outlier based intrusion detection approach to detect cyber intrusions. A specific dataset was taken to measure the presence of intrusions by using the neighborhood outlier factor method. The use of limited dataset and training model are the two weakness of this approach.

In [12] presented an approach that monitors malicious activities at the network to prevent known and first-hand attacks using unsupervised neural networks. This real hierarchical intrusion time solution uses Principal Components Analysis neural nets to avoid the limitations of sing lelevel structures. It relies on finding difference among data attributes which are classified into environmental attributes and indicator attributes. This method detects anomalous if any deviation in the predefined value of indicator attributes. However, it does not consider environmental attributes in few cases. The precision of this method precisely depends on its learning phase. In [13] proposed a rough set theory and then a k-NN classifier mechanism to determine network intrusions with the intention of increasing detection rate of the system and minimal false alarm rate.

In [14] aimed at constructing a model which produces fuzzy association rules with reference to classifiers and use them for detecting general network intrusions. The fuzzy sets theory provides an effective way to categorize different classes of normal and/or anomalous. A training dataset that belongs to a particular type is validated by using matching parameters produced by the proposed approach. If the compatibility of a test sample falls the predefined threshold, then it is considered as anomalous. In [15] presented a network intrusion detection system that rely on fuzzy rules to recognize the occurrence of specific or general exceptional network patterns. However, training instances play a vital role to decide the detection accuracy of the system. The devoted to the development of network intrusion detection system that uses genetic algorithms to construct detection rules. A chromosome of individual genes mapped to various aspects such as the root-user attempt, type of service attempt to use, or logged in or not. The author concludes that malware attacks that are common can be traced easily compared to unusual characteristics.

3. Randomized Recursive Traversal Algorithm for the Code Reuse Attack

With the proposed model code reuse attack is prevented using the PreorderRTA integrated with the Recursive Traversal Algorithm. It is assumed that most malicious malwares are developed by inheriting characteristics from its previous version. For example, the various versions of TDSS rootkit are: TDL1 which was designed to load and run at the time of booting the operating system which was designed with the intention of infecting system drivers. TDL2 appears to be same as TDL1. However, it includes different names with random string and also imports new technique to avoid detection and removal. In order to obtain control over the victim computer, TDL3 patches the disk controller driver. Some features of TDL2 were updated to make detection and removal more difficult. The aim of TDL4 variant is the same as that of TDL3, but patched master boot record to make infection of computers with 64-bit processor. The overall flow diagram of the PreorderRTA approach is given figure 1.



Identification of API call
Identification of API call
Sample for Malware
Compute malware sample
Computation of the Randomized Preorder values
Computation of Similarity Values
Similarity Value > Threshold
Yes
Benign
No
Malware

Fig. 1: Flow chart of Preorder RTA

A directed edge (u,v) in E represents a function call of the program, u→v. PreorderRTA attempts to discover the malicious code attacks which integrate API detouring technique to launch their illegal activities using an API call graph approach. As API function call is a finite set of sequence of invocations with ordered parameters and also they communicate with the use of handles (Unique identifier), PreorderRTA can identify all necessary resources to construct an API call graph for a corresponding function call as presented in figure 2.

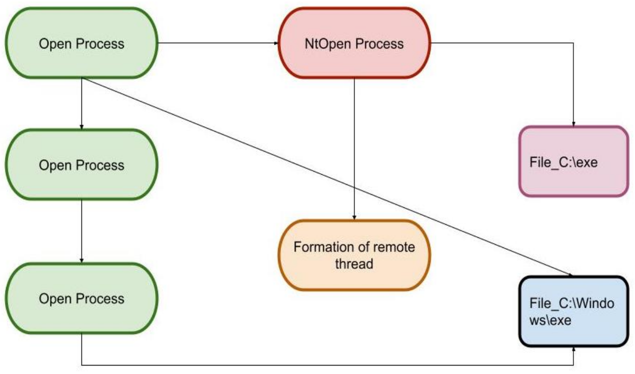


Fig. 2: Randomized Traversal Algorithm

The proposed randomized PreorderRTA uses the randomized code for the prevention of the code reuse attack. The proposed PreorderRTA comprises of the two graph call (GC) and Approach graph (AG). The determination of the subgraph approximation is computed as in equation (1)

$$S_a = Paramax_G . sim(GC, AG) \quad (1)$$

S_a - Maximize {Simval(CG,MG) = 1} then CG is isomorphic to MG.

where CG⊆ G and sim(CG,MG) represents the level of matching between CG and MG.

Algorithm 1: PreorderRTA for the code reuse attack prevention

```
/* Algorithm for Generating dependent graph API */
Begin
        Compute the all executable function calls
            Select the function from the API
             If
                  Compute the API call in the transversal algorithm
    {
 Compute the node ← name of the function;
Get parameters(name of the function);
attack.node ← recursive(analysis of pointer);
// Randomization function call
  Any point generate graph ();
}
End if
Store the attack database
End
```

The Preorder assignment for the Recursive Traversal algorithm is presented. The proposed PreorderRTA compute the features in the different malware in the computer network security.

Consider the item set number for the attack prevention as $I = \{x_1, x_2, \ldots\ldots, x_n\}$ fro the non empty subset I in the itemset. The computed randomized value with the PreorderRTA compute the characteristics of the code reuse attack denoted as $TID = \{t_1, t_{2,} \ldots\ldots, t_n\}$ for every order as O incorporates the itemset value those comprises of the I. For the each randomized value in the attack α count of preorder value is computed with α as TID_α support the α set.

The itemset for the code reuse attack is evaluated based in the items α as presented in equation (2)

$$|\alpha| = |\{x_i / x_i \in \alpha\}| \quad (2)$$

Definition 1: Frequent itemset in the code reuse attack

For every dataset data in the α itemset depends on the $\frac{|TID_\alpha|}{|TID|} \geq \sigma$, in this α in TID supports the $\frac{|TID_\alpha|}{|TID|}$ with computation of the threshold value as $0 \leq \sigma \leq 1$.

It is assumed that proposed PreorderRTA evaluated with the supp (α) for the pattern itemset in the data. The pattern in the data is computed as α and α' for the sub-pattern if $\alpha \complement \alpha'$ of α.

Definition 2: Frequent Pattern

The pattern of the frequent is closed only it has super frequent pattern those are same set.

Lemma 1: F or itemsets α and α' if $\alpha \complement \alpha', then\, TID_\alpha \in TID_{\alpha'}$

i.e., if a = {m,n,s} and a' = {m, n, s, p, q}.let TID_α= {tl, t2,.... tio} then it may be TIDa' ={ ti, t4, ts}.Hence a smaller itemset have lot more chances of occurring in transactions than a larger one.

From lemma 1, it can be known that for a pattern a, TIDa = flpcaTIDp.

The important characteristics of prodigious pattern computed in the PreorderRTA

1. The length of the code sequence are natural
2. The minimal number of variables are considered in the database.
3. The generated randomized code is eliminated from the pattern without any support set. The larger pattent set are computed for its prominent size for the observed robustness.

Today, a malware writer can develop a malware by updating new features and techniques with its predecessor rather than coding from scratch. This information can help the defender to reduce the complexity of considering all kinds of addiction while inquiry the approach graph. The objective of the post-processing stage is to generate a subgraph of the data graph by referring the approach graph. The randomized code comprises of the edge graph dependencies with the location of the recursive traversal algorithm with the edge node as Call graph (CG) and Approach Graph (AG). The proposed PreorderRTA is simplified with the matching subgraph for the exact identification of the code reuse attack. Through randomization process the subgraph are computed for the CG and AG matching to compute the similarity normalized value as 0 and 1.

4. Experimental Setup

The evaluation results are obtained by conducting simulation experiments for comparing the proposed PreorderRTA method against existing methods using some common parameters such as true positive, false positive, detection rate, and accuracy rate. These parameters are defined and calculated below.

True Positive (TP) occurs when a malware is correctly detected as a malware.

False Positive (FP) occurs when a legitimate sample is caught to be a malware.

Detection Rate $DR = \frac{TP}{TP+FN}$

Accuracy Rate $AR = \frac{TP+TN}{TP+TN+FP+FN}$

False Positive Rate $FPR = \frac{FP}{TN+FP}$

• Receiver Operating Characteristic (ROC) curve – It is a two dimensional graph used to visualize the performance of the proposed approach by plotting TPR on the X axis against FPR on the Y axis.

The proposed PreorderRTA model concentrated on the computation of the accuracy and detection for the conduction experiment to prevent code reuse attack. In the table 1 presented the comparative examination of the minimum and maximal similarity value for every calculated attack values. The table 1 the average similarity values are computed for the detection capability in the every group compared with the proposed PreorderRTA values.

Table 1: Comparison of Maximal and Minimal Value

Technique	**Maximum SV**	**Minimum SV**	**Number of malware samples not detected**
Family: Rootkits			
[11]	87.74	31.28	3
[12]	92.68	40.56	1

[13]	97.73	58.62	2
Proposed PreorderRTA	98.23	58.08	1
Family: Worms			
[11]	82.21	12.18	1
[12]	88.10	19.23	1
[13]	81.69	12.01	2
Proposed PreorderRTA	92.08	42.34	0
Family: Trojans			
[11]	80.86	31.90	1
[12]	94.79	18.64	1
[13]	79.32	33.83	2
Proposed PreorderRTA	90.29	43.56	0

Table 1 shows that the proposed Proposed PreorderRTA method has achieved an average of 93.20 similarity value. Among all, the [11] method failed to detect 8 malware samples in total produces lowest performance. Whereas in [12] method undetected only 6 malware samples. The [13] method failed to detect an average of one malware sample but Proposed PreorderRTA undetected only one malware sample and surpasses the method proposed in [13]. Another important consideration of PreorderRTA approach is to evaluate its effective against the detection rate benign samples. The figure 3 provides the false positive rate measured for the proposed PreorderRTA model.

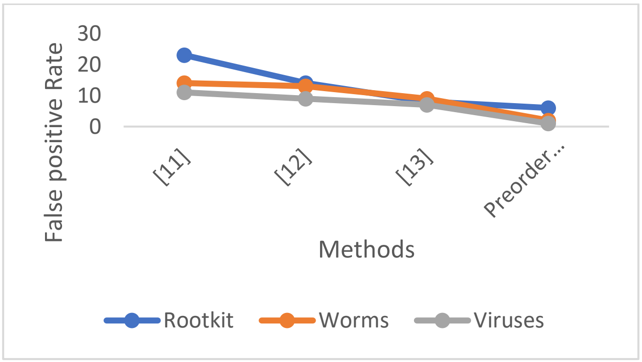


Fig. 3: Comparison of False Positive Rate

The figure 4 provides the comparative analysis of the proposed PreorderRTA model detection rate for the Trojan and Worm. The rootkit group analysis stated that for the worm and Trojan the detection rate is measured as 100% and for the rootkit the detection rate is measured as 97.68%. The existing technique achieves the detection rate of 97.59% for the dataset.

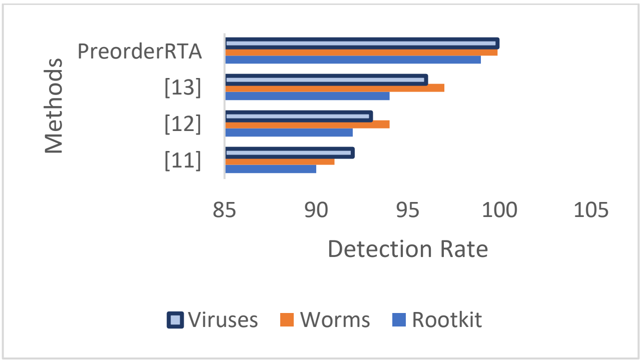


Fig. 4: Comparison of Detection Rate

Figure 4 shows the overall accuracy rate of Proposed PreorderRTA approach and other existing techniques against all 250 malware samples. The Proposed PreorderRTA approach achieved the highest accuracy rate of 98 % against all 50 rootkit malware samples, but achieved an average of 100 % in the Trojans and worms groups. In [13] the method has achieved next best result than other comparable techniques with an average of 98.21% AR. Moreover, the methods proposed in [11] and [12] have achieved an average accuracy rate of 95.09 % and 96.53 % respectively.

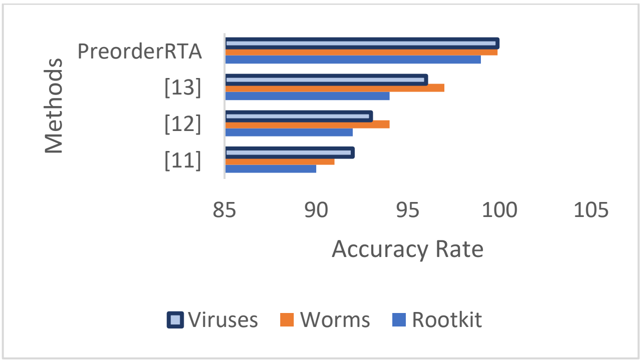


Fig. 5: Comparison of Accuracy Rate

From the figure 5 above experimental results and discussion, it is clear that Proposed PreorderRTA approach outperforms than the rest of discussed existing approaches in all aspects. A game-theoretic approach is also used to ensure the optimization of resources consumed by the PreorderRTA approach. The game theoretic model dynamically selects a specific API targeted by stealthy rootkit malware based on the expected attack scenario.

5. Conclusion

In the API environment the detection rate of the code is discovered the API hook for the suspicious system with the proposed PreorderRTA. The proposed model comprises of the randomized model for the code reuse attack through call graph and mapping technique. With the proposed PreorderRTA similarity rate is computed. The experimental analysis stated that proposed PreorderRTA achieves the higher detection rate of 98% - 100% those are significantly higher than the existing model. The proposed PreorderRTA model achieves the ~3% improved performance than the existing model.

Reference

[1] Mishra, S., & Polychronakis, M. (2021, April). SGXPecial: Specializing SGX Interfaces against Code Reuse Attacks. In *Proceedings of the 14th European Workshop on Systems Security* (pp. 48-54).

[2] Lin, K., Xia, H., Zhang, K., & Tu, B. (2021, September). AddrArmor: An Address-based Runtime Code-reuse Attack Mitigation for Shared Objects at the Binary-level. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)* (pp. 117-124). IEEE.

[3] Wang, J., Zhang, Z., Ma, B., Yao, Y., & Ji, X. (2021, May). Research on SSTI attack defense technology based on instruction set randomization. In *2021 2nd International Conference on Artificial Intelligence and Information Systems* (pp. 1-5).

[4] Nikolaev, R., Nadeem, H., Stone, C., & Ravindran, B. (2022, February). Adelie: continuous address space layout re-randomization for Linux drivers. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (pp. 483-498).

[5] Potteiger, B., Cai, F., Zhang, Z., & Koutsoukos, X. (2022). Data space randomization for securing cyber-physical systems. *International Journal of Information Security*, *21*(3), 597-610.

[6] Schloegel, M., Blazytko, T., Basler, J., Hemmer, F., & Holz, T. (2021, October). Towards Automating Code-Reuse Attacks Using Synthesized Gadget Chains. In *European Symposium on Research in Computer Security* (pp. 218-239). Springer, Cham.

[7] Novković, B. (2021). A Taxonomy of Defenses against Memory Corruption Attacks. In *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)* (pp. 1196-1201). IEEE.

[8] Yoon, H., & Lee, M. (2022). SGXDump: A Repeatable Code-Reuse Attack for Extracting SGX Enclave Memory. *Applied Sciences*, *12*(15), 7655.

[9] Shrivastava, R. K., Singh, S. P., Hasan, M. K., Islam, S., Abdullah, S., & Aman, A. H. M. (2022). Securing Internet of Things devices against code tampering attacks using Return Oriented Programming. *Computer Communications*, *193*, 38-46.

[10] Xu, S., & Wang, Y. (2022). Defending against Return-Oriented Programming attacks based on return instruction using static analysis and binary patch techniques. *Science of Computer Programming*, *217*, 102768.

[11] Ying, H., Zhou, H., Degani, A., & Sacks, R. (2022). A two-stage recursive ray tracing algorithm to automatically identify external building objects in building information models. *Computer-Aided Civil and Infrastructure Engineering*, *37*(8), 991-1009.

[12] Huang, X., Yan, F., Zhang, L., & Wang, K. (2021). Honeygadget: A deception based approach for detecting code reuse attacks. *Information Systems Frontiers*, *23*(2), 269-283.

[13] Lee, S., Kang, H., Jang, J., & Kang, B. B. (2021). Savior: Thwarting stack-based memory safety violations by randomizing stack layout. *IEEE Transactions on Dependable and Secure Computing*.

[14] Zuo, Z., Fang, Y., Huang, Q., Liao, Y., Wang, Y., & Wang, C. (2021, October). Derivation and Formal Proof of Binary Tree Depth Non-Recursive Algorithm. In *2021 5th International Conference on Communication and Information Systems (ICCIS)* (pp. 191-196). IEEE.

[15] Zhang, C., Bonifati, A., Kapp, H., Haprian, V. I., & Lozi, J. P. (2022). A Reachability Index for Recursive Label-Concatenated Graph Queries. *arXiv preprint arXiv:2203.08606*.