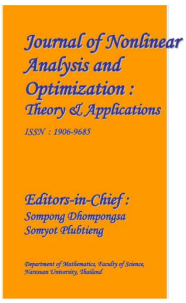
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Issue. 1, No.15 : 2024
ISSN : **1906-9685**



SUBSTANTIAL AUTHORIZATION RECOGNITION IN MACHINE LEARNING-DRIVEN DETECTION OF MALICIOUS

Dr.M.Swapna , Assistant professor CSE, Vaagdevi College of Engineering(Autonomous),India
Nandre shravya,UG Student ,CSE, Vaagdevi College of Engineering(Autonomous),India
Mohammad adil ahmed ,UG Student ,CSE, Vaagdevi College of Engineering(Autonomous),India
Pastham shiva raj ,UG Student ,CSE, Vaagdevi College of Engineering(Autonomous),India
Poreddy sumanth reddy ,UG Student ,CSE, Vaagdevi College of Engineering(Autonomous),India

ABSTRACT

The alarming growth rate of malicious apps has become a serious issue that sets back the prosperous mobile ecosystem. A recent report indicates that a new malicious app for Android is introduced every 10 s. To combat this serious malware campaign, we need a scalable malware detection approach that can effectively and efficiently identify malware apps. Numerous malware detection tools have been developed, including system-level and network-level approaches. However, scaling the detection for a large bundle of apps remains a challenging task. In this paper, we introduce Significant Permission Identification (SigPID), a malware detection system based on permission usage analysis to cope with the rapid increase in the number of Android malware. Instead of extracting and analyzing all Android permissions, we develop three levels of pruning by mining the permission data to identify the most significant permissions that can be effective in distinguishing between benign and malicious apps. SigPID then utilizes machine-learning-based classification methods to classify different families of malware and benign apps. Our evaluation finds that only 22 permissions are significant. We then compare the performance of our approach, using only 22 permissions, against a baseline approach that analyzes all permissions. The results indicate that when a support vector machine is used as the classifier, we can achieve over 90% of precision, recall, accuracy, and F-measure, which are about the same as those produced by the baseline approach while incurring the analysis times that are 4-32 times less than those of using all permissions. Compared against other state-of-the-art approaches, SigPID is more effective by detecting 93.62% of malware in the dataset and 91.4% unknown/new malware samples.

1. INTRODUCTION

Android is currently the most used smart-mobile device platform in the world, occupying 85% of market share . As of now, there are nearly 3 million apps available for downloading from Google Play, and more than 65 billion downloads to date [1] . Unfortunately, the popularity of Android also spurs interests from cyber-criminals who create malicious apps that can steal sensitive information and compromise mobile systems. Unlike other competing smart-mobile device platforms, such as iOS, Android allows users to install applications from unverified sources such as third party app stores and file sharing websites. The malware infection issue [2] has been so serious that a recent report indicates that 97% of all mobile malware target Android devices . In 2016 alone, over 3.25 million new malicious Android apps have been uncovered. This roughly translates to an introduction of a new malicious Android app every 10 seconds . These malicious apps are created to perform different types of attacks in the form of trojans, worms, exploits, and viruses. Some notorious malicious apps have more than 50 variants [3], which makes it extremely challenging to detect them all . To address these elevating security concerns, researchers and analysts have used various approaches to develop Android malware detection tools.

For example, RISKRANKER [4] utilizes static analysis to discover malicious behaviors in Android apps. However, static analysis approaches generally assume more behaviors are possible than actually would be, which may lead to a large number of false positives. To improve the analysis accuracy, researchers have also proposed various dynamic analysis methods to capture real-time execution context. For example, TAINTDROID [5] dynamically tracks multiple sensitive data source simultaneously using tainting analysis. However, dynamic analysis approaches, in general, need adequate input suites to sufficiently exercise execution paths. As we can use cloud computing to satisfy those requirements, the privacy concerns then become potential problem . Petra et al. show that there is another broad range of anti-analysis techniques can be employed by advanced malware to successfully evade dynamic analysis based malware detection [6]. Recently, more efforts have been spent on analyzing apps' behavioral data both in the Android system and other online data process system . The apps' requested permissions have been used to enforce least.

DREBIN [7] combines static analysis and machine learning techniques to detect Android malware. The experimental result shows that DREBIN can achieve high detection accuracy by incorporating as many features as possible to aid detection. However, using more features leads to increased modeling complexity, which increases the computational overhead of their system. Considering the large amount of new malicious apps, we need a detection system that can operate efficiently to identify these apps. Google also identifies 24 permissions out of the total of more than 300 permissions as “dangerous” [8] . At first glance, the list of dangerous permissions can be used as a guideline to help identify malicious applications [9]. Yet, as will be shown in this paper, using just this list to identify malware still yields suboptimal detection effectiveness. In this paper, we present SIGPID [10], an approach that extracts significant permissions from apps, and uses the extracted information to effectively detect malware using supervised learning algorithms. The design objective of SIGPID is to detect malware efficiently and accurately. As stated earlier, the number of newly introduced malware is growing at an alarming rate [11]. As such, being able to detect malware efficiently would allow analysts to be more productive in identifying and analyzing them. Our approach analyzes permissions and then identifies only the ones that are significant in distinguishing between malicious and benign apps. Specifically, we propose a multi-level data pruning approach including permission ranking with negative rate, permission mining with association rules and support based permission ranking to extract significant permissions strategically.

Then, machine learning based classification algorithms [12]are used to classify different types of malware and benign apps. The results of our empirical evaluation show that SIGPID can drastically reduce the number of permissions that we need to analyze to just 22 out of 135 (84% reduction), while maintaining over 90% malware detection accuracy and Fmeasure when Support Vector Machine (SVM) [13] is used as the classifier. We also find that the number of significant permissions identified by our approach is lower than the number of “dangerous” permissions identified by Google . Moreover, only 8 permissions jointly appear on our list and their list.This is because, as a data-driven approach, SIGPID dynamically determines significant permissions based on actual usage by the applications instead of statically defining dangerous permissions based on their intended services. This fundamental difference allows our approach

to detect more malware than the approach that uses the dangerous list alone. To show the generality of this approach, we also test SIGPID with 67 commonly used supervised algorithms and find that it maintains very high accuracy with all these algorithms.

Furthermore, we compare the accuracy and running [14]time performance of our approach against two state-of-theart approaches, DREBIN, and existing virus scanners. Again, we find that our approach can detect more malware samples than the other approaches with significantly less overhead. In summary, our paper makes the following contributions .We develop SIGPID, an approach that identifies an essential subset of permissions (significant permissions) that can be used to effectively identify Android malware. By using our technique, the number of permissions that needs to be analyzed is reduced by 84%. We evaluate the effectiveness of our approach using only a fifth of the total number of permissions in Android. We find that SigPID [15] can achieve over 90% in precision, recall, accuracy, and F-measure. These results compare favorably with those achieved by an approach that uses all 135 permissions as well as just the dangerous permission list. Compare with other state-of-the-art malware detection approaches, we find that SIGPID is more effective by detecting 93.62% of malicious apps in the data set and 91.4% unknown malware.To show that the approach can work generically with a wide range of supervised learning algorithms, we apply SIGPID with 67 commonly used supervised learning algorithms and a much larger dataset (5,494 malicious and 310,926 benign apps). We find that 55 out of 67 algorithms can achieve F-measure of at least 85%, while the average running time can be reduced by 85.6% compared with the baseline approach.

2. LITERATURE SURVEY

Smartphone sales have recently experienced explosive growth. Their popularity also encourages malware authors to penetrate various mobile marketplaces with malicious applications (or apps). These malicious apps hide in the sheer number of other normal apps, which makes their detection challenging. Existing mobile anti-virus software are inadequate in their reactive nature by relying on known malware samples for signature extraction. In this paper, we propose a proactive scheme to spot zero-day Android malware. Without relying on malware samples and their signatures, our scheme is motivated to assess potential security risks posed by these untrusted apps. Specifically, we have developed an automated system called RiskRanker

[16] to scalably analyze whether a particular app exhibits dangerous behavior (e.g., launching a root exploit or sending background SMS messages). The output is then used to produce a prioritized list of reduced apps that merit further investigation. When applied to examine 118,318 total apps collected from various Android markets over September and October 2011, our system takes less than four days to process all of them and effectively reports 3281 risky apps. Among these reported apps, we successfully uncovered 718 malware samples (in 29 families) and 322 of them are zero-day (in 11 families). These results demonstrate the efficacy and scalability of RiskRanker to police Android markets of all stripes.

Most existing malicious Android app detection approaches rely on manually selected detection heuristics, features, and models. In this paper, we describe a new, complementary system, called DroidMiner [17], which uses static analysis to automatically mine malicious program logic from known Android malware, abstracts this logic into a sequence of threat modalities, and then seeks out these threat modality patterns in other unknown (or newly published) Android apps. We formalize a two-level behavioral graph representation used to capture Android app program logic, and design new techniques to identify and label elements of the graph that capture malicious behavioral patterns (or malicious modalities). After the automatic learning of these malicious behavioral models, DroidMiner can scan a new Android app to (i) determine whether it contains malicious modalities, (ii) diagnose the malware family to which it is most closely associated, (iii) and provide further evidence as to why the app is considered to be malicious by including a concise description of identified malicious behaviors. We evaluate DroidMiner using 2,466 malicious apps, identified from a corpus of over 67,000 third-party market Android apps, plus an additional set of over 10,000 official market Android apps. Using this set of real-world apps, we demonstrate that DroidMiner achieves a 95.3% detection rate, with only a 0.4% false positive rate. We further evaluate DroidMiner's ability to classify malicious apps under their proper family labels, and measure its label accuracy at 92%.

Mobile malware attempts to evade detection during app analysis by mimicking security-sensitive behaviors of benign apps that provide similar functionality (e.g., sending SMS messages), and suppressing their payload to reduce the chance of being observed (e.g., executing only its payload at night). Since current approaches focus their analyses on the types of security-

sensitive resources being accessed (e.g., network), these evasive techniques in malware make differentiating between malicious and benign app behaviors a difficult task during app analysis. We propose that the malicious and benign behaviors within apps can be differentiated based on the contexts that trigger security-sensitive behaviors, i.e., the events and conditions that cause the security-sensitive behaviors to occur. In this work, we introduce AppContext [18], an approach of static program analysis that extracts the contexts of security-sensitive behaviors to assist app analysis in differentiating between malicious and benign behaviors. We implement a prototype of AppContext and evaluate AppContext on 202 malicious apps from various malware datasets, and 633 benign apps from the Google Play Store. AppContext correctly identifies 192 malicious apps with 87.7% precision and 95% recall. Our evaluation results suggest that the maliciousness of a security-sensitive behavior is more closely related to the intention of the behavior (reflected via contexts) than the type of the security-sensitive resources.

The emergence of malicious apps poses a serious threat to the Android platform. Most types of mobile malware rely on network interface to coordinate operations, steal users' private information, and launch attack activities. In this paper, we propose an effective and automatic malware detection method using the text semantics of network traffic. In particular, we consider each HTTP flow generated by mobile apps as a text document, which can be processed by natural language processing to extract text-level features. Then, we use the text semantic features of network traffic to develop an effective malware detection model. In an evaluation using 31 706 benign flows and 5258 malicious flows, our method outperforms the existing approaches, and gets an accuracy of 99.15%. We also conduct experiments to verify that the method is effective in detecting newly discovered malware, and requires only a few samples to achieve a good detection result. When the detection model is applied to the real environment to detect unknown applications in the wild, the experimental results show that our method performs significantly better than other popular anti-virus scanners with a detection rate of 54.81%. Our method also reveals certain malware types that can avoid the detection of anti-virus scanners. In addition, we design a detection system on encrypted traffic for bring-your-own-device enterprise network, home network, and 3G/4G mobile network. The detection model is integrated into the system to discover suspicious network behaviors.

A recent report indicates that there is a new malicious app introduced every 4 seconds.

This rapid malware distribution rate causes existing malware detection systems to fall far behind, allowing malicious apps to escape vetting efforts and be distributed by even legitimate app stores. When trusted downloading sites distribute malware, several negative consequences ensue. First, the popularity of these sites would allow such malicious apps to quickly and widely infect devices. Second, analysts and researchers who rely on machine learning based detection techniques may also download these apps and mistakenly label them as benign since they have not been disclosed as malware. These apps are then used as part of their benign dataset during model training and testing. The presence of contaminants in benign dataset can compromise the effectiveness and accuracy of their detection and classification techniques. To address this issue, we introduce PUDROID (Positive and Unlabeled learning-based malware detection for Android) to automatically and effectively remove contaminants from training datasets, allowing machine learning based malware classifiers and detectors to be more effective and accurate. To further improve the performance of such detectors, we apply a feature selection strategy to select pertinent features from a variety of features. We then compare the detection rates and accuracy of detection systems using two datasets; one using PUDROID [18] to remove contaminants and the other without removing contaminants. The results indicate that once we remove contaminants from the datasets, we can significantly improve both malware detection rate and detection accuracy.

3. PROBLEM STATEMENT

IBM QRadar [19] is a Security Information and Event Management (SIEM) [19] system that combines log management, threat detection, and incident response into a single platform. It collects and correlates data from various sources, including logs, network traffic, and endpoints. QRadar uses predefined rules, anomaly detection, and machine learning to identify potential security incidents and unauthorized access. The system provides real-time visibility into the organization's security posture and enables security teams to respond to threats effectively.

Few Disadvantages of the Existing System are:

- False Positives and Negatives: Existing systems often struggle with balancing false positives (incorrectly flagging legitimate activities as malicious) and false negatives (failing

to detect actual threats), which can lead to inefficient use of resources and potential security gaps.

- Resource Intensiveness: Implementing and maintaining machine learning-driven systems can be resource-intensive in terms of computational power, data requirements, and ongoing updates, making them costly and complex to manage.
- Adversarial Vulnerabilities: These systems are susceptible to adversarial attacks, where malicious actors manipulate data to deceive the system. This vulnerability can undermine the system's effectiveness and reliability in real-world security scenarios.

4. PROPOSED SYSTEM

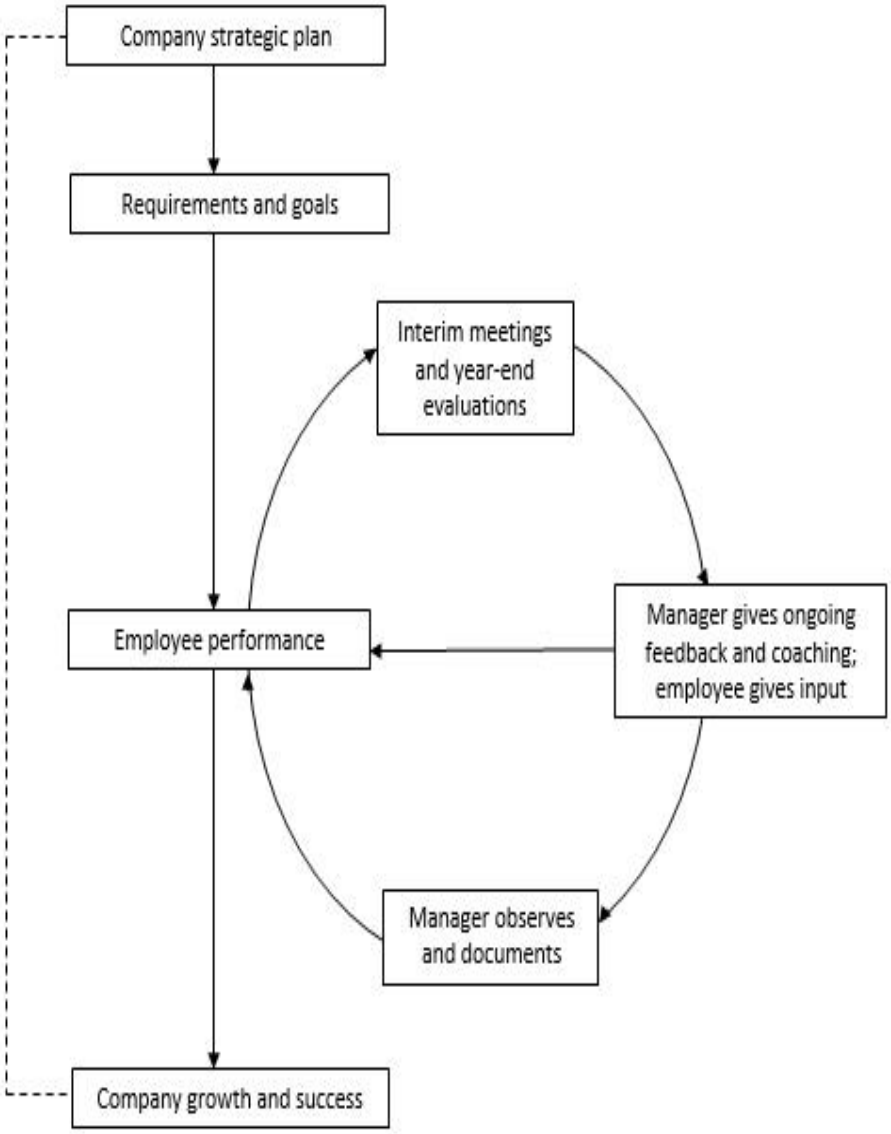
A proposed system could be a deep learning-based access control system designed to enhance substantial authorization recognition [20]. This system would use deep neural networks to analyze user behavior patterns and authorization requests. It would establish a baseline of normal behavior for users and entities within a network and continuously monitor for deviations from this baseline. When anomalies or suspicious access requests are detected, the system could trigger alerts or automatically enforce access controls to prevent unauthorized access. The advantage of a deep learning-based approach is its ability to adapt and learn from evolving threats, making it more effective in detecting novel,

Few Advantages of Proposed System are:

- The proposed deep learning-based access control system offers several advantages for substantial authorization recognition in machine learning-driven detection of malicious activities.
- It leverages deep neural networks to continuously analyze and adapt to user behavior patterns and authorization requests, providing a dynamic and context-aware approach to security.
- This adaptability enhances the system's ability to detect novel and sophisticated threats effectively, reducing false positives and false negatives. Furthermore, its deep learning foundation enables it to learn from evolving threat landscapes, improving over time without extensive manual rule management.

This not only enhances the accuracy of threat detection but also reduces the burden on security teams, making it a more efficient and proactive solution for safeguarding critical assets and data.

5. SYSTEM ARCHITECTURE



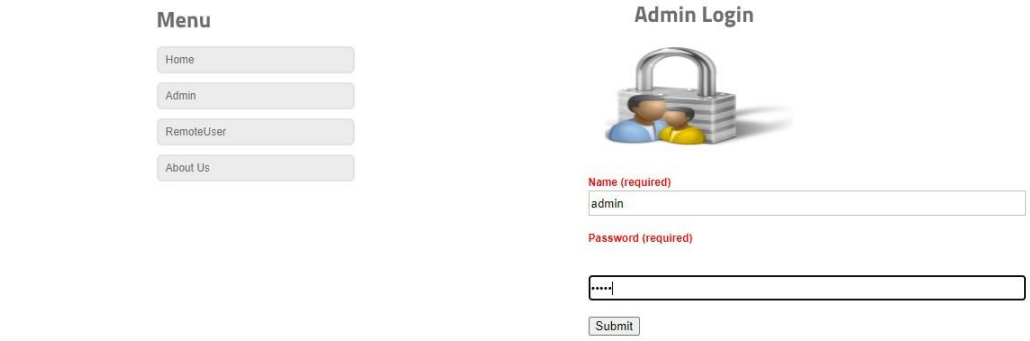
6. IMPLEMENTATION

6.1 Admin module: The admin module in a Java project serves as the backbone for managing system functionality, user roles, and data administration. It typically includes features such as user authentication, authorization, and access control, allowing administrators to regulate

user permissions and privileges. Through an intuitive user interface, administrators can add, edit, or remove users, configure system settings, and monitor system activities. Additionally, the admin module facilitates the management of various resources, such as files, databases, and configurations, ensuring smooth operation and security compliance. With robust error handling and logging mechanisms, it provides administrators with insights into system performance and facilitates troubleshooting.

6.2 Remote User: The Remote User Module in a Java project facilitates remote user authentication and management. It typically includes functionalities for user login, registration, password reset, and session management, all accessible over a network. This module utilizes Java's networking libraries to establish secure connections between client and server, ensuring data integrity and confidentiality. Employing encryption algorithms, it safeguards sensitive user information during transmission. Additionally, it often integrates with authentication protocols like OAuth or LDAP for enhanced security and interoperability. Through standardized interfaces and protocols, the Remote User Module enables seamless integration into diverse Java-based applications, offering a robust and scalable solution for remote user management.

7. EXPECTED RESULTS


Menu
Home
Admin
RemoteUser
About Us
Admin Login
Name (required)
admin
Password (required)
Submit


Admin Menu
View All Document
View All Comments
View Malware Propagation User Details
View Users
View Android Users and Give Permission
View Friend Request / Responses
View Blocked Users
Log Out
Welcome Admin Main

Admin Menu

- View All Document
- View All Comments
- View Malware Propagation User Details
- View Users
- View Android Users and Give Permission
- View Friend Request / Responses
- View Blocked Users
- Log Out

Welcome Admin Main

All Malware Propagation User Files

Malicious User Pic	OwnerName	Malware Name	Title	Uses	Description FileName	Description	MAC	SK	Date
Submit	Rohith	gal1.jpg	MjAxOV9FbGVjdGlvbg==	dG8ga25vdyBhYm91dCBFbGVjdGlvbg==	connect.ade	¤ DestinationA ¤¤ ¤java/lang/Object ¤¤ ¤java/awt/event/ActionListener¤ ¤f¤ ¤Ljava/awt/Font;¤ ¤T4¤ ¤Ljavax/swing/JLabel;¤ ¤tf¤ ¤Ljavax/swing/JTextA	-48565562d9bf37ed2f0cef2527fd94291e35877	[B@4b0bbb	26/12/2018 13:30:34
Submit	Manjunath	gal6.jpg	SmF2YQ==	VG8ga25vdyBhYm91dCBKYXZh	Focusdata.inf	ï¿½ï¿½ï¿½ï¿½ 2¤,¤ ¤¤ ¤Attacker¤ ¤¤ ¤javax/swing/JFrame¤ ¤¤ ¤java/awt/event/ActionListener¤ ¤mac¤ ¤Ljava/lang/String;¤ ¤path¤ ¤Ljava/io/File;¤ ¤len¤ ¤I¤ ¤encoder¤	5eb314418372b849f9d934c983f41a27690066df	[B@10651eb	26/12/2018 17:42:03

All User Details !!!

Profile Pic	User Name	Email	Mobile	Address	DOB	Gender	Location
Submit	Rohith	Rohith.123@gmail.com	9535866270	#6276,14th Cross,Malleshwaram	05/06/1987	Male	Bangalore
Submit	Suresh	Suresh.123@gmail.com	9535866270	#7827,4th Cross,Rajajinagar	05/06/1987	Male	Bangalore
Submit	Manjunath	tmksmanju13@gmail.com	9535866270	#7827,4th Cross,Rajajiangar,Namgalore-21	05/06/1987	Male	Bangalore

Back

User Friend Request and Response

Request By	Request To	Status	Date
Suresh	Rohith	Accepted	26/12/2018 13:33:05
Manjunath	Rohith	Accepted	26/12/2018 17:38:41

Back

Remote User Menu

- Upload
- Search Users
- Search Files
- View My Search History
- Find Malicious Document
- View My Friends
- View Friend Request
- View My Files
- View and Scan My Friends Files
- Log Out

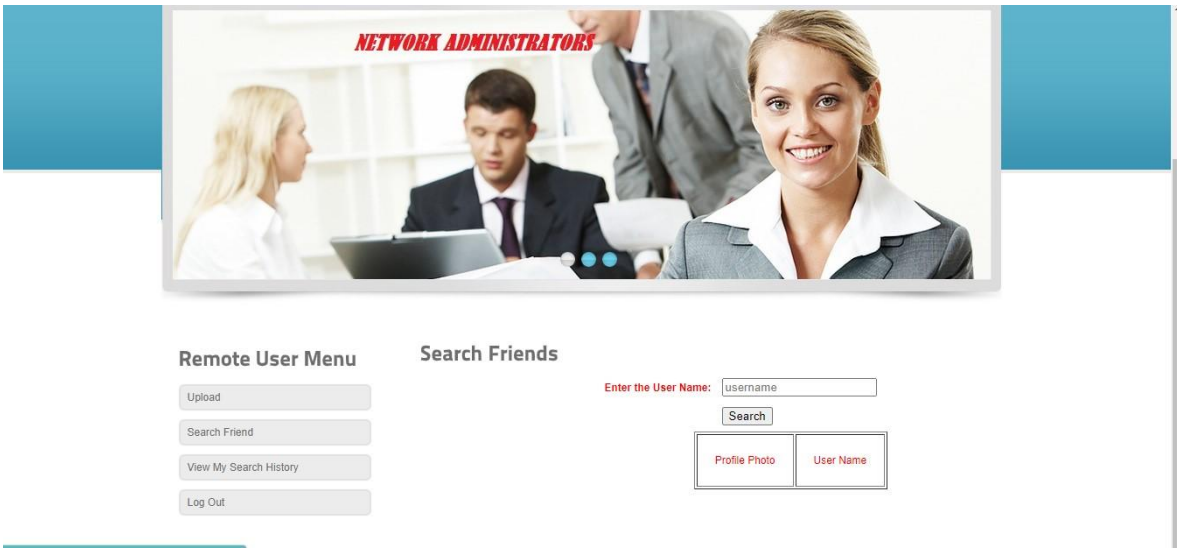
Welcome Remote User Main: suresh

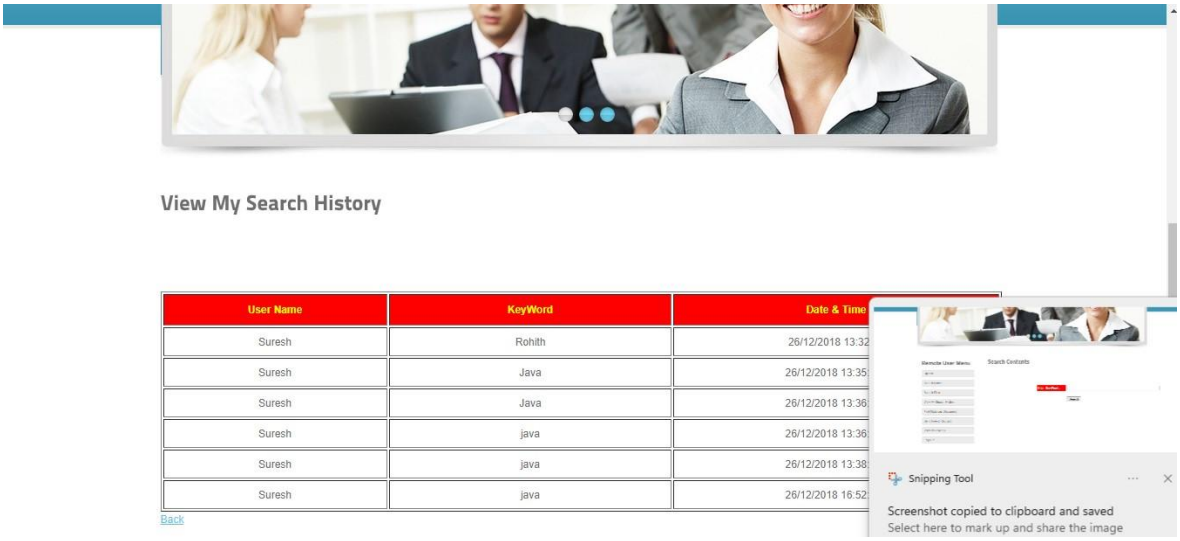
Remote User Menu

- Upload
- Search
- View My Search History
- Send Friend Request
- Log Out

Upload File

Select Doc Image:-	Choose File No file chosen
Title :-	
Uses :-	
Select Description File :-	Choose File No file chosen
Description :-	
MAC :-	


NETWORK ADMINISTRATORS
Remote User Menu
Upload
Search Friend
View My Search History
Log Out
Search Friends
Enter the User Name:
username
Search
Profile Photo
User Name


View My Search History
User Name
KeyWord
Date & Time
Suresh
Rohith
Suresh
Java
Suresh
Java
Suresh
java
Suresh
java
Suresh
java
Back
Snipping Tool
Screenshot copied to clipboard and saved
Select here to mark up and share the image

Malicious Content !!!

Malicious User Pic	OwnerName	FileName	Title	Uses	Malware File	Description	MAC	S
Submit	Rohith	gal1.jpg	MjAxOV9FbGVjdGlvbg==	dG8ga25vdyBhYm91dCBFbGVjdGlvbg==	connect.ade	ASAMRGVzdGluYXR pb25BICAEASAQam F2YS9sYW5nL09ia mVjdCAgBgEgHkph dmEvYXd0L2V2ZW5 0L0fjdGlvbkxpc3 RlbmVyASABZgEgD 0xqYXZhL2F3dC9G	-48565562d9bf37ed2f0cef2527fd94291e35877	[B@4
Submit	Manjunath	gal6.jpg	SmF2YQ==	VG8ga25vdyBhYm91dCBKYXZh	Focusdata.inf	77+977+977+977+ 9ICAgHgIsByACAS AIQXR0YWNrZXIHI AQBIBJqYXZheC9z d2luZy9KRnJhbWU HIAYBIB1qYXZhL2 F3dC9ldmVudC9BY 3Rpb25MaXN0ZW51	5eb314418372b849f9d934c983f41a27690066df	[B@10

Back

8. CONCLUSION

In this paper, we have shown that it is possible to reduce the number of permissions to be analyzed for mobile malware detection, while maintaining high effectiveness and accuracy. SIGPID has been designed to extract only significant permissions through a systematic, 3-level pruning approach. Based on our dataset, which includes over 2,000 malware, we only need to consider 22 out of 135 permissions to improve the runtime performance by 85.6% while achieving over 90% detection accuracy. The extracted significant permissions can also be used by other commonly used supervised learning algorithms to yield the F-measure of at least 85% in 55 out of 67 tested algorithms. SIGPID is highly effective, when compared to the state-of-the-art malware detection approaches as well as existing virus scanners. It can detect 93.62% of malware in the data set, and 91.4% unknown/new malware.

9. FUTURE SCOPE

Substantial authorization recognition in machine learning-driven detection of malicious activities holds significant potential for the future, particularly in enhancing cybersecurity measures. Here are some aspects to consider. Machine learning algorithms can be trained to recognize patterns indicative of malicious behavior within authorization processes. This can include detecting anomalies in user access patterns, unusual privilege escalation attempts, or suspicious activity during authentication.ML-driven systems can provide real-time monitoring of authorization requests and activities, allowing for swift detection and response to potential threats. This

proactive approach can help prevent security breaches before they escalate. Ad By continuously learning from new data, ML algorithms can adapt and evolve to stay ahead of emerging threats. This adaptability is crucial in the dynamic landscape of cybersecurity, where attackers are constantly evolving their tactics. Traditional security systems often generate a high number of false positives, leading to alert fatigue and potentially overlooking genuine threats. ML algorithms can help reduce false positives by learning to distinguish between normal and abnormal authorization behavior more accurately over time.

10. REFERENCES

[1] IDC, "Smartphone os market share, 2017 q1." [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os

[2] Statista, "Cumulative number of apps downloaded from the google play as of may 2016." [Online]. Available: https://www.statista.com/ statistics/281106/number-of-android-app-downloads-from-google-play/

[3] G. Kelly, "Report: 97% of mobile malware is on android. this is the easy way you stay safe," in Forbes Tech, 2014. [4] G. DATA, "8,400 new android malware samples every day." [Online]. Available: https://www.gdatasoftware.com/blog/2017/04/29712-8-400-new-android-malware-samples-every-day

[5] Symantec, "Latest intelligence for march 2016," in Symantec Official Blog, 2016.

[6] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "Riskranker: scalable and accurate zero-day android malware detection," in Proceedings of the 10th international conference on Mobile systems, applications, and services. ACM, 2012, pp. 281–294.

[7] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011, pp. 627–638.

[8] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an informationflow tracking system for realtime privacy monitoring on smartphones," ACM Transactions on Computer Systems (TOCS), vol. 32, no. 2, p. 5, 2014.

[9] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and ¨ C. Siemens, "Drebin:

Effective and explainable detection of android malware in your pocket," in Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS), 2014.

[10] C. Yang, Z. Xu, G. Gu, V. Yegneswaran, and P. Porras, "Droidminer: Automated mining and characterization of fine-grained malicious behaviors in android applications," in European Symposium on Research in Computer Security. Springer, 2014, pp. 163–182.

[11] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. Van Der Veen, and C. Platzer, "Andrubis–1,000,000 apps later: A view on current android malware behaviors," in Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2014 Third International Workshop on. IEEE, 2014, pp. 3–17.

[12] W. Yang, X. Xiao, B. Andow, S. Li, T. Xie, and W. Enck, "Appcontext: Differentiating malicious and benign mobile app behaviors using context," in Software engineering (ICSE), 2015 IEEE/ACM 37th IEEE international conference on, vol. 1. IEEE, 2015, pp. 303–313.

[13] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "Textdroid: Semantics-based detection of mobile malware using network flows."

[14] Z. Li, L. Sun, Q. Yan, W. Srisa-an, and Z. Chen, "Droidclassifier: Efficient adaptive mining of application-layer header for classifying android malware," in International Conference on Security and Privacy in Communication Systems. Springer, 2016, pp. 597–616.

[15] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang, "Machine learning based mobile malware detection using highly imbalanced network traffic," Information Sciences, 2017.

[16] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "Detecting android malware leveraging text semantics of network flows," IEEE Transactions on Information Forensics and Security, 2017.

[17] J. Z. L. H. P. S. Y. Lichao Sun, Xiaokai Wei and W. Srisa-an, "Contaminant removal for android malware detection systems," in Proceedings of IEEE International Conference on Big Data, 2017.

[18] L. Sun, Y. Wang, B. Cao, P. S. Yu, W. Srisa-an, and A. D. Leow, "Sequential keystroke behavioral biometrics for mobile user identification via multi-view deep learning," in Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD), 2017.

[19] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "Textdroid: Semantics-based

detection of mobile malware using network flows," in IEEE INFOCOM 2017 Workshop: MobiSec 2017, Atlanta, GA, USA, May 2017.

[20] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," Computer and Security, vol. 72, pp. 1–12, 2018.