I J C T A, 10(9), 2017, pp. 43-47 ISSN: 0974-5572

Signature Based Intrusion Detection by using RIPPER Algorithm in Android OS

Chetan J. Shelke*, Pravin Karde and V. M. Thakre*****

ABSTRACT

As in recent era of internetworking, phone's are used to in human daily life. These Smart phones having various apps and features but these up to date features give chance to new malwares & threats. Android is newer OS since it is tough to detect and prevent that viruses and malwares by using some old methods.

So security of these Smart phones is now becoming an issue of researchers. So to overcome these various pitfalls we proposed an smart malicious app detector as a security concern. It uses signature based method and makes the app malicious free.

Keywords: Android OS, Smart phones, Malwares, User feedback, Applications Security.

1. INTRODUCTION

With the rapid expansion of Android phone during the protect the network from malwares.. One of the techniques used for making the network secure and detecting malware is user feedback method. It is a mechanism that detects unauthorized and malicious activity from the app. Signature based detection implements pattern matching techniques against a frequently updated past few decades, security has become a crucial issue in the today's era. The purpose of android security is to data. creation of virus is very fast so many time signature is not present .

2. LITERATURE REVIEW

Detecting malicious code isn't recent issue in security. first methods takes signatures to detect it. That was consists of many features: name of file, strings, or machine code. work also focused on protecting the system from the security perspectives that it created.

Persons were employed to analyse programs work. Using own knowledge, signature was found made a malicious code example vary from other malicious code. this type of work was done by Spafford [24], analyzed the Internet Worm & provide notes on attack over the Internet.

Although clear, this is over costly, and slow. If little code will ever given then that will work fluent, but Wildlist [22] is repeatedly change and expand. The Wildlist is a list of code that are recently circulating at all times.

At IBM, Kephart & Arnold [9] developed a method to extract malicious code signature, based on verbal communication identification algorithms & shown to perform almost best a individual expert at detect identified malicious code.

Lo et al. [15] offered a routine to filter hateful code based on "tell-tale signs" were manually engineered based on observing the features of malicious code.

* Asst. Professor, Dept.of IT, P.R.Patil College of Engineering, Amravati, India

** Asst. Professor, Dept of IT, Govt. Polytechnic, Amravati, India

*** HOD, Dept.of Computer science SGBAU Amravati University Amravati, India

Unfortunately, a new malicious program can not have any accessible signature so old methods can't spot a new malicious code. In challenge to solve this issue, the antivirus industry creates heuristic classifiers by hand [8]. This can be even more costly than generating signatures, so finding self method to generate code has been the topic of research in the antivirus community. To solve this issue, IBM researchers use ANNs to find boot sector malicious binaries [25]. An ANN is a classifier who binds neural networks explored in human cognition. By the drawbacks of the implementation of own classifier, they didn't analyze anything boot sector viruses which comprise about 4-5% of all malicious binaries.

Recognition system by Lee et al. [13][14]. Their method was applied to system calls and network data to study how to detect new intrusions. They report good detection rates as a result of applying data mining Using an ANN classifier with all bytes from the boot sector malicious code as input, IBM researchers were able to identify 80–85% of unfamiliar boot sector malicious code successfully with a low false positive rate (< 2%). They were incapable to find a way to relate ANNs to the other 95% of computer malicious binaries.

In similar work, Arnold and Tesauro [1] applied the same technique to Win32 binaries, but as of boundaries of the ANN classifier they were unable to have the analogous accuracy over new Win32 binaries.

Our method is different because we analyze the en-tire set of malicious code instead of only boot-sector viruses, or only Win32 binaries.

Our technique is similar to data mining techniques that have already been applied to Intrusion to the problem of IDS.

3. SIGNATURE METHODS

System detect the malicious application by using its signature system firstly scan the particular application for signature. Pattern of instruction present in android application downloaded from android market recognize as the signature of app. If the signature is matched with global signature dataset of malicious application then the system inform the user by notifying about malicious app. few points to be in consideration that signature dataset of malicious app always updated for the new application i.e signature of new malicious app is not present in global signature dataset then system uses RIPPER algorithm which can detect new malicious app done by some hypothesis and few rule based condition.

Signature detector program extract the opcode from decompiled apk file opcode pattern consider as the signature of app then the specific signature is matched with global dataset if match is found the input app belong to family of malicious app.

1) *Signature of malicious app present in local signature datasets*: In signature based detection firstly signature extracted from the application by using signature extraction method once the signature is extracted from application that signature is matched with the local signature dataset to which contain signature of malicious application if the signature is match then the application is blacklisted .

2) *Signature is not present in local signature dataset*: if the signature of malicious app is not present in local dataset then system uses RIPPER algorithm to analyze the application is malicious or not by different condition: Ripper algorithm predict the malicious app by ruled based hypothesis. i.e if, then .model is trained on the basis of training a data set which have various condition like

- Has the extracted application taken more permission than required
- Has the application using malicious system call
- Has the application Background playing

The important property of any inductive learner is that no a priori assumption has been complete regarding the final concept.

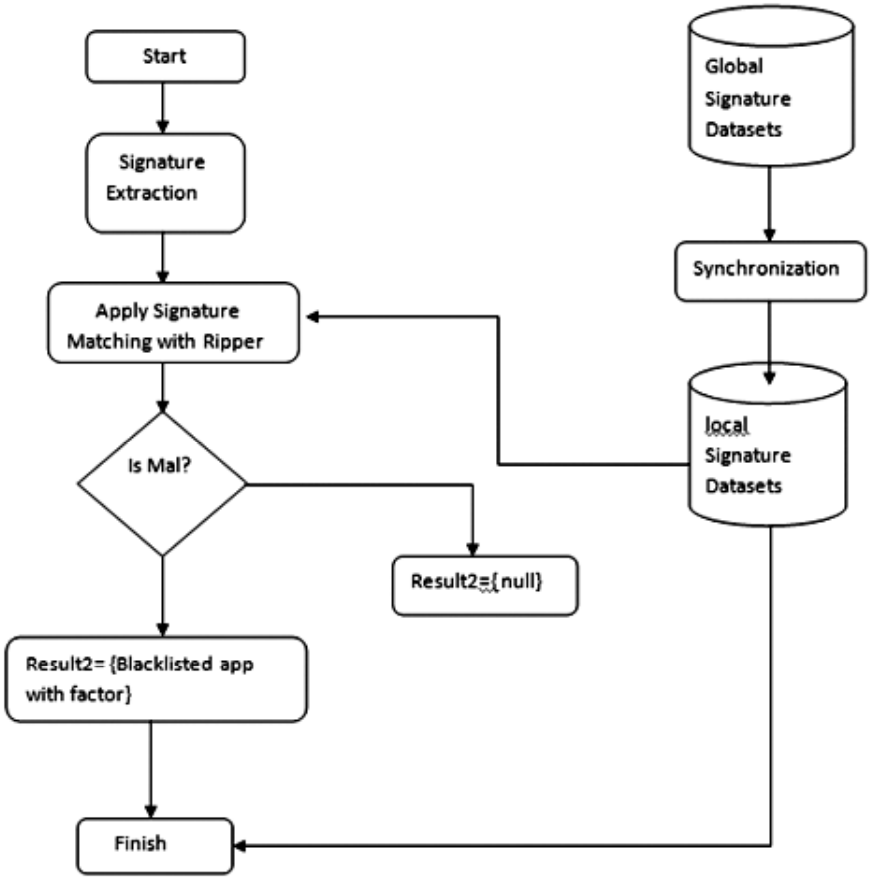

Figure 1: Signature Based Detection

RIPPER looks at both positive and negative examples to generate a set of hypotheses that more closely approximate the target concept.

4. RESULT ANALYSIS

4.1. Battery power use

As we be familiar with any app that runs on a Smart Phones be supposed to use some amount of battery power depending on its activities. Most of the android malware finding applications are runs in a background, so it will consume additional battery power.

In proposed System, cloud is use as the main mace, all the processes like decompilation of a apk file, Training the database, Testing a new APK file via Pattern Matching of signatures after uploading are done on the cloud so no one process run on the background. Hence it takes less battery of user's smartphones.

4.2. Less Storage space

Android applications which will installed on a smart phones requires different storage space to for storing different important files required by these applications. The different malware detection applications available requires certain amount of storage space to stores the signatures or information about the malwares, as we know that new malwares are introduced daily so we have to update the database of a installed malware application. Once we update the database it will increases the size of the application, so the required space by application becomes increasing and it will causes the availability of storage space for the user.

To overcome such problems, proposed system maintains the database of the signatures on a cloud. That means we can update the application database by training on a cloud, so it will minimizes the storage space of the application.

4.3. Centralized database

As the proposed system uses the cloud as a server and it contain database of all the signatures while training and for the testing purpose proposed system uses the database for pattern matching so all users access the remote server from different places. Hence database is centralized.

4.4. Comparison amongst the Existing Systems

Table 1
Comparison of systems

Cases	*Avast Antivirus*	*Avg Antivirus*	*Implemented system*
Storage space on device	12.8 mb	27 mb	1.16 mb
Battery consumption	7 %	6%	2%
Database	Database is on device	Database is on device	Database is on cloud
Ram	30 mb	22 mb	17.74 mb
Updation of database	User has to install	User has to install	Admin has to install
Decompilation of app	No	No	Yes
Background playing	Yes	Yes	No
Centralized database	No (only for one user)	No (only for one user)	Yes – available for all user

5. CONCLUSION

The proposed system uses signature based detection for the detection of malicious application from Smartphone proposed system detect malicious app by extracting signature and comparing that signature from global database if the signature is not found on the centralized server then for new application system uses ripper algorithm which use rule based prediction as per the behavior of the application. We tried to overcome the problem of existing malware detection app i.e existing method not catch the malicious app if the signature is not updated or signature is not present in the global dataset our system uses the ripper algorithm to catch new malicious program by training the program which uses the rule base approach for new application.

REFERENCES

[1] William Arnold and Gerald Tesauro. Automatically Generated Win32 Heuristic Virus Detection. Pro-ceedings of the 2000 International Virus Bulletin Conference, 2000

[2] Fred Cohen. A Short Course on Computer Viruses. ASP Press, 1990.

[3] William Cohen. Learning Trees and Rules with Set-Valued Features. American Association for Artificial Intelligence (AAAI), 1996.

[4] R. Crawford, P. Kerchen, K. Levitt, R. Olsson, M. Archer, and M. Casillas. Automated Assistance for Detecting Malicious Code. Proceedings of the 6th International Computer Virus and Security Conference, 1993.

[5] Cygnus. GNU Binutils Cygwin. Online publication,1999.http://sourceware.cygnus.com/cygwin.

[6] D.Michie, D.J. Spiegelhalter, and C.C.Taylor D. Machine learning of rules and trees. In Machine Learning,Neural and Statistical Classification. Ellis Horwood,1994.

[7] Eleazar Eskin, William Noble Grundy, and Yoram Singer. Protein Family Classification using Sparse Markov Transducers. Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, 2000.

[8] Dmitry Gryaznov. Scanners of the Year 2000: Heuristics.Proceedings of the 5th International Virus Bulletin,1999.

[9] Jeffrey O. Kephart and William C. Arnold. AutomaticExtraction of Computer Virus Signatures. 4th Virus Bulletin International Conference, pages 178-184, 1994.

[10] P. Kerchen, R. Lo, J. Crossley, G. Elkinbard, and R. Olsson. Static Analysis Virus Detection Tools for UNIX Systems. Proceedings of the 13th National Computer Security Conference, pages 350–365, 1990.

[11] Zou KH, Hall WJ, and Shapiro D. Smooth nonparametric ROC curves for continuous diagnostic tests. Statistics in Medicine, 1997.

[12] R Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. IJCAI,1995.

[13] W. Lee, S. J. Stolfo, and P. K. Chan. Learning patterns from UNIX processes execution traces for intrusion detection. AAAIWorkshop on AI Approaches to Fraud Detection and RiskManagement, pages 50–56. AAAI Press, 1997.

[14] R.W. Lo, K.N. Levitt, and R.A. Olsson. MCF: a Malicious Code Filter. Computers & Security, 541 566., 1995.

[15] MacAfee. Homepage - MacAfee.com. Online publication,2000. http://www.mcafee.com.

[16] Microsoft. Portable Executable Format. Online publication,1999.http://support.microsoft.com/support/kb/articles/Q121/4/60.asp.

[17] Peter Miller. Hexdump. Online publication, 2000.http://www.pcug.org.au/millerp/hexdump.html.

[18] MIT Lincoln Labs. 1999 DARPA intrusion detection evaluation.

[19] Tom Mitchell. Machine Learning. McGraw Hill,1997.

[20] Kamal Nigam, Andrew McCallum, Sebastian Thrun,and Tom Mitchell. Learning to Classify Text from Labeled and Unlabled Documents. AAAI-98, 1998.

[21] Wildlist Organization. Virus descriptions of viruses in the wild. Online publication, 2000. http://www.fsecure.com/virus-info/wild.html.

[22] REUTERS. Microsoft Hack Shows Companies Are Vulnerable. New York Times, October 29, 2000.

[23] Eugene H. Spafford. The Internet worm program: an analysis. Tech. Report CSD–TR–823, 1988. Department of Computer Science, Purdue University.

[24] Gerald Tesauro, Jeffrey O. Kephart, and Gregory B.Sorkin. Neural Networks for Computer Virus Recognition.IEEE Expert, **11**(4):5–6. IEEE Computer Society,August, 1996.

[25] Steve R. White. Open Problems in Computer Virus Research. Virus Bulletin Conference, 1998.

[26] Steve R. White, Morton Swimmer, Edward J. Pring,William C. Arnold, David M. Chess, and John F.Morar. Anatomy of a Commercial-Grade Immune System. IBM Research White Paper, 1999.