Volume 7, Issue 3, March 2017 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Classification System for Malware Identification and Detection

Tushar Punjabi, Udayraj Biradar, Dnyanashwar Kawade, Akash Battin, Prof. A.A. Pund

Department of Information Technology, D.V.V.P.C.O.E.A., Ahmednagar,
Maharashtra, India

DOI: 10.23956/ijarcsse/V7I3/0172

Abstract— Signature based malware detection systems have been a tremendously utilized reaction to the unavoidable issue of malware. Distinguishing proof of malware variations is basic to a detection system furthermore, is made conceivable by distinguishing invariant qualities in related examples. To characterize the packed and polymorphic malware, this paper proposes a novel system, named malwise, for malware characterization utilizing a quick application level emulator to switch the code pressing change, furthermore, two flowgraph coordinating algorithms to perform characterization. A correct flowgraph coordinating algorithm is utilized that utilizations string based signatures, and can recognize malware with close constant execution. Furthermore, a more powerful surmised stream chart coordinating algorithm is recommended that uses the decompilation method of organizing to produce string based signatures managable to the string alter remove. We utilize genuine and manufactured malware to show the adequacy what's more, productivity of Malwise. Utilizing more than 15,000 genuine malware, gathered from honeypots, the viability is approved by demonstrating that there is an 88% likelihood that new malware is distinguished as a variation of existing malware. The effectiveness is shown from a littler example set of malware where 86% of the specimens can be classified in under 1.3 seconds.

Keywords— component; Computer security, malware, control flow, structural classification, structured control flow, unpacking.

I. INTRODUCTION

Malware, short for malicious software, implies a assortment of types of antagonistic, nosy or irritating software or program code. Malware is an inescapable issue in appropriated PC and system systems. As per the Symantec Internet Threat Report [1],499,811 new malware tests were gotten in the second 50% of 2007. F-Secure moreover detailed, "As much malware [was] created in 2007 as in the previous20 years through and through" [2]. Detection of malware is imperative to a protected conveyed figuring environment. The transcendent procedure utilized as a part of business anti malware systems to recognize a case of malware is using malware signatures. Malware signatures endeavour to catch invariant attributes on the other hand designs in the malware that extraordinarily distinguishes it. The examples used to build a signature have customarily got from strings of the malware's machine code and crude record substance [1] [2]. String based signatures have stayed mainstream in business systems because of their high effectiveness; however can be ineffectual in recognizing malware variations. Malware variations frequently have particular byte level representations while in chief have a place with the same group of malware. The byte level substance is distinctive since little changes to the malware source code can bring about fundamentally extraordinary accumulated question code. In this paper we depict malware variations with the umbrella term of polymorphism. Polymorphism depicts related malware sharing a typical history of code. Code sharing among variations can be inferred from self-sufficiently self mutating malware, or physically replicated by the malware maker to reuse beforehand wrote code.

Static examination joining n-grams, alter separations, API call groupings, and control stream [3] have been proposed to distinguish malware what's more, their polymorphic variations. In any case, they are either incapable or wasteful in grouping packed what's more, polymorphic malware. A malware's control stream data gives a trademark that is identifiable crosswise over strains of malware variations. Inexact coordinating of flow graph based attributes can be utilized as a part of request to distinguish a more prominent number of malware variations. Detection of variations is conceivable notwithstanding when more noteworthy changes to the malware source code are presented. Control stream has demonstrated compelling [2][3][4], and quick algorithms have been proposed to distinguish correct isomorphic entire program control stream charts what's more, related data [4], yet rough coordinating of program structure has appeared to be costly in runtime costs [5]. Poor execution in execution speed has brought about the nonappearance of inexact coordinating in end have malware detection. To thwart the static investigation fundamental for control stream examination, the malware's genuine substance is as often as possible shrouded utilizing a code change known as pressing [5][6]. Pressing is not exclusively utilized by malware. Pressing is additionally utilized as a part of software security plans and record pressure for honest to goodness software, yet the larger part of malware additionally utilizes the code pressing change. In one month amid 2007, 79% of recognized malware was packed [7]. Also, very nearly half of new malware in 2006 were repacked forms of existing malware [7][8].This article has been acknowledged for distribution in a future issue of this diary, yet has not been completely altered. Substance may change preceding last distribution.

II. LITERATURE SURVEY

Mechanized unloading utilizing entire system imitating was proposed in Renovo and Pandora's Bochs. Entire system imitating has been illustrated to give viable outcomes against obscure malware tests, yet is not totally impervious to novel assaults [9].Renovo and Pandora's Bochs both distinguish execution of dynamically created code to decide when unloading is finished and the concealed code is uncovered. An option algorithm for identifying when unloading is finished was proposed u sing execution histograms in Hump - and-dump [10]. The Hump-and-dump was proposed as conceivably attractive for mix into an emulator. Polyunpack proposed a blend of static furthermore, dynamic examination to dynamically recognize code at runtime which can't be distinguished amid an underlying static investigation. The primary refinement isolating our work from beforehand proposed mechanized unpackers is our utilization of use level imitating and an forceful methodology to confirm that unloading is finish[11]. The upside of use level copying over entire system imitating is altogether more prominent execution. Application level copying for mechanized unloading has had business intrigue [12] yet has acknowledged few scholastic distributions assessing its adequacy what's more, execution. Dynamic Binary Instrumentation was proposed as an other option to utilizing an instrumented emulator [13] utilized by Renovo what's more, Pandora's Bochs. Omni pack and Saffron proposed computerized unloading utilizing local execution and equipment based memory insurance highlights. This outcomes in superior in contrast with copying based unloading [14]. The disservice of unloading utilizing local execution is obvious on EMail entryways on the grounds that a virtual machine or emulator is required to execute the malware. A virtual machine way to deal with unloading, utilizing x86 equipment expansions, was proposed in Ether. The utilization of such a virtual machine and similarly to an entire system emulator is the necessity to introduce a permit for every visitor working system. This limits desktop appropriation which commonly has a solitary permit. Virtual machines are additionally repressed by moderate start-up times, which again are risky for desktop utilize. The use of a virtual machine likewise keeps the system being cross stage, as the visitor and host CPUs must be the same[14][15]. Our exploration varies from past flow graph grouping look into by utilizing a novel estimated control stream diagram coordinating algorithm utilizing organizing. We are the first to utilize the approach of organizing and decompilation to create malware signatures. This permits us to utilize string based methods to handle generally infeasible chart issues[15]. We utilize a correct coordinating algorithm which performs in close constant while as yet having the capacity to recognize surmised matches at an entire program level. The novel set closeness seek we perform empowers the continuous order of malware from a substantial information base. No earlier related investigate has performed continuously.

III. SYSTEM ARCHITECTURE

A malware order system is accepted to have propelled access to an arrangement of known malware. This is for development of an underlying malware database. The database is built by distinguishing invariant attributes in each malware and creating a related signature to be put away in the database. After database introduction, typical utilization of the system starts. The system has as information a formerly obscure binary that will be classified as being malicious or non malicious. The information binary and the underlying malware doubles may have also experienced a code pressing change to thwart static examination[14][15][16].

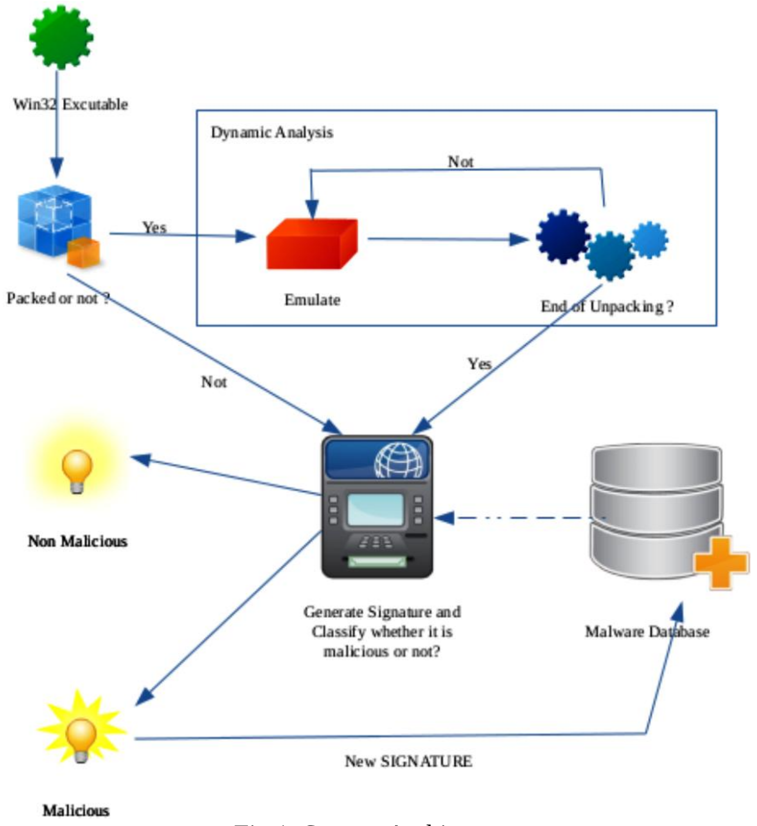


Fig 1: System Architecture

The classifier figures likenesses between the information binary and each malware in the database. The comparability is measured as a genuine number between 0 also, 1 – 0 showing not under any condition comparative and 1 demonstrating an indistinguishable or fundamentally the same as match. This closeness is a based on the comparability between malware attributes in the database. On the off chance that the closeness surpasses a given limit for any malware in the database, then the info binary is esteemed a variation of that malware, and hence malicious. Our approach utilizes both dynamic and static investigation to group malware[16]. Entropy examination at first figures out whether the binary has experienced a code pressing change. On the off chance that packed, dynamic examination utilizing application level imitating uncovers the shrouded code utilizing entropy examination to distinguish when unloading is finished. Static investigation at that point recognizes attributes, building signatures for control stream diagrams

Two methodologies are utilized to produce and analyze Stream chart signatures. The system configuration is exhibited in figure 1.Two stream diagram coordinating techniques are utilized to accomplish the objective of either viability or productivity[16][17]. A concise presentation is given here. Correct Matching: A requesting of the hubs in the control stream diagram is utilized to create a string based signature invariant of the flow graph. String balance between chart invariants is utilized to appraise isomorphic charts. Estimated Matching: The control stream chart is organized in this approach. Organizing is the procedure of decompiling unstructured control stream into larger amount, Source code like builds including organized Conditions and cycle. Every signature speaking to the organized control stream is spoken to as a string. These signatures are then utilized for questioning the database of known malware utilizing a rough lexicon look.

IV. SYSTEM DESIGN

In this paper we have cosine similarity for approximate matching. Done using set of similarities .database will be comprised of signature of known malware, input will be a binary. A similarities is constructed between the binary's flowgraph string and each set of flowgraph associated with malwares in the database, considers the weights associated with signature as well. the process results with a similarity value for each set of signature in the malware[17]. Values ranges between 0 and 1.

- Value > 0.95 =>isomorphs
- Value < 0.6 => no similarity
- 0.6 > value < 0.95 =>variant

Detection rate was rounded to be about 57.8%, earlier approaches achieved maximum up to 39.6%, resilience to false positives. Less than 0.61% of samples were incorrectly identified as malwares. At least 10 procedures should be present in the flow graph matching. For exact flow graph matching at least 15 procedures should be present. The threshold values were fixed after through pilot study.

A. Cosine Similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

A_i and B_i are components of vector A and B respectively. The resulting similarities ranges from -1 meaning exactly same the same, with 0 indicating orthogonality and in between values indicating intermediate similarities or dissimilarities for text matching the attribute vector A and B are usually the term frequency vectors of documents. The cosine similarity can be seen as a method of normalizing document length during comparison.

B. Similarity Thresholds

The threshold to determine if two programs are similar, in either exact flow graph matching or approximate flow graph matching, is empirically decided in Malwise. Likewise as is the similarity ratio between flow graphs. The actual figures are decided by investigating a huge number of real-life malware samples. This approach is currently adopted by most antivirus systems[17]. It is a desirable feature that the malware classification system can adaptively select the thresholds. Machine learning-based approach can be taken

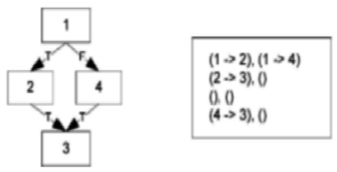
C. Exact flow graph matching

Only exact replicas or isomorphism are identified, signatures are created by ordering the nodes of the control flow graph in depth first order. Signature will consist of list of graph edges for ordered nodes, Matching done using dictionary lookup[17][18].

Weight is found by –

Weight (x) = $\frac{Bx}{\sum i\ Bi}$ where Bi = no of basic blocks in binary

Depth first ordered flow graph



```
(1 -> 2), (1 -> 4)
(2 -> 3), ()
(), ()
(4 -> 3), ()
```

Fig 2: flow graph matching

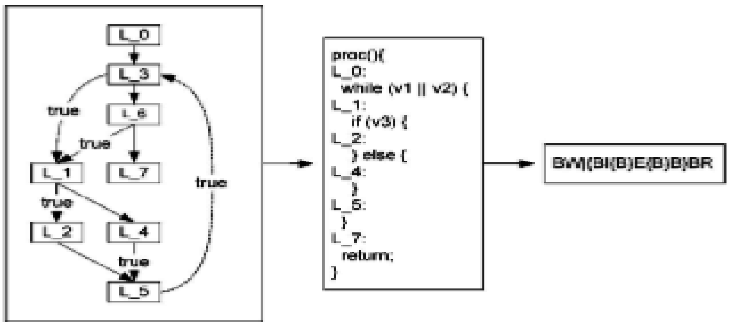
D. Approximate flow graph matching

Approximate matches of control flow graphs are considered. Enables detection of variants, structuring is used to generate signature, the output of string character tokens representing high level structured constructs[18]

Weight is found by-

$$\text{Weight}(x) = \frac{\text{len}(Sx)}{\sum i\ \text{len}(Si)}$$

Where si=signature of S in binary.



```
proc(){
L_0:
  while (v1 || v2) {
L_1:
    if (v3) {
L_2:
    } else {
L_4:
    }
L_5:
  }
L_7:
  return;
}
```

```
BW|(BI{B}E{B}B)BR
```

Fig 3: control flow graph high level structured graph signature.

V. RESULTS

Following snapshots shows the implementation results of the proposed system.

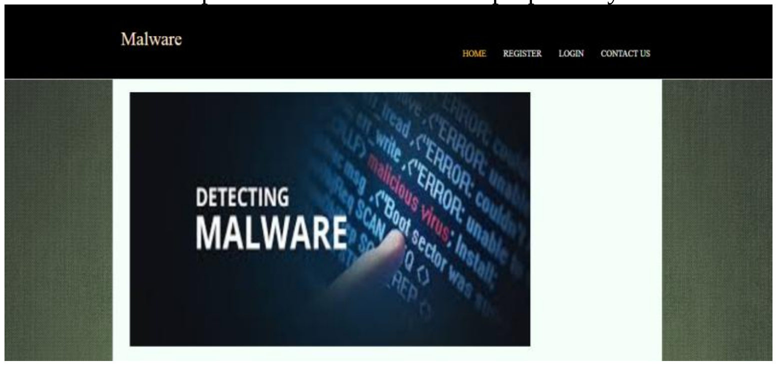
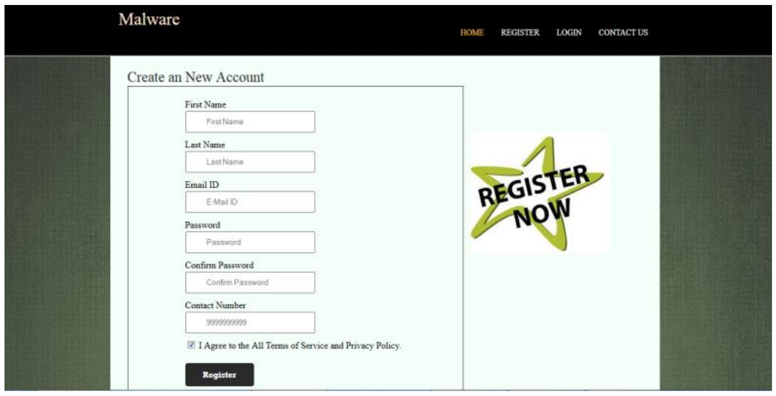


Fig 4: Home Page



Create an New Account

First Name: First Name

Last Name: Last Name

Email ID: E-Mail ID

Password: Password

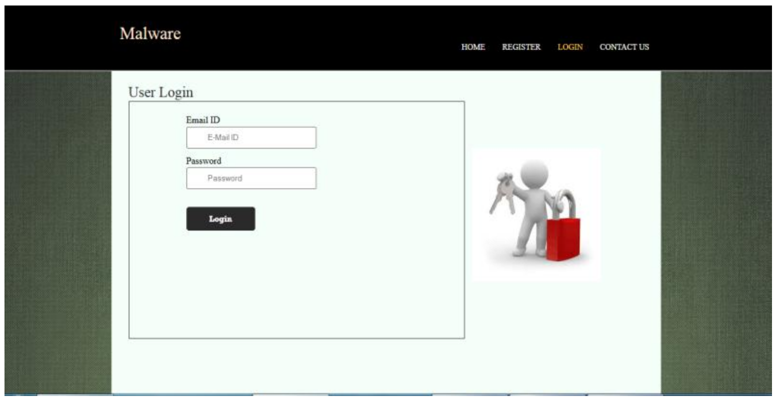
Confirm Password: Confirm Password

Contact Number: 9999999999

☑ I Agree to the All Terms of Service and Privacy Policy.

Register

Fig 5: Registration Form



User Login

Email ID: E-Mail ID

Password: Password

Login

Fig 6: Login Form

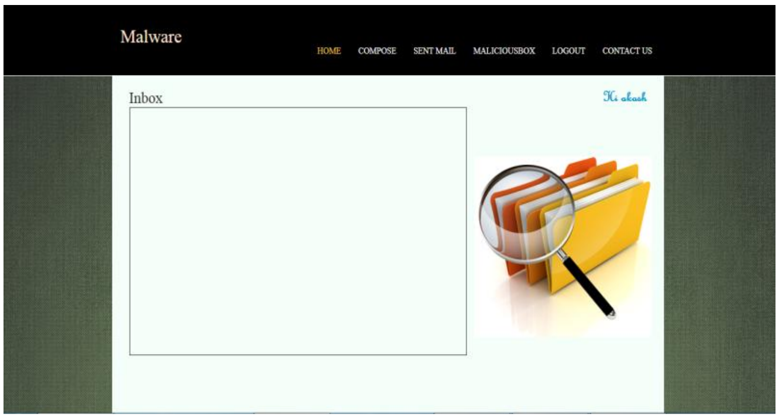
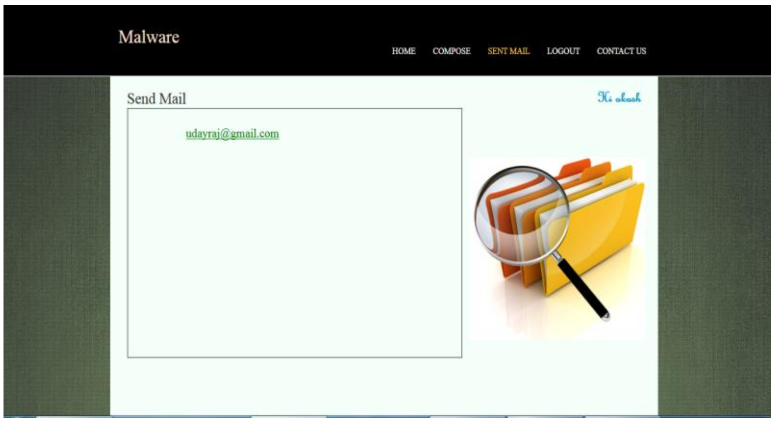
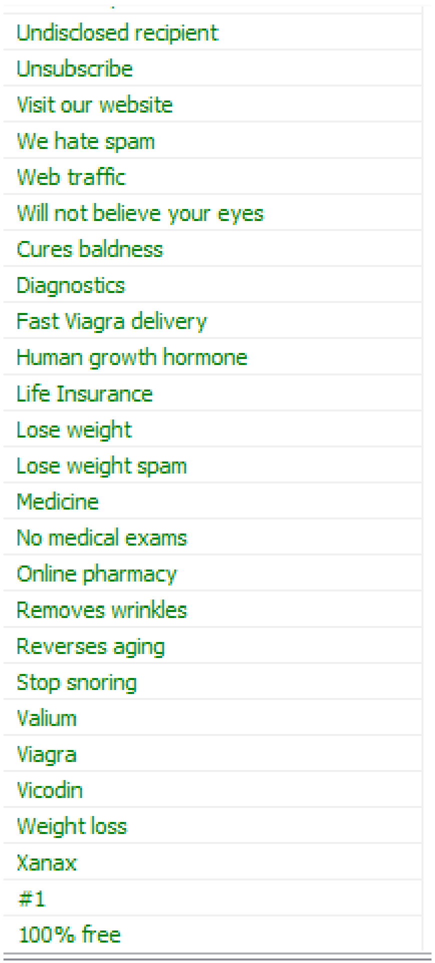


Fig 7: Inbox of User



udayraj@gmail.com

Fig 8: Sent Mail



Undisclosed recipient
Unsubscribe
Visit our website
We hate spam
Web traffic
Will not believe your eyes
Cures baldness
Diagnostics
Fast Viagra delivery
Human growth hormone
Life Insurance
Lose weight
Lose weight spam
Medicine
No medical exams
Online pharmacy
Removes wrinkles
Reverses aging
Stop snoring
Valium
Viagra
Vicodin
Weight loss
Xanax
#1
100% free

Fig 9: Malicious Dataset

VI. CONCLUSION

Malware can be classified by closeness in its flow graphs. This investigation is made more testing by packed malware. In this paper we proposed distinctive algorithms to unload malware utilizing application level imitating. We also proposed performing malware arrangement utilizing either the alter remove between organized control stream charts, on the other hand the estimation of isomorphism between control stream diagrams.

We executed and assessed these approaches in a completely functionally system, named Malwise. The computerized unloading was exhibited to conflict with a promising number of manufactured examples utilizing known pressing apparatuses, with rapid. To recognize the culmination of unloading, we proposed and assessed the utilization of entropy examination. It was demonstrated that our system can viably distinguish variations of malware in tests of genuine malware. It was likewise demonstrated that there is a high likelihood that new malware is a variation of existing malware. At long last, it was shown the productivity of unloading and malware grouping warrants Malwise as reasonable for potential applications counting desktop and Internet entryway and Antivirus systems.

REFERENCES

[1] Vidhate, Deepak A and Kulkarni, Parag "Innovative Approach Towards Cooperation Models for Multi-agent Reinforcement Learning (CMMARL)", Springer Nature series of Communications in Computer and Information Science, Vol. 628,pp. 468-478,2016

[2] Vidhate, Deepak A and Kulkarni, Parag "New Approach for Advanced Cooperative Learning Algorithms using RL Methods (ACLA)" Proceedings of the Third International Symposium on Computer Vision and the Internet, ACM, pp 12-20, 2016

[3] K. Griffin, S. Schneider, X. Hu, and T. Chiueh, "Automatic Generation of String Signatures for Malware Detection," Proc. 12th Int'l Symp. Recent Advances in Intrusion Detection (RAID '09), 2009.

[4] Y. Ye, D. Wang, T. Li, and D. Ye, "IMDS: Intelligent Malware Detection System," Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, 2007.

[5] S. Cesare and Y. Xiang, "Classification of Malware Using Structured Control Flow," Proc. Eighth Australasian Symp. Parallel and Distributed Computing (AusPDC '10), 2010.

[6] Vidhate, Deepak, A; Kulkarni, Parag (2014):"Improvement In Association Rule Mining By Multilevel Relationship algorithm" in International Journal of Research in Advent Technology, 2(1), pp.366-373

[7] Briones and A. Gomez, "Graphs, Entropy and Grid Computing: Automatic Comparison of Malware," Proc. Virus Bull. Conf., pp. 1- 12, 2008.

[8] G. Bonfante, M. Kaczmarek, and J.Y. Marion, "Morphological Detection of Malware," Proc. IEEE Int'l Conf. Malicious and Unwanted Software, pp. 1-8, 2008.

[9] R.T. Gerald and A.F. Lori, "Polymorphic Malware Detection and Identification via Context-Free Grammar Homomorphism," Bell Labs Technical J., vol. 12, pp. 139-147, 2007.

[10] X. Hu, T. Chiueh, and K.G. Shin, "Large-Scale Malware Indexing Using Function-Call Graphs," Proc. ACM Conf. Computer and Comm. Security, pp. 611-620, 2009.

[11] Vidhate, Deepak A and Kulkarni, Parag "Single Agent Learning Algorithms for Decision making in Diagnostic Applications", SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE), Vol.3, No.5,pp.46-52,2016

[12] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee, "Polyunpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware," Proc. Computer Security Applica- tions Conf., pp. 289-300, 2006.

[13] M.G. Kang, P. Poosankam, and H. Yin, "Renovo: A Hidden Code Extractor for Packed Executables," Proc. Workshop Recurring Malcode, pp. 46-53, 2007.

[14] Vidhate, Deepak A and Kulkarni, Parag "Implementation of Multiagent Learning Algorithms for Improved Decision Making", International Journal of Computer Trends and Technology (IJCTT) Vol 35, No 2,pp 60-66, 2016

[15] L. Sun, T. Ebringer, and S. Boztas, "Hump-and-Dump: Efficient Generic Unpacking Using an Ordered Address Execution Histo- gram," Proc. Int'l Computer Anti-Virus Researchers Organization (CARO) Workshop, 2008.

[16] L. Martignoni, M. Christodorescu, and S. Jha, "Omniunpack: Fast, Generic, and Safe Unpacking of Malware," Proc. Ann. Computer Security Applications Conf. (ACSAC), pp. 431-441, 2007.

[17] Vidhate, Deepak A and Kulkarni, Parag "A Step toward Decision making in Diagnostic Applications using Single Agent Learning Algorithms", International Journal of Computer Science and Information Technologies (IJCSIT),Vol.7,No.3, pp 1337-1342,2016

[18] Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: Malware Analysis via Hardware Virtualization Extensions," Proc. 15th ACM Conf. Computer and Comm. Security, pp. 51-62, 2008.