POLITECNICO DI MILANO

V Facolta di Ingineria

Dipartimento di Elettronica e Informazione

Corso di Laurrea Specialistica in Ingegneria Informatica

(Master of Science in Computing Systems Engineering)

COMPARISON AND BENCHMARKING OF AUTOMATIC MALWARE UNPACKING TECHNIQUES

Supervisor: *Prof. Stefano ZANERO*

Master's Thesis By:

Tewfik Adem, Getu

Matr. N°. : 737138

Academic Year 2009-2010

ABSTRACT

Analyzing and detection of “malicious software” (malware), such as viruses, worms and botnet clients, whether fully automated or human assisted is a critical step in defending against the threat such malware poses. The challenge will be more when malware writers misuse the novel idea of software packing, to bypass detection from malware analyzers and antivirus software.

As a matter of fact, nowadays 80% malware is often transmitted in packed or encrypted form to prevent examination by anti-virus software [1]. To analyze new malware, researchers typically resort to automatic and dynamic code analysis techniques to unpack the code for examination. Unfortunately, these dynamic techniques are susceptible to a variety of anti-monitoring defenses, as well as “time bombs” or “logic bombs,” which can be slow and tedious to identify and disable.

This thesis work, compares and benchmark currently existing automatic malware unpacking techniques, and explores new approaches to design automated malware unpackers. It basically starts by assessing research works related to malware analysis and detection, and focus on packed malware analysis techniques.

To beat the challenges posed by malware writers a packed malware analyzer should be transparent to the analyzed malware, it should be able to detect different layers of packing and more over able to extract and reconstruct both syntactic and semantic behaviors of the packed malware.

ACKNOWLEDGMENT

First and foremost I offer my sincere gratitude to my supervisor, **Professor Stefano ZANERO**, who has supported me throughout my thesis with his patience and knowledge. I attribute the level of my Masters degree to his encouragement and dedication to advice me.

Special thanks to **Prof. Colombetti Marco**, who consistently and tirelessly dedicated his time to encourage, guide and help me throughout my stay at Politecnico Di Milano.

Thanks to Anteneh Teferi and Mohammed Mahmudur Rahman for their support and morale in the course of this thesis, and all the dear friends nearby in helping and encouraging me during my stay.

Finally, not only for the thesis, but for the whole opportunity I have to pursue this masters program to an end, I would like to thank Politecnico di Milano and **DSU scholarship Program**, who provided me with all the necessary fees, education and helpful stuffs.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

1. INTRODUCTION

Malware is a collective term for any malicious software which enters a computer system without authorization of a system user. The term is created from merging the words ‘malicious’ and ‘software’. Malware is a very big threat in today’s computing world and IT research.

Analyzing and detection of malware, such as viruses, worms and botnet clients, whether fully automatically or with human assistance is a critical step in defending against the threat such malware poses. The challenge will be more when malware writers misuse the novel idea of software packing to bypass detection from malware ware analyzers and antivirus software.

As a matter of fact, nowadays 80% malware is often transmitted in packed or encrypted form to prevent examination by anti-virus software [1]. To analyze new malware, researchers typically resort to automatic and dynamic code analysis techniques to unpack the code for examination. Unfortunately, these dynamic techniques are susceptible to a variety of anti-monitoring defenses, as well as “time bombs” or “logic bombs,” which can be slow and tedious to be identified and disabled. Moreover, these different techniques are focusing on solving different aspect of this problem in which there isn’t clear bisection between them.

This thesis work, compares and benchmark currently existing automatic malware unpacking techniques, and explores new approaches to design automated malware unpackers. It basically start on assessing research works related to malware analysis and detection, and focus on packed malware analysis techniques.

1.1. MOTIVATION

Despite the day-to-day increase in packed malwares in the wild, there are very few works on “automatic malware unpacking”. These days, researchers are working hard and trying their level best to come up with a generic solution to the problem of packed malwares analysis. But as the usual nature of any research these different studies are focusing on various aspects of the problem in which sometimes there is some overlapping and in its worst cases one might “solve” a problem already solved. This is due to the lack of a clear study that shows the SWOT analysis between these studies.

In addition to above mentioned reason the main motivation behind this thesis is to study: what is the consideration and main target, specific techniques used, experiments done and contributions of each of these different published unpacking techniques. Thus one can have clear idea of the current state of the art and try to come up with a better solution.

1.2. OBJECTIVES

As a result of this thesis work we want to come up with a comprehensive summary of comparison and benchmarking of currently existing malware unpacking techniques. This work includes assessment of the existing malware analysis techniques, followed by selection of techniques specifically focusing on packed malware unpacking techniques.

Finally, we would like to do a SWOT analysis that assesses the strength, weakness, opportunities and threats of each of these techniques, and as a consequence categorization and ranking within these techniques.

1.3. STRUCTURE OF THE REPORT

The content of this thesis is structured as follows. *Chapter two* introduces the preliminary concepts that are crucial to understand the rest of the report. It basically discusses what are malicious software (malware), malware analysis and detection techniques, software packing and malware packing and explains the general idea of packed malware analysis and unpacking techniques.

Chapter three starts with literature review of existing works on analysis of malware unpacking techniques with their basic strength and pitfalls. Followed by, literature review of currently existing works on malware analysis and detection. In which subset of them are selected for further study and analysis. Finally a brief overview of the selected techniques is presented.

Chapter four presents the analysis result of the selected unpacking techniques according to some critical drivers such as main target, specific technique, specific requirement, experiment and contribution of each technique. It also shows the SWOT analysis detail, along with the categorization, raking and benchmarking of the selected techniques.

Finally in *chapter five*, summary and conclusions of the whole thesis work, limitations and problems on the study, and possible future works are presented. Advanced readers with the basic idea of malware analysis and detection can skip chapter two and directly start from chapter three.

CHAPTER TWO

2. PRELIMINARIES

This chapter introduces preliminary concepts that are essential elements to understand the discussion that follows in the coming chapters. The second section explains the definition and types of malware which draw the clear picture of what malware is and its different types. The second section discusses the importance and concept of malware analysis and detection techniques and its categories. Followed by, a discussion of software packing and malware packing. Finally discuss packed malware analysis and unpacking techniques.

2.1. MALICIOUS SOFTWARE (MALWARE)

Malware, short for malicious software, is software designed to infiltrate a computer system without the owner's informed consent. The expression is a general term used by computer professionals to mean a variety of forms of hostile, intrusive, or annoying software or program code.

Though a computer virus itself is one of the types of malware that can reproduce itself, the term is often seen missed to refer to the entire category of malware. Instead Malware can be broadly classified into following main categories.

I. Viruses

Computer virus refers to a small program with harmful intent & has ability to self replicate. Its mode of operation is through appending virus code to an executable file. When file is run, virus code gets executed. The original virus may evolve into new variants by modifying itself as in case of metamorphic viruses. A virus may spread from infected computer to other through network or corrupted media such as floppy disks, USB drives. Viruses have targeted binary executable file (such as .COM, .EXE , PE files in Windows etc.), boot records and/or partition table of floppy disks & hard disk, general purpose script files, documents that contains macros, registry entries in Windows, buffer overflow, format string etc.

II. Worms

Worms are self replicating programs. It uses network to send copies of itself to other systems invisibly without user authorization. Worms may cause harm to network by consuming the bandwidth. Unlike virus the worms do not need the support of any file. It might delete files, encrypt files in as crypto viral extortion attack or send junk email. Example Sasser, MyDoom, Blaster, Melissa etc.

III. Spyware

Spyware is a collective term for software which monitors and gathers personal information about the user like the pages frequently visited, email address, credit card number, key pressed by user etc. It generally enters a system when free or trial software is downloaded.

IV. Adware

Adware or advertising-supported software automatically plays, displays, or downloads advertisements to a computer after malicious software is installed or application is used. This piece of code is generally embedded into free software. The problem is, many developers abuse ad-supported software by monitoring Internet users' activities .The most common source of adware programs are free games, peer-to-peer clients like KaZaa, BearShare etc.

V. Trojans

Trojan horses emulate behavior of an authentic program such as login shell and hijacks user password to gain control of system remotely. Other malicious activities may include monitoring of system, damages system resources such as files or disk data, denies specific services.

VI. Botnet

A botnet is remotely controlled software – collection of autonomous software robots. It is usually a zombie program (Worms, Trojans) under common control on public and private network infrastructure. Botnets are usually used to send spam /spyware remotely. Botnets doesn't sit around on machine (infected machine) waiting for the instruction from a third party instead it looks for the communication with similar instances of bots awaiting instructions.

2.2. MALWARE ANALYSIS AND DETECTION TECHNIQUES

The need for dynamic malware analysis arise from the fact that traditional signature-based malware detection techniques (used by many of the antivirus software) rely on byte sequences, called signatures, in executable for signature-matching. However, Modern malware authors can bypass signature-based scanning by employing the recently emerged technology of code obfuscation for information hiding like packing and emulation.

Malware Analysis and detection method can be classified in to different classes based on the type of malware they are targeting and the specific techniques used for the analysis and detection of the malware. The following figure demonstrates the basic taxonomy of malware analysis and detection techniques.

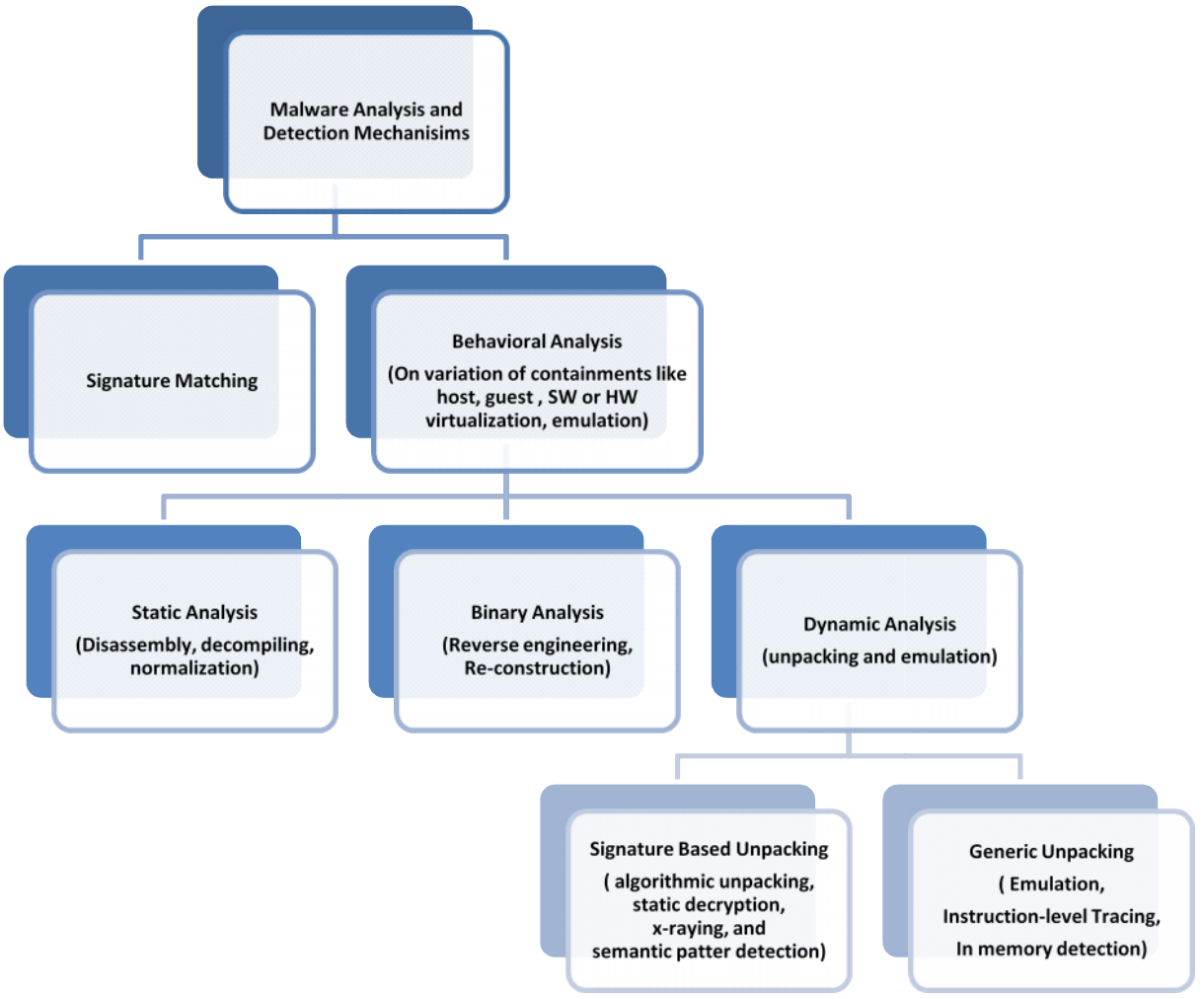

Figure 2.1 Taxonomy of malware analysis mechanisms

2.3. SOFTWARE PACKING AND MALWARE PACKING

Packers are software programs that compress and encrypt other executable files in a disk and restore the original executable images when the packed files are loaded into memories. Packing is applied on legitimate software to reduce the size of executable files and to protect the intellectual property that is distributed with the code. The figure below demonstrates packed executable unpacking.

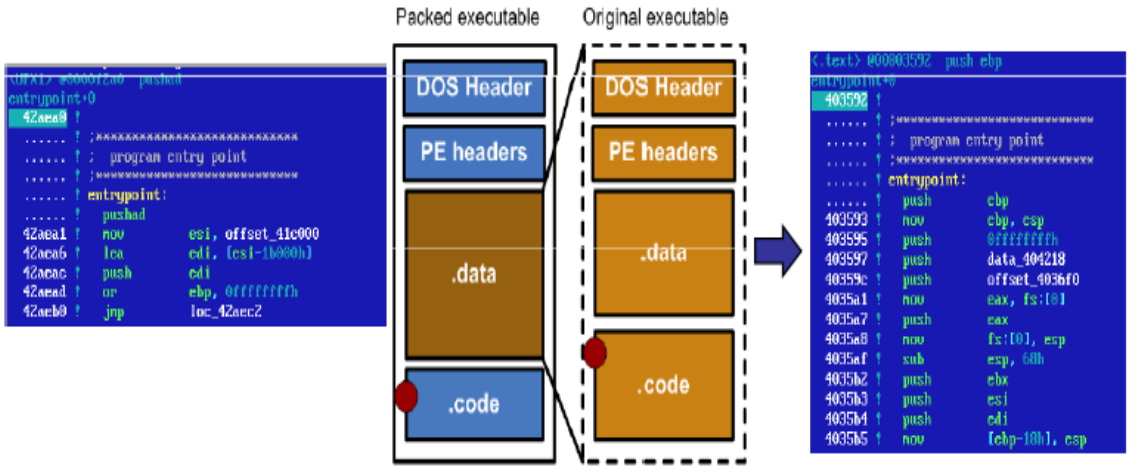

Figure 2.2 Packed executable unpacking demonstration

As mentioned above, packers were first written in order to provide a mechanism to shrink executables so that they take less space to store and less time to transfer over slow channels. Malware writers use this novel idea of packing to bypass signature-based detection, as packing completely modifies the binary foot-print of a program. In packed malware the malicious code resides in executable file in encrypted form, and is not exposed until the moment the executable is run.

2.4. PACKED MALWARE ANALYSIS AND UNPACKING TECHNIQUES

As already discussed in the above paragraphs, a commonly used obfuscation technique these days is packing in which actual code stays hidden till runtime (when the executable is unpacked) making it immune to static analysis. As a result a packed malware analysis should be preceded by unpacking phase in order to reveal the actual semantic of the program code.

Unpacking is the process of stripping the packer layer (or layers) off packed executables to restore the original contents so that AV programs and security researchers can inspect and analyze the original executable signatures. There are many different methods used by malware analyzers and antivirus software to unpack packed malwares. The three common categories of these unpacking methods are discussed below:

2.4.1. STATIC UNPACKING TECHNIQUE

This category of unpacking technique basically rely on the assumption that malware packer use a common standard packing algorithm in which there exists a standard unpacking algorithm to unpack the code. In this kind of unpacking the task of unpacking will only be determining the packer and unpacking of the malware code using the standard unpacking algorithm of that specific packer.

Apart from the difficulty of packer identification, static unpacking technique works well with known packing signatures. It is so obvious that this technique is easy and fast in terms of execution complexity. Moreover static unpacking is system-independent and unpacks without actually running the file, which makes it safe.

However; a very simple modification to the standard packing algorithm makes this technique fail. This is the case in most of modern malware packing tricks; to the extent that they are packed with new brand packing algorithm. The figure below shows a generic model of static unpacking technique:

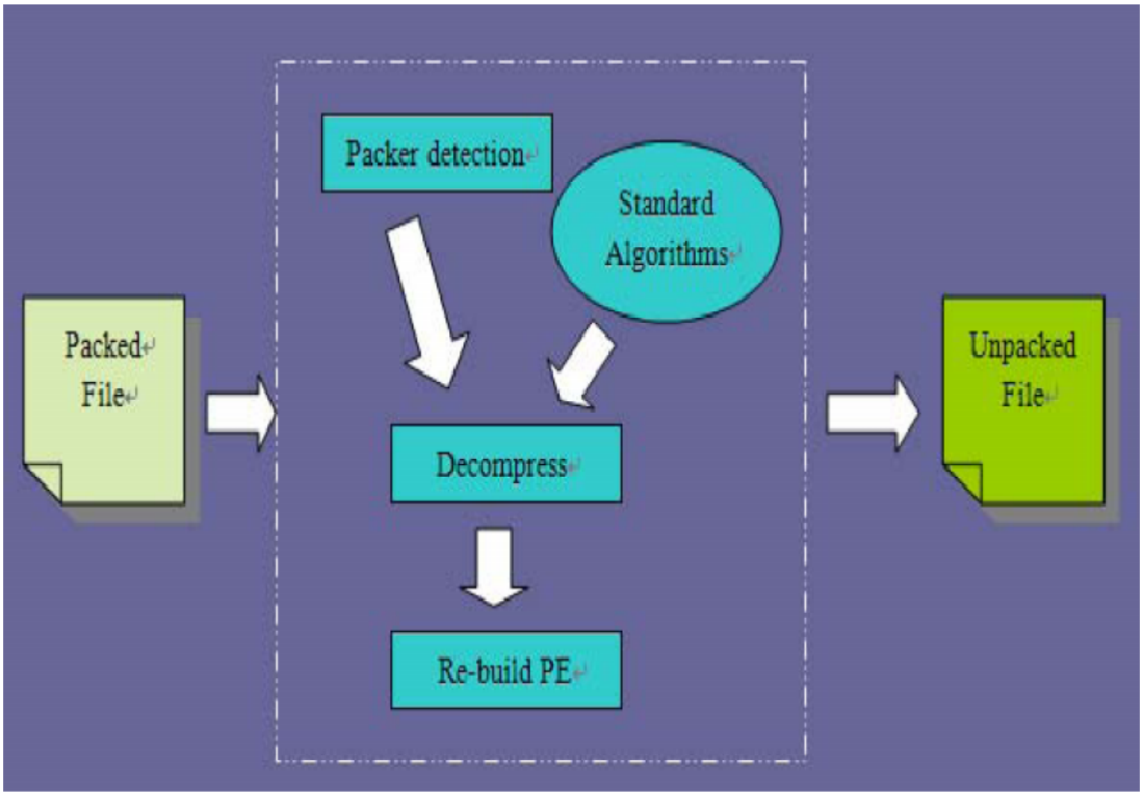

Figure 2.3 Generic model of static unpacking technique

2.4.2. DYNAMIC (LIVE) UNPACKING

Dynamic analysis techniques make use of the fact that no matter what packing technique is applied to the executable, the actual code or its equivalent will ultimately be available in memory sooner or later, and it will execute at some point of run-time. In most case dynamic unpacking let the program run on a real system and unpack itself.

Though they are not exclusive alternatives, there are different mechanism to do dynamic unpacking such as (dynamic unpacking with debugger, run and dump unpacking with memory dumping, emulation and virtual machine based). The following figure demonstrates the generic dynamic unpacking based on virtual environment.

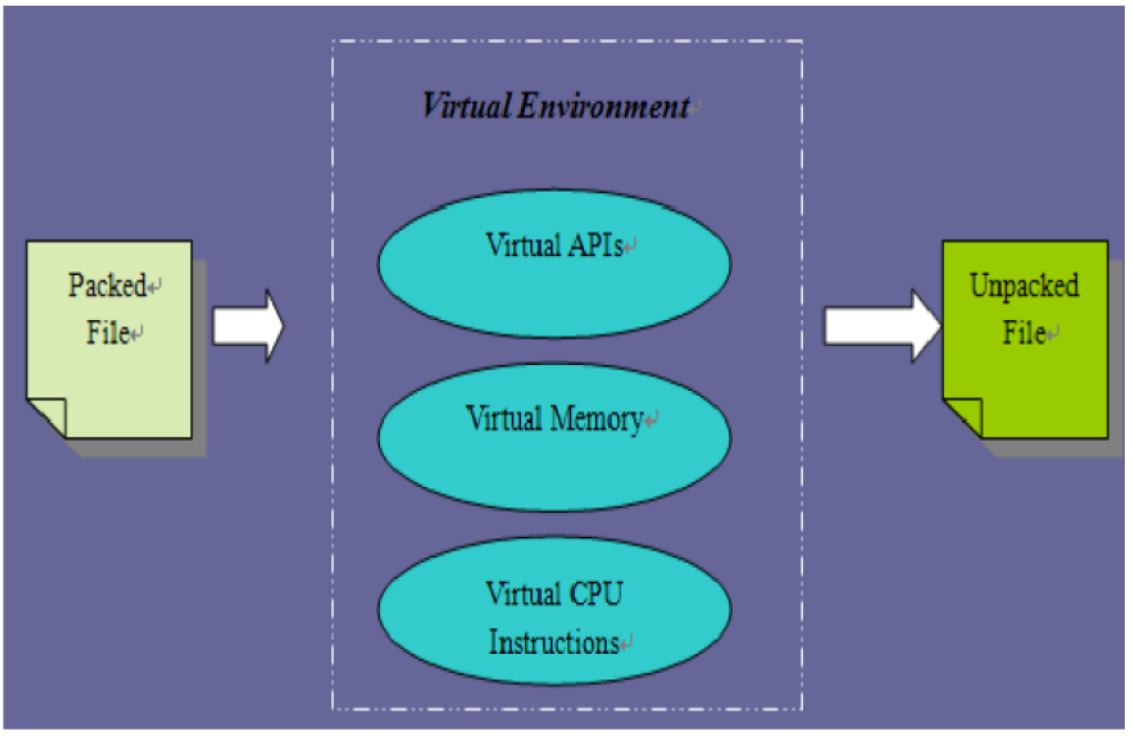

Figure 2.4 Generic model of dynamic unpacking technique based on virtual environment

2.4.3. HYBRID UNPACKING TECHNIQUE

Taking into consideration the weakness of static unpacking and the complexity of dynamic unpacking, it will be a better solution to use static unpacking combined with the dynamic unpacking. The static unpacking can cope with the standard packers rapidly, and the dynamic unpacking could be a complementary part, to handle the modified packers or unknown packer issue.

Though it is obvious that integrating static analysis with dynamic analysis introduces additional complexity to the system, but it is an efficient approach to handle the issues of unknown malware packer's signature.

CHAPTER THREE

3. LITERATURE REVIEW

This section briefly discusses the existing works on analysis of "malware unpacking techniques" with their basic strength and pitfalls followed by review of "state of the art" on currently published works on "malware analysis and detection". Finally, subset of them are selected and presented for further study and analysis.

3.1. EXISTING WORKS ON ANALYSIS OF MALWARE UNPACKING TECHNIQUES

Despite the importance and need of analysis of "malware unpacking techniques" as discussed in the motivation section, very few papers have addressed it in very restricted and limited manner. Among them two very related publications are discussed below:

"Survey on Malware Detection Methods [Vinod P., V.Laxmi, and et. Al., 2008]". In this survey work a series of malware analysis and detection techniques has been well presented. The problems related to traditional signature based analysis and detection method is also highlighted in contrast to that of automated and dynamic techniques.

However, this survey in addition to its being very generic survey it doesn't take into consideration a specific malware unpacking techniques for comparison or benchmarking. This thesis will address this limitation by taking subset of specific malware unpacking techniques compare and benchmark them based on SWOT analysis.

"Generic Unpacking Techniques [Komal B., and Faiza K., *2009]"*. This work tries to assess currently published different categories of dynamic malware analysis mechanisms in its generic and broad sense. Apart from mentioning some specific techniques as an example of each category this work doesn't do any cross comparative analysis within different techniques. It is more survey like work than comparison and benchmarking. Moreover, it doesn't suggest some sort of ordering among the specified example techniques under the categories. We strongly believe that this thesis solves the above limitation by doing cross comparisons and raking between specific malware unpacking techniques

3.2. CURRENTLY PUBLISHED WORKS ON MALWARE ANALYSIS AND DETECTION

Many studies have been carried out and published in the area of malware analysis and detection during the past few years. However, very few of them are mainly targeted on packed malware analysis or unpacking. In this section we will present the set of scientific works collected and selection and brief overview of specific works for further analysis and comparison in the following sections.

The following table presents list of published scientific papers on malware analysis and detection. The remark column of the table gives some remarks on the publication with respect to its importance to this thesis work.

Table 3.1 List of scientific papers on malware analysis and detection

NO	Technique (Paper Title)	SOURCE DETAIL (AUTHOR(S) AND PUBLISHER)	REMARK
1	OmniUnpack: Fast, Generic, And Safe Unpacking Of Malware	Lorenzo Martignoni, Mihai C., and Somesh Jha In Computer Security Applications Conference, **2007**. ACSAC 2007. Twenty-Third Annual , vol., no., pp.431-441, 10-14 Dec. 2007 doi: 10.1109/ACSAC.2007.15	100% Related to the thesis topic Totally about malware unpacking Able to get the implemented unpacker
2	PolyUnpack: Automating The Hidden-Code Extraction Of Unpack-Executing Malware	P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee. In Proceedings of **2006** Annual Computer Security Applications Conference (ACSAC), pages 289–300, Washington, DC, USA, 2006. IEEE Computer Society	100% Related to the thesis topic Totally about malware unpacking Able to get the implemented unpacker
3	AGUnpacker : A Unpacking And Reconstruction System	Yu San-Chao and Li Yi-Chao; Computer Network and Multimedia Technology, **2009**. CNMT 2009. International Symposium on , vol., no., pp.1-4, 18-20 Jan. 2009	100% Related to the thesis topic Totally about malware unpacking Not able to get the implemented unpacker
4	Renovo: A Hidden Code Extractor For Packed Executables	Kang, M. G., Poosankam, P., and Yin, H. In Proceedings of the **2007** ACM Workshop on Recurring Malcode WORM '07. ACM, New York, NY, 46-53. DOI= http://doi.acm.org/10.1145/1314389.1314399	100% Related to the thesis topic Totally about malware unpacking Not able to get the implemented unpacker

5	Ether: Malware Analysis Via Hardware Virtualization Extensions	Dinaburg, A., Royal, P., Sharif, M., and Lee, In Proceedings of the 15th ACM Conference on Computer and Communications Security (Alexandria, Virginia, USA, October 27 - 31, 2008). CCS '08. ACM, New York, NY, 51-62. DOI= http://doi.acm.org/10.1145/1455770.1455779	90% Related to the thesis topic About malware analysis techniques Able to get the implemented malware analyzer
6	Rotalum´e: Automatic Reverse Engineering Of Malware Emulators	Sharif, M., Lanzi, A., Giffin, J., and Lee, In Proceedings of the 2009 30th IEEE Symposium on Security and Privacy (May 17 - 20, **2009**). SP. IEEE Computer Society, Washington, DC, 94-109. DOI= http://dx.doi.org/10.1109/SP.2009.27	90% Related to the thesis topic Totally about reverse engineering of packed and emulated malware Not able to get the implemented tool
7	EERM: Emulating Emulation-Resistant Malware	Kang, M., Yin H., Hanna, S., McCamant, S., and Song In Proceedings of the 1st ACM Workshop on Virtual Machine Security (Chicago, Illinois, USA, November 09 - 09, **2009**). VMSec '09. ACM, New York, NY, 11-22. DOI= http://doi.acm.org/10.1145/1655148.1655151	90 % Related to the thesis topic About malware analysis and defense techniques Not able to get the implemented malware Analyzer

8	MmmBop : Generic Unpacking Of Self-Modifying, Aggressive, Packed Binary Programs	Piotr Bania. http://piotrbania.com/all/articles/ pbania-dbi-unpacking2009.pdf, **2009**	90% Related to the thesis topic About malware analysis Based binary instrumentation Not able to get the implemented tool
9	Automatic Static Unpacking Of Malware Binaries	Coogan, K.; Debray, S.; Kaochar, T.; Townsend, G.; Reverse Engineering, **2009**. Wcre '09. 16th Working Conference On , Vol., No., Pp.167-176, 13-16 Oct. 2009doi: 10.1109/Wcre.2009.24	70 % Related to the thesis topic About malware analysis method No implemented malware Analyzer
10	Malware Obfuscation Detection Via Maximal Patterns	Li, J., Xu, M., Zheng, N., And Xu, J. **2009**. In Proceedings Of The 3rd International Conference On Intelligent Information Technology Application (Nanchang, China, November 21 - 22, 2009). Q. Luo And M. Zhu, Eds. Ieee Press, Piscataway, Nj, 324-328.	60 % Related to the thesis topic About malware analysis method based on patterns matching
11	Eureka: A Framework For Enabling Static Malware Analysis	Harif, M., Yegneswaran, V., Saidi, H., Porras, P., And Lee, In Proceedings Of The 13th European Symposium On Research In Computer Security: Computer Security (Málaga, Spain, October 06 - 08, **2008**). S Doi= Http://Dx.Doi.Org/10.1007/978-3-540-88313-5_31	60% related to the thesis topic About malware unpacking but based on static analysis

12	Cobra: Fine-Grained Malware Analysis Using Stealth Localized-Executions	Vasudevan, A.; Yerraballi, R.; , Security And Privacy, **2006** Ieee Symposium On , Vol., No., Pp.15 Pp.-279, 21-24 May 2006 Doi: 10.1109/Sp.2006.9	60% Related to the thesis topic Malware static analysis and disassembly
13	Run-Time Detection Of Malwares Via Dynamic Control - Flow Inspection	Park, Y., Zhang, Z., And Chen, S. In Proceedings Of The **2009** 20th Ieee International Conference On Application-Specific Systems, Architectures And Processors (July 07 - 09, 2009). Asap. Ieee Computer Society, Washington, Dc, 223-226. Doi= Http://Dx.Doi.Org/10.1109/Asap.2009.30	60% Related to the thesis topic Not about unpacking and more over it is based on control flow inspection
14	Panorama: Capturing System-Wide Information Flow For Malware Detection And Analysis	Yin, H., Song, D., Egele, M., Kruegel, C., And Kirda, E. In Proceedings Of The 14th Acm Conference On Computer And Communications Security (Alexandria, Virginia, Usa, October 28 - 31, **2007**). Ccs '07. Acm, New York, Ny, 116-127. Doi= Http://Doi.Acm.Org/10.1145/1315245.1315261	60% Related to the thesis topic Too general no specific method mentioned
15	Architecture Of A Morphological Malware Detector	Bonfante G., Kaczmarek M., Marion J.Y.: J. Comput. Virol. 5(3), 263–270 (**2008**)	60% Related to the thesis topic No specific unpacking method recommended

16	Malware Behaviour Analysis	Wagener, G., State, R. & Dulaunoy, A. Journal In Computer Virology **2008** Vol. 4(4), Pp. 279-287	60% Related to the thesis topic Too specific, not exactly oriented to packed malware
17	Software Transformations To Improve Malware Detection	Mihai Christodorescu; Somesh Jha ; Johannes Kinder ; Stefan Katzenbeisser ; Helmut Veith Journal In Computer Virology (November **2007**), 3 (4), Pg. 253-265	60% Related to the thesis topic No specific method is recommended
18	Static Analysis Of Executables To Detect Malicious Patterns	Hristodorescu, M. And Jha, S In Proceedings Of The 12th Conference On Usenix Security Symposium - Volume 12 (Washington, Dc, August 04 - 08, **2003**). Usenix Security Symposium. Usenix Association, Berkeley, Ca, 12-12.	30% Related to the thesis topic Patter and signature based static analysis method
19	Pe File Header Analysis-Based Packed Pe File Detection Technique (Phad)	Choi, Y., Kim, I., Oh, J., And Ryou, J. In Proceedings Of The International Symposium On Computer Science And Its Applications (October 13 - 15, **2008**). Csa. Ieee Computer Society, Washington, Dc, 28-31. Doi= Http://Dx.Doi.Org/10.1109/Csa.2008.28	30% Related to the thesis topic Specific to PE and it is static analysis, only to detect if an executable is packed or not

20	Malware Examiner Using Disassembled Code (Medic)	Sulaiman, A.; Ramamoorthy, K.; Mukkamala, S.; Sung, A.H.; , Information Assurance Workshop, **2005**. Iaw '05. Proceedings From The Sixth Annual Ieee Smc , Vol., No., Pp. 428- 429, 15-17 June 2005 Doi: 10.1109/Iaw.2005.1495985	30% Related to the thesis topic Signature based and specific to assembly language instrumentation
21	MapMon: A Host-Based Malware Detection Tool	Shih-Yao Dai; Sy-Yen Kuo; , Dependable Computing, **2007**. Prdc 2007. 13th Pacific Rim International Symposium On , Vol., No., Pp.349-356, 17-19 Dec. 2007 Doi: 10.1109/Prdc.2007.23	30% Related to the thesis topic Static and signature based malware analysis tool
22	Metaaware: Identifying Metamorphic Malware	Qinghua Zhang And Douglas S. Reeves. Computer Security Applications Conference, Annual, 0:411–420, **2007**.	30% Related to the thesis topic Automated Static analysis method
23	Malware Detection Using Adaptive Data Compression	Zhou, Y. And Inge, W. M. In Proceedings Of The 1st Acm Workshop On Workshop On Aisec (Alexandria, Virginia, Usa, October 27 - 27, **2008**). Aisec '08. Acm, New York, Ny, 53-60. Doi= Http://Doi.Acm.Org/10.1145/1456377.1456393	30% Related to the thesis topic Malware analysis based on learning and adaptation

24	A New Generic Taxonomy On Hybrid Malware Detection Technique	Y. Robiah, S. Siti Rahayu, M. Mohd Zaki, S. Shahrin, M. A. Faizal, R. Marliza International Journal Of Computer Science And Information Security, Ijcsis, Vol. 5, No. 1, Pp. 56-61, September **2009**, Usa, Report Number:Issn 1947 5500	30% Related to the thesis topic Signature based and more of analysis
25	The Threat To Identity From New And Unknown Malware	Hodgson, P. W. Bt Technology Journal 23, 4 (Oct. **2005**), 107-112. Doi= Http://Dx.Doi.Org/10.1007/S10550-006-0012-2	30% Related to the thesis topic Discussion only, no specific method recommended
26	A Heuristic Approach For Detection Of Obfuscated Malware	Treadwell, S.; Mian Zhou; , Intelligence And Security Informatics, 2009. Isi '09. Ieee International Conference On , Vol., No., Pp.291-299, 8-11 June **2009** Doi: 10.1109/Isi.2009.5137328	20% Related to the thesis topic Specific to PE files , and static analysis method
27	Semantics-Aware Malware Detection	Christodorescu, M.; Jha, S.; Seshia, S.A.; Song, D.; Bryant, R.E.; , Security And Privacy, **2005** Ieee Symposium On , Vol., No., Pp. 32- 46, 8-11 May 2005 Doi: 10.1109/Sp.2005.20	20% Relate to the thesis topic Based on patter matching (which needs to have signatures detection)

28	A New Malware Detection Method Based On Raw Information	Qiao-Ling Han; Yu-Jie Hao; Yan Zhang; Zhi-Peng Lu; Rui Zhang; , Apperceiving Computing And Intelligence Analysis, 2008. Icacia **2008**. International Conference On , Vol., No., Pp.307-310, 13-15 Dec. 2008 Doi: 10.1109/Icacia.2008.4770030	20% Related to the thesis topic Based on static analysis of raw data Not related to automatic unpacking
29	Research And Implementation Of Compression Shell Unpacking Technology For Pe File	Li Lu; Liu Qiuju; Xu Tingrong; , Information Technology And Applications, **2009**. Ifita '09. International Forum On , Vol.1, No., Pp.438-442, 15-17 May 2009 Doi: 10.1109/Ifita.2009.545	10% Related to the thesis topic Too specific and not exactly related to packed malwares
30	An Improved Clustering Validity Index For Determining The Number Of Malware Clusters	Wang, Y., Ye, Y., Chen, H., And Jiang, Q In Proceedings Of The 3rd International Conference On Anti-Counterfeiting, Security, And Identification In Communication (Hong Kong, China, August 20 - 22, **2009**). Ieee Press, Piscataway, Nj, 544-547.	10% Related to the thesis topic It discuss just about malware clustering

31	Malware Self Protection Mechanism	Alsagoff, S.N.; , Itsim 2008. International Symposium On , Vol.3, No., Pp.1-8, 26-28 Aug. **2008** Doi: 10.1109/Itsim.2008.4631981	10% Related to the thesis topic Too general, not specifically about packed malware unpacking
32	CIMDS: Adapting Postprocessing Techniques Of Associative Classification For Malware Detection	Yanfang Ye; Tao Li; Qingshan Jiang; Youyu Wang; , Systems, Man, And Cybernetics, Part C: Applications And Reviews, Ieee Transactions On , Vol.40, No.3, Pp.298-307, May **2010** Doi: 10.1109/Tsmcc.2009.2037978	10% Related to the thesis topic Not about unpacking and moreover it is based on mining algorithm
33	A Parameter-Free Hybrid Clustering Algorithm Used For Malware Categorization	Han, Z., Feng, S., Ye, Y., And Jiang, Q In Proceedings Of The 3rd International Conference On Anti-Counterfeiting, Security, And Identification In Communication (Hong Kong, China, August 20 - 22, **2009**). Ieee Press, Piscataway, Nj, 480-483.	10% Related to the thesis topic Malware Clustering algorithm and basically analyses malware types
34	Limits Of Static Analysis For Malware Detection	Moser, A.; Kruegel, C.; Kirda, E.; , Computer Security Applications Conference, 2007. Acsac **2007**. Twenty-Third Annual , Vol., No., Pp.421-430, 10-14 Dec. 2007 Doi: 10.1109/Acsac.2007.21	10% Related to the thesis topic Discussion on static analysis limitations

35	A Static Method For Detection Of Information Theft Malware	Jiajing Li; Tao Wei; Wei Zou; Jian Mao; , Electronic Commerce And Security, **2009**. Isecs '09. Second International Symposium On , Vol.1, No., Pp.236-240, 22-24 May 2009 Doi: 10.1109/Isecs.2009.148	10% Related to the thesis topic Static analysis of Information theft, not about packed malware analysis
36	Scenario Based Worm Trace Pattern Identification Technique	S. Siti Rahayu, Y. Robiah, S. Shahrin, Mohd M. Zaki, R. Irda, M.A. Faizal Cryptography And Security (Cs.Cr), International Journal Of Computer Science And Information Security, Ijcsis, Vol. 7, No. 1, Pp. 1-9, January 2010, Usa Report Number:Computer Science Issn 19475500	10% related to the thesis topic Too specific too worm detection method
37	Detection And Prevention Of New And Unknown Malware Using Honeypots	Shishir Kumar, Durgesh Pant Networking And Internet Architecture (Cs.Ni); Cryptography And Security (Cs.Cr) Ijcse Volume 1 Issue 2 **2009** 56-61	10% Related to the thesis topic Too specific network oriented malwares

38	Analysis Of Virus Algorithms	Jyoti Kalyani, Karanjit Singh Kahlon, Harpal Singh, Anu Kalyani Journal Of Computer Science 2 (10): 785-788, 2006 Issn 1549-3636 February 2006	10% Related to the thesis topic More on virus analysis development
39	SBMDS: An Interpretable String Based Malware Detection System Using Svm Ensemble With Bagging	Yanfang Ye; Lifei Chen; Dingding Wang; Tao Li; Qingshan Jiang; Min Zhao Journal In Computer Virology (November **2009**), 5 (4), Pg. 283-293	10% Related to the thesis topic Too specific and not dynamic analysis
34	Auto-Sign: An Automatic Signature Generator For High-Speed Malware Filtering Devices	Gil Tahan Chanan Glezer Yuval Elovici Lior Rokach **2010** 6 Journal In Computer Virology 2 Http://Dx.Doi.Org/10.1007/S11416-009-0119-3 Db/Journals/Virology/Virology6.Html#Tahanger10	10% Related to the thesis topic Automated way signature analysis and generation m

3.3. SELECTED TECHNIQUES FOR FURTHER ANALYSIS

Among the above listed scientific papers, eight of them are selected and listed below for the purpose of further analysis and comparison.

No.	Technique	Remarks
1	**Omniunpack**: Fast, Generic, And Safe Unpacking Of Malware	Year of Publication : 2007 implemented
2	**Polyunpack**: Automating The Hidden-Code Extraction Of Unpack-Executing Malware	Year of Publication : 206 implemented
3	**AGUnpacker** : A Unpacking And Reconstruction System	Year of Publication : 2009 implemented
4	**Renovo**: A Hidden Code Extractor For Packed Executables	Year of Publication : 2007 implemented
5	**Ether**: Malware Analysis via Hardware Virtualization Extensions	Year of Publication : 2008 implemented
6	**Rotalum´e**: Automatic Reverse Engineering of Malware Emulators	Year of Publication : 2009 implemented
7	**EERM:** Emulating Emulation-Resistant Malware	Year of Publication : 2009 Not implemented
8	**MmmBop** : Generic Unpacking of Self-modifying, Aggressive, Packed Binary Programs	Year of Publication : 2009 implemented

Table 3.2 List of selected scientific papers for further analysis and comparison

3.4. BRIEF DESCRIPTION OF THE SELECTED TECHNIQUES

3.4.1. OmniUnpack: Fast, Generic, and Safe Unpacking of Malware

OmniUnpack monitors the execution of a program in real-time and detect when the program has removed the various layers of packing and directly provides the unpacked malicious payload to the malware detector.

The trick here is to monitor the program execution and track written as well as written then-executed memory pages and when the program makes a potentially damaging system call, invokes a malware detector to scan all the written memory pages.

The basic algorithm is, all memory writes and the program counter are tracked. If the program counter reaches a written memory address, it is know that some form of unpacking, self-modification, or code generation occurs in the program. All written-then executed (or written-and-about-to-be-executed) memory locations should then be analyzed by a malware detector.

OmniUnpack is implemented as a kernel driver for Microsoft Windows XP executing on an Intel IA-32 processor, where it simulate non-executable pages in software. It is basically derived from OllyBone, a plug-in for the well known debugger OllyDbg. The Malware detector is based on the ClamAV open source anti-virus. The following figure shows the implementation architecture of OmniUnpack:

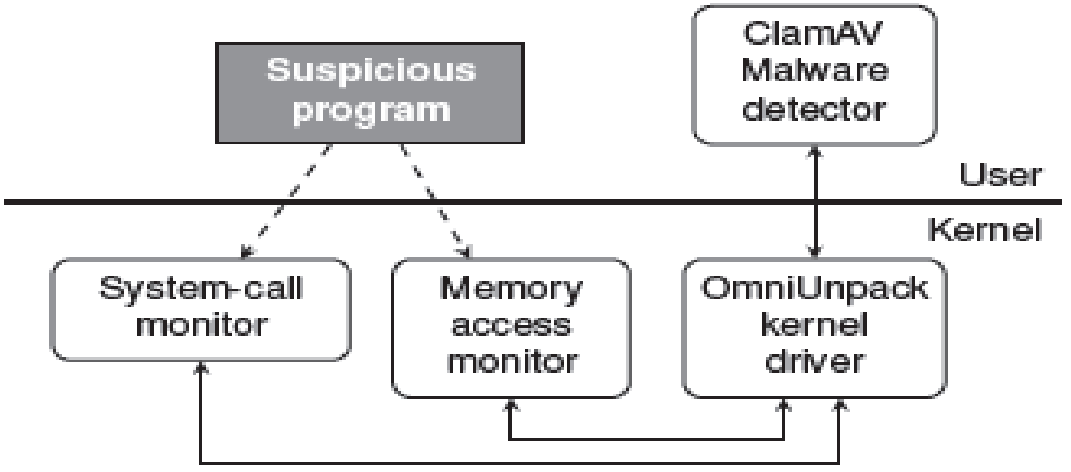

Figure 4.1 Implementation architecture of OmniUnpack

3.4.2. PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing

PolyUnpack proposes a technique for automating the process of extracting the hidden-code bodies of malware. This approach is based on the observation that sequences of packed or hidden code in a malware instance can be made self-identifying when its runtime execution is checked against its static code model.

This work formally defines unpack-execute behavior that malware exhibits and devise an algorithm for identifying and extracting its hidden-code. It also provides details of the implementation and evaluation of the extracting technique.

This approach basically uses trace of behavior of the program execution. Uses a combination of static and dynamic analysis, to automate the process of extracting the hidden-code of unpack executing malware. Then finally implemented an EXTRACT UNPACKED CODE tool (i.e., PolyUnpack) as a command-line tool that operates over x86 Microsoft Windows executables. The following figure shows the implementation architecture of PolyUnpack:

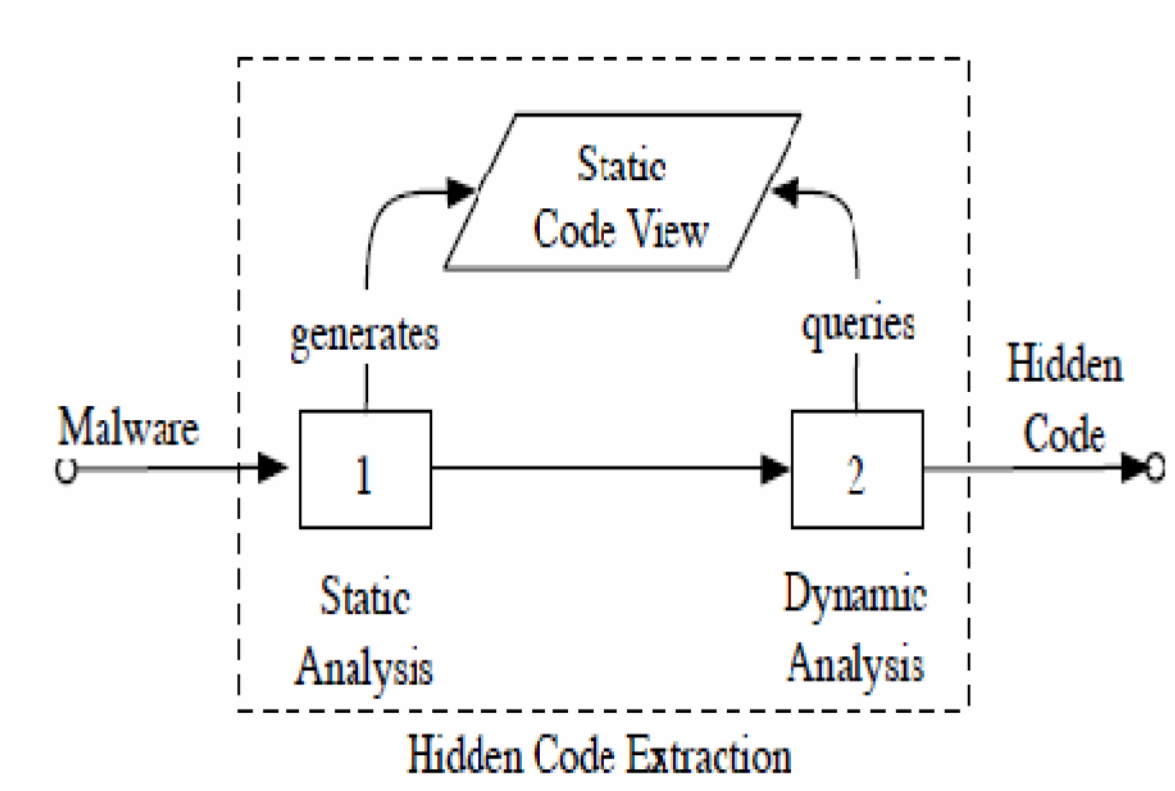

Figure 4.2 Implementation architecture of PolyUnpack

3.4.3. AGUnpacker : A Unpacking and Reconstruction System

AGUnpacker, based on primary behaviors of packing which are code obfuscation, PE formats modification and Anti-technique, proposes a solution Automatic and Generic unpacker (AGUnpacker) to extract and reconstruct the hidden code of packed malware.

For code obfuscation, AGUnpacker decides when the object program has decrypted itself completely in memory on the basis of stack balance role, intersection jump role and the characteristics of entrance. For PE formats modification, after locating Import Address Table (IAT) by monitoring all of the call instructions, a forensics tracing technique to restore the items in IAT, which are unmatched with Export Table items of DLL, is presented to obtain a runnable binary. In order to bypass anti-technique, the system is implemented by taking over exceptions through common ways and finally reconstructs an executable routine.

The implementation architecture of AGUnpacker consists of three main components: Packer Detector, Magic Jump Detector and Reconstruct Component. The figure below shows the interaction between these three components and the packed object program:

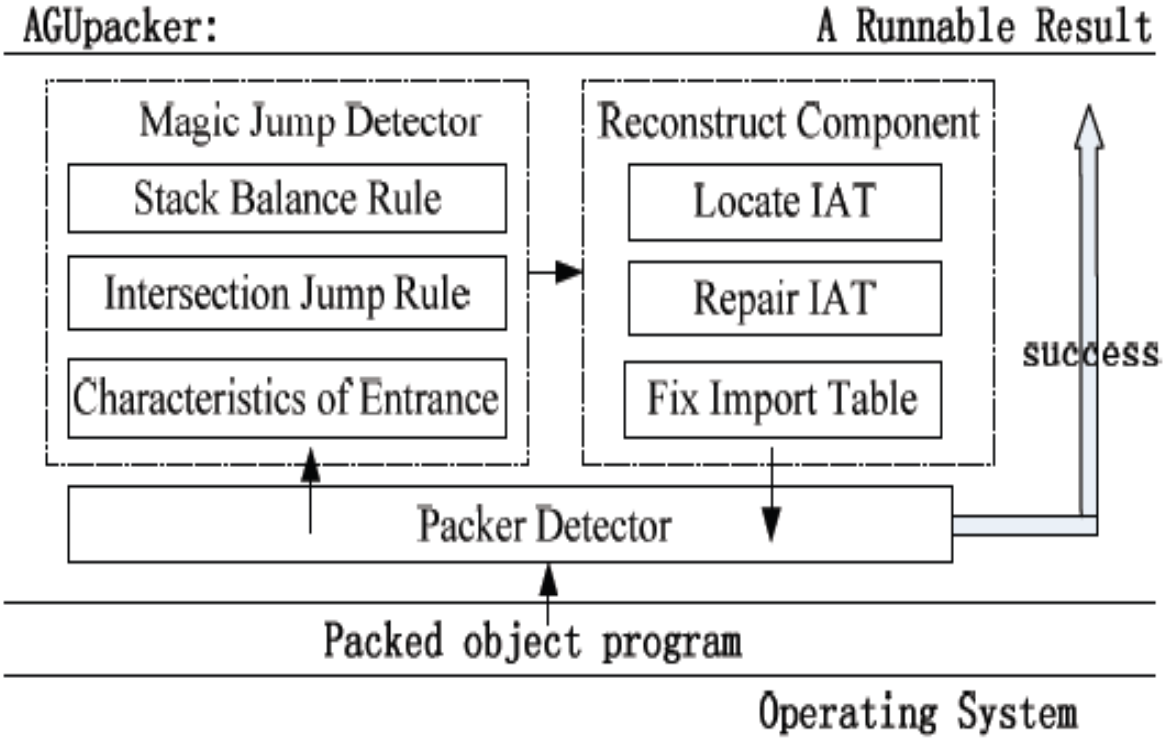

Figure 4.3 Implementation architecture of AGUnpacker

3.4.4. Renovo: A Hidden Code Extractor for Packed Executables

Renovo is a fully dynamic approach that captures an intrinsic nature of hidden code execution and additional information useful for further analysis that the original code should present in memory and executed at some point at run-time.

This approach monitors program execution and memory writes at run-time, determines if the code under execution is newly generated, and then extracts the hidden code of the executable. In addition to extracting the hidden code, this technique can also provide additional information on the packed binaries.

Finally implement and a tool Renovo, an automated framework for extracting hidden code, and evaluate it with a large number of real-world malware samples. Renovo is built on top of TEMU, which is a dynamic analysis component of the BitBlaze binary analysis platform. The following figure shows the architecture and assumption taken into consideration to develop Renovo.

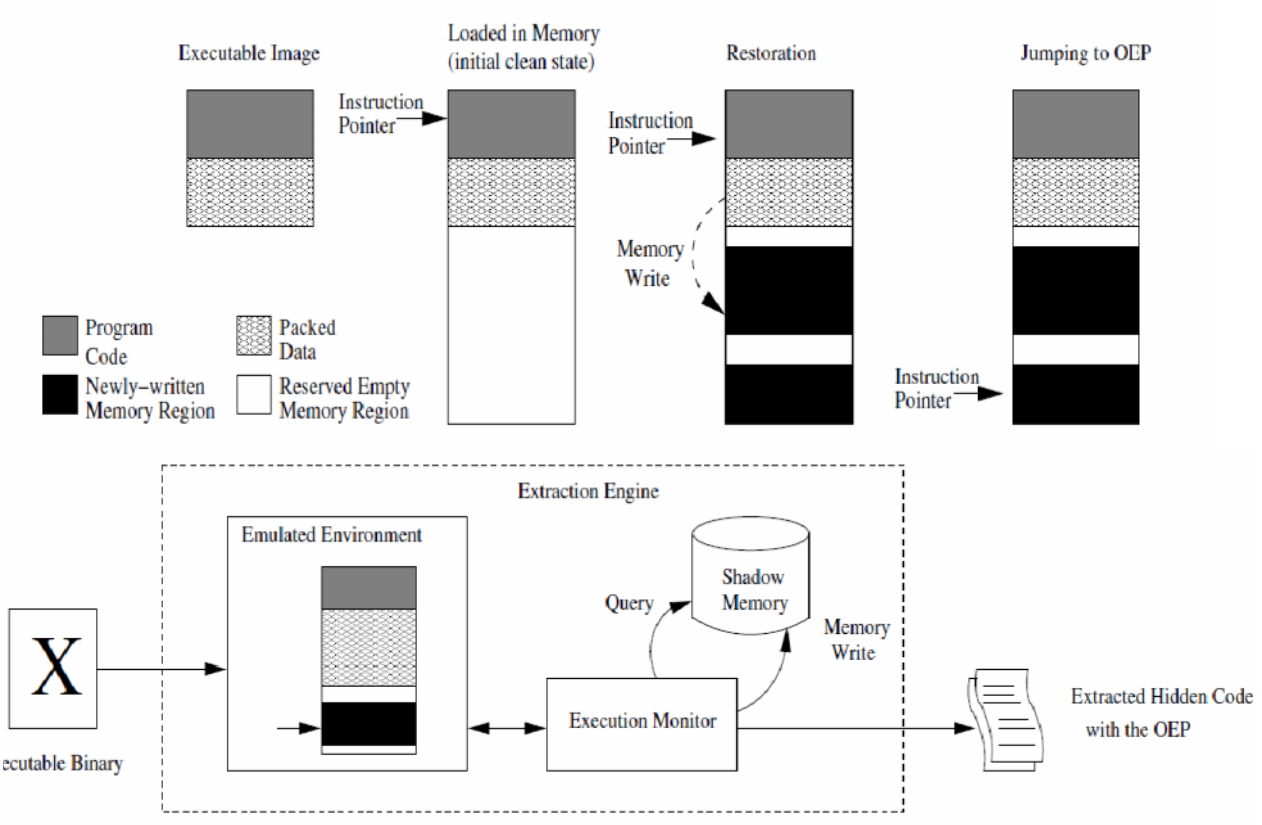

Figure 4.4 Implementation architecture of Renovo

3.4.5. Ether: Malware Analysis via Hardware Virtualization Extensions

Ether is a transparent and external approach to malware analysis, which is motivated by the intuition that for a malware analyzer to be transparent, it must not induce any side-effects that is unconditionally detectable by malware.

Ether, is based on application of hardware virtualization extensions, and resides completely outside of the target OS environment. Thus, there are no in-guest software components vulnerable to detection, and there are no shortcomings that arise from incomplete or inaccurate system emulation.

This approach basically first formally defines transparency requirements, i.e. obtaining an execution trace of a program identical to that if it were run in an environment with no analyzer present. Then implements the requirements with help of above mentioned hardware virtualization extensions.

The implementation of ether is based on architecture with an environment of hardware virtualization extension Intel VT, and software that can utilize hardware virtualization extensions, i.e. Xen hypervisor version 3.1.0. Target operating system Windows XP (Service Pack 2). The Following figure shows the implementation architecture of ether:

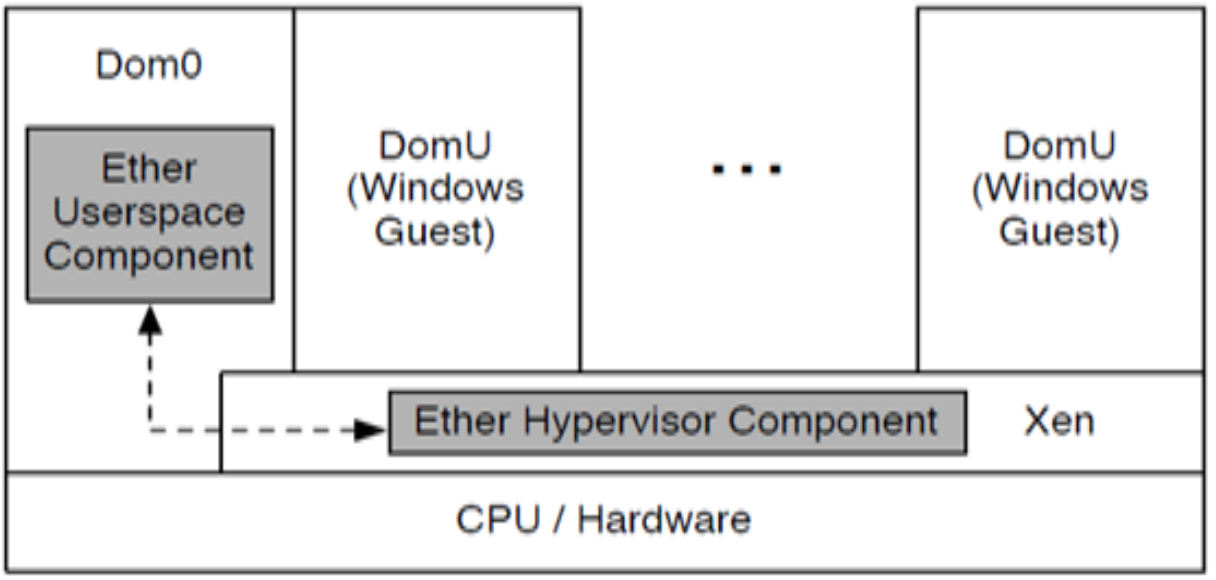

Figure 4.5 Implementation architecture of Ether

3.4.6. Rotalum´e: Automatic Reverse Engineering of Malware Emulators

Rotalum´e is the first work in automatic reverse engineering of malware emulators based on dynamic analysis. Rotalum´e executes the emulated malware in a protected environment and record the entire x86 instruction trace generated by the emulator.

Rotalum´e basically analyses the trace of program execution to extract the syntactic and semantic information about the byte code instruction set. Finally, it generates data structures of the analysis output, which provide the foundation for subsequent malware analysis.

Also Implemented a proof-of-concept system called Rotalum´e and evaluated it using both legitimate programs and malware emulated by VMProtect and Code Virtualizer. Rotalum´e accurately reveals the syntax and semantics of emulated instruction sets and reconstructs execution paths of original programs from their bytecode representations.

Rotalum´e uses a QEMU based component to perform dynamic analysis. The output of system is the extracted syntax and semantics of the source bytecode language suitable for subsequent analysis using traditional malware analyses. The following figure shows the implementation architecture of Rotalum´e:

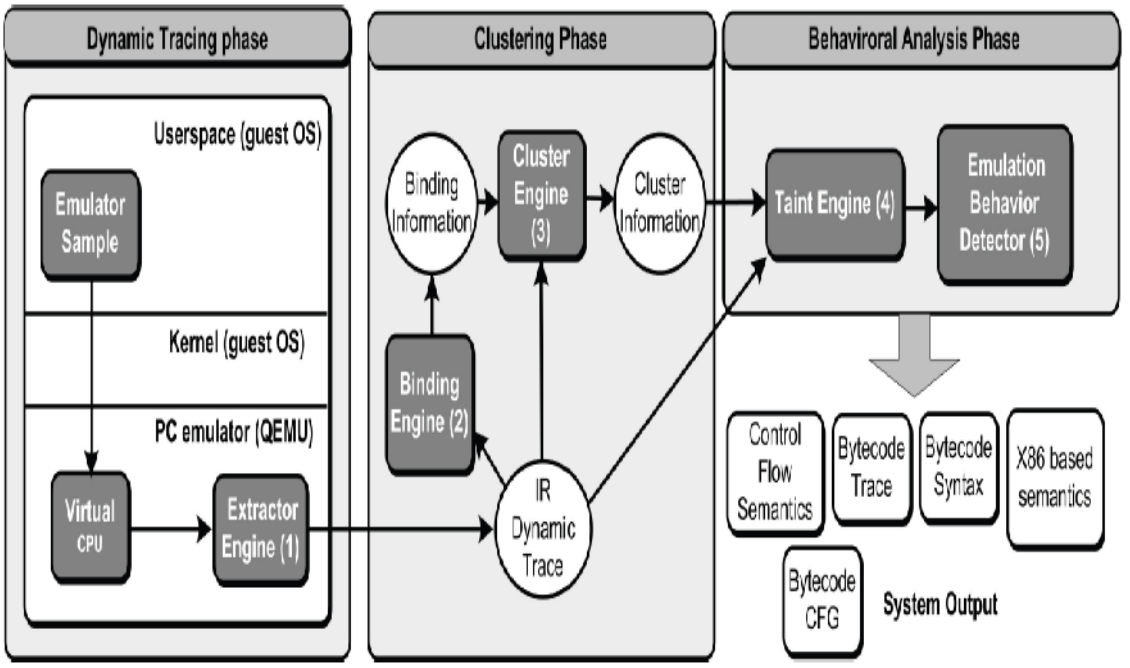

Figure 4.6 Implementation architecture of Rotalum´e

3.4.7. EERM: Emulating Emulation-Resistant Malware

EERM is an automated technique to dynamically modify the execution of a whole-system emulator to fool a malware sample's anti-emulation checks.

This approach uses a trace matching algorithm to locate the point where emulated execution diverges, and then compares the states of the reference system and the emulator to create a dynamic state modification that repairs the difference and fools the malware anti emulation check.

It is implemented in the form of enhancements to, the popular open-source whole-system emulation system, QEMU and virtual execution system based on Intel VT hardware virtualization. The following figure shows the implementation architecture of EERM:

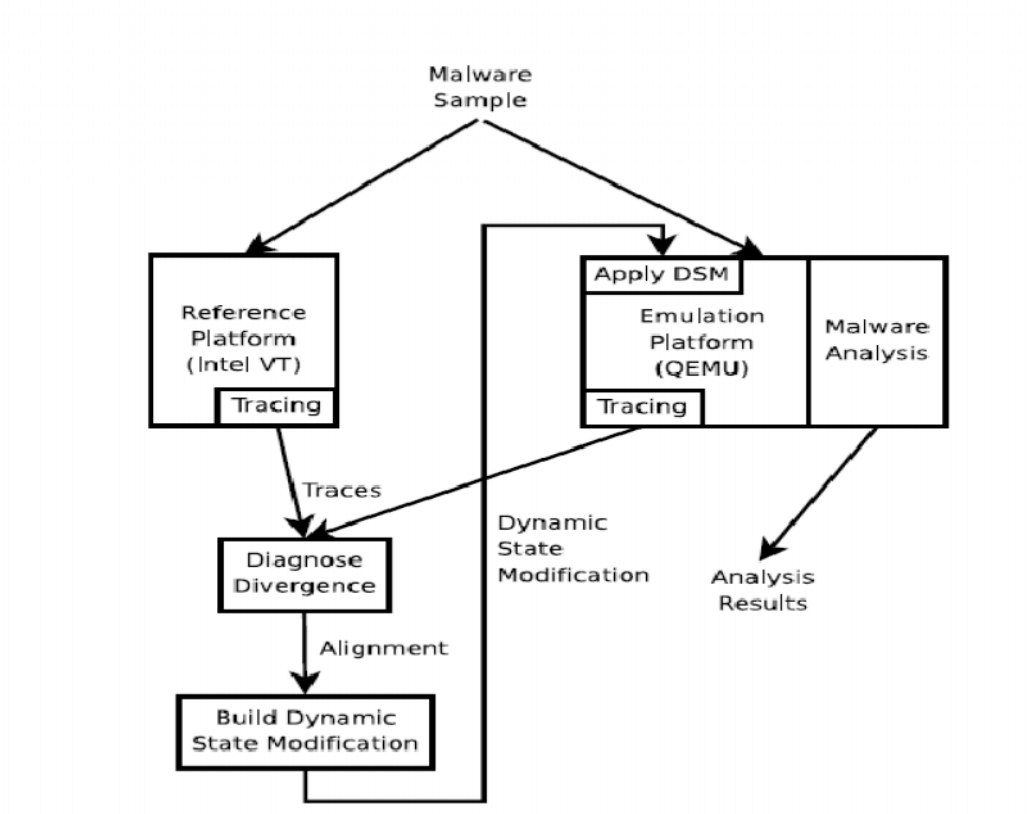

Figure 4.7 Implementation architecture of EERM

3.4.8. MmmBop: Generic Unpacking of Self-modifying, Aggressive, Packed Binaries

This technique proposes MmmBop as a relatively new concept of using dynamic binary instrumentation techniques for unpacking and by-passing detection, by self-modifying and highly aggressive packed binary code. Present the method for by-passing packed, obfuscated, armored layers and a couple of methods for finding original entry point (OEP).

Implementation of MmmBop consists of two separate modules: Injector and DBI-Engine. MmmBop supports the IA-32 architecture and it is targeted for Microsoft Windows XP, some of the further deliberations will be referring directly to this operating system. The following figure shows the implementation architecture of MmmBop:

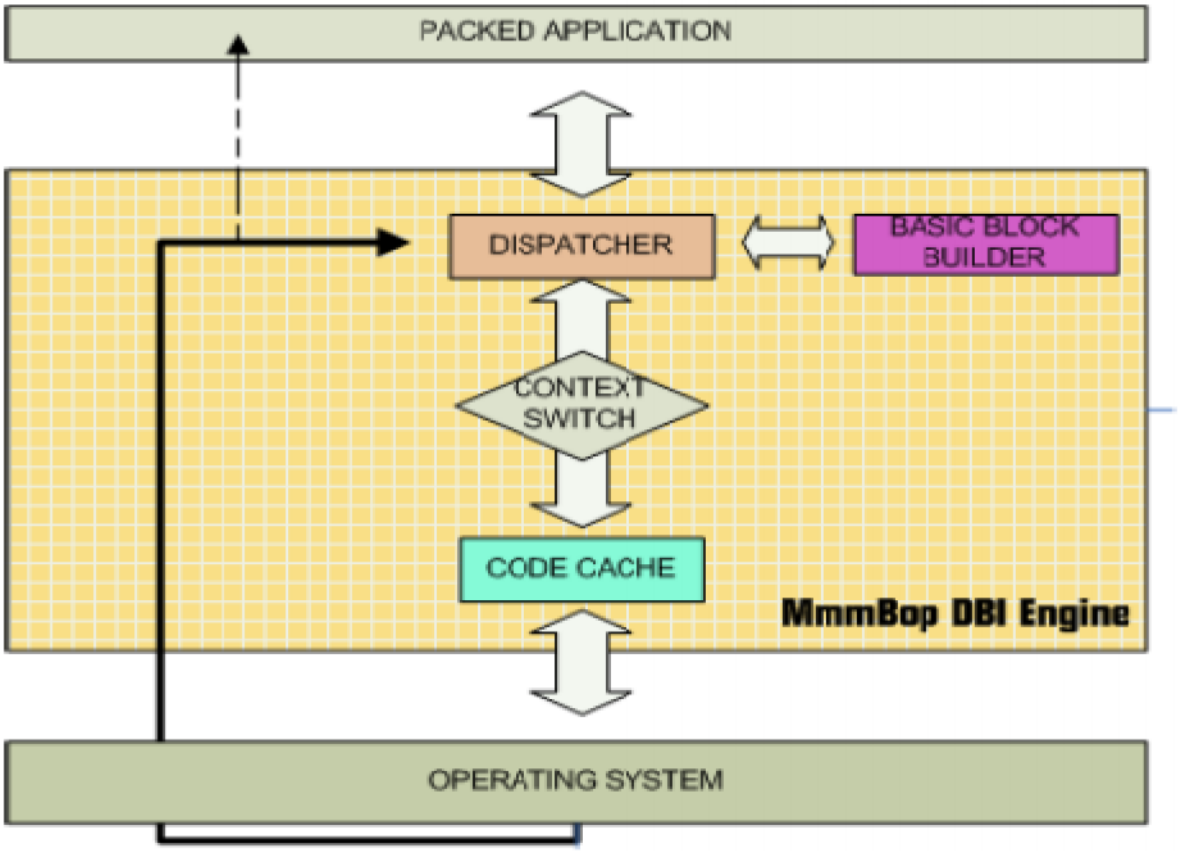

Figure 4.8 Implementation architecture of MmmBop

CHAPTER FOUR

4. ANALYSIS OF THE UNPACKING TECHNIQUES

This section is the important chapter of the thesis which presents the analysis detail of the malware unpacking techniques short listed and discussed in the previous chapter. First part presents a tabulated analysis of each technique with critical drivers, such as main target (considerations), specific technique used, experiments done and summary of contributions done by each of the specific technique. Based on these drivers the techniques are categorized to different type of malware detection and analysis categories discussed in the previous sections.

The second part of this chapter discusses the SWOT analysis detail in which the strength, weakness, opportunities and threats of each of the selected malware unpacking techniques are presented. Finally, it presents categorization, comparative raking and benchmarking of the techniques.

4.1. ANALYSIS BASED ON CRITICAL DRIVERS

Table 4.1 Tabulated analysis of each technique with respect to critical drivers

No.	Technique (Paper title)	Consideration (Main target)	Specific Technique Used	Specific Requirements	Experiments	Contributions
1.	**OmniUnpack: Fast, Generic, and Safe Unpacking of Malware**	Packed malicious programs Both compressed and encrypted malwares Single step, or multiple times unpacks Self modifying malwares (polymorphic/ metamorphic) Known and unknown packers	Execution of program in a contained yet accurate environment Real time tracking of written as well as written then-executed memory pages Blacklisting of System calls as **potentially damaging system call** for detection trigger Use of non-executable pages or equivalent hardware mechanisms for memory monitoring Invoking of Malware detector only once to scan all written memory pages	System with virtual memory capabilities Hardware to manages the translation between virtual and physical addresses and memory-protection facilities at the page level Customization of ClamAV, open source anti-virus, for malware detection Specific Requirements of Signatures from the malware detector	Experimental evaluation using: - both known and unknown packers - and random sample of self modifying code A comparative experiment with PolyUnpack, and ClamAV Unpacker - Significantly faster - Handled 80% of packed malware - Relatively smaller overhead	A General-purpose unpacker In-memory malware detection strategy Set of experimental results A full pseudo code of the unpacker algorithm

No.	Methods (Paper title)	Consideration (Main target)	Specific Technique Used	Specific Requirements	Experiments	Contributions
2.	**PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware**	Packed malware with hidden-code bodies	Behavior analysis based approach, using a combination of static and dynamic analysis	Knowledge of the instance's static code model	On several thousands of malware binaries	Formal description of unpack-executing programs
		Encrypted and polymorphic viruses	Tracing the unpack-executing behavior of malware and extracting the hidden-code bodies	Sterile, isolated environment for malware execution part	On samples that exhibited a wide variety of unpack-execute behavior	Algorithm for behavior-based hidden-code extraction
		Unpack executing based malware	Checking runtime execution of the malware code against static code model to indentify unpacking.	Customized x86 Microsoft Windows command-line interaction	A comparative Experiments using ClamAV and McAfee Antivirus	Implementation of an extraction tool PolyUnpack
					Demonstrate a good reduction in false negatives and shows that PolyUnpack identifies more unpack executing malware	Implementation and use of a framework for testing technique against large sets of malware samples.

No.	Methods (Paper title)	Consideration (Main target)	Specific Technique Used	Specific Requirements	Experiments	Contributions
3.	**AGUnpacker : A Unpacking And Reconstruction System**	Malware with - Code obfuscation composed of packing or encryption	Complete decryption control on the basis of: - Stack balance role analysis, - Intersection jump role - Entrance characteristics	No specific and critical requirement specified.	On suspect repository which was composed 10 different types of packers including unknown (custom) Packers	- Comprehensive Summary of packers behaviors and its different categories
		- PE formats modification specifically IAT	Import Address Table (IAT) monitoring and matching with Export Table items of DLL		A sample set of 150 versions or different object software of each packer	- An Automatic and Generic Unpacker (AGUnpacker).
		- Anti-technique mechanisms like anti debugging, anti virtual and anti reverse engineering)	Careful handling of exceptions in order to bypass the anti technique methods imposed by the malware		Results show that it is faster than existing unpackers like PolyUnpack	- Details of Magic Jump Detector and Reconstruct Component - Detail of Experimental evaluation results

No.	Methods (Paper title)	Consideration (Main target)	Specific Technique Used	Specific Requirements	Experiments	Contributions
4.	**Renovo: A Hidden Code Extractor for Packed Executables**	Malwares with Anti-reverse engineering techniques	Trace and analysis of binary code to identify and extract the hidden packed code	Emulated environment for the purpose of tracing execution	On large number of real-world malware samples	- Fully dynamic unpacking method which monitors currently-executed instructions and memory writes at run-time.
		Packed executables both compressed and encrypted	Capturing and analysis of an intrinsic nature of the program execution	Full access to shadow memory from the host Operating System	Malware samples with more than one hidden layer.	- A mechanism to extract additional information useful for further code analysis
		Detection of novel samples and modified packing techniques.	Generating a memory map and monitoring for possible write followed jumps to marked memories locations Monitoring of execution from the outside (host), consulting the shadow memory of the process, determine if a hidden code is being executed.		Synthetic sample programs generated by using 14 different packing tools. A comparative experiment with Universal PE Unpacker and PolyUnpack	- Implementation and evaluation result of Renovo (an automated framework for hidden code extraction)

No.	Methods (Paper title)	Consideration (Main target)	Specific Technique Used	Specific Requirements	Experiments	Contributions
5.	**Ether: Malware Analysis via Hardware Virtualization Extensions**	Packed malwares with some anti-technique mechanisms	Transparent and external approach based on hardware virtualization extensions.	- Higher Privilege to the analyzer	On 25,000 recent malware samples, in which ether remain transparent	Framework for describing program execution
		Malware with myriad of anti-debugging, anti-instrumentation, and anti-VM	Hiding the possible side-effects that are unconditionally detectable by malware	- Able to manage non-privileged side effects	A comparative Experiment with Renovo and PolyUnpack	Framework for analyzing the requirements for transparent malware analysis.
		Transparency (able hide a malware analyzer from analyzed malware)	Use of higher privilege over the OS kernel in the virtual machine	- Transparent Exception Handling mechanism		Implementation of Ether, an external, transparent malware Analyzer
				- Able to provide an Identical Measurement of Time		Broad-scale evaluation of current approaches (Copies of discrete samples referenced in the paper and the 25,000 malware sample)

No.	Methods (Paper title)	Consideration (Main target)	Specific Technique Used	Specific Requirements	Experiments	Contributions
6.	**Rotalum´e : Automatic Reverse Engineering of Malware Emulators**	Packed and Emulated Malwares	Execution of the emulated malware in a protected environment and record the entire x86 instruction Followed by dynamic data-flow and taint analysis and extract the syntactic and semantic information about the bytecode instruction set Multi-path exploration, across the bytecode program Identifying the fundamental characteristic of decode-dispatch emulation	Protected guest OS environment(QEMU) Two important requirements for the run-time environment for the dynamic tracing phase: - Instruction-level tracing, - Isolation from malware and attacks. A customizable Traditional malware detector.	Evaluated on legitimate programs and on malware emulated by VMProtect and Code Virtualizer. Synthetic and real test program obfuscated with emulation. Able to identify and reconstruct the bytecode buffers in the emulated malware	Formulate the research problem of automatic reverse engineering of malware emulators. Framework and working prototype system to identify: - Candidate memory regions containing bytecode - Dispatch and instruction execution blocks - Method for discovering bytecode instruction syntax and semantics.

No.	Methods (Paper title)	Consideration (Main target)	Specific Technique Used	Specific Requirements	Experiments	Contributions
7.	**EERM: Emulating Emulation-Resistant Malware**	Malwares with anti emulation (emulation-resistance) techniques Transparency of malware emulators	Creation of a reference platform and trace matching between the emulator and the reference platform Dynamic state modification of the emulator based on the above result	Fully accessible and traceable reference platform. Hardware virtualization in single step mode	On real malware samples collected from the wild, with anti-emulation technique By building an implementation into an emulator used for malware analysis	- Specific formulation of the problem of how to ameliorate anti-emulation checks - Framework of amelioration approach - Implementation of the technique

No.	Methods (Paper title)	Consideration (Main target)	Specific Technique Used	Specific Requirements	Experiments	Contributions
8.	**MmmBop: Generic Unpacking of Self-modifying, Aggressive, Packed Binary Programs**	Self-modifying and highly aggressive packed binary code Packed, obfuscated, malwares with armored layers Finding the hidden original entry point (OEP) and bypassing of anti-technique mechanisms Reduction of packer detection false positive	Tracing the execution flow of malware based dynamic binary instrumentation	No specific and critical requirement specified.	Samples collected from different well know malware packers	- A formal description of Methods for by-passing packed, obfuscated, armored layers - A couple of methods for finding original entry point (OEP) - MmmBop unpacker

4.2. CLASSIFICATION

Classification of the selected techniques to the "malware detection and analysis taxonomy", discussed above, will help us to cluster the techniques to some categories in the following sections. The following table shows the classification to which the selected techniques belong:

<table>
<tr><th colspan="5">Malware Analysis and detection mechanisims</th></tr>
<tr><th rowspan="2">Signature matching</th><th colspan="4">Behavioral Analysis</th></tr>
<tr><th>Static Analysis</th><th>Binary analysis</th><th colspan="2">Dynamic analysis</th></tr>
<tr><td></td><td></td><td></td><td>Signature based unpacking</td><td>Generic unpacking</td></tr>
<tr><td></td><td>• Polyunpack *</td><td>• Rotalum´e</td><td>• ASUM</td><td>• Polyunpack*
• Renovo
• Ether
• MmmBop
• OmniUnpack
• EERM
• AGUnpacker</td></tr>
</table>

Table 4.2 Classification of the selected techniques to malware detection taxonomy

4.3. SWOT ANALYSIS

This section presents the SWOT analysis of the techniques, which will make us able to have cross comparison between the techniques based on the strength, weakness, opportunities and threat of each of the techniques. The following table illustrates the SWOT analysis detail.

Table 4.3 SWOT analysis of each technique

NO	Methos (Paper title)	Strengths (Best features)	Weaknesses (Limitations)	Opportunities	Threats
1.	**OmniUnpack: Fast,Generic, and Safe Unpacking of Malware**	Being generic unpacker and able to: - Unpack both known and unknow packers - Both compressed and encrypted malware - Self modifying malwares (polymorphic/ metamorphic) Able to detect various levels of unpacking and self modification layers Postponed of malware detection scan until a dangerous system call is invoked Malware Detction at the end of all unpackign layer only and Low overheads during unpacking (due to page level trace) Integrates with any malware detection engine, any operating system (basically with the requirments) Runs the program on the native OS, does not use debugging, virtual machine, or emulation mechanisims Resilient to antidebugging, anti-VM, anti-emulation and SEH attacks	The performance and efficiency of the unpacker depends on the efficiency of the malware detector Only works for running processes, but is not suitable for at-rest file scanning Tough, it reduce overhead suffer from the imprecision of page-level tracking, insteade of istraction level tracking Consideration of a dangourse system call as end of unpackign indicator Choice and listing of dangerous system calls might be incomplete Memory access exception as indication of unpack, reduce transparency to the malware Continuous scanning and monitoring overhead	The availablity of software solution to monitor memeory using non-executable pages (equivalent hardware mechanisms) Reuse of concepts of OllyBone, a plugin for debugger OllyDbg, for pagelevel break-on-execute A lesson from PaX PAGEEXEC for the purpose of tracking memory write and execute accesses	Malware writers able to evade malicious payload in layers of compression or encryption. Termination of the unpacking routine is undecidable Malwares with multiple processes Automatic generation of signatures that satisfy the requirements imposed

NO	Methos (Paper title)	Strengths (Best features)	Weaknesses (Limitations)	Opportunities	Threats
2.	**PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware**	Reduce the time required to analyze packed malware	Single step debugging(tracing) repeatedly queried overhead since malware's execution is paused after each instruction	The possiblity to execute instruction in a sterile, isolated environment	Programs call to Dynamic Link Library (DLL)
		Improve the performance of malware detection tools	Attachment of an instruction-execution time out or bound n in EXTRACT UNPACKED CODE	Windows API calls to single-step execute a program;	Accurate and successful is disassembly not always easy
		Uses of combination of static and dynamic analysis	The way it handles multiple level packing not effective	Static and dynamic disassembly using a 80x86 32-bit disassembler library.	Disassembly of variable length instructions and non-code regions of malware
		Pretesting of packed and not packed programs for efficiy purpose	Not transparent to malware so that it can easy be attacked by malwares with anti debugging or anti trace mechanisms	For outputting the complete executable version used memory dumper	
		Able produce an output of a plain-text disassembly of the unpacked code, a binary dump of the code, or a complete executable version	Not good(fast) for interactive users, and not Resilient to anti-debugging and SEH attacks		

NO	Methos (Paper title)	Strengths (Best features)	Weaknesses (Limitations)	Opportunities	Threats
3.	**AGUnpacker : A Unpacking And Reconstruction System**	Can handle both known and unknown packer independent of packing algorithms Result binary can run and be analyzed dynamically Faster than existing unpackers by reducing unpacking time effectively Able to recover the PE architecture and restore the original Import Address Table (IAT) unlike many of the other packers AGUnpacker can improve the performance and effectiveness of unpacking significantly	The assumptions taken for candidate identification of Magic Jump is weak (too Specific to some conditions) Not applicable in case of more complicated PE format modifications done by malware packers	Usage of virtual memory for protection exception control and candidate indetification.	Implementation limitations of the unpacker Undecidability of Magic Jump and time-cost

NO	Methos (Paper title)	Strengths (Best features)	Weaknesses (Limitations)	Opportunities	Threats
4.	**Renovo: A Hidden Code Extractor for Packed Executables**	In addition to extracting the hidden code, Provides additional information (like OEP, layering mechanism used, and so on) from the packed binaries.	As stated in the paper "Dealing with emulate malwares is beyond the scope of this paper"	Emulated environments with virtually Merrories	Emulated malware, circumventing the emulated environment
		Does not depend on the program specific disassembly or the known signatures of packing techniques used by malware	Always assumes that hidden code should eventually be written and executed at run-time	Full access to shadow memory from the host OS	Exploiting the time-out, a possible threat to Renovo
		Able to Extract information on multiple hidden layers and handle any sort of packing techniques applied to the binaries	Since it runs on emulated environment, it suffers from anti-emulation malwares.	TEMU to reason about OS-level semantics	

NO	Methos (Paper title)	Strengths (Best features)	Weaknesses (Limitations)	Opportunities	Threats
5.	**Ether: Malware Analysis via Hardware Virtualization Extensions**	Being transparent and external approach to malware analysis Completely outside of the target OS, so that no in-guest software components vulnerable to detection, and there are no shortcomings that arise from incomplete or inaccurate system emulation	Specific platform or Hardware Requirements Current implementation has still threats and limitation of Hardware Architecture to deal with	Hardware virtualization extensions such as Intel VT Software that can utilize hardware virtualization extensions, i.e. Xen hypervisor	Current architectural restrictions - Intel VT flushes the TLB on every VMExit - Memory hierarchy detection methods

NO	Methos (Paper title)	Strengths (Best features)	Weaknesses (Limitations)	Opportunities	Threats
6.	**Rotalum´e: Automatic Reverse Engineering of Malware Emulators**	Able to reveal the syntax and semantics of emulated instruction sets and reconstructs execution paths of original program Using dynamic analysis, Able to extract execution paths in the bytecode program and the syntax and semantics of the bytecode instructions used in those paths Automatic reverse engineering of unknown malware emulators.	Assumes a decode-dispatch emulation model only, thus, malware authors may implement variations or alternative approaches Malware using decode-dispatch emulation may attempt to evade accurate analysis by targeting specific properties of the analysis Limitation of utilizing the discovered syntax and semantics to completely convert bytecode to native instructions	Code protection tools such as Code Virtualizer and VMProtect	Malware emulators with sophisticated approaches: - using a threaded approach - dynamic translation based emulation Recursive emulation, which converts the emulator itself to another bytecode language and introduces an additional emulator to emulate it.

NO	Methos (Paper title)	Strengths (Best features)	Weaknesses (Limitations)	Opportunities	Threats
7.	EERM: Emulating Emulation-Resistant Malware	Able to clearly determines the root cause of differing behavior and a way to ameliorate it automatically	Over head of creating reference platform and whole-system emulation	whole-system analysis environment TEMU	Attacks for which there is no reference platform
		Allows a general-purpose emulator, with a relatively minimal runtime Overhead	Cost of diagnosis, due to the alignment algorithm	Intel XED library to disassemble and obtain an operand list for each instruction, necessary for accurate slicing	Too many divergence points, and obfuscation of data flow
		Built a practical implementation of this technique into an emulator used for automatic malware analysis	Not general solution since it Considers a few number of anti-emulation attacks		
		Automatically correct the emulation failures, with robust DSMs, allowing an automated analysis to reveal the malware's malicious activities	It is basically based on the possible behavior that malware can exhibits to do anti-emulation		

NO	Methos (Paper title)	Strengths (Best features)	Weaknesses (Limitations)	Opportunities	Threats
8.	**MmmBop : Generic Unpacking of Self-modifying, Aggressive, Packed Binary Programs**	Able to deal with most of the known and unknown packing algorithms Suitable to successfully bypass most of currently used anti-reversing tricks Performance does not depend on any other 3rd party software. Developed entirely in user mode (ring3),and does not use any debugging API, virtual machine or emulation	Not able to handle packers based on Virtual Machines (VM) approach Suffers from anti-technique malware, since dynamic binary instrumentation solutions need to modify target process address space	Availability of binary instrumentation techniques	The fast evolution of the evading and anti-debugging techniques Multi-threading loader stubs and more aggressive packers

4.4. CROSS COMPARISON AND CATEGORIZATION

In this section, based on the above presented "drivers based analysis" and "SWOT analysis", we would like to make cross comparison and dominance ranking among the techniques. According to the main target of the techniques and specific requirements we have classified them in to two categories.

The first category of techniques is "GENERIC UNPACKING AND ANALYSIS TECHNIQUES". These techniques mainly target on generic unpacking of packed malwares and no specific attention is given to anti-technique defense during this unpacking. Moreover, these techniques are generic in a sense that they are not targeting to specific type of obfuscation mechanism.

The second category of techniques is "MALWARE ANALYZERS WITH ANTI TECHNIQUE DEFENSE". These techniques, in addition to unpacking of a packed malware, also handles defense to anti technique such as anti debugging, anti emulation and anti virtual machine. The table below presents these two categories of techniques with their corresponding dominance raking and some basic justifications.

Table 4.4 Categorization and ranking of the techniques

NO	CATEGORY	DOMINANCE RANK	TECHNIQUES	JUSTIFICATION OF THE DOMINANCE RANKING
1	GENERIC UNPACKING AND ANALYSIS TECHNIQUES	1ST	**AGUnpacker**	- Generic unpacker with anti-technique mechanism and able to reconstruct and generate runnable binary - The limitations and threats are tolerable relative to others. - No such difficult and unachievable requirement is specified - Relatively very recent work (2009) - Completely dominates the other methods listed below
		2ND	**MmmBop**	- Generic unpacker able to handle self modifying packed malwares - The limitations and threats are tolerable relative to others with the exception of handling anti-technique mechanism - No such difficult and unachievable requirements is specified - relatively very recent work (2009) - Completely dominates the other methods listed below
		3RD	**Omniunpack**	- Generic unpacker targeted at detecting multilayer unpack with no consideration of anti technique - The limitations and threats are NOT tolerable relative to others
		4TH	**Polyunpack**	- Generic unpacker with no consideration of multilayered and anti technique embedded packed malware - The limitations and threats are NOT tolerable relative to others

NO	CATEGORY	DOMINANCE RANK	TECHNIQUES	JUSTIFICATION OF THE DOMINANCE RANKING
2	MALWARE ANALYZERS WITH ANTI TECHNIQUE DEFENSE	1ST	**Ether**	- Generic malware analyzer with main target of detecting anti technique - Transparent to malware and able to handle most of the current anti technique mechanisms - The limitations and threats are tolerable relative to others - Relatively recent work (2008) - Completely dominates the other methods listed below
		2ND	**EERM**	- Malware analyzer with specific target of detecting anti-emulation malwares by emulating them - Too specific relative to other methods listed - The limitations and threats are NOT tolerable relative to others - Relatively very recent work (2009)
		3RD	**Rotalum´e**	- Malware analyzer with specific target of reverse engineering of emulated malwares - Too specific relative to other methods listed - The limitations and threats are NOT tolerable relative to others - Recent very work relatively(2009)
		4TH	**Renovo**	- Malware analyzer with specific target of detecting anti-reverse engineering malwares - The limitations and threats are NOT tolerable relative to others - Too specific relative to other methods listed - Relatively not recent work(2007)

CHAPTER FIVE

5. CONCLUSIONS AND FUTURE WORKS

In this thesis work we have reviewed research works on malware analysis and detection techniques with special focus on comparison and analysis of the “automatic malware unpacking techniques”. A literature review of existing similar works has also been done. Out of the very few research works in this area two of them are presented and discussed with respect to this thesis work.

Firstly, we have made a review of currently published research works on “malware analysis and detection” in which eight techniques out of the total set is selected based on their relation with our thesis topic. These eight techniques mainly discuss about packed malware unpacking and analysis. A brief overview of each of this technique is presented in the literature review section of this report.

An analysis based on some critical drivers, such as main target (considerations), specific technique used, experiments done and summary of contributions, on each of the selected techniques is done and discussed in the analysis section of this report. Based on this analysis detail the eight techniques are classified to different taxonomy of malware detection and analysis mechanism in order to be able to categorize the techniques according to the taxonomy.

We have also analyzed the strength, weakness, opportunity and threat of each of these selected techniques. Based on this SWOT analysis and the “driver based” analysis we have categorized and made a dominance raking among the techniques. The eight selected techniques are classified in to two categories. of four techniques each.

AGUnpacker dominates all first category techniques by its being a generic unpacker equipped with anti technique defense, able to reconstruct and generate runnable binary and some more justification discussed in the report. **Ether** dominates all second category techniques by its being transparent and generic malware analyzer with a main target of anti technique detection and some more justifications discussed in the report.

A very systematic combination of the above mentioned two techniques, with omission off overlapping and reduction of some requirements discuss above, will be a promising solution to the different packing and obfuscation challenges imposed by malware writers.

One of the critical limitations we faced during this study is the access to the implemented tool for the above discussed techniques. This limitation basically restricted the scope of our study from testing the tool and doing some experimental analysis to come up with better results.

A possible extension of this thesis work could be to do an experimental analysis by implementing the tools for, at least the best ranked and more dominant, techniques. And moreover to come up with a new technique to do an automatic unpacking of packed malware that will obviously address the limitation of the techniques discussed in this thesis work.

APPENDIX

A. LIST OF MALWARE PACKERS

NO.	PACKER NAME	BRIEF DESCRIPTION	REMARK
1.	**ACprotect**	Protect Windows executable files (PE files) against piracy. Using public keys encryption algorithms (RSA) to create and verify the registration keys and unlock some RSA key locked code.	OmniUnpack, AGUnpacker
2.	**Armadillo**	Commercial protection for any Win32 program. Also adds as you might expect some anti-debugging code and the simple fact that its mostly encrypted means you can forget disassembling	OmniUnpack, PolyUnpack, AGUnpacker, Renovo, Ether
3.	**ASpack**	ASPack is an advanced Win32 executable file compressor, capable of reducing the file size of 32-bit Windows programs by as much as 70%. http://www.aspack.com/	OmniUnapck, PolyUnpack, AGUnpacker, Renovo, Ether
4.	**CExe**	CExe is a Win32 based .exe compressor. It has a framework for multiple compressors and tries them all, choosing the compressor that result in the smallest size. http://www.scottlu.com/Content/CExe.html (download)	OmniUnapck
5.	**ExeStealth**	EXE Stealth can protect most of the executable files that are compatible with PE format. Manage your serials for your shareware. Protect your exe files against cracking with crypto technologies http://www.webtoolmaster.com/download/ExeStealth.exe (download)	OmniUnapck, PolyUnpack

6.	**EZip**	EZIP is a free packer with some features: Compressed EXEs are typically 30-50% their original size. Compressed EXEs run as normal. No special files or drivers need. Compressed programs are more difficult to reverse.	PolyUnpack
7.	**FSG**	FSG - F[ast] S[mall] G[ood] is a perfect compressor for executable files, its decompression code is only 158 bytes long, it's compatible with Windows 95 / 98 / ME / 2K / XP / Vista / 7.	OmniUnpack, PolyUnpack, AGUnpacker, Renovo, Ether
8.	**MEW**	Mew is an exe-packer application, based on ApPack and LZMA methods, written in Visual C and MASM 32. Originally it was designed for small files (4k, 64k intros), but it supports bigger files too.	OmniUnpack, PolyUnpack, Renovo, Ether
9.	**MoleBox**	MoleBox is a runtime exe packer for Windows applications. It bundles the executable together with the DLL and data files into a single EXE file, without losing the ability to run the application. MoleBox compresses and encrypts all the application files. http://www.molebox.com/molebox-vs-features.shtml	OmniUnpack, PolyUnpack, Renovo, Ether
10.	**nPack**	nPack is a Win32 executable file compressor. Features:(Support for all types of PE files (exe, dll, ocx),Compression of program code, data, and resources, Section naming support,Fast decompression routines, Save overlay support, Relocation support) http://petools.org.ru/npack.zip	OmniUnpack
11.	**nSPack**	Nspack is an executable compressor for Windows, which exceeds other similar products in features and compression ratio. Not only can it compress exe,dll, ocx, and scr files, but it can compress 64-bit executables, and executables created for the .net platform. http://www.download32.com/go/56085/http%3A%2F%2Fwww.nsdsn.com%2Fenglish%2Fnspack.zip/	OmniUnpack, AGUnpacker, Ether

12.	**Obsidium**	Allow you to protect your program from unauthorized modifications (i.e. "cracking") and provides you with a reliable yet easy to implement licensing system. It is compatible with any 32-bit Windows OS.	PolyUnpack, Renovo, Ether
13.	**PEComapct**	PECompact is a next generation Windows executable compressor designed for software developers and vendors. Commonly termed an 'executable packer', such utilities compress executables and modules (i.e. *.EXE, *.DLL, *.OCX, *.SCR). http://www.bitsum.com/pecompact.php	PolyUnpack, AGUnpacker, Renovo, Ether
14.	**PESpin**	PESpin is a Windows executable files (EXE, DLL) protector, compressor coded in pure assembly using MASM. It allows compression of the whole executable - code, data and resources, leaving them executable and protects against patching and disassembling. http://pespin.w.interia.pl/	MmmBop
15.	**Pex**	PeX is simple PE packer & protector. It's compatible with Win95/98/NT. http://github.com/qhoxie/rcrypt	PolyUnpack
16.	**PKLite**	PKLITE - An executable file compression utility for MS-DOS from PKWARE, Inc.. PKLITE compresses the body of the executable and adds a small, fast decompress routine in the header	OmniUnpack, Ether
17.	**RCryptor**	RCrypt is a simple RSA encryption library for your Ruby scripts. It aims at being a no-frills encrypt-decrypt solution	Ether
18.	**RLPack**	RLPack combines best of the breed compression and protection elements to give you the best possible protection against software cracking, IP theft and software tampering. http://www.reversinglabs.com/products/RLPack.php	Ether OmniUnpack

19.	**teLock**	tElock is a PE-File Encryptor/Compressor tool that was designed to process most .exe, .ocx and .dll files. It compresses and encrypts those file types while leaving them executable and protects against patching and disassembling. http://www.telock.com-about.com/	OmniUnpack, MmmBop, Ether
20.	**Themida**	Powerful Windows Software Protector. Designed for software developers who wish to protect their applications against advanced reverse engineering and software cracking.	OmniUnpack, Renovo, Ether
21.	**Upack**	Upack is a packer similar to UPX, but it uses LZMA compression and is designed with a focus on anti-unpacking.	PolyUnpack, Ether
22.	**UPX**	Ultimate Packer for Xecutables (UPX) is an extendable software high-performance executable file compression packer software for a number of diverse executable file formats. It achieves an excellent file compression ratio and fast running decompression. http://upx.org/	OmniUnpack, PolyUnpack, AGUnpacker, MmmBop, Renovo, Ether

REFERENCES

[1] - Babar, K.; Khalid, F.; , "*Generic unpacking techniques*," Computer, Control and communication, 2009. IC4 2009. 2nd International Conference on , vol., no., pp.1-6, 17-18 Feb. 2009

[2] - Vinod, P.; Jaipur, R.; Laxmi, V. & Gaur, M. "*Survey on Malware Detection Methods*",. (2009) Hack. 74

[3] - Martignoni, L.; Christodorescu, M.; Jha, S.; , "*OmniUnpack: Fast, Generic, and Safe Unpacking of Malware*," Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual , vol., no., pp.431-441, 10-14 Dec. 2007

[4] - Royal, P.; Halpin, M.; Dagon, D.; Edmonds, R.; Wenke Lee; , "*PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware*," Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual , vol., no., pp.289-300, Dec. 2006

[5] - Yu San-Chao; Li Yi-Chao; , "*A Unpacking and Reconstruction System-AGUnpacker*," Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on , vol., no., pp.1-4, 18-20 Jan. 2009

[6] - Min Gyung Kang, Pongsin Poosankam, and Heng Yin., *"Renovo: A Hidden Code Extractor for Packed Executables"* In Proceedings of the 5th ACM Workshop on Recurring Malcode (WORM), October 200

[7] - Artem Dinaburg , Paul Royal , Monirul Sharif , Wenke Lee, "*Ether: malware analysis via hardware virtualization extensions"*, Proceedings of the 15th ACM conference on Computer and communications security, October 27-31, 2008, Alexandria, Virginia, USA

[8] - Monirul Sharif , Andrea Lanzi , Jonathon Giffin , Wenke Lee, *"Automatic Reverse Engineering of Malware Emulators"*, Proceedings of the 2009 30th IEEE Symposium on Security and Privacy, p.94-109, May 17-20, 2009

[9] - Min Gyung Kang, Heng Yin, Steve Hanna, Steve McCamant, and Dawn Song. *"Emulating Emulation-Resistant Malware"* In Proceedings of the 2nd Workshop on Virtual Machine Security, November 2009.

[10] - Piotr Bania. "*Generic Unpacking of Self-modifying, Aggressive, Packed Binary Programs*" http://piotrbania.com/all/articles/pbania-dbi-unpacking2009.pdf, **2009**