International Journal of Technical Innovation in Modern Engineering & Science (IJTIMES)
Impact Factor: 3.45 (SJIF-2015), e-ISSN: 2455-2585
Volume 4, Issue 5, May-2018

New trends in Computer Virology and Malware Analysis

[1]Suhel Ahamed
[1]*Assistant Professor, Department of Information Technology, Guru Ghasidas Vishwavidyalaya, Central University, Bilaspur, Email id: suhel.yusuf@gmail.com*

Abstract— This paper illustrates the advancement in the field of computer virology. The purpose is to help computer professionals understand the new trend of virus development and the threats it poses. It should serve as a starting point for individuals and researchers who would like to get an idea about the next generation viruses, antivirus techniques and malware analysis. This paper would also discus the threats to antivirus techniques and malware analysis. Understanding of threats to computer security and practice of safe and good computing can be active defense against security threats. There are some important research problems, the solutions to any of which will significantly improve our ability to deal with the virus problems of the near future.

Keywords: polymorphism, CPU emulation, generic decryption, Static and Dynamic Analysis, X-ray, Retrovirus.

I. INTRODUCTION

Since the appearance of the first computer virus in 1980's, a significant number of new viruses have appeared every year. This number is growing and it threatens to outpace the manual effort by anti-virus researchers and malware analysts in designing solutions for detecting them and removing them from the system. Many anti-virus scanners primarily detect viruses by looking for simple virus signatures within the file being scanned. The signature of a virus is typically created by disassembling the virus into assembly code, analyzing it, and then selecting those sections of code that seem to be unique to the virus. The binary bits of those unique sections become the signature for the virus. However, this approach can be easily subverted by polymorphic viruses, which change their code (and virus signature) every time they're run.

The increasing speed of attack has placed a greater emphasis than ever before on the speed at which antivirus vendors respond to new threats. In the 'good old days' referred to above, quarterly updates were enough for most customers. Later, monthly updates became standard. Then, in the wake of Melissa and Love-Letter, most anti-virus vendors switched to weekly virus definition updates. Now, several of them offer daily updates. Although virus definitions are available faster than ever before, there is still a gap between the appearance of a new threat and the means of blocking it, a gap that allows a virus or worm to spread unchecked. And this brings us back to the questions posed earlier. Can anti-virus products cope with the threats of today and tomorrow? Is the signature scanner obsolete? Does the future belong to behavioral analysis or other generic technologies? In order to answer these questions, we need to take a look at the proposed alternatives.

II. COMPUTER VIROLOGY:

The word "computer virus"[1] comes from the seminal theoretical works of Cohen and Adleman in the mid-1980.The first question is what a virus is? In his PhD thesis[2], Cohen defines virus with respect to Turing Machines. Roughly speaking, a virus is a word on a Turing machine tape such that when it is activated, it duplicates or mutates on the tape. The name "virus" was coined by Len Adleman[3]. Adleman took a more abstract formulation of computer viruses based on recursive function in order to have a definition independent from computation models. A recent article of Zuo and Zhou completes Adleman's work, in particular in formalizing polymorphic viruses. In both approaches, a virus is a self replicating device or piece of code that attaches itself to other programs and usually requires human interaction to propagate. So, a virus has the capacity to act on a description of itself. [1] A worm shares several characteristics with a virus. Worms are standalone, and do not rely on other executable code, also worms spread from machine to machine across networks without any human intervention. Most of us are familiar with the basic types of computer virus like file infectors, Boot-Sector infector, macro virus etc. also we are aware of some basic antivirus techniques.

There are four basic means of virus detection:

1. Signature based scanning,
2. Emulation,
3. Heuristics search,
4. Behavioral analysis and integrity checker.

A. Advancement in Computer Virology:

Here's no question that today's threats are faster than ever before. Where it used to take weeks, or even months, for a virus to achieve widespread circulation, today's threats can achieve worldwide distribution in hours - riding on the back of our business-critical email infrastructure and exploiting the increasing number of system vulnerabilities that give them a springboard into the corporate enterprise.

1. Vulnerability Exploited! [4]

The use of system exploits has now become commonplace. In fact, some threats have avoided the use of 'traditional' virus techniques altogether. Lovesan, Welchia and, more recently, Sasser are examples of Internet worms pure and simple. There's no mass-mailing, there's no requirement for a user to run an infected program. Instead, these threats spread directly across the Internet, from machine to machine, using various system vulnerabilities. Lovesan exploited the MS03-026 vulnerability ('Buffer Overrun in RPC Interface Could Allow Code Execution'). Welchia exploited the same vulnerability, plus MS03-007 ('Unchecked Buffer in Windows Component Could Cause Server Compromise'). Sasser exploited the MS04-011 vulnerability (a buffer overflow in the Windows LSASS.EXE service).

Others have combined the use of system exploits with other infection methods. Nimda, for example, incorporated several attack mechanisms. As well as the mass-mailing aspect of the virus outlined above, Nimda also appended viral exploit code (in the form of infected Java code) to HTML files. If the infected machine were a server, a user became infected across the web when they accessed the infected pages. Nimda went even further in its efforts to spread across the corporate network by scanning the network for accessible resources and dropping copies of itself there, to be run by unsuspecting users. On infected machines, the virus also converted the local drive(s) to open shares, providing remote access to anyone with malicious intent. For good measure, Nimda also used the MS00-078 exploit ('Web Server Folder Traversal') in Microsoft IIS (Internet Information Server) to infect vulnerable servers by downloading a copy of itself from already infected machines on the network. Nimda's multi-faceted attack strategy, coupled with its use of system vulnerabilities, led many to refer to this as a 'compound' threat.

Not all successful threats make use of system vulnerabilities. The mass-mailing technique pioneered by Melissa in 1999 continues to prove successful, particularly where 'social engineering' is used to conceal the malicious intent of the e-mail borne virus or worm. Social engineering refers to a non-technical breach of security that relies heavily on human interaction, tricking users into breaking normal security measures. In the context of viruses and worms specifically, it means any trick used to beguile naïve users into running the malicious code, typically by clicking on an attached file. The results can be very effective for the virus author. The Swen worm (appeared September 2003) masqueraded as a special patch from Microsoft that would close all security vulnerabilities. Swen was very convincing, not just because it used realistic dialog boxes, but because it came hard on the heels of threats like Lovesan and Welchia and appeared at a time when the patching vulnerable systems was being given a high profile.

2. The large profit chain! [5]

w32.fujacs (see Symantec security response) W32.Fujacks (Chinese name: Panda Shao Xiang) is a worm and file-infector virus, which is written in Borland Delphi. TheW32.Fujacks outbreak started in November 2006 and its author has consistently released a considerable number of new variants. W32.Fujacks and its variants have spread widely throughout China since then. Finally, on 3rd Feb 2007, the author Mr. Jun Li was arrested by Hubei Police Department. Unlike most other common viruses, W32.Fujacks and its variants were updated with an astonishing frequency. These variants are not only repacked with a new harder to detect packer – obviously, Mr. Li and his friends are driven by profit they also attempted to add more and more functions to make it spread widely and steal more valuable assets. When W32.Fujacks infects a computer, it downloads several Trojan horses from the Internet. It then attempts to steal account information for several online games. It also attempts to steal account information from the Instant Messenger tool "QQ", which is one of the most popular IM tools in China. Furthermore, it also gains full control of the compromised computer and uses it to build a botnet. The hacker will use these compromised computers for further attacks.

W32.Fujacks enumerates windows and all running processes. If any specified security application is found, W32.Fujacks will end that application. W32.Fujacks searches for and deletes all files with an extension of .gho, which is a hard disk image file, created by Symantec Ghost. By doing this, the backup data will be lost and it will be harder to restore the system.

Unlike most traditional viruses, the outbreak of W32.Fujacks is beneficiated and driven by its large profit chain. Mr. Li sold roughly 20 copies of W32.Fujacks for 500 to 1000 RMB (Yuan Renminbi) per copy (approximately 64 to 129 US dollars). On 24th Jan 2007, his classmate, Mr. Lei, sold 2000 compromised computers for him. Mr. Shun Zhang is in charge of reselling and spreading W32.Fujacks. From the time Mr. Lei contacted Mr. Li until he was arrested, he transferred between 3500 RMB and 6500 RMB ($452-$839) to Mr. Li's account daily. Mr. Li has received more than 150,000 RMB ($19,374) from Mr. Lei. Compared to Mr. Li, Mr. Shun Zhang has made even more money. The question is "who is paying for W32.Fujacks?"Some partners who buy this worm usually register a domain and server with a fake name or use a vulnerable Web site for spreading The Trojan horse and collecting stolen information. They use it to steel credit card and other valuable information and these computers can be controlled by the partner. The partner may then use the computer for ad clicking. These clicks may generate a huge amount of money for the partner. Partner can resell this information to gain more money. This way it goes

There are lessons to be learned from W32.Fujacks and we all need to learn them fast.

3. *Polymorphism***:**

One of the few solid theoretical results in the study of computer viruses is Cohen's 1987 demonstration that there is no algorithm that can perfectly detect all possible viruses. [6] Polymorphic computer viruses are the most complex and difficult viruses to detect, often requiring anti-virus companies to spend days or months creating the detection routines needed to catch a single polymorphic. Polymorphic virus is a computer virus that can mutate itself every time it runs. Polymorphic virus uses a Mutation Engine with infinite number of decryptor loop to encrypt and decrypt the virus body and mutation engine as well. Although polymorphism is independent of encryption, it is easier to use encryption to hide the main body of the virus and implement a polymorphic decryptor.

The first known polymorphic virus was written by Mark Washburn [7]. The virus, called 1260, was written in 1990. A better-known polymorphic virus was invented in 1992 by the Bulgarian cracker Dark Avenger [7] (a pseudonym) as a means of avoiding pattern recognition from antivirus software. In 1992, Dark Avenger, author of Maltese Amoeba, distributed the Mutation Engine, also known as MtE, to other virus authors with instructions on how to use it to build still more polymorphic.

Two polymorphic [8] — One Half and Natas — rank among the 20 most-prevalent computer viruses, according to the 1996 Computer Virus Prevalence Survey conducted by the National Computer Security Association (NCSA).

One Half slowly encrypts a hard disk. Natas, also known as SatanBug.Natas, is highly polymorphic, designed to evade and attack anti-virus software. It infects .COM and .EXE program files.

A few modern polymorphic viruses use encryption in a more powerful way, by encrypting parts of themselves and *not* including the decryption key within the virus. Without the decryption key, a human analyst cannot determine what that part of the virus would do if it were to be decrypted and executed. Viruses store their decryption key on an anonymous text file whose checksum is used to trace it by virus to decrypt itself.

III. ADVANCEMENT IN ANTIVIRUS TECHNIQUES:

There are several researches going on for new and advance antivirus techniques to emerge I am illustrating here few of them.

1. *The digital immune system* [9]

Enterprise-Grade Anti-Virus Automation in the 21st century (see Symantec's technical brief) IBM® and Symantec designed the Digital Immune System as a closed-loop automated system to deal with Melissa-class threats, orders-of-magnitude increases in submissions, floods, and denial-of-service attacks.

The Digital Immune System:

1. Detects a high percentage of new or unknown threats at the desktop, server, and gateway.
2. Makes the full system highly scalable.
3. Provides secure submission of virus samples and secures distribution of new definitions.
4. Provides intelligent filtering of submissions to focus system resources on the most critical threats.
5. provides high-speed analysis capabilities.
6. Reduces instances of false positives.
7. Provides full end-to-end automation of submission, analysis, and distribution of new definitions.
8. Provides real-time status updates on all submissions.
9. Manages common flooding conditions and denial of service attacks.
10. Provides the administrator with the ability to set the level of automation.

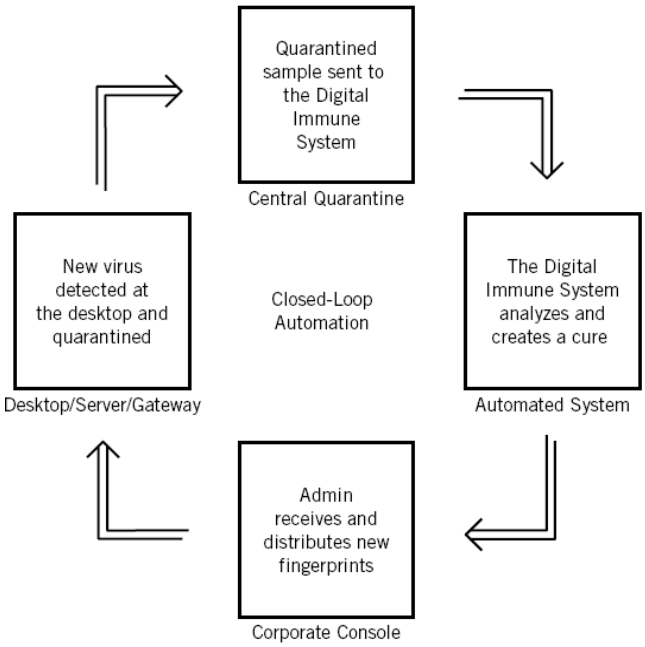

Fig.1: A high-level state diagram of the Digital Immune System closed-loop system.

2. *Polymorphic detection* techniques:

It is still possible to detect a polymorphic virus using signatures, but the virus body must be decrypted first. There are several methods that are widely used in the AV Industry for the purpose of decrypting polymorphic viruses: cryptanalysis (also known as x-ray), dedicated decryption routines, CPU emulation, etc. Today, virtually every anti-virus product uses a CPU emulator [8] to detect polymorphic computer viruses. Basically there are two main techniques for this 1) generic decryption, 2) heuristic based generic decryption.

A scanner that uses generic decryption loads the file into a self-contained virtual computer created from RAM. Inside this virtual computer, program files execute as if running on a real computer. When a scanner loads a file infected by a polymorphic virus into this virtual computer, the virus decryption routine executes and decrypts the encrypted virus body. This exposes the virus body to the scanner, which can then search for signatures in the virus body that precisely identify the virus strain. The key problem with generic decryption is speed. Generic decryption is of no practical use if it spends five hours waiting for a polymorphic virus to decrypt inside the virtual computer.

3. *Heuristic-Based Generic Decryption:*

It employs "heuristics," a generic set of rules that helps differentiate non-virus from virus behavior. A Heuristic-based generic decryption looks for such inconsistent behavior that increases the likelihood of infection and prompts a scanner that relies on heuristic-based rules to extend the length of time a suspect file executes inside the virtual computer, giving a potentially infected file enough time to decrypt itself and expose a lurking virus. Unfortunately, heuristics demand continual research and updating. Also we may be forced to rewrite the complex heuristic set of rules for new viruses, as virus writers continue trying to make viruses look like clean programs, heuristics can easily reach to the point where almost any program might share attributes that trigger the scanner to lengthen the time it takes to examine a file, simply decreasing the system performance.

4. *X-ray* [10] :

This works by attempting to find the decryption key by using the known decryption algorithm and a fragment of decrypted code, which is part of the signature. For each key, an equation is written, that expresses this key as a function of the encrypted code, decrypted code and the other keys. Solving the system will produce the correct set of keys required to decrypt the virus. The method is fairly simple to implement and has good performance for a single given decryption algorithm. In some cases, it is also able to detect fragments of the virus code, even if the entire virus is not present or is not functional. However, the method does not scale well with the number of detected viruses, because it needs to be run for each different encryption algorithm. Also, it can only handle simple algorithms, as the equation system becomes impossible to solve for more complex ones. Its usefulness is very limited in the case of viruses having multiple encryption layers.

Dedicated decryption routines can be developed to detect any virus, and performance for any given virus is usually better compared to the x-ray method. Unfortunately, writing such a routine requires that the virus is analyzed completely and the developer understands completely all the possible variants of encryption that the virus can generate. A thorough analysis of the malware and then developing and testing a specific detection routine could take a lot of work and a lot of time to accomplish. Therefore, the response time when using this solution can often be quite long. Additionally, this method does not scale well with the number of detected viruses, as each file must be checked with all the available routines.

5. *Symantec's Striker system [8]*

Like generic decryption, Striker loads the file into a self-contained virtual computer created from RAM. The program executes in this virtual computer as if it were running on a real computer. However, Striker does not rely on heuristic guesses to guide decryption. Instead, it relies on virus profiles or rules that are specific to each virus, not a generic set of rules that differentiate non-virus from virus behavior. When scanning a new file, Striker first attempts to exclude as many viruses as possible from consideration For example, different viruses infect different executable file formats. Some infect only .COM files. Others infect only .EXE files. Some viruses infect both. Very few infect .SYS files. As a result, as it scans an .EXE file, Striker ignores polymorphic that infect only .COM and .SYS files. If all viruses are eliminated from consideration, then the file is deemed clean. Striker closes it and advances to scan the next file. If this preliminary scan does not rule out infection, Striker continues to run the file inside the virtual computer as long as the behavior of the suspect file is consistent with at least one known polymorphic or mutation engine.

6. *Integrity verification systems such as tripwire* [11]:

It is an integrity verification system is a program that scans all of the executables in a system, and for each executable, generates a checksum. From time to time, the integrity verification system recalculates the checksum and generates an exception if the check sum changed. The problem here is that viruses can hide in. dlls, in libraries, or in scripts. Also, building a standardized mechanism for storing checksums invites viruses, which attack that storage mechanism. Finally, tripwires require a rigorous separation of read-only code and configuration information, which is the antithesis of design in the MS-Windows and Windows/NT systems.

7. *Behavior blocking* [4]:

One of the technologies often seen as a successor to signature-based scanning is behavior blocking. It's not a new idea. Some security vendors adopted this approach in the early 1990's; Traditional antivirus scanners hold signatures of malicious code in a database and cross-check against this database during the scan process. Behavior blockers, by contrast, determine whether an application is malicious or not according to what it does. If an application does something that falls outside the range of acceptable actions, its operation is restricted. For example, trying to write to certain parts of the system registry, or writing to pre-defined folders, may be defined as a threat.

The action can be blocked, or the user notified about the attempted action. An alternative behavioral method is to 'wrap' a downloaded application and restrict its action on the local system - the application is run in a protective 'sandbox' (or 'playground', or 'secure cache') to limit its actions according to a pre-defined policy. The activity performed by the program is checked against a set of rules. The chief benefit of a behavior blocker, according to its proponents, is that it's able to distinguish between 'good' or 'bad' programs without the need for a professional virus researcher to analyze the code.

However, there have always been potential drawbacks with behavioral analysis. One major problem lies in the fact that there's a grey area between actions that are clearly 'bad' and those that are legitimate. What's bad in a hostile program may be good in a legitimate program. For example, the low-level disk writes carried out by a virus, worm or Trojan, perhaps to erase data from your hard disk, are also used legitimately by the operating system. And how is a behavior blocker deployed on a file-server to know whether a modification to (or deletion of) a document is being carried out legitimately by a user or is the result of a hostile program on the infected user's machine? After all, a virus or worm is simply a program that copies itself. Beyond this, it may do what any other normal program does. As a result, it can be very difficult to determine what rules you should use to define something as 'bad'. And there's clearly a risk of false alarms, flagging an application or process as 'bad' when it's perfectly legitimate.

8. Intrusion prevention systems (IPS) [4]:

It may be considered as the successor of the behavioral analysis techniques discussed above, although some IPS systems also offer signature-based detection. Host-based IPS, designed to protect desktops and servers, typically employ behavioral analysis to detect malicious code. They do this by monitoring all calls made to the system and matching them against policies based on 'normal' behavior. Network-based IPS, deployed inline on the network, filters packets for malicious code, looking for abnormal bandwidth usage or for non-standard traffic (like malformed packets). Network-based IPS is particularly useful for picking up DoS attacks, or the traffic generated by network-based worms.

One of the key problems with IPS, as with older behavioral analysis programs, is the risk of false alarms. The down-side to this is that they need to be carefully tuned, so there's a much bigger administrative overhead than with traditional anti-virus protection.

IV. MALWARE ANALYSIS:

Malware analysis is the process of determining the purpose and functionality of a given malware sample (such as a virus, worm, or Trojan horse). This process is a necessary step to be able to develop effective detection techniques for malicious code. In addition, it is an important prerequisite for the development of removal tools that can thoroughly delete malware from an infected machine. In addition, the knowledge about the functionality of malware is important for removal. That is, to be able to cleanly remove a piece of malware from an infected machine, it is usually not enough to delete the binary itself. It is also necessary to remove the residues left behind by the malicious code (such as unwanted registry entries, services, or processes) and undo changes made to legitimate files. All these actions require a detailed understanding of the malicious code and its behavior. The traditional approach to analyze the behavior of an unknown program is to execute the binary in a restricted environment and observe its actions. The restricted environment is often a debugger, used by a human analyst to step through the code in order to understand its functionality. Unfortunately, anti-virus companies receive up to several hundred new malware samples each day. Clearly, the analysis of these malware samples cannot be performed completely manually. One way to automate the analysis process is to execute the binary in a virtual machine or a simulated operating system environment. While the program is running, its interaction with the operating system (e.g., the native system calls or Windows API calls it invokes) can be recorded and later presented to an analyst. This approach relieves a human analyst from the tedious task of having to manually go through each single malware sample that is received.

The analysis method can be divided into two broad categories: *static analysis* and *dynamic analysis* techniques.

1. Static Analysis [12]:

It is the process of analyzing a program's code without actually executing it. In this process, a binary is usually disassembled first, which denotes the process of transforming the binary code into corresponding assembler instructions. Then, both control flow and data flow analysis techniques can be employed to draw conclusions about the functionality of the program. Static analysis has the advantage that it can cover the complete program code and is usually faster than its dynamic counterpart.

Its main weakness is that the code analyzed may not necessarily be the code that is actually run. In particular, this is true for self-modifying programs that use polymorphic (Szor, 2005; Yetiser, 1993) and metamorphic (Szor, 2005) techniques and packed executables that unpack themselves during run-time (Oberhumer& Molnar, 2004). Also, malicious code can make use of obfuscation techniques (Linn & Debray, 2003) to thwart the disassembly step. The reason is that for certain instruction set architectures (most notably, Intel x86), it is difficult to distinguish between code and data bytes in a file. Static analysis can be applied on Real computer as well as Virtual computer using emulator.

2. Dynamic Analysis [13]:

In contrast to static techniques, a dynamic technique analyzes the code during run-time. While these techniques are non-exhaustive, they have the significant advantage that only those instructions are analyzed that the code actually executes. Thus, dynamic analysis is immune to obfuscation attempts and has no problems with self-modifying programs. When using dynamic analysis techniques, the question arises in which environment the sample should be executed.

Dynamic Analysis can be done by two methods:

2.1 Real Analysis:

first is The Real Analysis, running malware directly on the analyst's computer, which is probably connected to the Internet, could be disastrous as the malicious code could easily escape and infect other machines. Furthermore, the use of a dedicated standalone machine that is reinstalled after each dynamic test run is not an efficient solution because of the overhead that is involved.

2.2 Virtual Analysis

The second method is Virtual Analysis, Running the executable in a virtual machine, such as one provided by VMware [14], is a popular choice. In this case, the malware can only affect the virtual PC and not the real one. After performing a dynamic analysis run, the infected hard disk image is simply discarded and replaced by a clean one (i.e., so called *snapshots or clones*). Virtualization solutions are sufficiently fast. There is almost no difference to running the executable on the real computer, and restoring a clean image is much faster than installing the operating system on a real machine.

Unfortunately, a significant drawback is that some executable, which is to be analyzed, may determine that it is running in a virtualized machine and, as a result, modify its behavior. In fact, a number of different mechanisms have been published that explain how a program can detect if it is run inside a virtual machine. Of course, these mechanisms are also available for use by malware authors.

V. THREATS TO ANTIVIRUS TECHNIQUES AND MALWARE ANALYSIS

A. Retroviruses [15]

A virus that actively tries to disable anti-virus software running on an infected machine is referred to as a *retrovirus*, this is a generic term for a virus employing this type of active defense, and doesn't imply that any particular technique is used. A more aggressive retrovirus can target the antivirus software on disk as well as in memory; so that antivirus protection is disabled even after the infected system is rebooted. For example, Ganda kills processes that appear to be anti-virus software, using the above list-based method; it also examines the programs run at system startup, looking for anti-virus software using the same list of names. If Ganda finds anti-virus software during this examination, it locates the executable image on disk and replaces the first instruction with a "return" instruction. This causes the anti-virus software to exit immediately after starting.

B. Attacks on Virtual Machine Emulators [16]

The simplest attack that malicious code can perform on a virtual machine emulator is to detect it. As more security researchers rely on virtual machine emulators, malicious code samples have appeared that are intentionally sensitive to the presence of virtual machine emulators. Those samples alter their behavior (including refusing to run) if a virtual machine emulator is detected. A harsher attack that malicious code can perform against a virtual machine emulator is the denial-of-service; specifically, this type of attack causes the virtual machine emulator to exit. Finally, the most interesting attack that malicious code can perform against a virtual machine emulator is to escape from its protected environment.

C. Other self-defense techniques in malwares **[**17**]:**

There are many different kinds of malware self-defense techniques and these can be classified in a variety of ways. We are naming here few of techniques.

1) Packers. 2) Root-kit. 3) Code-obfuscation 4) anti-debugging. 5) Anti- Disassembly 6) Blocking file access. 7) Modifying host files. 8) Using streams. 9). Auto-click and several other methods coming day by day.

VI. OPEN PROBLEMS AND SUGGESTIONS

There are several open problems in the field of computer virology .we are discussing here some of them, solving any of them may improve our ability to deal with the threats of coming future.

1. The traditional signature detection system is not sufficient to catch the viruses of new and complex type: so some stronger signature based detection system must be developed.
2. Virtually all of our anti-virus technology relies on detecting and removing viruses from a File system but there is always some hidden damage that viruses cause and there residues always left behind or they can make your system even more vulnerable: so some techniques to remove all the residues of malwares and to repair the system must be deployed.
3. Operating systems and computer network architectures have several open vulnerabilities and malwares can exploit them very easily: so study on removing Vulnerabilities must be conducted, and New even more secure operating systems must be deployed
4. Current anti-virus technology is largely reactive, relying on finding a particular virus before being able to deal with it well. Modern programming environments can give rise to viruses that spread increasingly rapidly, and for which a reactive approach becomes ever more difficult: So proactive techniques must be deployed in a large scale, but there is another open problem with it that proactive techniques are a headache for normal desktop user: so some proactive techniques to handle threat without much assistance must be developed so a user with less expertise may use it
5. Behavioral blockers and integrity checkers must be improved to deal with the complex viruses and the problems that we have discussed must be removed from these techniques.
6. New techniques to detect and remove malwares at network level must be developed
7. As cryptography is used by polymorphic viruses to subvert detection, virus cryptanalysis can be used to understand and detect this kind of viruses. We can also use cryptography to encrypt programs and executables to save it from viruses.

VII. SIGNIFICANCE AND CONCLUSION

Computer viruses are a controversial and taboo topic, despite having such a huge impact on our society. Some people believe that there is no longer any interesting research to do in the field of protection from computer viruses - that all of the important technology has already been developed - that it is now a simple matter of programming to keep up with the problem. Others believe that "virus research" simply means "analyzing viruses." This is just a misimpression.

As we have discussed the new trend or advancement in the field of computer virology, we have seen that there is no doubt many advance techniques for anti viruses has been evolved but the malware writers are a step ahead they have also created threats to antivirus and malware analysis techniques like retrovirus, detection of CPU emulation, anti-disassembly and anti-debugging etc.

Also we have discussed some important research problems in the area of computer virology, solving of these problems can significantly improve our understanding of malware and thereby helps improve security against malwares.

REFERENCES

[1]. Toward an abstract computer virology, G. Bonfante, M. Kaczmarek, and J-Y Marion, Loria, Calligramme project, B.P. 239, France.

[2]. F. Cohen. Computer Viruses. PhD thesis, University of Southern California, January 1986.

[3]. L. Adleman. An abstract theory of computer viruses. In Advances in Cryptology— CRYPTO'88, volume 403. Lecture Notes in Computer Science, 1988.

[4]. http://www.viruslist.com/en/analysis- Traditional antivirus solutions - are they effective against today's threats? - David Emm.

[5]. The Panda Outlaw: W32.Fujacks- Robert X Wang, Symantec Security Response, Dublin.

[6]. An Undetectable Computer Virus- David M. Chess and Steve R. White, IBM Thomas J. Watson Research Center, Hawthorne, New York, USA.

[7]. http://en.wikipedia.org/wiki/Polymorphic_code.

[8]. Understanding and Managing Polymorphic Viruses - The Symantec Enterprise Papers- Carey Nachenberg.

[9]. The Digital Immune System- TECHNICAL BRIEF, Symantec corp. USA.

[10]. DEFEATING POLYMORPHISM: BEYOND EMULATION - Adrian E Stepan, Microsoft Corp., One Microsoft Way, Redmond, WA 90852, USA.

[11]. Understanding Polymorphic Viruses -By Jeff Silverman.

[12]. TTAnalyze: A Tool for Analyzing Malware- Ulrich Bayer & Christopher Kruegel & Engin Kirda, Ikarus Software & Technical University of Vienna.

[13]. Dynamic analysis of malicious code - Ulrich Bayer · Andreas Moser · Christopher Kruegel · Engin Kirda.

[14]. http://www.vmware.com/

[15]. Computer Viruses and Malware by John Aycock, University of Calgary, AB, Canada: ISBN-10: 0-387-30236-0 (c) by Springer publisher.

[16]. Attacks on Virtual Machine Emulators - Peter Ferrie, Senior Principal Researcher, Symantec Advanced Threat Research.

[17]. http://www.viruslist.com/en/analysis- Traditional antivirus solutions - The evolution of self-defense technologies in malware, Alisa Shevchenko.

*****