ISSN: 2321-7782 (Online)
Impact Factor: 6.047

Volume 4, Issue 10, October 2016

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Introduction to Malware and Malware Analysis: A brief overview

Anusmita Ray[1]
Department of Computer Science
St. Xavier's College(Autonomous)
Kolkata – India

Dr. Asoke Nath[2]
Department of Computer Science
St. Xavier's College(Autonomous)
Kolkata – India

Abstract: ***One of the most widespread threats to cyber security in today's world of unlimited Internet access is malware. In recent times, the malware being designed are polymorphic and metamorphic, with the ability to transform their code and to hide quietly in the systems of the unsuspecting users. Malware analysis is the process of performing analysis of the malware and understanding its actions and behavior. It is of two types- static and dynamic analysis. Static analysis is performed by observing the source code of the malware and drawing conclusions based on it. Dynamic analysis is the analysis performed by executing the piece of code and noting its actions. Malware analysis is an important and relevant task, for the advanced forms of malware these days are often not even detectable by commonly available anti-virus software. In the present paper, the authors have made a systematic study on different issues in malware and analysis of malware.***

Keywords: Malware, Viruses, Static Analysis, Dynamic Analysis, Classification, Security.

I. INTRODUCTION

Malware, abbreviated from malicious software, is any kind of software that "deliberately fulfills the harmful intent of an attacker." Aycock (2006) defined malware as "software whose intent is malicious, or whose effect is malicious". [26] It can have any number of vindictive purposes such as disrupting the normal computer operations, gathering sensitive and confidential information from an unwitting user, gaining access to private computer networks and showing unwanted advertisements or spam. The term 'malware' was coined by Yisrael Radai in 1990, however, preceding that these types of software were prevalently known as computer viruses [22]. Malware is a general term, used to refer to a variety of intrusive and harmful software, including computer viruses, Trojan horse, worms, adware, and spyware. [25] Malware may be stealthy pieces of software, quietly stealing private information or they may cause direct harm like sabotage and even extort payment from the unsuspecting victim.

With the rise of broadband Internet access, simple computer viruses, which were initially written as mere pranks, have given way to malicious software which, in turn, has taken the form of a widespread epidemic. In recent times, malware has been written and designed keeping profits in mind. Since 2003, the majority of worms and viruses, which have become widespread, have been written in order to perform illicit activities on users' computers. McAfee [2] catalogs over 100,000 new malware samples every day means about 69 new threats every minute or about one threat per second. Preliminary results from Symantec [3] published in 2008 suggested that "the release rate of malicious code and other unwanted programs may be exceeding that of legitimate software applications." F-Secure [4] noted how the quantity of malware produced in 2007 was as much as in the previous 20 years. This, however, has seen a rise in highly specific and advanced tools to counter such malware. In a cause-effect situation, this, in turn, has led to the new generation cyber security threats and attacks to be more targeted, unknown, stealthy, personalized and zero day. This is in stark contrast to the traditional malwares which were broad, known, open and one time. Once these advanced malwares find their way into the users' computer systems, they hide, replicate and disable host

protections. After installation, they call their command and control servers for further instructions, which could be to steal data, infect other machines, and allow reconnaissance. [1]

Current systems to detect malicious code (most prominently, virus scanners) are largely based on syntactic signatures. That is, these systems are equipped with a database of regular expressions that specify byte or instruction sequences that are considered malicious. A program is declared malware when one of the signatures is identified in the program's code. [5] Malware authors use a variety of techniques to expose the vulnerabilities in different web services, operating systems, browsers, or in versions of browser plug-ins and exploit these weaknesses. Some commonly used techniques implemented include dead code insertion, register reassignment, subroutine reordering, instruction substitution, code transposition and code integration to evade detection by traditional defenses like firewalls, antivirus and gateways which typically use signature based techniques and are unable to detect the previously unseen malicious executables. Syntactic signatures ignore the semantics of instructions and hence the syntactic properties of code are largely ignored, resulting in such malware being resilient against common defence mechanisms. Commercial antivirus vendors are not able to offer immediate protection for zero day malwares as they need to analyze these to create their signatures. [1]

Thus with the usual signature based defence mechanisms failing in the face of advanced modern malware, the need for malware analysis is abundantly clear. Malware analysis can be of two types – static and dynamic. Static malware analysis refers to analyzing malicious software without executing, by merely observing and inspecting the techniques of the malware. Dynamic malware analysis involves analyzing a given program while it is being executed.

II. Different Types of Malware

The following paragraphs are solely here for the purpose of a brief introduction to the different terminology associated with malware. The various types of malicious software are very shortly discussed. It is to be noted that the classes that are going to be mentioned here are not mutually exclusive and a piece of malware may be displaying characteristics and traits of multiple such classes. In depth discussions of malicious code can be found in Szor (2005).

Viruses: "A computer program usually hidden within another seemingly innocuous program that produces copies of itself and inserts them into other programs or files, and that usually performs a malicious action (such as destroying data)".[6]

Worms: Spafford (1989) defines a worm as "a program that can run independently and can propagate a fully working version of itself to other machines." This type of harmful code is predominantly prevalent in networks such as the Internet.

Trojan horses: The term is derived from the Ancient Greek story of the wooden horse that helped Greek troops invade the city of Troy by stealth and deception. Like the name, Trojan horses are basically programs that misrepresent themselves as helpful software, such as plug-ins or downloadable games, while secretly perform malicious activities in the background.

Spyware: Spyware refers to software that access confidential information from the user and pass it on to another entity without informing the user of this action. Thus it takes control over the user's system without asking for his/her consent regarding the matter.

Backdoor: The method of evading usual authentication procedures, particularly over connections such as the Internet, is known as backdoor. One or more backdoors may be installed into a system without the user's knowledge to make the system susceptible to outside attacks.

Rootkits: It is important for a piece of malicious software to stay concealed once it has been installed in a system. This is to avoid detection, which defeats the purpose of malware. Rootkits are software packages that aid in the hiding of malware in the system by modifying the user's operating system ensuring that the software remains concealed.

In recent times, a sizeable portion of malware uses a number of techniques to avoid detection. The most prevalent evasion technique implemented in these days is by fingerprinting the environment as soon as the malware is executed. Another common

technique is by implementing stolen certificates to disable anti-virus protection. This is done by certain adware; technical remedies are available to deal with such adware.

III. STATIC MALWARE ANALYSIS

Analyzing malicious software without executing it, but by merely inspecting the program is called static or code analysis. It is usually performed by taking each part of the binary file and studying each component thoroughly without actually executing it. The different detection patterns used in case of static analysis [1] include string signature, byte-sequence n-grams, syntactic library call, control flow graph and opcode (operational code) frequency distribution etc.

Basic static analysis involves the following steps [7]. First, the suspicious executable is run through different anti-virus software and solutions to detect common malware. The applications detected by the anti-virus are noted down as is the information of what the anti-virus detects the malware as. Once the scanning of malware using anti-virus is completed, the second step of basic static analysis is executed. The malware is opened up in a hex editor to see what type of software it is and whether it is using some kind of packer application (such as Ultimate Packer for Executables or UPX). A packer such as UPX uses a simple data compression algorithm which allows for decompression of the file in a few hundred bytes of code. On unpacking the malware, other tools can be run against it. A copy of the malware must be made, for if the unpacking is done incorrectly, the malware may be unable to function. Microsoft has a useful utility called Strings, which checks for ASCII, Unicode or both characters in a file. Strings searches the executable while ignoring context and formatting and thus helps in finding out protocols, ports, IP addresses and other such information that provide important clues about the functionality of the malware. Strings searches for a three-letter or greater sequence of ASCII and Unicode characters, followed by a string termination character. Once the Strings search is done, it is time to disassemble the malware. The executables are reverse compiled using debuggers or disassemblers. The disassembled, decrypted malware code provides great insight into the working of the software.

Often malware writers use various obfuscation techniques to prevent the above reverse engineering and thus make static analysis unpredictable, unreliable and expensive. Binary obfuscation turns malware binaries into self-compressed and uniquely structured binary files. Furthermore, essential information about the malware is lost while trying to work with such binary executables and hence the process of malware analysis is made more difficult. [1] Moser et al. discussed the drawbacks and shortcomings of static analysis. In their paper, they introduced a scheme based on code obfuscation that demonstrated how static analysis alone is not enough to detect malware. They also proposed that dynamic malware analysis along with static analysis is necessary to make the process less susceptible to code obfuscation techniques and method.

IV. DYNAMIC MALWARE ANALYSIS

Once static malware analysis is completed, it is time to move on to dynamic analysis. As mentioned before, for proper analysis of malware, dynamic analysis must complement the process of static analysis. In dynamic analysis, the process is executed and the system is observed while the changes that occur are noted. It is to be kept in mind that the malware needs to be executed in a safe environment, preferably in a virtual machine. It is also wise to take a snapshot of the virtual machine before the malware binaries are executed so as to ensure that the safe state can be returned to and the proper changes can be noted. Before performing dynamic analysis, it is to be ensured that the virtual machine networking is not connected to any other networks apart from the host, for there is a great chance of introducing the malware to other networks if this step is not carried out. Other examples of controlled environments where the malware can be executed are simulators, emulators and sandboxes. Also, before executing the malware, certain monitoring tools such as Process Monitor [8] and Capture BAT [9] (for file system and registry monitoring), Process Explorer [10] and Process Hackerreplace [11] (for process monitoring), Wireshark [12] (for network monitoring) and Regshot [13] (for system change detection) are installed and activated. [1] The various techniques that are employed to carry out dynamic analysis are discussed ahead.

Function Call Monitoring: Functions used in programs consist of codes that perform a specific task, such as sorting a set of data. Functions are useful in case of programming, for they help in code reusability and also make the code easier to maintain. One characteristic of functions is that they provide an aspect of abstraction in carrying out the task. For example, a sorting function would ensure that the primary concern is the correct sorted output, without indulging in the details of the algorithm that is employed to reach the aforementioned result. It does not truly matter in the larger picture where the algorithm used is a merge sort or a bubble sort as long as the result reached by the function is correct. These abstractions help in creating a summary of the behavior and pattern of the program while analyzing code. One way to aid such analysis is to intercept the calls to such functions. The term 'hooking' is used to describe this process of intercepting function calls. The code is analyzed and alongside the required functions, a 'hook' function is invoked which helps to implement the respective analysis functionality, such as analyzing input parameters or keeping records of invocations to log files. [14] System calls are used by user-mode software to request the operating system to perform certain functions on its behalf. Usually, malware invokes system calls to interact with kernel space while executing in the user space. This makes the dynamic analysis of this interface quite interesting.

Function Parameter Analysis: In static analysis, function parameter analysis tries to deduce the types of the parameters or the set of their values in a static manner. However in dynamic analysis, the actual values of the parameters, that are passed when a function is called, are of concern. For instance, the return value of a CreateFile system call may be used later for a WriteFile call, and this correlation is of notable importance in dynamic malware analysis. [14]

Information Flow Tracking: The purpose of information flow tracking/analysis is to elaborate on the flow of 'interesting' data throughout the system, while the program manipulating it is being executed. Dynamic taint analysis helps in tracking the flow of data between the 'source' and the 'sink'. Any value in the program that depends on computation using data from a 'tainted' source is known as 'tainted' data. Yin et al [15] proposed a method called whole-system fine-grained taint analysis which uses a whole-system emulator to capture the intrinsic properties of a variety of malware and thus offering a great amount of improvement to automatic malware detection and analysis. As noted in their paper, their proposed system showed how the method helped in detecting a large number of malware of different classes including backdoors.

Instruction tracing: Instruction tracers, or tracers, are noted for recording every single instruction and related state while executing a piece of code. This tracing is then analyzed by a trace analyzer for extraction of relevant information. Bangerter et al [16] introduced a new tracer called Helios which involved a lot of optimizations such as automatically skipping irrelevant code parts that are also computationally expensive. The basic technique behind Helios is straightforward in that it interrupts the execution flow every time control transfer instruction (CTI) occurs, records the instruction between two CTIs, and then carries on with the execution from the destination address of the CTI.

Apart from this, there are various other techniques like autostart extensibility points that are also part of dynamic malware analysis. It must be noted that while dynamic analysis is far more efficient than static malware analysis, it is also much more time intensive and resource consuming, thus leading to elevated scaling issues.

V. TOOLS USED FOR MALWARE ANALYSIS

Before discussing the different malware analysis tools available popularly, a few terms need to be clarified. Malware analysis uses a tool called sandbox commonly, in order to run the unauthorized and possibly harmful piece of code or program without harming the host system. In computer security, a sandbox refers to a specified, separate environment, analogous to a container, with strict restrictions and permissions, where computer code can run without being able to inflict any damage or cause infection. Anything outside the sandbox is beyond the reach of the suspicious computer code. It is to be noted that a sandbox and a virtual machine are not the same thing. When a program runs in a sandbox, it has the permission to execute as though it was not in a sandbox. Any changes attempted by the application are lost when the application stops running. In contrast, anything changed or created by the application is allowed to remain in a virtual machine, and all actions stay within it.

The main job of the sandbox is to enable "users to automate the sample submission process; completely analyze any threat; and quickly act to protect sensitive data". [17]

Several online automated tools exist for the purpose of dynamic analysis of malware. Only a few of them shall be discussed here.

Norman SandBox [18]: "The Norman SandBox Analyzer is a utility meant to automate, simplify, and speed up the information gathering process when analyzing malware." [19] The sandbox provides a highly controlled environment and can thus be considered as a specific form of virtualization. Sandboxes are thus frequently used to test codes with malware, without causing harm to the host computer. They function by restricting the resources used by the execution of the piece of malicious code. One of the many advantages of using Norman SandBox is that as the malware gets executed in a simulated system, obfuscation cannot hinder the process of malware analysis itself. This aids in the detection of viruses and worms spread over email or via P2P networks. Alongside this, a general malware detection algorithm is also run in order to capture other kinds of malicious software.

Anubis: Anubis is developed by the International Secure Systems Lab and is capable of analyzing both files and URLs. Unknown binaries are analysed in an emulated environment of a Windows XP operating system in this project. The analysis is carried out by monitoring the system calls and Windows API functions. Function parameters are also tracked and monitored in this malware analysis project [20].

CWSandbox: CWSandbox is a tool for malware analysis that satisfies the three design conditions of automation, effectiveness and correctness. Dynamic analysis of malware is done to achieve automation. The program or code is executed in a simulated environment, a sandbox. Effectiveness is ensured by using the technique of API hooking. The calls to the Windows application programmers' interface (API) are sent to the monitoring software for analysis before the actual API code is called. API hooking ensures that all nuances of malware behavior are noted for which the API calls are hooked. Correctness of the tool is achieved by implementing the technique of DLL code injection. Briefly, DLL code injection can be described to allow API hooking to be implemented in a reusable and modular way. It is worthy of mention that although this tool can comment on the visible actions of the malware, it cannot describe how the malware was programmed. However, in spite of this drawback, the information gathered from executing malware using a CWSandbox is valuable and more often than not, sufficient to diagnose the dangers associated with the particular malware [21].

VI. Results and Discussion

It is with alarm that experts are noting how malware seems to be getting more and more sophisticated by the day. A rather bleak picture is easily painted, considering that compared to the advancements in the technology behind malware, operating systems and web browsers are not launched that fast.

One recent trend in the rise of malware attacks is noticeable in the field of instant messaging. As the various systems start allowing interaction between them, the number of malware attacks on them will increase. A similar pattern is noted among Massive Multiplayer Online Roleplaying Games (MMORPG), where malware authors take control of the accounts of the unsuspecting users and use them to their own malicious benefit. Top corporations these days are mostly worried about Trojan Horses. Using the help of carefully placed keyloggers or screen-scraping software, cybercriminals have taken to attacking specific computers, aiding in their vindictive pursuits of industrial espionage or similar financially motivated crimes. Such targeted attacks often remain undetected, for standard software are not capable of identifying them. Recent reports note many new and improved attacks. There was a malware specifically written to steal intellectual property. What was anomalous about the malware was its ability to crawl through different file types (Excel, PDF, etc), encrypt the intellectual information and send the stolen data to a remote server. Another method used by cybercriminals these days is known as "hack-back", as noted by Gunter Ollmann, vice-president of research for Damballa. This feature detects if and when a researcher is studying the malware,

and immediately compromises the researcher's machine. Similar steps are taken by several other botnet malware which activate DDoS – denial of service – attacks on researchers, if they come too close to the C&C (command-and-control) system. The Conficker worm actually blacklists users investigating the malware, who attempt to access the botnet server.

Although statistics show that there are about 30% more CVEs (common vulnerabilities and exposures) in Microsoft Office as opposed to PDFs, it is seen that the number of attacks on PDFs hugely overshadows the number of attacks on MS Office. [23] Prior to October, 2007, PDF attacks were non-existent; however, at the end of September that year, CVE-2007-5020 was released, exposing a major vulnerability in Adobe's PDF software. And thus began the bias of security attacks on PDFs over MS Office. Attackers use three major ways to compromise PDFs – mass mailing, drive-by downloads and targeted attacks. Mass mailing involves sending malicious PDFs via email and using social engineering in order to tempt users to open the file. On the other hand, drive-by downloads silently deliver the PDFs to the unsuspecting users' machines when they visit malicious websites. Targeted attacks are like mass mailings, except that the malicious PDFs are not sent in bulk, but are sent to a specific person or organisation.

Malicious PDFs can be enhanced in three distinct ways by the attackers. The first is it containing one specific exploit. The second one is the possibility of a malicious PDF having several exploits. And the third method of infection involves detecting the version of PDF software installed on the computer that is targeted. That way only the appropriate exploit shall be used. Lesser the number of exploits, greater is the chance of a successful attack.

Example of a de-obfuscated malicious JavaScript code showing how the PDF software version is detected:

```
function PDF(admwn.p collab)
{
var lv=Pdf1.GetVersions();
var fi=/EScript=([^,]+),/;
lv=lv.match(fi)[1].split('.');
lv=parseInt(lv.join(' '));
if(lv<=812)
{
    SHOWPDF(collab);
}
else
 {
    SHOWPDF(admwnp);
}
}
```

This piece of code identifies the type of PDF software and delivers a malicious PDF accordingly.

When analysing malicious PDFs, we can generally classify them into two categories: JavaScript based and Non JavaScript based. Because of its flexibility and ease of use, JavaScript is widely used in malicious PDF, exploiting a vulnerable JavaScript API and setting up the PDF reader program's memory with malicious code. Although the majority of malicious PDFs observed

in the wild use JavaScript, other techniques are also observed. One prevalent method is to embed Flash objects in the PDF. In addition to various distribution methods, attackers have also improvised different techniques to evade detection of the malicious content. Some of the malicious PDFs contain junk parts of code to throw off antivirus software. Others crash the PDF reader so as to give the user the illusion that they are corrupted files, while silently carrying out the vicious activities. Furthermore, various parts of a PDF can be obfuscated, thus not allowing detection. Obfuscation of PDF file format is much easier compared to other file formats such as MS Word, etc. A very simple obfuscation technique by which a vulnerable API call string is broken into smaller strings is demonstrated below.

```
try{eval("thi"+"s.m"+"ed"+"ia"+".n"+"ew"+"Pl"+"ay"+"er(n"+"ull)");}
catch(e){}
```

To avoid being infected, a user can perform the following precautions. Users can disable JavaScript support where possible. They should keep up-to-date with all the software patches available for the PDF reader software. Antivirus and IPS definitions should be up-to-date, and finally, users should always exercise caution while opening PDFs from an untrustworthy source.

Another major exploitation trend was spotted in Java as recently as 2012. [24] According to Oracle, 1.1 billion desktops run Java, and hence vulnerability in Java is bound to have a widespread effect on user security. The vulnerability in question had been fixed before the first appearance of the malware. However a researcher disclosed details of the vulnerability on his web page, which was shortly followed by the reveal of the malware. Shortly afterwards, it was noted that the exploitation of this vulnerability took over all other pre-existing Java vulnerabilities. This vulnerability is currently the number one vector for all drive-by exploits. Categorizing past Java vulnerabilities, four categories are noted overall: type confusion, logic error, memory corruption and argument injection. Type safety is a very important feature of Java security. Type safety is ensuring that a variable with a certain data type is not treated as a different data type in a program. On the failure of type safety, type confusion takes place. It is analogous to identity theft in the real world. An example of a past type confusion vulnerability in Java component is CVE-2012-0507: Atomic Reference Array type confusion vulnerability. Logic errors can reside within Java system code. CVE-2011-3544: Java Rhino Script Vulnerability is an instance of logic error where Security Manager is disabled. Memory corruption issues, while no longer a trend, have occurred before, such as CVE-2010-0842: Sun Java Runtime Environment MixerSequencer. Argument injection is very popular with Java plug-ins, as was noted in CVE-2010-0886: Java Deployment Toolkit Component. To analyse such Java vulnerabilities, specific static and dynamic research tools are required for Java binaries and the platform. CVE-2012-0507 is currently the most common vulnerability for drive-by exploits.

VII. Conclusion and Future Scope

The ideas of malware, the different types of malware and malware analysis have been discussed in details here. It is inferred from the information collected that dynamic analysis is a better method of malware analysis than static analysis. Although dynamic analysis has the obvious flaw of analyzing only one execution of the malware, static analysis is rather difficult to do properly, for in most cases, the source code is not visible. And even if the source code was available, it can never be ensured that no modifications were done to the binary executables which remained undocumented by the source. Hence, due to many such drawbacks, dynamic analysis is preferred over static malware analysis. On studying malware analysis tools, it can be concluded that the sandbox environment is particularly conducive towards analyzing malware. Recent trends in malware attacks show highly advanced techniques being applied to secure sensitive information. It is with great alarm that experts are noting how malware attacks are being more and more sophisticated in a short period of time, whereas operating systems and other user software are not produced in such brief intervals.

References

1. Gandotra, E., et al. (2014) Malware Analysis and Classification: A Survey. Journal of Information Security, 5, 56-64. http://dx.doi.org/10.4236/jis.2014.52006
2. (2013) Infographic: The State of Malware. http://www.mcafee.com/in/security-awareness/articles/state-of-malware-2013.aspx
3. "Symantec Internet Security Threat Report: Trends for July–December 2007 (Executive Summary)" (PDF). XIII. Symantec Corp. April 2008: 29. Retrieved 11 May2008.
4. "F-Secure Reports Amount of Malware Grew by 100% during 2007" (Press release). F-Secure Corporation. 4 December 2007. Retrieved 11 December 2007
5. Andreas Moser, Christopher Kruegel, and Engin Kirda, Limits of Static Analysis for Malware Detection, Secure Systems Lab Technical University Vienna
6. "What are viruses, worms, and Trojan horses?". Indiana University. The Trustees of Indiana University. Retrieved 23 February 2015.
7. http://resources.infosecinstitute.com/malware-analysis-basics-static-analysis/
8. (2014) Process Monitor. http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx
9. Capture BAT. https://www.honeynet.org/node/315
10. (2014) Process Explorer. http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx
11. Process Hackerreplace. http://processhacker.sourceforge.net/
12. Wireshark. http://www.wireshark.org/
13. Regshot. http://sourceforge.net/projects/regshot/
14. Egele, M., Scholte, T., Kirda, E. and Kruegel, C. (2012) A Survey on Automated Dynamic Malware-Analysis Techniques and Tools. Journal in ACM Computing Surveys, 44, Article No. 6.
15. Whole-system Fine-grained Taint Analysis for Automatic Malware Detection and Analysis Heng Yin hyin@cs.wm.edu College of William and Mary Dawn Song dawnsong@cmu.edu Carnegie Mellon University http://bitblaze.cs.berkeley.edu/papers/malware-detect.pdf
16. Efficient and stealthy instruction tracing and its applications in automated malware analysis: Open problems and challenges Endre Bangerter, Stefan B¨uhlmann, and Engin Kirda Bern University of Applied Sciences, Switzerland endre.bangerter@jdiv.org Bern University of Applied Sciences and Joe Security, Switzerland stefan.buehlmann@bfh.ch Northeastern University, USA ek@ccs.neu.edu, http://dl.ifip.org/db/conf/ifip11-4/inetsec2011/BangerterBK11.pdf
17. http://cwsandbox.org/
18. Norman Sandbox. http://sandbox.norman.no
19. Gadhiya et al., International Journal of Advanced Research in Computer Science and Software Engineering 3(4), April - 2013, pp. 972-975
20. Anubis. Analysis of unknown binaries. http://anubis.iseclab.org
21. Toward automated dynamic malware analysis using CWSandbox. http://dl.acm.org/citation.cfm?id=1262675
22. Christopher Elisan (5 September 2012). Malware, Rootkits & Botnets A Beginner's Guide. McGraw Hill Professional. pp. 10–. ISBN 978-0-07-179205-9
23. Karthik Selvaraj and Nino Fred Gutierrez, The Rise of PDF Malware, Symantec Security Response.
24. Jeong Wook (Matt) Oh (jeongoh@microsoft.com), Recent Java exploitation trends and malware, Black Hat USA 2012 Las Vegas.
25. Imtithal A Saeed, Ali Selamat and Ali M A Abuagoub. Article: A Survey on Malware and Malware Detection Systems. International Journal of Computer Applications 67(16):25-31, April 2013. Full text available.
26. Verma, Aparna, M.S.Rao, A.K.Gupta, W. Jeberson, and Vrijendra Singh. "A Literature Review On Malware And Its Analysis." International Journal of Current Research and Review 5 (2013), 71-82.

AUTHOR(S) PROFILE

Anusmita Ray, is a final year student of B.Sc. in Computer Science. Apart from her usual studies, she is involved in research work in Cryptography and Network Security, Malware Software and Big Data Analytics.

Dr. Asoke Nath, is Associate Professor in the Department of Computer Science, St. Xavier's College (Autonomous), Kolkata, India. Currently he is actively involved in research works in various fields such as Cryptography and Network Security, Steganography, Ethical hacking, Visual Cryptography, Green Computing, Big Data Analytics, Data Science, Li-Fi technology, DNA Cryptography, Cognitive Radio, MOOCs, Mathematical modeling of Social networks, e-learning methodologies and so on. He is life time member of MIR Labs (US) and CSI Kolkata Chapter. Dr. Nath delivered keynote address/invited talk/tutorial/short courses in several International conferences in India, US and in Vietnam. He has published more than 201 publications in International Journals and proceedings of conferences.

