Edith Cowan University
Research Online

Australian Digital Forensics Conference | Conferences, Symposia and Campus Events

2010

Lessons Learned from an Investigation into the Analysis Avoidance Techniques of Malicious Software

Murray Brand
Edith Cowan University

Craig Valli
Edith Cowan University

Andrew Woodward
Edith Cowan University

DOI: 10.4225/75/57b2903540cd7
Originally published in the Proceedings of the 8th Australian Digital Forensics Conference, Edith Cowan University, Perth Western Australia, November 30th 2010
This Conference Proceeding is posted at Research Online.
http://ro.ecu.edu.au/adf/74

Lessons Learned from an Investigation into the Analysis Avoidance Techniques of Malicious Software

Murray Brand, Craig Valli and Andrew Woodward
secau - Security Research Centre
Edith Cowan University
Perth, Western Australia
m.brand@ecu.edu.au
c.valli@ecu.edu.au
a.woodward@ecu.edu.au

Abstract
This paper outlines a number of key lessons learned from an investigation into the techniques malicious executable software can employ to hinder digital forensic examination. Malware signature detection has been recognised by researchers to be far less than ideal. Thus, the forensic analyst may be required to manually analyse suspicious files. However, in order to hinder the forensic analyst, hide its true intent and to avoid detection, modern malware can be wrapped with packers or protectors, and layered with a plethora of anti-analysis techniques. This necessitates the forensic analyst to develop static and dynamic analysis skills tailored to navigate a hostile environment. To this end, the analyst must understand the anti-analysis techniques that can be employed and how to mitigate them, the limitations of existing tools and how to extend them, and how to employ an appropriate analysis methodology to uncover the intent of the malware.

Keywords
Anti-forensics, anti-analysis, malware, digital forensics

INTRODUCTION

Malicious software (malware) can be considered as any software which has a malicious intent or effect (Aycock, 2006). This goes beyond consideration of viruses, worms, bots, trojans and similar categorizations of malware that attack computers on an almost indiscriminate basis. It extends to malware tailored to attack an individual or organisation to extract confidential information. Research shows that malware signature detection is far less than ideal and is limited to recognition of malware that has already been extensively analysed (Chouchane, Walenstein, & Lakhotia, 2007; Masood, 2004). It is very unlikely that a signature will exist to detect customised malware that will be of benefit to the forensic analyst in the field. In such cases, the analyst may be required to conduct static and dynamic analysis to determine the intent of the malware. However, an extensive variety of anti-analysis techniques exists to hinder the analyst from analysing the malware. Modern malware incorporates stealth techniques to hide it from the analyst, deception techniques to hide its true intent, and active techniques to defeat common analysis tools in their default configurations (Grugq, 2003; Harbour, 2007; Rutkowska, 2006a, 2006b). Such techniques are commonly referred to as anti-forensics and are becoming a very important consideration for the digital forensic analyst, as the majority of modern malware employs these analysis avoidance techniques (Falliere, 2007; Ferrie, 2008; Yason, 2007). A simple reality is that detailed analysis often cannot begin without mitigating these anti-analysis techniques.

It was reported in an online article that a speaker at the Australian IT Security in Government Conference claimed that 65% of new malware employs these anti-forensics techniques to avoid detection before, during and after an event (Kotadia, 2006). Further, anti-forensics has also been described as any attempt to hinder the forensic process by negatively impacting available evidence (Rogers 2006). Anti-forensics is arguably becoming a very important consideration for the digital forensic analyst. Malware is becoming increasingly stealthy and more likely to incorporate deception to stay on the target of interest in an attempt to remain undetected. This could be attributed to the substantial illicit financial gain that can be achieved from the employment of malware (Larsson, 2007; Newman, 2006).

LESSONS LEARNED

This research had two main avenues of enquiry. The first was a study of the effectiveness of the analysis avoidance techniques malware can employ, together with an investigation into how the use of these techniques can be detected and mitigated. The most significant papers on the employment of these techniques present code snippets of assembly language programs together with brief discussions on how the techniques can be mitigated

(Falliere, 2007; Ferrie, 2008; Yason, 2007). These same techniques were then incorporated by this research into small, standalone assembly language programs so that the effectiveness of the techniques could be determined using popular debugging tools in a series of quasi-experiments. All of the employed techniques were found to be effective. Popular debuggers are able to be scripted to automate analysis of disassemblies and this can be useful for static analysis. However, malware is often so heavily obfuscated, it must be allowed to run so that the instructions and data are de-obfuscated at run time to be able to be read and analysed. Debuggers are also able to be scripted to interact with the program at run time to assist in this endeavour. This feature is very useful in order to detect and mitigate anti-forensic techniques. The second line of enquiry was to analyse network based malware collected from a honeynet and examine how effective anti-virus (AV) software is at detecting malware and also to examine the use of anti-forensic techniques employed by the malware. It was essentially found that AV signature detection is less than ideal, and that the collected malware invariably contained anti-analysis techniques. These avenues of enquiry resulted in a number of lessons learned, and the key lessons learned as a result of this research are presented below.

Malware Signature Detection its Far Less Than Ideal

An examination of 738 malware specimens collected for the purposes of this research shows that even though the majority of the malware collected had been observed in the field for periods of time approaching or even exceeding one year, the unanimous detection by a collection of thirty six AV detection engines was only 10.4%. The particular AV engine used by the Anubis (International Secure Systems Lab, Vienna University of Technology, Eurecom France, & UC Santa Barbara, 2008) online virus analyser only recorded a 73.7% detection rate of the samples as malicious. This is a significant and potentially alarming result as it indicates that even though the use of AV software is considered mandatory for both personal and enterprise users, detection of all malware could be highly unlikely. This supports the findings of other researchers (Masood, 2004; Mohandas, n.d.; Skoudis & Zeltser, 2004; Szewczyk & Brand, 2008).

Presentation of a Taxonomy of Analysis Avoidance Techniques

A review of the literature uncovered an extensive range of techniques, mostly published by three key researchers who each provide their own, slightly differing taxonomies of analysis avoidance techniques (Falliere, 2007; Ferrie, 2008; Yason, 2007). Note that these papers have only been published within the past year or two of this research and this could be indicative of the problems encountered by the increased spectrum of techniques malware is now employing to hinder analysis. Their work is supplemented by other researchers whose online articles focus on more individual techniques and provide greater detail with respect to implementation and analysis (Anthracene, 2006; Gordon, n.d.; Rolles, 2007; Smidgeonsoft, 2005; Smith & Quist, 2006; xC, 2007). The work of Rolles in particular focuses on leading edge techniques such as malware that uses its own virtual machines to severely hinder detailed analysis. Such malware is difficult to analyse because the custom virtual machines have their own instruction sets, and these customised instruction sets have to be determined before detailed analysis can commence. A proposed taxonomy by the authors of this research combines elements of the taxonomies of Falliere, Ferrie and Yason, and appears in

Table 1, in an attempt to provide a more complete coverage of techniques. Note that each technique listed in the taxonomy is the highest level stratum and could be further stratified.

Analysis Avoidance Techniques are Very Effective

An extensive range of anti forensic techniques can be implemented in malware as indicated by

Table 1. The literature and search of reverse engineering web sites revealed more than 80 fundamental techniques. Note that these techniques can have multiple variations and can be used in various combinations. A number of these techniques were implemented in small standalone assembly language programs and tested against popular analysis tools in a series of quasi-experiments. All of the techniques were found to be effective against the tools in their default configurations. With the appropriate settings and/or with appropriate scripts or plugins, these techniques could be detected and mitigated. However, significant analysis skills are still required in order to successfully ascertain the modus operandi of each piece of malware.

Analysis Tools Have Deficiencies

A number of analysis tools are utilized by malware forensic analysts, with static and dynamic analysis representing two significant methodologies that can be used to analyse malware (Aquilina, Casey, & Malin,

2008). Software disassemblers and debuggers such as IDA Pro (Hex-Rays, 2008) and OllyDBg (Yuschuk, 2008) can be used to perform a detailed analysis of the malware code and provide an internal view of the malwares functionality (Valli & Brand, 2008). This is referred to as static analysis. In contrast, dynamic analysis runs the malware and observes the interaction of the running malware with the computer from a behavioural point of view. A number of plug-ins that extend the functionality of IDA Pro and OllyDBg include IDA Stealth (Newger, 2008) and Olly Advanced (MaRKuS, 2006) respectively to work with malicious code that employ anti-analysis techniques. The intention of such plug-ins is to provide functionality to hide their associated tools

Table 1: A proposed taxonomy of techniques employed by malware in order to avoid analysis

Technique	Description
Anti Emulation	A range of techniques exist to detect that the malware is running inside popular VM's such as VMWare or Virtual PC.
Anti Online Analysis	A variety of techniques exist for malware to determine if it is running in a specific online analysis engine such as Anubis or Norman Sandbox.
Anti Hardware	Techniques that target hardware such as the CPU including the debug registers to determine if it is being debugged.
Anti Debugger	Target the way debuggers work and take advantage of these to take control of the flow of execution. This gives malware the opportunity to incorporate deception.
Anti Disassemblers	Target the way disassemblers work and take advantage of this to produce a false disassembly.
Anti Tools	Detect the presence of specific analysis tools and enter a deceptive mode.
Anti Memory	Target the way memory is used when a process is being debugged and take advantage of this as well as the way processes can be dumped from memory including the use of stolen bytes.
Anti Process	Target the way processes are handled when being debugged and take advantage of this including structured exception handling.
Anti-Analysis	Target the way analysis is conducted. Use junk code, code camouflage, check sum checks, destruction of the Import Address Table and other deceptive techniques to make analysis harder.
Packers and Protectors	Use run time packers and protectors to obfuscate code and data and make it hard to unpack to find the original entry point. This includes packers that use their own virtual machines such as HyperUnpackme2.
Rootkits	Insert rootkits at Ring 0 to take control of the way the operating system manages processes and use deception to hide malicious processes.

from the malware they are analysing. The research conducted in this work showed that the number of anti forensic techniques covered by such plug-ins is much less than the number of techniques that are available to be implemented by malware. In addition, this research shows that although the plug-ins successfully hides the debugger or disassembler, the tools do not provide any information to the analyst about having detected the use of analysis avoidance techniques. This is significant for two major reasons. Firstly, because detection of the use of anti-analysis techniques in software may be of assistance to a digital forensic investigator to show that deception was used to hide malicious intent. Secondly, a false sense of security from using the plugins may lead to the analyst not conducting a thorough analysis of the malware and being the resultant subject of deception. This suggests a deficiency in existing tools that the analyst must be aware of.

A variety of scripting languages and Application Programming Interfaces (API) exist to extend popular debuggers in order to successfully detect and mitigate the use of anti-analysis techniques. Given the claim by

this research that existing plugins have severe limitations due to their lack of coverage of anti-analysis techniques and lack of logging functionality to show discovery of the use of these techniques, scripting of debuggers is an essential skill required for analysing malicious software.

Detection and Mitigation Techniques can be Effective

This research shows that the use of scripting for debuggers and disassemblers extends the functionality of the tools to facilitate the detection and mitigation of analysis avoidance techniques employed by malware. This research recommends that the development of debugger and disassembly scripting skills is requisite to being able to detect and counter analysis avoidance techniques of malware. This contribution exists at the current front line of research in the detection of malware.

The detection of anti-analysis techniques features far less in the literature than does the discussion of the employment of anti-analysis techniques. Detection of anti-analysis techniques in code would not only assist the analyst in investigation of malicious intent and the discovery of any attempts at deception, it also appears that detection of anti-analysis techniques may be a very good indicator that the code has a malicious intent. That is not to say that the use of anti-analysis techniques does not have a place in protecting the intellectual property of legitimate software. However, this research shows that malware invariably incorporates anti-analysis techniques and that detection of such techniques may warrant further investigation, even if the detailed analysis may have to be conducted by a specialist.

An Extensive Knowledge Domain is Required

A significant body of knowledge is required to obtain detailed information from manual analysis in order to determine the in-depth functionality of the malware (Valli & Brand, 2008). A short, non-exhaustive, requisite skills list for the analysis of Windows-based malware analysis indicated by Valli & Brand could include:

Assembly language programming ability
Program debugging skills
Static code analysis techniques
Dynamic code analysis techniques
Windows Applications Programming Interface (API) programming skills
Windows Operating System internals knowledge
Computer networking and network programming skills
Malware techniques knowledge
Reverse engineering skills
Knowledge of analysis avoidance techniques

This research shows that the extent of knowledge required to analyse malware is extensive. A proposed Malware Analysis Body of Knowledge (MABOK) was initiated by the conduct of this research where the treatment of anti-analysis techniques is a key and vital component. The reality is, that because malware extensively incorporates anti-analysis techniques, detailed analysis cannot start until the anti-analysis techniques are mitigated.

Packers and Protectors are Extensively used by Malware

Run time packers are utilised by network based malware to compress malware and to act as a counter measure to signature based AV software via obfuscation (Sun, Ebringer, & Boztas, 2008). The packed malware has to be unpacked before an investigator or specialist can perform a detailed static analysis, because packed malware obfuscates the malware code. Knowledge of the packer used assists in the process of unpacking because the appropriate unpacking methodology can be employed to unpack it. Software tools are available that attempt to determine the name of the packer that was used to pack the malware by signature recognition. This research showed that two popular packer detectors that were used by this researcher did not agree on the names of any of the packers that were used. This is significant because it indicates uncertainty could be associated with the determined packer signatures and that more in depth, manual analysis is required to validate the type of packing that was employed. Generally, once the packer signature has been determined, the appropriate algorithm can be applied to unpack the malware to arrive at the original entry point (OEP), which is the original entry point of the program before it was packed. However, if conflicting packer signatures are determined from two or more packer signature detectors, both algorithms may have to be applied to arrive at the OEP, and there is no guarantee that either one of them is correct without validation from a manual analysis perspective. This has

implications with respect to the consumption of the time available to the analyst and certainly benefits the malware writer whose objective is to prevent or hinder analysis of the malicious code.

The line of this research was extended to examine entropy (randomness) measurements of the packed malware as a method of determining if the collected malware was packed or not. Entropy measurements are shown in this research to be a very good indicator that malware has been packed.

Derivation of an Appropriate Analysis Methodology

A review of the literature on malware analysis methodologies found that the most effective methodologies take the presence of analysis avoidance techniques into account (Skoudis & Zeltser, 2004; Zeltser, 2007). Zelter (2007) presents an incremental, static and dynamic spiral analysis methodology for analysing malware which additionally moulds the analysis environment as understanding of the malware is attained. A simple example of Zelter's methodology begins by performing a basic static analysis of the malware specimen, such as performing a virus scan, determining the type of file and the type of packer used. This is followed by setting up a suitable environment to examine the specimen in, such as Windows XP in a virtual machine if the type of malware was a Microsoft Windows executable file. This is followed by running the malware and observing its behaviour with dynamic analysis tools. Using knowledge gained from this phase, static analysis can then be used to focus on sections of code or data of interest. This spiral based approach continues until as much detail on the malware has extracted as required. This methodology was adapted to include the detection and mitigation of anti-analysis techniques in each phase.

Figure 3 depicts a proposed analysis methodology for analysing malware that facilitates the discovery and mitigation of analysis avoidance techniques as an extension to Zelter's analysis methodology. The advantage of this adapted methodology is that when anti forensic techniques are encountered, they can be detected and mitigated before proceeding with the analysis. The analysis begins with a preliminary static analysis, such as determination of the file type of the malicious executable under investigation, and an appropriate static analysis environment is then established. Anti-analysis techniques can be categorized to target static and dynamic analysis techniques, therefore, it makes sense to focus on detecting and mitigating static analysis avoidance techniques before conducting a detailed static analysis. This phase is followed by tailoring an appropriate dynamic analysis environment using any relevant information found by the previous phase. Any dynamic analysis avoidance techniques are then detected and mitigated before proceeding with a more detailed dynamic analysis. The information discovered during this cycle is then used as input to determine how to proceed with the next static analysis phase. This spiral cycle continues until enough satisfactory information about the malware has been extracted.

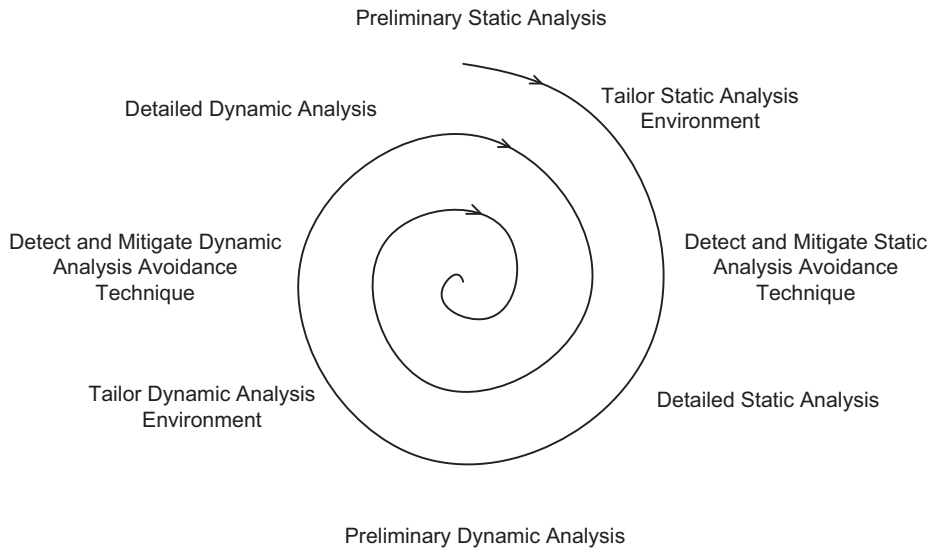

Figure 3: An extension of Zelters spiral analysis methodology proposed as a means of analysing and classifying avoidance techniques employed by malware.

An Alternate Paradigm for Malware Detection is Required

Anti-virus software typically uses signature matching and recognition of heuristics to detect malware. This approach generally requires the malware to have been collected from targeted or victim computers, subsequently analysed, and signatures downloaded to client computers to approach any level of effectiveness. Significant damage to computers could occur between the time of collection and the point at which signature updates have been performed. In addition, it is very unlikely that AV software will detect custom malware that has not been set loose on the internet, but targeted against an individual or a corporation, because it will not have been analyzed and a signature will not have been obtained by an AV company. Anti-virus software that uses this approach is seen to be fighting a losing battle, both in the literature and from observations made during this research (Mila Dalla, Mihai, Somesh, & Saumya, 2008; Zhou & Meador Inge, 2008). This research supports a proposal for a new paradigm for malware detection. In particular, this research proposes that detection of deception and anti-analysis techniques in software should flag the software as potentially malicious and delegate for further in depth analysis or removal.

AVENUES FOR FURTHER RESEARCH

A number of avenues exist for furthering this research, with suggestions to include plugin development for analysis tools, collation of analysis avoidance techniques, improved packer and protector identification and a new paradigm for malware detection. These suggestions are expanded upon in the following sub sections.

Plugin Development

This research noted that plugins such as IDAStealth and OllyAdvanced provide coverage for only a subset of analysis avoidance techniques. A limitation of the existing plugins is that their focus is on hiding the debugger and do not have the ability at this point in time to detect and log the use of anti-analysis techniques. Additional research could be conducted on extending the coverage of techniques of such plugins. The detection and logging of techniques as they are discovered during forensic analysis of malware could assist in the collection of evidence suitable for a court of law.

Collation of Analysis Avoidance Techniques

This research revealed an extensive range of analysis avoidance techniques that is currently distributed amongst research papers, hacking, and reverse engineering web sites. Detection and mitigation techniques are not heavily represented in academic literature, as compared to the information available on hacking and reverse engineering sites. A very useful contribution to the field of malware analysis research could be to collate analysis avoidance techniques, together with their corresponding detection and mitigation techniques into a central library and to further develop an encompassing taxonomy.

Improved Packer Signature Detection

Packer signature detection has been revealed in this research to be an area that requires further and most likely, continual research. This also extends to the area of unpacking packed malware as well, because malware can use multiple packers not only from a sequential sense. For example, pack the entire malware specimen with packer A and then pack the result with packer B, but firstly pack sections of code with packer A and then pack the result with packer B. This last scenario is another deception technique that is generally only uncovered once manual analysis is conducted. It is possible that an automated analysis process may miss the second, or subsequent iterations, of packing. This remains an area of research that lacks published work, and should be a focus for future research in order to increase levels of detection in general.

A New Paradigm for Malware Detection

This research has shown AV software to be less than fully effective at detecting malware. Research could continue into investigating a new paradigm for malware detection, particularly by detecting the use of anti-analysis techniques in scanned software and flagging it for more detailed attention.

CONCLUDING REMARKS

Malware can use anti-forensic techniques and use deception to hide its real purpose whilst being analysed. If it does not perform any malicious action while it is being analysed, it may be accepted on the system as being safe, or excluded from the evidence collection process. Once free from analysis, the malware can perform its original, malicious objective.

Some considerations must be made in order to closely analyse malware. Firstly, totally relying on AV software to classify the malware could be a mistake because signature based detection is far from ideal as it is unlikely to recognise customised malware that has not been analysed before. This leads to the necessity of the digital forensic analyst to analyse the malware manually. It must be noted that a significant number of anti-analysis techniques exist covering the entire spectrum of the computational mechanics of computers. These techniques are very effective at hindering analysis, can be compounded by additional factors, and include deficiencies in analysis tools that do not cover the number of anti-analysis techniques that are available to be employed. Analysis is made more difficult by the number of packers and protectors that malware can use. This makes it hard because a typical technique to unpack the malware is to use known algorithms to let the malware unpack itself to reach the OEP. In doing so, control is given to the malware and an opportunity exists for the malware to detect that it is being analysed and provides the opportunity of the malware to employ deception. An additional consideration is that a very extensive knowledge of programming, debugging and operating system internals is required that arguably exceeds the level attained even by competent software engineers. On the positive side, the use of anti-analysis techniques can be detected and mitigated if significant analysis skills have been attained. This can be assisted by using an appropriate methodology where static and dynamic methods are combined in such a way that the view of the malware transitions from a high level of detail down, to a low level of detail, mitigating the anti-analysis techniques as analysis progresses in a spiral analysis methodology. Although legitimate software uses anti-analysis techniques to protect itself from reverse engineers, malware is almost certain to use anti-analysis techniques. So much so, the detection of the use of anti-analysis techniques may be a very good indicator of the presence of malware.

As suggested previously, if more work is not undertaken in this field, there are a number of adverse consequences. It is possible that the use of these anti-forensic techniques will not only make the job of the forensic investigator difficult, there is also the potential to render anti-virus software next to useless as a means of protecting computers and networks.

REFERENCES

Anthracene. (2006). Unpacking with Anthracene. Retrieved August 21, 2007, from http://www.tuts4you.com/download.php?list.18

Aquilina, J., Casey, E., & Malin, C. (2008). Malware Forensics Investigating and Analyzing Malicious Code. Burlington, MA: Syngress.

Aycock, J. (2006). Computer Viruses and Malware. New York: Springer

Chouchane, M., Walenstein, A., & Lakhotia, A. (2007). Statistical signatures for fast filtering of instruction-substituting metamorphic malware. Paper presented at the Proceedings of the 2007 ACM workshop on Recurring malcode.

Falliere, N. (2007). Windows Anti-Debug Reference. Retrieved October 1, 2007 from http://www.securityfocus.com/infocus/1893

Ferrie, P. (2008). Anti-Unpacker Tricks. Paper presented at the 2nd International Caro Workshop. from http://www.datasecurity-event.com/uploads/unpackers.pdf

Gordon, J. (n.d.). Win32 Exception handling for assembler programmers. Retrieved Feb 10, 2008 from http://win32assembly.online.fr/Exceptionhandling.html

Grugq. (2003, August 26, 2006). The Art of Defiling, Defeating Forensic Analysis on UNIX File Systems. Paper presented at the Black Hat Asia 2003, Singapore.

Harbour, N. (2007). Stealth Secrets of the Malware Ninjas. Retrieved October 20, 2007 from https://www.blackhat.com/presentations/bh-usa-07/Harbour/Presentation/bh-usa-07-harbour.pdf

Hex-Rays. (2008). IDA Pro.

International Secure Systems Lab, Vienna University of Technology, Eurecom France, & UC Santa Barbara. (2008). Anubis: Analyzing Unknown Binaries. Retrieved October 4, 2008, from http://anubis.iseclab.org/

Kotadia, M. (2006). Beware 'suicidal' malware, says CyberTrust. Retrieved August 27, 2006 from http://software.silicon.com/malware/0,3800003100,39160966,00.htm

Larsson, L. (2007). Meeting the Swedish Bank Hacker. Retrieved April 14, 2007 from http://computersweden.idg.se/2.2683/1.93344

MaRKuS. (2006). Olly Advanced.

Masood, S. G. (2004). Malware Analysis for Administrators. Retrieved 17 March, 2007 from http://www.securityfocus.com/infocus/1780

Mila Dalla, P., Mihai, C., Somesh, J., & Saumya, D. (2008). A semantics-based approach to malware detection. ACM Trans. Program. Lang. Syst., 30(5), 1-54.

Mohandas, R. (n.d.). Hacking the Malware – A reverse-engineer's analysis. Retrieved 17 March 2007, from geocities.com/rahulmohandas/hacking_the_malware.pdf

Newger, J. (2008). IDA Stealth Plugin.

Newman, R. (2006). Cybercrime, identity theft, and fraud: practicing safe internet - network security threats and vulnerabilities. Paper presented at the Proceedings of the 3rd annual conference on Information security curriculum development.

Rolles, R. (2007). Defeating HyperUnpackMe2 With an IDA Processor Module. Retrieved Feb 28, 2008 from http://www.openrce.org/articles/full_view/28

Rutkowska, J. (2006a). Fighting Stealth Malware - Towards Verifiable OSes. Journal. Retrieved from http://www.invisiblethings.org/papers/towards_verifiable_systems.ppt

Rutkowska, J. (2006b). Introducing Stealth Malware Taxonomy. Journal. Retrieved from http://www.invisiblethings.org/papers/malware-taxonomy.pdf

Skoudis, E., & Zeltser, L. (2004). Malware Fighting Malicious Code. New Jersey: Prentice Hall.

Smidgeonsoft. (2005). SetUnhandledExceptionFilterTrick. Retrieved Feb 11, 2008, from http://www.openrce.org/forums/posts/45

Smith, S., & Quist, D. (2006). Hacking Malware: Offense is the new Defense. Retrieved July 24, 2007 from http://www.offensivecomputing.net/dc14/valsmith__dquist_hacking_malware_us06.pdf

Sun, L., Ebringer, T., & Boztas, S. (2008). Hump-and-Dump: efficient generic unpacking using an ordered address execution histogram. Journal. Retrieved from http://www.datasecurity-event.com/uploads/hump_dump.pdf

Szewczyk, P., & Brand, M. (2008). Malware Detection and Removal: An Examination of Personal Anti-Virus Software. Paper presented at the 6th Australian Digital Forensics Conference, Edith Cowan University, Mount Lawley Campus, Western Australia.

Valli, C., & Brand, M. (2008). Malware Analysis Body of Knowledge. Paper presented at the 6th Australian Digital Forensics Conference, Edith Cowan University, Mount Lawley Campus, Western Australia.

xC. (2007). Defeating Anubis File Analyzer. Retrieved Jul 21, 2007 from http://www.ryan1918.com/viewtopic.php?p=68714&sid=354448fa02136b766d94dfcea11b4e2d

Yason, M. (2007). The Art of Unpacking. Retrieved Feb 12, 2008 from https://www.blackhat.com/presentations/bh-usa-07/Yason/Whitepaper/bh-usa-07-yason-WP.pdf

Yuschuk, O. (2008). OllyDbg.

Zeltser, L. (2007). Reverse Engineering Malware: Tools and Techniques Hands-On. Bethesda: SANS Institute.

Zhou, Y., & Meador Inge, W. (2008). Malware detection using adaptive data compression. Paper presented at the Proceedings of the 1st ACM workshop on Workshop on AISec.