Evading Malicious Code with Concurrent Programming in Parallel Architectures and Their Protection Methods

Caglar SAYIN

Master's Thesis Project Description
Master of Science in Information Security
5 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2013

Avdeling for
informasjonssikkerhet
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Information Security
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Revision history

Version #	Description of change (why, what where - a few sentences)
0.1	First version is made available via Fronter

Abstract

Abstract (1/2 page) This document provides format and guidelines for the MSc project descriptions. The document has been produced using MikTeX and TeXnicCenter.

The objective of the abstract is to provide the reader with an understanding of the work to be done and put him in the position to make a 'correct' decision regarding reading/not reading the report.

The abstract of the project description *must* include

- a summary of the problem description,
- motivation and
- a summary of the planned contribution from the master project in terms of *new* results.

Control questions

1. Does the abstract have a 'reasonable' length?
2. Is it clear to a non expert (e.g. a typical reader of a newspaper) what problem is addressed?
3. Does a person that has been working in the field find the text informative?
4. Do the results that might be obtained have the potential to be interesting to a lot of people? How interesting to how many and why?
5. Would a decision maker/manager be willing to pay NOK 400.000 to have the project completed (estimated salary costs + overheads) after having read the abstract? Why/why not?

Contents

1 Introduction (1-2 pages)

The purpose of introduction chapter is giving the readers blueprint of the subject, the problems that we face, the change in the solutions, as well as motivation of its importance. In addition, It also purpose to form proper research question which will guide thesis.

1.1 Topic covered by the project

The thesis purposes an architecture of the malware which process parallel, access memory concurrently, conceal itself systematically, shortly that it is likely to be rocket science. However, everything actually started with a simple mathematical theory by John Von Neumann [1] and the first example of practical malware is writen by Bob Thomas at BBN, and it was called Creeper

The malware is abbreviation of malicious software. It could be any piece of code which is defined malicious. There is no formal definition of malicious, it could be some software advertise without any contest or it could be self-producing code piece which aim to distribute itself and steal your private information, and it turned an arm race between power holders today.

With development of the first malware, their counter software are created and anti malware software have evolved with them so far. In this race, malware authors are always one step further, because of security's nature. This race between black and white side raised the bar above. The motivation of the information amount and severity influence both today, and that information can be sometimes vital.

1.2 Keywords

Security, Concurrent Malware Design, Malware, Concurrent, Parallelism and Concurrency

1.3 Problem description

The one of the main and indecipherable problem in security discipline is formulating general threat definition and recognizing malicious activity and all this problems unsurprisingly reflect on information and computer security concept. Security is defined by system's identification, which involve with purpose, crowd, design structures, network model and so on, and today's information system which is designed with various architectural forms is protected against malware by general purpose protection tools. In the market, The anti malware tools producers focused on pragmatic solutions to survive, but it leads to that most of these tools are utterly reverse engineering process which works on result instead of reason.

With usual and pragmatic signature based methods, there are two mainstream techniques to detect malicious code which are called static and dynamic analysis.Static analysis identifies malwares mainly with code flow graph and data flow graph on stored file which is not processing. However, On the dynamic side it is a bit more tricky to analyze process, because you are working on the running pieces of codes without knowledge of

structures and worse than this, it must concern race condition and memory coherency flaws.

The detection methods and techniques have been adequately worked so far because of the simplicity of architectures and usage of the massive generic computers, However, with increasing of the not standardized, parallel and popular devices like arm's SoC, it is not hard to estimate their new challenges. It is really likely to evade and obfuscate properly your on-the-fly processes with using uncertain charactership of parallel processing, complexity of concurrent programming, and structure of "Non Uniform Memory Architecture".

1.4 Justification, motivation and benefits

If malware designing is superficially considered, you could fall in usual fallacy that It is not beneficial and exactly opposite. However, if we can design it, there is always more skillful author who already abuse this vulnerabilities on the black side of the moon. The work we are obligated to actually proof this vulnerabilities and design counter measure against them. In this way, our blessed motivation is finding possible vulnerabilities, and mitigate or eliminate their risk. Otherwise; if we confront with unknown attack, it could be too late to fix and analyze it. For example, some of the most sensational and beneficial papers are criticize malware as same as the thesis ([2],[3],[4]), and their values are undoubted today.

1.5 Research questions

1. Can a malware model be designed with using parallel and concurrent architecture in order to conceal its presence from detection mechanism?
2. If we can design the mentioned malware, can we build a detection mechanism against these kind of malware's presence?
3. If we build the detection mechanism, What is detection complexity of the algorithms?

1.6 Planned contributions

This Master thesis is looking for better understanding on concurrent malware abilities and their counter-measure. Especially, It will try to show how possible to abuse concurrent memory accessing and how durable recent detection kits. It is quite unique work which we have to consider on the future. Ultimate goal is to eliminate any uncertainties which detection methods encounter with concurrent memory accessing.

2 Related work (3-10 pages)

This chapter will give an overview of researches about Malware‘s self-defense technique, the methods to analyze them, and their application on concurrent architecture.

2.1 Malware Self-Defense

This section will try to give the literature about malware evasion techniques. This techniques are generally antonym solution which are found by malware authors, however, there are enough surveys about known technique. We classified all these methods in six categories which are code obfuscation, code reuse, anti debugging, anti emulator and visualization and covert channel over network traffic. This taxonomy is well defined by Jonathan A.P Marpaung, et al [5], yet malware authors used them to protect their own properties.

Code obfuscation was originally found for protecting intellectual property[6], but It aims to puzzle code‘s binary against merely static analysis and disassembling[7]. The first known obfuscation method used encryption in order to hide its content. It was called Cascade which is seen first 1986[8]. This simple architecture of the obfuscation is called packing[9]. It involve with two part of binary which are slub part, in order to decipher and encipher.[5]. Cascade was using simple XOR encryption and that was increasing performance.

Early of the 1990s , oligomorphism and polimorphism started to show up[8]. The main idea behind them is basicly transforming their slub part in each attempt of encryption process[7]. Today, there are two type of polymorphic approach to generate different variants of slub.[10]

- Rewriting the code each time from pseudo-code so it differs code synthetically which is actually transformation based obfuscation.
- Self-cipher itself different, order of these ciphers and using different keys.

One of the other important milestone of polymorphic malware is Mutation Engine(MtE) is writen by a Bulgarian virus Author, called The Dark Avanger. It was automated obfuscation tool which actually considered impossible in those times.[11]

There are also several methods to prevent unpacking process. These methods are collected carefully by Peter Ferrie [12]. These methods are especially obstacle for automated analysis.

Compare with polymorphic methods, metamorphic approach is more complicated. It is transformation based method instead of encryption approach.[13] Fundamentally, it produce different codes which doing same blue printed semantic. That just mitigate detection possibility because of lack of static code.

Network traffic, which malware produce are generally Achilles heel for malware, because they are generally adequately unique traffics to be identified[5]. They usually cover their overt malicious traffic with covert channel methods.[14]

Code reuse attacks are strong attacks because they do not inject any code in them as obfuscation methods did. They aim to use legitimate software to evade themselves.

There are there well known applied version which are return-into-libc, return oriented programming and Frankenstein.

Return into libc attacks were demonstrated by solar designer in 1997 as a method of bypassing non executable stack to executable libc libraries[15]. It's object is to change the "ret" infrastructure argument to the known address possibly libc library(stdio, system, etc). However, this attack is limited with libc libraries, which we improved with return oriented programming.

Return oriented programming is more flex version of retur-into-libc attack, which is introduced by Shacham in 2007[16]. Return oriented programming purpose a programming language with small gadgets(instruction bound) which involve all ability of Turing's machine[17]. Frankenstein is one of the novel application of return oriented programming by Vishwath Mohan and Kevin W. Hamlen[18].

Anti debugging and anti emulator methods are really usual for today's malware. The survey of Chen Xu et al. showed us in 2008, majority of 6900 on-the-air malware could evade their self with exhibiting benign behavior in sandboxes, debuggers, and virtual machines.[19]. VM and debuggers are most important element of dynamic analysis techniques in autonomous sector, because it must run the file just before it touch the working environment. Yet, it is not that knotty to determine whether working environment is virtual or not. Fuzzing cpu bechmarks and comparing results entropy is a good way to determine virtual machines.[20]

Rootkits are the piece of malicious code which aims integrity of the system state.

2.2 Malware analysis methods

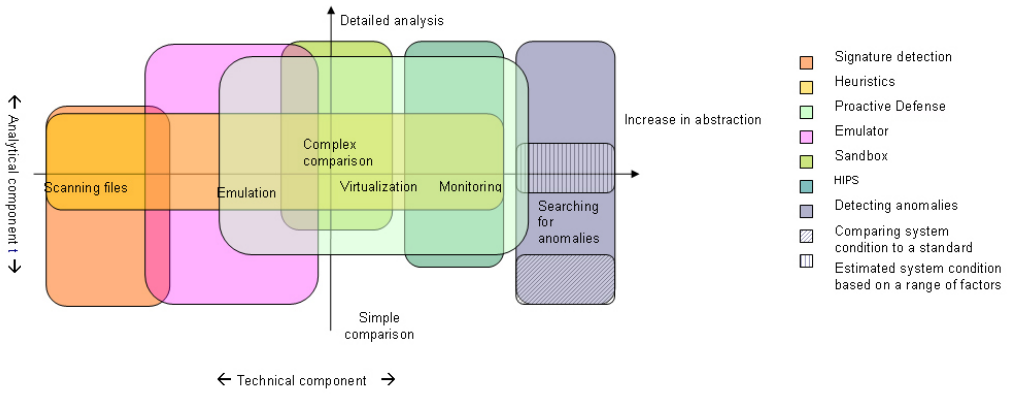

Figure 1: Detection models [**?**]

Malware analysis medhods could be considered two dimensional plane which are "anomaly, signature based detection technique" and "Statistic and dynamic analysis method".

There are also several applied techniques, which combine terminology above.

N-gram It is a anomaly based heuristic detection method algorithm. It is a bit costly process and not practical for client side analysis. It could be fit for honey pot analysis [21] [22] [23]. They are capable against to zero time malwares and that could makes it futures malware detection system.

Sequential approach on system and funtion call This approach is anomaly based dynamic analysis and observing and recording the flow graph of systems and function calls and try to analysis anomaly behaviors.[24]

Taint It is also called data flow analysis or data flow graph. It is basic tracking marked data values during execution.[25][25][26]

Control Flow Graph They are one of the most important arm of commercial autonomous malware detection tools[27] [28] [29]. After the invention of the polymorphic and metamorphic, syntactic analysis could not bear with them. Then we moved to upper layer of information, semantic layer. Semantic can be representation of code flow, and the routes of the code are adequate to produce signature to identify malware. This methods are a member of static analysis and disassembling and source code analysis job.

Network Monitoring Malware intention of the communication over network actually big clue to detect them. They generally use unique hostname, ip adress or specific protocol with particular way [5].

3 Choice of methods (2-5 pages)

This section is to include a description of the methods to be used, including references to literature describing the methods to be used (e.g. qualitative, quantitative, interviews, surveys, questionnaire, model building etc.) For each of the research questions to be addressed, the chapter is to explain why the method is

- appropriate
- likely to provide the desired knowledge/information.

4 Milestones, deliverables and resources (2-5 pages)

The purpose of this chapter is to convince the reader that you know exactly what to do. This chapter gives a description of how the project is to be broken down into smaller parts and activities.

1. What is it you have to do in order to obtain the desired knowledge?
2. What deliverables are to be produced (MSc thesis report, software,...)
3. When are the various deliverables going to be available?

For each deliverable, identify 4 versions, having an 'increasing' degree of completeness/quality. Students are strongly recommended to review each others drafts. For each version of a deliverable explain why and how this version is to be better/more complete. E.g. v1.0: my first draft - chapter text includes 1/2 page summaries only. v.2.0: Like v1.0, but comments by NN(who? fellow student) has been incorporated. v3.0:....

This section is to include a preliminary table of contents for the MSc thesis (only include 2 levels).

For each of the activities identified, specify

1. the time you need to complete each activity both calendar time and 'man-hours'.
2. hours needed by you
3. things you need to buy (consumables)
4. equipment, lab space or facilities you need access to
5. contributions from others (e.g. survey/interview participants) and how much each will have to contribute in terms of resources (probably time)

At the beginning of this section, provide a 2-3 line summary of the resource requirements. This is particularly useful if you have broken down the task into a lot of small tasks.

5 Feasibility study (1/2-3 pages)

An analysis of why it is likely that the desired results can be produced within the given time and resource bounds. This may include a description of

- similar projects completed by others and their 'resource consumption',
- an attempt to answer parts of the research questions
- the 'difficult' elements of the work and an explanation of why/how these problems can be solved. Alternatively you can explain an 'approximate' solution.

6 Risk analysis (1/2-2 pages)

In this project, there five inevitable risks which we can face during development.

- The thesis is highly dependent on the hardware, and the cost of the hardware constitute risk on its own. Any case of hardware defect leads to comprise obstacle.
- Hardware dependency is also leads to logistical and time consuming risk which could result with latency on submit time.
- Firmware codes which we are planning to work on are mostly undocumented. We could discover their usage by proper reverse engineering and fuzzing process when required, however it is obviously manpower.
- Most important and highlighting risk is there isn't proper research on this particular area. That means there are strongly possibly hidden risks which could cause other mental and physical result.
- During testing and purification part, Anti-malware tools could come out with unreliable result. To analyze result properly, we may need to inspect mentioned tools with reverse engineering process which could violate proper usage agreement. To mitigate that kind of risks, we could request research agreement from companies.

7 Ethical and legal considerations (1/4-1 page)

The content of this document could be used in order to malicious purpose, but any matter or information could be misused in the life and ignorance is not known well as a defense strategy. In this purpose, this thesis aims to enlighten security specialist and system developers against recent way of the possible attacks.

However, in order to act ethical responsibility, we tried to eliminate practice of tools and piece of codes which could leads malicious usage. In any case, there is no doubt that it is critical to discover and publish vulnerabilities which could cause deep impact before malicious people discover and abuse them.

> "Virus don’t harm, ignorance does."
> - VxHeaven

Bibliography

[1] Von Neumann, J., Burks, A. W., et al. 1966. Theory of self-reproducing automata.

[2] Moser, A., Kruegel, C., & Kirda, E. 2007. Limits of static analysis for malware detection. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, 421–430. IEEE.

[3] Cavallaro, L., Saxena, P., & Sekar, R. 2008. On the limits of information flow techniques for malware analysis and containment. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, 143–163. Springer.

[4] Egele, M., Scholte, T., Kirda, E., & Kruegel, C. 2012. A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys (CSUR)*, 44(2), 6.

[5] Marpaung, J. A., Sain, M., & Lee, H.-J. 2012. Survey on malware evasion techniques: state of the art and challenges. In *Advanced Communication Technology (ICACT), 2012 14th International Conference on*, 744–749. IEEE.

[6] Balakrishnan, A. & Schulze, C. 2005. Code obfuscation literature survey. *CS701 Construction of Compilers*, 19.

[7] Nachenberg, C. 1996. Understanding and managing polymorphic viruses. *The Symantec Enterprise Papers*, 30, 16.

[8] You, I. & Yim, K. 2010. Malware obfuscation techniques: A brief survey. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, 297–300. IEEE.

[9] Team, I. S. Bypassing anti-virus scanners. *Packet Storm Security*.

[10] Li, X., Loh, P. K., & Tan, F. 2011. Mechanisms of polymorphic and metamorphic viruses. In *Intelligence and Security Informatics Conference (EISIC), 2011 European*, 149–154. IEEE.

[11] anonymous. Polymorphic generators. In *VxHeaven*.

[12] Ferrie, P. 2008. Anti-unpacker tricks. In *Amsterdam: CARO Workshop*.

[13] Konstantinou, E. & Wolthusen, S. 2008. Metamorphic virus: Analysis and detection. *Retrieved on February*, 22, 2011.

[14] Rutkowska, J. 2006. Rootkits vs. stealth by design malware. *Black Hat, Europe*.

[15] Designer, S. 1997. Getting around non-executable stack (and fix).

[16] Shacham, H. 2007. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM conference on Computer and communications security*, 552–561. ACM.

[17] Roemer, R., Buchanan, E., Shacham, H., & Savage, S. 2012. Return-oriented programming: Systems, languages, and applications. *ACM Transactions on Information and System Security (TISSEC)*, 15(1), 2.

[18] Mohan, V. & Hamlen, K. W. 2012. Frankenstein: Stitching malware from benign binaries. In *WOOT*, 77–84.

[19] Chen, X., Andersen, J., Mao, Z. M., Bailey, M., & Nazario, J. 2008. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, 177–186. IEEE.

[20] Franklin, J., Luk, M., McCune, J. M., Seshadri, A., Perrig, A., & Van Doorn, L. 2008. Remote detection of virtual machine monitors with fuzzy benchmarking. *ACM SIGOPS Operating Systems Review*, 42(3), 83–92.

[21] Reddy, D. K. S. & Pujari, A. K. 2006. N-gram analysis for computer virus detection. *Journal in Computer Virology*, 2(3), 231–239.

[22] Abou-Assaleh, T., Cercone, N., Keselj, V., & Sweidan, R. 2004. N-gram-based detection of new malicious code. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, volume 2, 41–42. IEEE.

[23] Abou-Assaleh, T., Cercone, N., Keselj, V., & Sweidan, R. 2004. Detection of new malicious code using n-grams signatures. In *PST*, 193–196.

[24] Kendall, K. & McMillan, C. 2007. Practical malware analysis. In *Black Hat Conference, USA*.

[25] Saxena, P., Sekar, R., & Puranik, V. 2008. Efficient fine-grained binary instrumentationwith applications to taint-tracking. In *Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization*, 74–83. ACM.

[26] Smith, G. 2007. Principles of secure information flow analysis. In *Malware Detection*, 291–307. Springer.

[27] Lee, J., Jeong, K., & Lee, H. 2010. Detecting metamorphic malwares using code graphs. In *Proceedings of the 2010 ACM symposium on applied computing*, 1970–1977. ACM.

[28] Christodorescu, M. & Jha, S. Static analysis of executables to detect malicious patterns. Technical report, DTIC Document, 2006.

[29] Christodorescu, M., Jha, S., Seshia, S. A., Song, D., & Bryant, R. E. 2005. Semantics-aware malware detection. In *Security and Privacy, 2005 IEEE Symposium on*, 32–46. IEEE.