# Obfuscation: The Hidden Malware

**ESE546: Computer Communication & Networks**

**Final Project**

**Jaya Godugu**

**108209853**

# Obfuscation: The Hidden Malware

This issue from the journal Security & Privacy talks about obfuscation of malware and various techniques of obfuscation.

* What is a **Malware**?

Malware stands for “Malicious software”, and is generally a code snippet, scripts etc .. which is designed to disrupt operation, steal personal /sensitive data or gain unauthorized access to system’s resource or execute abnormal behavior. Some of the most commonly found malwares are:

* **Computer Viruses**: Program that can replicate itselfand spread from one computer to another
* **Worms**: Self-replicating program, which uses a computer network to send copies of itself to as well as other nodes on the network without any intervention.
* **Trojan horses**: Software that appears to perform a desirable function for the user prior to run or install, but actually steals information or harms the system.
* **Spyware**: A program which collects small pieces of information about users without their knowledge
* **Rootkits**: Software that enables continued privileged access to a computer while actively hiding its presence from administrators by subverting standard operating system functionality or other applications.
* Why does one need **Obfuscation**?

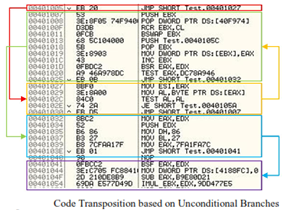
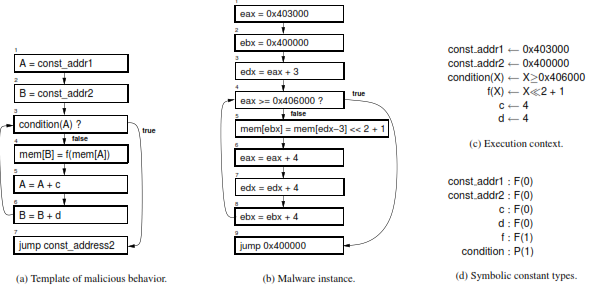
In order for a malware to spread and evolve, it must remain undetected for a long period of time .In fact the longer the malware conceals itself the more it evolves. Obfuscation helps the malware to survive for a longer duration by concealing it in a way that the anti-malwares fail to detect it i.e. match the pattern.

* What is **Obfuscation of Malware**?

It is the technique where in a given program (in this case the malware) is converted to a version hard to understand (encrypted), though keeping the functionality same. This technique was primitively used by software organizations to protect their intellectual property/ sensitive data. With burgeoning technology it has also been used widely by malware authors to prevent detection of malwares by antivirus scanners

Antimalware software’s have been designed to cease the spread of these malwares by detecting and removing completely or neutralizing them. Some of the most prevalent **anti-malware techniques** are mentioned below:

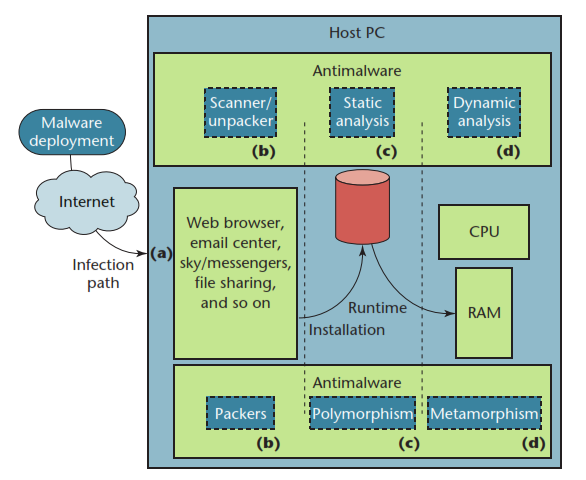
* **Signature detection:** Here the source of malwares such as files, browser plugins are checked against a database of known malware signatures. If a match is found an alert is placed by the anti-malware for further .The drawback with this method is that the entire file contents need to be checked in order to detect the signature also this method fails against 0-day-threats (i.e. attacks of unknown signatures)
* **Semantic based detection**: Signature based detection fails when code obfuscation takes place i.e. when a malicious code is inserted into the original functional code. Suppose a signature is defined by a certain sequence of instruction these instructions can be replaced by equivalent malicious instructions. Hence it is important to have a technique where such obfuscation transformations can be handled. Sematic based detection analyses the program for threatening behavior by using a code template, which makes this approach less susceptible to code obfuscation.
* **Semantic analysis**: Here files with inappropriate resource use and analyses programs for activities that use known security vulnerabilities.
* **File Emulation**: Here the malware/infected code is run on a secure virtual environment to determine if the program is trying to access privileged operation which isn’t normally associated with the program.



Some obfuscation techniques

* **Anti-Malwares V/S Malwares: Weapon’s race:**

As and when newer malwares are brought out onto the network/system anti-malwares are being parallel developed to counter them. The below description discuss what is known as the Weapon’s race between the Anti-malwares and the malwares:



**Phase1:**

The malware is injected into the system/medium.

**Phase2:**

Anti-Malware used: Signature Detection: - Here the signature detection technique was used to identify the malwares

Malware introduce: As a counter attack the malware was “packed” i.e. compressed

**Phase3:**

Anti-Malware Technique: Statistic Analysis: - It is a non-signature based technique. This works with the machine level code & disassembles instructions, hence helping in identifying the hidden malware code before run-time. As it is static scanning the overhead of scanning is reduced as it could be schedule offline during idle time.

Malware introduced: Polymorphic malwares are introduced which avoid from being detected by generating new signatures each time the malware is executed.

**Phase4:**

Anti-Malware Technique: Dynamic Analysis: - Here the behavior of the program is monitored at runtime with the help of an API sequence.

Malware introduced: Metaphoric Malwares: The malware changed its opcode each time during execution which resulted in new signatures being produced each time. This made its detection by the dynamic analysis.

* What are the **various obfuscation techniques?**

3 primary obfuscation techniques are discussed in this journal:

1. **Packers:** Here the malware is packed into a compressed set of executable used to run software. An example that has been discussed in the journal issue is the Polypack which supported 10 packers & 10 anti-Malware engines.

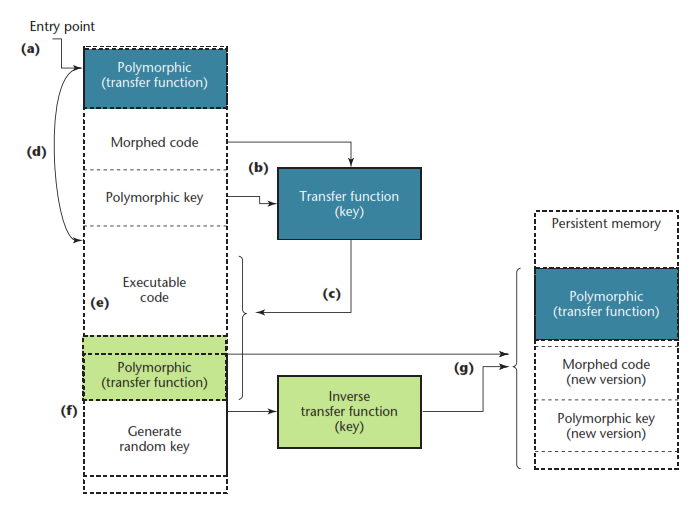
Drawbacks of packing: packers could be identified 2 ways:

* Signature Analysis

For e.g. PEiD

* Entropy Analysis: As the file is packed & encrypted the entropy proportionally increases. Based on an entropy analysis a high degree of confidence would help in detecting the packed files. For e.g. PHAD,PE-Probe etc.

2. **Polymorphism:** Here the binary executable is itself changed in order to avoid detection. With countless number of signatures produced with each run of the malware by techniques such as dead-code insertion, register reassignment etc.. it becomes easy for the signature analysis to fail as the signature sequences created wouldn’t match the ones present in the database.



Polymorphic engine

A polymorphic engine can be implemented as follows:

1. Gain entry into the host
2. The transfer function is executed which transfers the mutated malware into the host using the polymorphic key
3. The executable is loaded into the memory
4. The malware containing code is run
5. The notorious code segment is executed
6. A new key is generated in order to generate new version of mutated code
7. Inverse transfer function is executed that transforms opcode back to the mutated code.

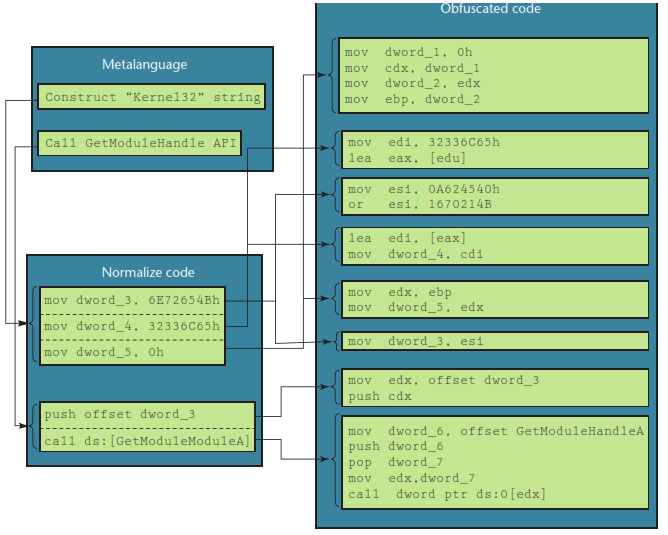
Drawbacks: Can be detected in 2 ways:

* As the opcode/body of the code remains constant for each instance, detection through emulation (such as “sandbox”) can be done while loading the malware into the memory at runtime.
* Neural pattern recognition

3. **Metamorphism:** This technique makes the best use of code obfuscation. The malware changes the opcode present in the memory each time it runs i.e. evolving its body into new generation (which functionally remains the same) and writes back the new version of the malware to the host. As the native opcode sequence doesn’t remain constant detecting by signature scanning/emulation would be an arduous task.

There can be 2 kinds of metamorphic malware:

* Open- world Malware: It relies on external channels of communication such as the websites etc. to mutate. For example : Conficker
* Close-world Malware: It doesn’t need an external media to mutate in fact the executable mutates the binary code itself. For example : Win32/Apparition



Metamorphic code

The process:

1. Decoding opcodes by the disassembler

2. Shrinker compresses the code to prevent the continuous growth of the program.

3. Permutator reorders instructions by placing them in a random order

4. Expander reorders single opcode instructions into several instructions and inserts malicious opcode such as nop opcodes etc..

5. Assembler reorders the above & calculates jmp & call addresses

Drawbacks:

1. They are generally large in size especially when compared to polymorphic engines
2. As this engine produces more obfuscated opcode which in turn increases the CPU’s workload. This could in turn make it easier to detect by the Anti-Malware software by constant monitoring of CPU idle time& comparing it against a benchmark.

* **Future Trends?**

As and when the software, hardware technologies grow corresponding malware would also develop & flourish. Some of the popular malwares in the current times are:

* JS\_VIRTOOL which is a web malware which uses java script as a tool for code obfuscation.
* Rickrolling a smartphone malware which uses the user’s personal to inject itself into the host machine.
* **Conclusion:**

This report starts with discussing what a malware is and why & what obfuscation is. A race between anti-malwares & the malwares is also projected. Three major techniques of obfuscation namely packers, polymorphism, metamorphism along with each of their drawbacks is discussed .The report also presents a brief section regarding the recent trends in malwares. As a conclusion it could be said that obfuscation as technique that effectively hides the signs of a malware, hence it is important for the antimalware makers to focus on what the malware is doing rather than how it’s doing it.

References:

* IEEE: Security & Privacy journal Issue 5, Sept-Oct 2011

<http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=8013>

* IEE paper: Malware Obfuscation Techniques; A brief survey

<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5633410>

* IEEE paper: Limits of static analysis for malware detection

<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4413008&tag>=

* AMC paper: Semantics-aware malware detection

<http://dl.acm.org/citation.cfm?id=1190216.1190270>

* Blogs: Cisco/Security: A Brief history of malware obfuscation
* Metamorphic virus: Analysis & Detection
* <http://media.techtarget.com/searchSecurityUK/downloads/RH5_Evgenios.pdf>