Malware Obfuscation through Evolutionary Packers

Marco Gaudesi Andrea Marcelli Ernesto Sanchez Giovanni Squillero Alberto Tonda

Goal

Develop a new obfuscation mechanism based on evolutionary algorithms.

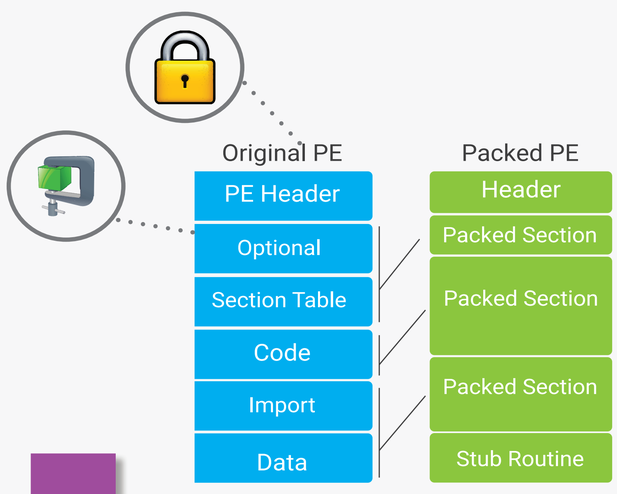
It can be used by security industries to stress the analysis methodologies and to test the ability to react to malware mutations.

Malware

/ malicious software /

- hides as long as possible
- communicates
- executes the payload
- propagates

Packer



A packer compresses or encrypts the instructions and data of a program generating a new executable version. At run time, the new executable decompress the original program in memory, and then jump into it.

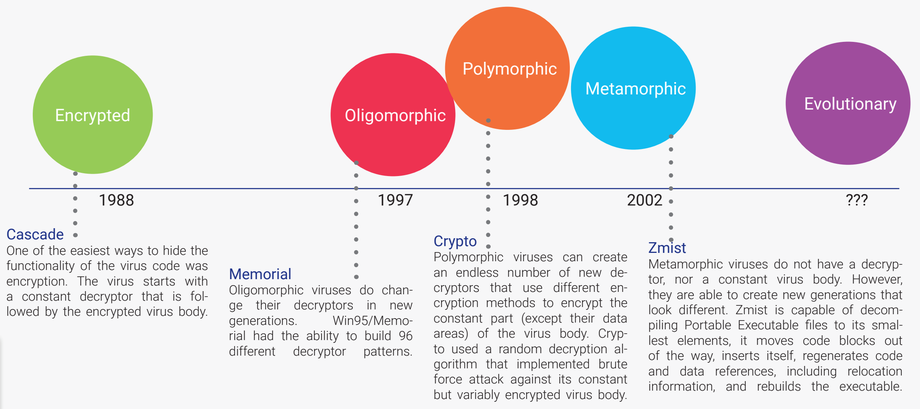
Packers have been originally designed to save disk space.
Then they have been introduced in the word of malicious software: the code must be decrypted before static analysis can be applied.
Moreover changing the encryption key produces a completely different executable.

The unpacking stub:

1) It decompresses and decrypts the original code.

2) It resolves the imports of the executable: if the import table is packed, the loader cannot resolve the imports and load the corresponding DLLs.

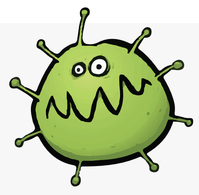
3) It transfers back the control to the Original Entry Point (OEP).



The idea of genetic selection for behaviours was first seen in 2002.
W32/Smile

Polymorphism using genetic algorithms was first seen in 2005.
W32/Zellome

Generating the code



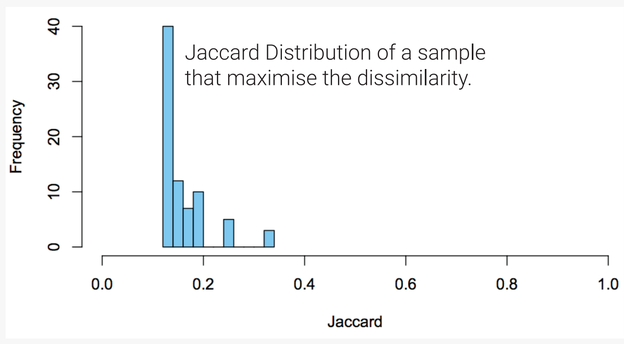
1. Generate an opcode sequence.
   Randomly-generated, variable-length sequence of x86 assembler instructions.
2. Test the sequence. Is it reversible?
   Encoding and decoding routines are applied subsequently to sequence of bytes.
3. Fitness evalutation with the Jaccard Index.
4. Code encryption.
   Creation of a new packer variant.
5. Reproduction.
   The decoding routine is embedded in the new executable. At run time it will restore the original program in memory.

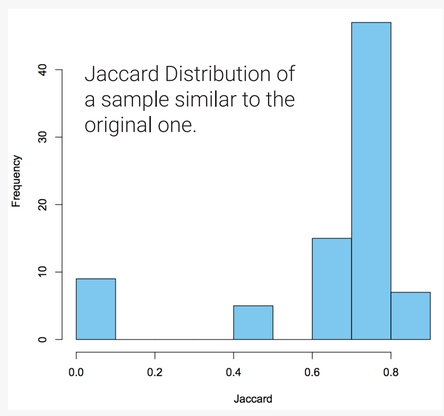
The malware uses an evolutionary algorithm to generate completely new obfuscating algorithms.
The individuals are a set of working packers and the ‘fitness’ is how similar the new executable is to the original one.

Jaccard Index

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

It is used to evaluate the similarity between a Malware sample and the original one.





Experimental Evaluation

Tcp bind shellcode from Metasploit.
Well-known AV signature.
328 byte length

```
\xfc\xe8\x82\x00\x00\x00
\x60\x89\xe5\x31\xc0\x64
\x8b\x50\x30\x8b\x52\x0c
\x8b\x52\x14\x8b\x72\x28
\x0f\xb7\x4a\x26\x31\xff
\xac\x3c\x61\x7c\x02\x2c
\x20\xc1\x0 \x00\x01\xc7
\xe2\xf2\x52\x57\x8b\x52
\x10\x8b\x4a\x3c\x8b\x4c...
```

High initial detection rate
+
Executable behavior susceptible to heuristic evaluation

virustotal — 57 AV engines

OPSWAT. Metascan® — 44 AV engines

\+

Further evaluation with locally installed AVs.

	Non encoded	Evo1	Evo2	Evo3
Virus Total	35/57	2/57	2/57	1/57
OPSWAT Metascan	25/44	4/44	3/44	1/44

Unencoded version of the executable.

Evo 1 uses a quite simple encrypting technique.

Evo 2 implements a sophisticated encoding mechanism with shuffled instructions.

Evo 3 makes use of several operations that aim to confuse heuristic engines.

Evolutionary botnet as whole prey-predator ecosystem.

Future Development

anti-debugging

Evolution of the anti-debugging techniques that are used in an attempt to slow down the analysis as much as possible.

hiding mechanism

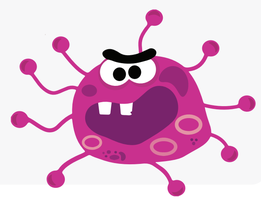
Further evolution and mutation of the executable structure, trying to increase the complexity of the analysis.

anti-disassembly

Evolution of anti-disassembly techniques that use specially crafted code or data to cause disassembly analysis tool to produce an incorrect program listing.

C&C communication

It is in charge of the mutation of redundant Command & Control channels through the usage of *variable port number*, *improper usage of existing protocols*, *randomized scanning* and *encrypted traffic*.



INRIA SCIENCE & IMPACT

POLITECNICO DI TORINO 1859 1906

GECCO