POLYMORPHIC VIRUS DETECTION USING GENETIC ALGORITHM

ROSHAN JAISWAL(ABV-IIITM Gwalior)

2023

ABSTRACT

Computer viruses have become an ever rising plague that constantly poses serious threats. Every virus is represented by a unique sequence of hexadecimal characters that current detection mechanisms look for in order to detect malware, however if this pattern is modified in the slightest, the aforementioned process is unsuccessful.In this paper, we propose a new type of virus signature definition, called evolutive signature, and a novel detection mechanism to deal with polymorphic viruses using Heuristic String Pattern matching algorithms and by inducing artificial evolution through Genetic Algorithms (GA) on existing virus definitions.

INTRODUCTION

Computer Malware is a program designed to infect, damage computer systems and replicate itself by using a machine resources without the owner's permissionPolymorphic malware maintains the same functionality and payload as their original version, while having apparently different structures. This polymorphic change (from A1 to A2) of a virus is referred to as a variant of the malware.Current antivirus software use primarily static signatures from a database to match against suspicious files and therefore detecting viruses. Each signature is said to be static given it represents a pattern and/or structure of program code which is present in the virus. For each malware variant, a respective static virus signature is needed for successful detection. Polymorphism invalidates such signatures. As a result, one of the aims of the proposed system has been to design a malware detection technique which is resilient to structure obfuscation through polymorphism by developing a Heuristic framework that evaluates using 3 different String Matching algorithms for pattern similarities between the suspect file and the known signatures.

LITERATURE REVIEW

Compared to the research work and quantity in static differencing, and not only until recent years , fewer works have been carried out in dynamic differencing. Through the use of GA in conjunction with Dependency Graph and Heuristics, Kim and Moon [16], successfully detect highly polymorphic Script viruses with no false positives shown in experiments. While such state-of-the-art technique has proven successful for source and byte-code malware, it requires each file tested to go through lengthy complex calculations that notoriously slow down the detection process, and therefore limiting the proposed system. Even though several techniques have been previously proposed for the detection of polymorphic malware, no research was found on the efficiency and manipulation of simple static virus signatures thought GA's, as well as on the generation of new signature definitions though evolutionary algorithms for polymorphic malware detection.

Furthermore, and quite the contrary to the objective of our proposed system, another research paper was found on the Evolution of Malware using GA [18]. This paper dealt with the idea of using GA's in order to evolve malware, rather than the malware detector. This scheme works on the same bases as ours, where each malware has a "genotype" representation based on its low-level code, however the authors studied the automatic creation/evolution of new malicious malware variants based on natural evolution. From the gathered results of their experiments they clearly show that malware can be successfully evolved using nothing more than a genetic algorithm with fine-tuned selection and crossover operators. This paper shined a bright light onto our proposed system, as it has already been demonstrated the ability to create new malware using GA's, therefore the opposite, which we aimed to accomplish, had a high probability of also being achievable. Additionally we learned that the selection and crossover operators play a fundamental role in the success of the mutation, and for those reasons we allow these values to be modified from the user interface of our program.

METHODOLOGY

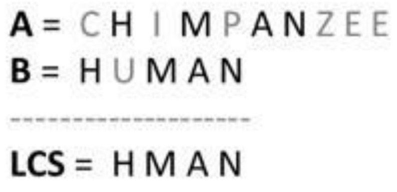
As explained earlier, a crucial component of the proposed system is the heuristic evaluation of token similarities between the suspicious file and a given virus definition. After our initial implementation, where we only used the Longest Common Subsequence algorithm, we learned that the accuracy of this algorithm alone will not suffice. We then proceeded to study and implement related string search algorithms with the aim that these will complement each other producing higher fidelity token matches and therefore increasing the probability of successfully generating a high fitness evolutionary signature.

2 algorithms used are

Longest Common Subsequence

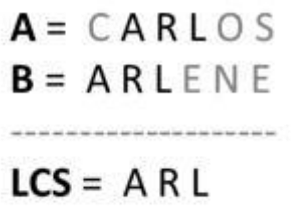
This algorithm is used to find the longest subsequence common to all sequences in a set of strings.

In other words, this algorithm finds a sequence of characters S that is present in the same continuous order in any given two strings A and B. For example :



Longest Common Substring

longest string S that is a sub-string of two or more strings. For example; given two strings A and B, of length X and Y respectively, find the longest string which is a substring of both A and B.

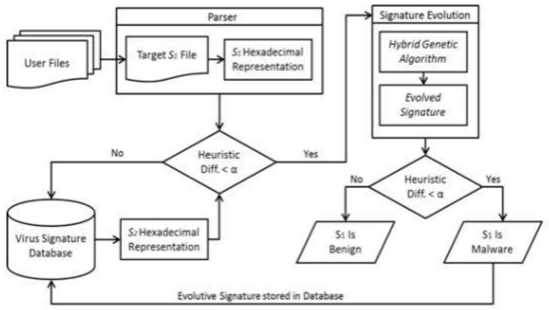


GENERAL WORKING :

1. The system parses and converts the target file S1 to be examined to a hexadecimal representation; this string is then broken down into strings of 10 hex characters. At this point S1 has not yet been classified as either benign or infected.
2. The system then proceeds to select a known static malicious signature S2 from the database.
3. Once S2 has been selected, it is broken into groups. The system then compares recurrently each hex group of S1 against each group of S2. In the process, S1 groups are given a grade individually according to their statistical similarity with S2. If the process is unsuccessful, the system returns to step 2.
4. If a match with statistical significance is found between S1 and S2 , then S2 will automatically becomes part of the “mark scheme” in the Genetic Algorithm, to which chromosomes will be evaluated against.
5. Using a Hybrid Genetic Algorithm, the system will draw the final decision, measuring the evolved signature Se against all of the elements of the GA mark scheme (using Substring and/or Subsequence).

6. If Se is re-evaluated using step 3, if a match is returned, then it is added to the known malicious signature database for future use.

A graphical representation of the proposed system



Heuristic Evaluators

The string matching framework is comprised by the 2 earlier mentioned algorithms; The Longest Common Subsequence, Longest Common Substring. From all of the above, the Longest Common Subsequence carries most of the weight when grading a token. The logic behind it is that as in polymorphic signatures the string is consistently modified and mostly only by a few values, the Subsequence problem return higher matches than the Longest Substring.Substrings are consecutives parts of a string, while subsequence's need not to be. Therefore, a Substring of a signature is always a subsequence of the signature, but a subsequence of a signature is not necessarily a substring of that same signature.

On that knowledge, the proposed system always checks first for the Longest Subsequence, when its length is >= 70% of the original string, we proceed to evaluate it again only this time we check for longest Substring and the edit distance, otherwise if less than 70% is discarded. In the situation the Longest Subsequence equals the length of the signature token, this token is no longer evaluated and is immediately stored on the system, otherwise it proceeds to be evaluated.

The percentage that each of the algorithms contributes to the final grade of the token on base 100 is the following;

Subsequence = 70% (100% if exact match)

Substring = 30%

Finally, another reason why the grading framework was developed in such way is that this process; of first checking for Subsequence's and then for the other algorithm, allows us to stop examining a token that will score poorly, resulting in a reduction of execution time.

Genetic Operators

Given the substantial number of possible variations of a virus signature, a Genetic Algorithm is appropriate method to cover in a relatively short period of time the search space of this problem. A GA will generate a set of initial solutions and evolves them in parallel over a set number of iterations. Said number of iterations can be shorten once the best solution is found (through a fitness/objective function), then the algorithm terminates. The proposed GA is explained in further detail bellow;

<u>Chromosome Representation</u>

As in all GA's, linear representation is required for the chromosomes. In the proposed system, the hexadecimal signature itself will represent the chromosomes. Each gene set in the chromosome is equivalent to the hexadecimal representation of a machine code statement.

<u>Fitness Function</u>

In order to evaluate how well each chromosome solves the problem at hand, all chromosomes are graded against an objective function that represents the problem itself. In this case, the longest common subsequence's and substrings combined when evaluated against the GA's mark scheme signatures. The higher the fitness value, the lower difference between Se and all the elements of the mark scheme.

Initialization

All chromosomes are created at random, all with equal length. This length is set on the recommended value of 10 however it can be modified on the user interface.

Selection

In order to determine offspring reproduction, this operator allocates all chromosomes in a "roulette wheel" with slots sized accordingly to their fitness value.

Crossover

This operator will randomly choose one or multiple locus (depending whether two-point crossover is selected) and exchange the content after (or between) the locus among the two chromosomes.

Mutation

According to mutation probabilities should remain significantly low, between Pm = 0.01 and Pm = 0.001. This will randomly select a locus in a chromosome and swap the character in that position for another randomly generated hexadecimal value. This parameter can also be fine-tuned on the user interface, and additionally multiple mutations per chromosomes can be selected for use.

Replacement

During each generation, we will replace the old individuals by performing roulette wheel selection, crossover and mutation to the offspring until the stated population size has been met.

Dataset

In order to try the correct functioning of the program we gathered 14.0 static virus signatures in the form of a .csv file as our virus knowledge base. Additionally we manually created polymorphic virus strings (based on those acquired signatures) which we used to embed on files using Hexadecimal Editors.

GRADING FRAMEWORK

Once we have organized and tokenized all the necessary data needed, the process to find the similarity between tokens begins. As extensively explained; this process first proceeds to evaluate each of the tokens in the suspect file/string against all of the tokens of each virus signature. The first algorithm to take place is The Longest Common Subsequence. Depending on the token match length, this will be either discarded (less than 70% the original token length) or it will proceed for further examination by means of the Longest Common Substring.

Once the genetic algorithm has finished and the best evolved chromosome identified, upon user request, the signature is re-evaluated using the Heuristic Grading Framework to corroborate that the evolution process has returned an equally high fidelity match for both the suspect file and the initially signature matches.

The outcome of this evaluation can be either; unsuccessful, low fidelity or high fidelity.

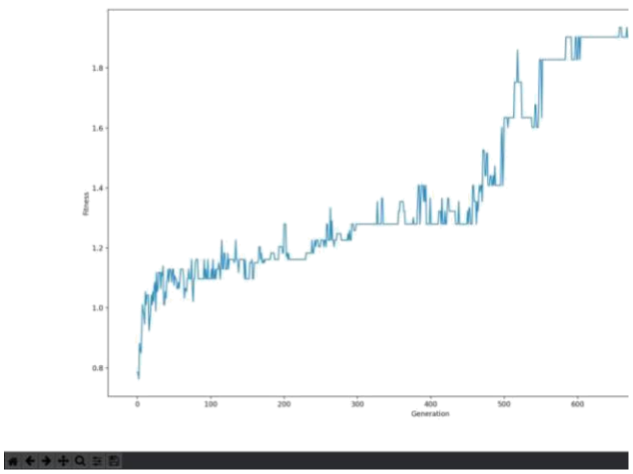
"No matches found", therefore end of program

"Low fidelity matches found" (< 65% and >40% avg. grade)

"High fidelity matches found"(>65%)

RESULTS

A number of trials were conducted with the aim to verify the accuracy and efficiency of the system to solve the problem in hand. we provide the results of genetic algorithm and of the system as a whole.



CONCLUSION

This is quite a successful achievement as it shows the potential as well as a respectable accuracy level of the combination of various string pattern matching algorithms. With the research performed to date, no strong grounds where found on the actual use of these algorithms, even less, the combination of the three to analyse and detect potential polymorphic malware. If these techniques were to be fine-tuned and implemented then both the accuracy and performance of this heuristic process could be improved.

REFERENCES

1. A New approach for Malware Detection Based on Evolutionary Algorithm. Farnoush Manavi. Ali Hamzeh
2. "Malware Detection using Evolutionary Algorithms: A Survey": https://www.researchgate.net/publication/316319824_Malware_Detection_Using_Evolutionary_Algorithms_A_Survey
3. "A Hybrid Approach for Malware Detection using Genetic Algorithm and Artificial Neural Networks": https://ieeexplore.ieee.org/document/8400658
4. S. Pearce. Viral polymorphism. Sans Institute
5. Malware Detection based on Dependency Graph using Hybrid Genetic Algorithm.Keehyung Kim.Byung-Ro Moon
6. M. Z. Shafiq, S. M. Tabish, and M. Farooq. On the appropriateness of evolutionary rule learning algorithms for malware detection. In GECCO '09: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation.