Malware Detection via Call Graphs Comparison

Gergely Erdelyi, Joris Kinable, Alexey Kirichenko, **Orestis Kostakis**, **Stefan Lundström**, Hamed Mahmoudi, Markus Miettinen, Kimmo Mustonen, Francois Nicola, Pekka Orponen.

Combinatorial Algorithms and Computation Group
Department of Information and Computer Science, Aalto University.

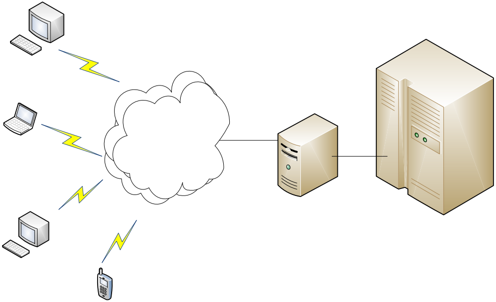
F-Secure Corporation.

Nokia Research Center.

The project is a part of WP6, Future Internet program of ICT SHOK.

February 15, 2011

Every day, anti-malware companies receive tens of thousands of executable files, sent by clients, partners, other security companies...



There are many interesting and important questions, of varying complexity, related to the “stream of samples”:

- Is a given sample malicious? If so, does it belong to a known malware family? (For instance, can it be disinfected by an existing disinfection method?)
- Is a given sample a version of a known benign application?
- Are a number of samples so similar that analyzing one (or a few) of those we can classify them all as malicious or benign?
- Can the sample classification and clustering tasks be automated?

Unfortunately, looking at the samples as binary files, it is hard to answer these questions. Our primary interest here is PE-format files, and

- small changes in the code or compiler/linker options may lead to significant changes in the resulting executable files
- many samples, including benign ones, are heavily obfuscated

One natural approach is to study structural properties of samples, specifically in the form of call graphs.

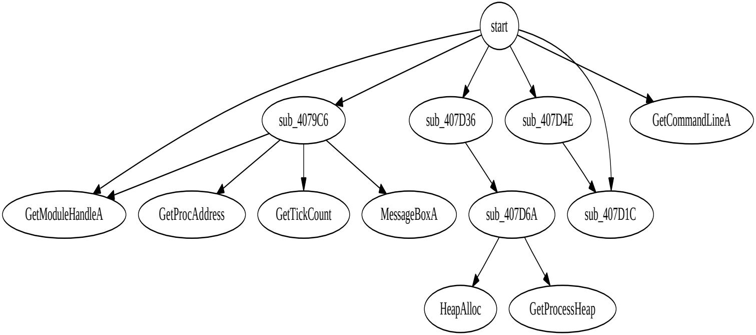


Figure: Example of a small callgraph; Bifrose variant

Many challenges are on the way:

A. extracting the call graphs.
B. defining “similarity” measure on the set of call graphs.
C. efficient computing of “distances” between graphs.
D. developing clustering and classification algorithms based on distances between call graphs and related performance problems.

(A) Samples are unpacked with F-Secure's “unpacker”, fed through IDA Pro to disassemble, and exported in Binary Export Annotation Format. (Numerous practical challenges!)

(B) For the similarity measure:

- Originally, heuristic measures (as in "Graph-based comparison of Executable Objects" by Thomas Dullien),
- We used the Graph Edit Distance (GED) measure, as in "Large-scale malware indexing using function-call graphs" by X. Hu, T. Chiueh, and K.G. Shin

(C) Efficient computing of ”distances” between graphs:

- computing GED is, predictably, an NP-hard problem.
- so we have to use approximation algorithms.
- In ”Large-scale malware indexing using function-call graphs”, Bipartite Matching is used.
- We use Simulated Annealing (SA), a local search method, and found it faster and more accurate

Simulated Annealing

- Local Search (hill climbing) algorithm.
- Basic Notion: Check random neighboring solution. If “better”, transition to it. Else, transition with certain probability.

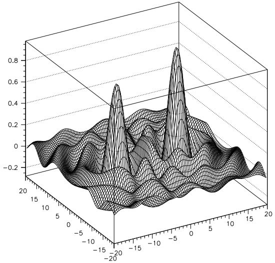


Figure: Example of a search space.

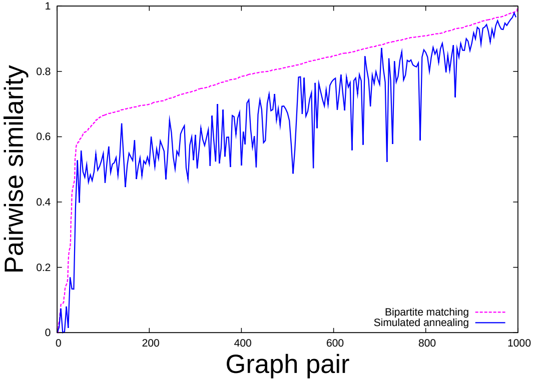


Figure: GED scores for 1000 random pairs of call-graphs. Comparison of methods; less is better.

(D) On clustering & classification:

- on small testing sets of call graphs, we used k-medoids and DBSCAN clustering algorithms, and the initial results were promising.
- Running more massive experiments with those is a part of the future work.
- In the real operating at the moment, we use an "iterative" clustering method, with a number of heuristic choices.

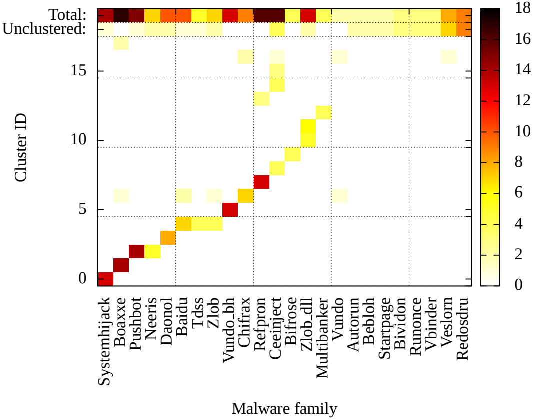


Figure: DBSCAN: Minpts = 3, Rad = 0.3. The colors depict the frequency of occurrence of a malware sample from a certain family in a cluster.

Now it is time for a demo by Stefan!!

Conclusions:

- GED appears a good call graph similarity measure; Simulated Annealing a good way of (approximately) computing it.
- Our algorithm finds meaningful clusters in the F-Secure's stream of samples.
- A significant step towards automating malware detection & classification.

Future work:

- study and optimize heuristic parts of the overall algorithm, especially clustering and classification.
- experiment with and possibly use for pre- and post-processing methods developed by Markus Miettinen, NRC, (SOM-based graphs pre-processing) and Kimmo Mustonen, F-Secure, (graph vertices comparison via opcode sequences)
- analyze the current ways to utilize “Classy” results.

References

- T. Dullien and R. Rolles.
  Graph-based comparison of executable objects.

- X. Hu, T. Chiueh, and K.G. Shin.
  Large-scale malware indexing using function-call graphs.
  In *Proceedings of the 16th ACM conference on Computer and Communications Security*, pages 611–620. ACM, 2009.

- J. Kinable and O. Kostakis.
  Malware Classification based on Call Graph Clustering.
  Journal in Computer Virology, 2011.

- O. Kostakis, J. Kinable, H. Mahmoudi, and K. Mustonen.
  Improved Call Graph Comparison Using Simulated Annealing.
  In *Proceedings of the ACM Symposium on Applied Computing*, 2011.