mant, can handle large string type reports as well as large amounts of reports and supports incremental analysis. Out of the output of the n-gram algorithm a behavioural profile is generated for each tree.

During data collection on non-infected systems a baseline of behaviour can be generated. A deviation from this baseline indicates unknown and possibly malicious behaviour.

Throughout our tests we monitored a baseline for a time period of about four and a half days and then tested this baseline against the trees of three distinct malicious traces. Those traces were deemed malicious by the algorithm alongside a total hit ratio of about 99.07 percent for all 7623 trees. That implies a low false positive rate of 0.93 percent or 71 falsely classified trees that slightly exceeded the distance threshold. Those rates are comparable to dynamic analysis that has been done before [2].

4. LESSONS LEARNED

In our earlier research [8] we were able to demonstrate that observing an entire system's behaviour is a promising approach for empirical malware research. The lessons learned from our research should help other researchers with implementing similar concepts.

The first important finding in our research was the enormous importance of working with real endpoint data. Getting access to event data from real workstations is certainly challenging, but essential for research building on that data. In this point our methodology stands in stark contrast to previous dynamic malware analysis approaches which require realistic client activity in a much smaller degree. Collecting this real data was more challenging than initially expected, because of technical limitations such as the impossibility of starting the data collection at an early stage during system boot. As a consequence, it is necessary to cope with a known incompleteness of the data set by observing distinctive subtrees instead of a single system process tree. This approach helps to maintain data integrity and keeps information loss to a minimum. Not all missing data can be reconstructed. Actions performed prior to the start-up of the monitoring tool cannot be accurately recovered and might lead to orphan process trees.

We discovered that only a very small subset of features from the data set is actually required for evaluating the maliciousness of activities while other features can be removed entirely from the set. This fact not only improves completeness of data but also reduces storage requirements. In our experimental setup, we were able to reduce to the total amount of collected endpoint activity data to well below 100 megabytes per day per workstation with an average system load.

5. CONCLUSIONS

The described methodology for empirical malware research works well as a proof of concept and encourages further research. It also shows that the idea of learning benign behaviour rather than malicious behaviour is not only possible but may as well become an important part of malware analysis in the future.

Based on the lessons learned from our earlier research we plan to improve the methodology in the future. Key research questions are the evaluation of automatic feature selection methods, incremental clustering of event data and the adaption of other classification approaches such as Naïve Bayes or graph matching algorithms.

Acknowledgments

The financial support by the Austrian Federal Ministry of Science, Research and Economy and the National Foundation for Research, Technology and Development is gratefully acknowledged.

Our gratitude extends to IKARUS Security Software for their invaluable support.

6. REFERENCES

[1] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan. Detection of New Malicious Code Using N-grams Signatures. In *PST*, pages 193–196, 2004.

[2] U. Bayer, E. Kirda, and C. Kruegel. Improving the efficiency of dynamic malware analysis. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1871–1878. ACM, 2010.

[3] R. S. Coutinho. D3.js drag and drop tree. `https://github.com/RodrigoSC/dndTree`.

[4] J. De Vries, H. Hoogstraaten, J. van den Berg, and S. Daskapan. Systems for Detecting Advanced Persistent Threats: A Development Roadmap Using Intelligent Data Analysis. In *Cyber Security, 2012 Intl. Conference on*, pages 54–61. IEEE, 2012.

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

[6] A. R. Grègio, D. S. Fernandes Filho, V. M. Afonso, R. D. Santos, M. Jino, and P. L. de Geus. Behavioral analysis of malicious code through network traffic and system call monitoring. In *SPIE Defense, Security, and Sensing*, pages 80590O–80590O. Intl. Society for Optics and Photonics, 2011.

[7] K. Jensen and L. M. Kristensen. *Coloured petri nets: modelling and validation of concurrent systems*. Springer, Dordrecht ; New York, 2009.

[8] S. Marschalek, R. Luh, M. Kaiser, and S. Schrittwieser. Classifying malicious system behavior using event propagation trees. ACM, iiwas2015 conference, 2015.

[9] K. Rieck, P. Trinius, C. Willems, and T. Holz. *Automatic analysis of malware behavior using machine learning*. Journal of Computer Security, 2011.

[10] A. S. Tanenbaum and H. Bos. *Modern operating systems*. Prentice Hall Press, 2014.

[11] M. Wagner, F. Fischer, R. Luh, A. Haberson, A. Rind, D. Keim, W. Aigner, R. Borgo, F. Ganovelli, and I. Viola. A Survey of Visualization Systems for Malware Analysis. In *Eurographics Conference on Visualization (EuroVis) State of The Art Reports*, pages 105–125. EuroGraphics.

[12] C. Willems, T. Holz, and F. Freiling. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security & Privacy*, (2):32–39, 2007.

[13] T. Wüchner, A. Pretschner, and M. Ochoa. DAVAST: data-centric system level activity visualization. pages 25–32. ACM Press, 2014.