October, 2013

Babble: Identifying Malware by Its Dialects

Aziz Mohaisen
Omar Alrawi
Andrew G. West
Allison Mankin

Babble: Identifying Malware by Its Dialects

Aziz Mohaisen, Omar Alrawi, Andrew G. West, and Allison Mankin
Verisign Labs, Reston, VA 20190
{*amohaisen* | *oalrawi* | *awest* | *amankin*}*@verisign.com*

I. Summary

Using runtime execution artifacts to identify whether code is malware, and to which malware family it belongs, is an established technique in the security domain. Traditionally, literature has relied on explicit features derived from network, file system, or registry interaction [4]. While effective, the collection and analysis of these fine-granularity data points makes the technique quite computationally expensive. Moreover, the signatures/heuristics this analysis produces are often easily circumvented by subsequent malware authors.

To this end, we propose *Babble*, a system that is concerned only with the *order* in which high-level system events take place. Individual events are mapped onto an alphabet and execution traces are captured via terse concatenations of those letters. Then, leveraging an analyst labeled corpus of malware, n-gram document classification techniques are applied to produce a classifier predicting malware family. This poster describes that technique and its proof-of-concept evaluation. This work concentrates only on network ordering and 3 malware families are highlighted. We show the technique achieves roughly 80% accuracy in isolation and makes non-trivial performance improvements when integrated with a baseline classifier of non-ordered features.

II. Babble: The Technique

We begin by summarizing the Babble workflow, as visualized in Fig. 1. Assume a sandboxed execution environment is capable of producing an ordered list of interesting events that take place during malware execution. Concentrating on unique events in that list, one can produce a mapping of events to an *alphabet* of characters (*i.e.,* a set of *grams*). With that, the trace can be rewritten as T, a string of characters/grams. Then, choosing parameter n, one can produce all adjacent strings of length n (*i.e.,* n-grams) in T. If the code trace is known to be part of a certain malware family, the presence or repetition of a given n-gram might be useful in classifying unlabeled malware in the future. Machine learning can be used to derive these patterns/models. From this simplified description, we now describe our practical implementation of the technique:

Deriving interesting events: To produce an ordered list of interesting events we use the AutoMal virtualized execution environment [4]. While also engineered to annotate memory, filesystem, and registry events – only *network* events are expressed in a time-ordered fashion in a `pcap` capture. From these files we identify 26 distinct network events (*e.g.*, DNS type `A` queries, outbound HTTP communication on port 80, *etc.*). We plan to alter AutoMal to enable time-ordered output of non-network events and leverage them via Babble.

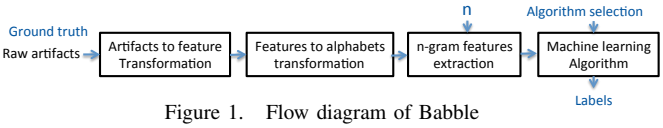

Figure 1. Flow diagram of Babble

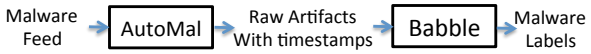

Figure 2. Flow diagram of Babble's use of AutoMal artifacts

Malware corpus: Our proof-of-concept evaluation of Babble utilizes three different malware families, chosen due to their diversity of network behavior:

- Zeus: A popular banking Trojan
- Darkness: Prevalent DDos botnet malware
- SRAT: Shady RAT [1], a targeted malware sample reported by a McAfee analyst

Malware family labeling is known to be a challenging task [2], [3]. With recent evidence of anti-virus systems producing inaccurate labels, we instead relied on manual analysts to verify family memberships (more details in [5]).

Tab. I shows the quantity of samples available per family. When we train/evaluate a family in Babble we do so in a binary fashion. For example, for "Zeus" we have 544 positive examples. We construct a balanced corpus by also adding 544 "non-Zeus" malware examples. These negative

Table I
Average number of events (characters/grams) per execution trace, by malware family

Family	Quantity	Alphabet Avg.
Zeus	1025	50.74
Darkness	544	61.47
SRAT	1130	52.74

Table II
Number of unique n-grams actually observed, by family

	1	2	3	4	5	6
Zeus	24	102	250	481	943	1690
Darkness	24	103	243	461	875	1503
SRAT	25	105	247	460	877	1536

Table III
PRECISION, RECALL, ACCURACY AND F1-SCORE FOR SELECTED N-GRAM VALUES.

	n-grams	1				4				8			
	Algorithms	P	R	A	F1	P	R	A	F1	P	R	A	F1
Zeus	KNN	80.79	79.68	81.48	79.97	79.07	83.90	82.25	81.35	78.29	78.17	79.64	78.09
	SVM	67.41	82.67	72.69	73.92	75.96	80.47	78.67	77.84	80.41	82.87	82.45	81.50
	Decision Trees	80.14	80.90	81.74	80.42	81.13	81.82	82.67	81.35	80.82	82.82	83.02	81.77
Dark.	KNN	76.22	73.13	76.08	74.56	80.40	71.52	77.70	75.57	71.38	69.58	71.65	70.20
	SVM	76.82	32.38	62.24	45.05	78.18	71.32	76.45	74.35	76.62	76.36	77.22	76.27
	Decision Trees	80.45	72.56	78.20	76.07	81.75	72.89	79.04	76.93	80.50	68.37	76.39	73.59
SRAT	KNN	81.38	76.78	82.78	78.45	83.87	81.83	85.51	81.95	83.99	74.28	82.93	78.16
	SVM	76.88	65.43	75.88	69.55	83.70	82.94	86.23	83.03	85.68	80.86	86.33	82.71
	Decision Trees	85.16	81.11	86.44	82.60	88.28	81.65	88.01	84.45	86.13	78.92	85.54	81.85

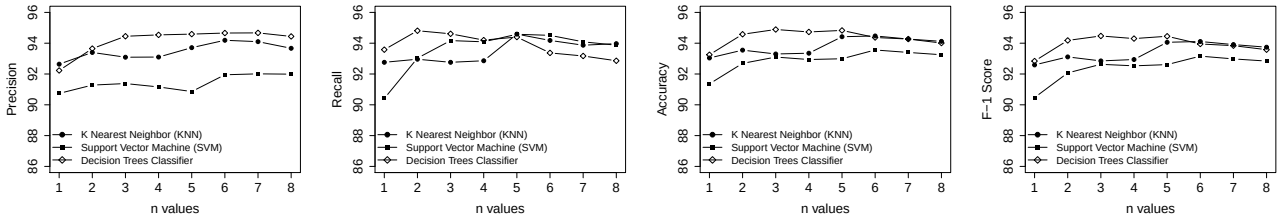

Figure 3. Precision, recall, accuracy, and F1-score for "Zeus" ordered classification atop a baseline classifier of filesystem features.

instances are not drawn from the other two test families, but a random larger set consisting of many malware families.

Computing n-grams: For a selected n it is straightforward to produce trace substrings in a sliding window fashion. Table II shows the number of unique n-grams actually observed. While there are 26^6 possible 6-$grams$, only 1690 (0.0005%) are actually observed for the Zeus malware family. A bag-of-words representation is extremely sparse and this hints at the underlying relationship/dependency between certain network actions.

Machine learning and metrics: We chose to evaluate three different machine learning algorithms, SVM, KNN, and Decision Trees. 10-fold cross validation is used to produce standard information-recall metrics such as recall, precision, accuracy, and F-score. Recall each class has a binary classifier and operates over a balanced corpus of instances. All metrics are previously studied, and the reader may refer to [5] for details.

III. RESULTS AND DISCUSSION

We now assess the performance of our preliminary Babble implementation from multiple perspectives:

Results in isolation: Tab. III shows performance for selected values (n=1,4,8) across all families and algorithms. To make sweeping generalizations across all families, Decision Tree classifiers tended to perform best with an average accuracy of ≈80%. Although the transition from $n = 1$ to $n = 4$ tended to produce noticeable performance increases, the performance ramifications of the next increase was more mixed. Between families we observe that "Darkness" malware performed most poorly overall, while "SRAT" performance fluctuated wildly based on the algorithm applied.

Atop a baseline classifier: While it is clear that Babble is able to independently predict malware family with reasonable accuracy, it is also desirable to see if it can contribute when paired with a baseline classifier. Fig. 3 shows this result, where the baseline classifier consists primarily of filesystem features. Compared to results presented in Tab. III and [4] we observe ordered features are capturing independent portions of the problem space and making non-trivial improvements to overall accuracy and other metrics.

Discussion: Leveraging ordered network features has proven itself moderately effective at predicting malware family. It is important to emphasize that network features represent a fraction of the time-dependent signals we would like to explore after better orchestrating our virtualized sandbox.

We also speculate that because we are aggregating high-level events that it may prove difficult for attackers to obfuscate around our detection scheme. Of course, we also have advantage in that we are the first to explore this classification strategy and attackers have not yet had the opportunity to do this for the malware samples in our corpora.

REFERENCES

[1] D. Alperovitch. Revealed: Operation Shady RAT, 2011. McAfee Report.

[2] M. Bailey, J. Oberheide, J. Andersen, Z. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of Internet malware. In *RAID*, 2007.

[3] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Krügel, and E. Kirda. Scalable, behavior-based malware clustering. In *NDSS*, 2009.

[4] A. Mohaisen and O. Alrawi. Unveiling Zeus: Automated classification of malware samples. In *WWW (Companion Volume)*, pages 829–832, 2013.

[5] A. Mohaisen, O. Alrawi, and M. Larson. AMAL: High-fidelity, behavior-based automated malware analysis and classification. Technical report, Verisign Labs, 2013.