IEICE TRANS. INF. & SYST., VOL.E100–D, NO.6 JUNE 2017

LETTER

Inferring Phylogenetic Network of Malware Families Based on Splits Graph

Jing LIU[†a)], ***Student Member***, **Yuan WANG**[†], **Pei Dai XIE**[†], ***and*** **Yong Jun WANG**[†], ***Nonmembers***

SUMMARY Malware phylogeny refers to inferring the evolutionary relationships among instances of a family. It plays an important role in malware forensics. Previous works mainly focused on tree-based model. However, trees cannot represent reticulate events, such as inheriting code fragments from different parents, which are common in variants generation. Therefore, phylogenetic networks as a more accurate and general model have been put forward. In this paper, we propose a novel malware phylogenetic network construction method based on splits graph, taking advantage of the one-to-one correspondence between reticulate events and netted components in splits graph. We evaluate our algorithm on three malware families and two benign families whose ground truth are known and compare with competing algorithms. Experiments demonstrate that our method achieves a higher mean accuracy of 64.8%.

key words: *malware phylogeny, splits graph, phylogenetic networks*

1. Introduction

Malware, short for malicious software, is a pervasive problem of network security. The volume of malware is growing at an exponential pace, it brings severe challenges to security vendors. However, the majority of new incoming instances are tweaked variants of previously encountered malware. They share the same functionality and exhibit characteristics of families.

Phylogeny model inference of malware aims at revealing the evolutionary relationships among families. It is not only helpful to malware classification, but also beneficial to forecast the evolution trend of families and thwart variations in advance. Given a collection of malware instances, phylogeny inference is different from generating a dendrogram based on similarity metric. It focuses on determining the temporal ordering among instances, the ancestor-descendent relationships between them. Besides, malware samples are usually captured in binary form with limited information, which makes phylogeny construction more challenging.

Previous researches of malware phylogeny inference mainly focused on tree-based model [1]–[6]. Karim et al. [1] used the UPGMA algorithm to generate phylogeny trees. Gupta et al. [6] proposed graph pruning techniques to establish phylogeny trees of malcode based on temporal informations. Seideman et al. [2] built phylogeny trees by computing the minimal spanning tree based on distance metric. However, phylogeny trees are not suitable to describe reticulate events, as Fig. 1 shows. If the ground truth is Fig. 1 (b) , instance '*e*' inherits code fragments from instance '*a*' and instance '*b*', there exists a reticulation in phylogenetic. Whereas phylogeny trees are either Fig. 1 (a) or (c), they cannot display reticulate edges in tree representation.

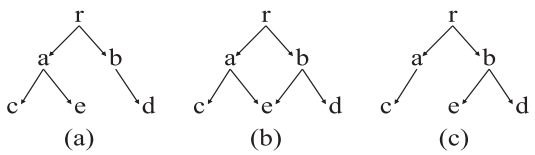

Fig. 1 An example of reticulation

Phylogenetic networks are more general to model evolutionary relationships. More recently, several researches have begun using directed acyclic graphs (DAGs) to display malware evolutions [7]–[9]. Jang et al. [7] used the minimal spanning tree to construct phylogeny DAGs of malware. And it added reticulate edges by preprocessing to find nodes with multiple parents. Andreson et al. [8], [9] employed the Bayesian network discovery algorithm to construct DAGs via statistical inference of conditional dependencies. While it demanded an informative prior about the partial ordering of instances.

In this paper, we propose a novel malware phylogenetic networks construction algorithm based on splits graph, without any prior ordering or post-processing, which refrains us from time consuming manual work and is more suitable for automatic analysis. It is based on the theorem that there exists an one-to-one correspondence between reticulations in network N and netted components of the splits graph $SG(N)$. Therefore, through generating splits graph $SG(T_1, T_2)$ from two phylogeny trees, if netted components occur in $SG(T_1, T_2)$, then we apply splits decomposition to construct phylogenetic networks with reticulations. Otherwise, any of the trees is the representation of family phylogeny. The number of trees can be easily expanded.

We conduct experiments on three malware families and two benign families whose ground truth are known and compare with MKLGC [8] and BNPrior [9]. Results demonstrate that our method performs better in both overall accuracy and precision. It achieves a mean accuracy of 64.8%.

The rest of the paper is organized as follows. Section 2 describes the proposed method. Experiments are demonstrated in Sect. 3. Conclusions are given in Sect. 4.

Manuscript received November 29, 2016.
Manuscript revised February 21, 2017.
Manuscript publicized March 22, 2017.

[†]The authors are with the College of Computer, National University of Defense Technology, Changsha, Hunan, China.
a) E-mail: liujing_nudt@nudt.edu.cn

DOI: 10.1587/transinf.2016EDL8230

2. Proposed Method

Given a collection of variants of a malware family, the phylogenetic network is built on a directed acyclic graph $G = (V, E)$. Each node v in the graph denotes an instance of malware, with one root node without incoming edges. And an edge $e = (u, w)$ denotes u is an ancestor of w and w is a descendant of u. Nodes with two or more than two incoming edges are called "reticulate node".

In this section, we describe our algorithm of constructing phylogenetic networks in detail. We first generate two phylogeny trees T_1, T_2 by a heuristic algorithm. T_1 is built from the graph edit distance of function-call graphs extracted from programs. And T_2 is built from Euclidean distance of instructions frequency. After that, we construct the splits graph $SG(T_1, T_2)$ of T_1 and T_2. According to the theorem [10], if there exists netted components in $SG(T_1, T_2)$, then we employ splits decomposition to find reticulations and construct final networks.

2.1 Phylogeny Trees Construction

In this work, we consider a heuristic algorithm to infer directed phylogeny trees. The trees are constructed from two different distance matrixes calculated between instances of a malware family. One of the metrics is the graph edit distance of function call graphs, the other is Euclidean distance of the frequency of instruction mnemonics.

2.1.1 Function Call Graph

Function call graph $G = (V, E)$ is a directed graph extracted from disassembly code of programs. It is a high-level structure feature and represents the functionality of a program. Each vertex in the graph represents a function and each edge represents a caller-callee relationship between functions. Function call graph has the advantage of covering all possible paths a program has. Moreover, it is resilient to low-level obfuscations, such as basic block reordering or register reassignment.

In our work, we use IDA Pro to acquire function call graphs of instances, which has achieved reasonable accuracy. There are two kinds of functions: local functions and external functions. Local functions are those written by program authors and external functions are statically linked or dynamically imported. We take control flow graphs as labels of local functions and function names for external functions. Then we use Hungarian algorithm [11] to calculate the graph edit distance between each pair of instances.

2.1.2 Instruction Frequency

Instruction mnemonics feature is a low-level semantic representation extracted from disassembly binary. Owning to its meaningful semantic characteristics and easy for generating, it is widely used in malware analysis. In our work, we utilize instruction mnemonics without operands. First, we use IDA Pro with python plugin to disassemble instructions of instances. Then we calculate the frequency of instructions, forming a feature vector $f_i = (f_1, f_2, \ldots, f_n)$, each f_i denotes frequency of a specific instruction. After that, we utilize Euclidean distance to measure the distance between two programs.

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \left(\frac{f_{ik} - f_{jk}}{S_k}\right)^2} \tag{1}$$

where, n denotes the number of instruction kinds comprised in a given malware family, S_k denotes the frequency variance of each instruction.

Getting the two distance matrixes, we use a heuristic algorithm to infer directed phylogeny trees. Let X denotes the instances set, we maintain an active nodes set S_a, which consists of all nodes in current tree T'. We first pick the real root node from ground truth being the first active node. At each step, we add a node x from $X - S_a$ that satisfy $\{ d_{min}(y, x) \mid y \in S_a,\ x \in (X - S_a) \}$ to T', and add a directed edge (y, x). Until $S_a = X$, the process stops.

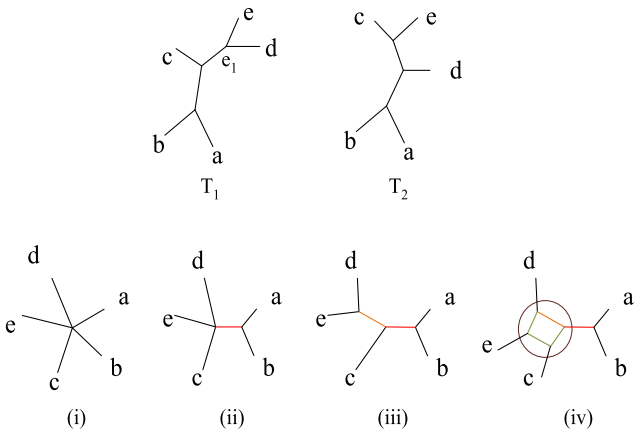

Fig. 2 Splits graph construction

2.2 Splits Graph Construction

For a given set X, a split $S = \frac{A_1}{A_2}$ is a partition of X into two non-empty and complementary sets and $A_1 = X - A_2$. A split is trivial if A_1 or A_2 has only one element. For a tree T, each edge of T defines a split. For example in Fig. 2, the split of edge e_1 in T_1 is $S_{T_1}(e_1) = \frac{\{a,b,c\}}{\{e,d\}}$. The splits set of T_1 is $\Sigma(T_1)$, contains five trivial splits: $\{\frac{\{a\}}{\{b,c,d,e\}}, \frac{\{b\}}{\{a,c,d,e\}}, \frac{\{c\}}{\{a,b,d,e\}}, \frac{\{d\}}{\{a,b,c,e\}}, \frac{\{e\}}{\{a,b,c,d\}}\}$, and two non-trivial splits: $\{\frac{\{a,b\}}{\{c,d,e\}}, \frac{\{d,e\}}{\{a,b,c\}}\}$.

Splits graph $SG(\sum)$ is representation of a set of splits. For the two phylogeny trees T_1 and T_2, let $\Sigma(T_1)$ and $\Sigma(T_2)$ denote the splits of trees, X denotes the collection of instances, Σ denotes the union of $\Sigma(T_1)$ and $\Sigma(T_2)$, S_t denotes trivial splits and S_n denotes non-trivial splits of Σ. Then $SG(T_1, T_2)$ is constructed as follows:

1. Separate Σ into S_t and S_n;
2. For each split $S_i = \frac{x_i}{X - x_i}$ in S_t, add a node x_i and an

edge e_i pointed to x_i in the graph, and create a star graph, as shown in Fig. 2 (i);

3. For each split $S_j = \frac{\{x_p,\ldots,x_q\}}{X-\{x_p,\ldots,x_q\}}$ belongs to S_n, find the shortest path $P = (x_p, e_p, \ldots, e_q, x_q)$ from x_p to x_q in graph. For each node x_j along the path except for x_p and x_q, add a new corresponding node x'_j and a new edge $\{x_j, x'_j\}$. For each edge e_j along the path except for e_p and e_q, add a new corresponding edge e'_j. Figure 2 (ii) shows the graph of adding split $\frac{\{a,b\}}{\{c,e,d\}}$, (iii) shows the graph of adding split $\frac{\{d,e\}}{\{a,b,c\}}$, and (iv) shows the final $SG(T_1, T_2)$ after adding split $\frac{\{c,e\}}{\{a,b,d\}}$.

Two splits $S_1 = \frac{A_1}{A_2}$ and $S_2 = \frac{B_1}{B_2}$ are called compatible if one of the four intersections $A_1 \bigcap B_1$, $A_1 \bigcap B_2$, $A_2 \bigcap B_1$ or $A_2 \bigcap B_2$ is empty. Given a splits set $\sum$, compatible relationships of each pair of splits among $\sum$ are described by conflict graph (CG). Each node i in CG denotes a split S_i, two nodes i and j are connected in the condition that S_i and S_j are incompatible. Connected components of CG have one-to-one correspondence with netted componets in splits graph [10]. Therefore, we can find splits contained in each netted component by looking through the corresponding connected component in CG. For example, $\sum(T_1, T_2)$ has only one edge $(\frac{\{d,e\}}{\{a,b,c\}}, \frac{\{c,e\}}{\{a,b,d\}})$ in CG, and the corresponding netted component is shown in Fig. 2 (iv).

If there exists netted components in $SG(T_1, T_2)$, as shown in Fig. 2 (iv), then for each netted component, we divide X' into two subsets R and B, where X' denotes nodes contained in the component, R denotes the reticulate nodes and B denotes the backbone nodes, which $B = X' - R$. We consider all possible subsets R of X', from size 1 to $|X|$. For each choice of R, if the splits of B after eliminating splits of R in the netted component correspond to a tree, the process stops. Then we modify the netted component into representation with backbone tree and reticulate nodes. Finally, the phylogenetic network is constructed.

3. Experiments

To evaluate the algorithm we proposed, we conduct experiments on five families, of which true phylogenetic graphs are described in [12]. Three of the families are malware: Net-Worm.Win32.Mytob, Net-Worm.Win32.Koobface and Email-Worm.Win32.Bagle, whose evolutions are depicted by several experts. And the dataset is acquired from VX Heavens [13]. The other two families are benign programs: NetworkMiner [14] and Mineserver [15]. They are gathered on open-source repositories.

We utilize three metrics to quantify our results, the *precision*, *recall* and *F-norm*. F-norm is an overall metric to measure the error on identified edges. They are defined as follows:

$$precision = \frac{true\ edges\ in\ graph}{the\ total\ number\ of\ edges\ in\ graph}$$

$$recall = \frac{true\ edges\ in\ graph}{the\ total\ number\ of\ edges\ in\ ground\ truth}$$

Table 1 Results comparison

Family	Method	F-norm	Precision	Recall
Mytob	MPNoSG	**2.8284**	**0.5833**	**0.5499**
	BNPrior	6.2450	0.2059	0.3684
	BN	7.0000	0.0833	0.1579
	MKLGC	7.9373	0.1563	0.5263
Koobface	MPNoSG	**3.4641**	**0.6251**	0.6783
	BNPrior	4.5826	0.4516	**0.7778**
	BN	5.8310	0.1923	0.2778
	MKLGC	5.2915	0.5812	0.5000
Bagle	MPNoSG	**3.0000**	**0.7143**	0.6667
	BNPrior	4.6904	0.5263	**0.8333**
	BN	7.4162	0.1026	0.1667
	MKLGC	5.7446	0.2000	0.3333
Mineserver	MPNoSG	**2.8284**	**0.7692**	0.6667
	BNPrior	2.8284	0.6818	**0.9375**
	BN	5.3852	0.0667	0.0625
	MKLGC	4.0000	0.7222	0.8125
NetworkMiner	MPNoSG	**4.2426**	**0.55**	0.55
	BNPrior	4.3589	0.5128	**1.0000**
	BN	6.1644	0.2632	0.5000
	MKLGC	4.5826	0.4857	0.8500

$$\|A - B\|_{F-norm} = \sqrt{\sum_i \sum_j (A_{ij} - B_{ij})^2}$$

where, true edges denote correctly identified edges, A, B denote adjacency matrices of the phylogeny graph we constructed and the ground truth graph. Each element A_{ij} (or B_{ij}) is either 0 or 1, it denotes whether there exists a directed edge from instance i to instance j.

We compare our method with MKLGC proposed in [8], BN and BNPrior proposed in [9], which are all the most recent researches of malware phylogeny networks reconstructions. Table 1 shows the results. As the table in bold demonstrates, our algorithm outperforms other algorithms in F-norm, the overall accuracy of all families. Besides, it has the best precision of all families. For the lower recall than BNPrior, we analysed the reason. Edge directions in the phylogeny networks of malware families are difficult to infer. Parts of wrong identified edges are because of the wrong directions. Therefore, BNPrior with prior information about correct edge directions acquired from human experts attains better recall than ours. However, BNPrior has a number of additional wrong edges, therefore, the precision and overall accuracy of BNPrior are lower than ours.

We found that the distance metrics we used are both symmetric. While in phylogeny tree construction, one mistake may cause a series opposite direction edges because of symmetric distance. Figure 3 shows the phylogenetic graphs of NetworkMiner. Because $d(1.3, 1.4) = d(1.4, 1.3)$, so the

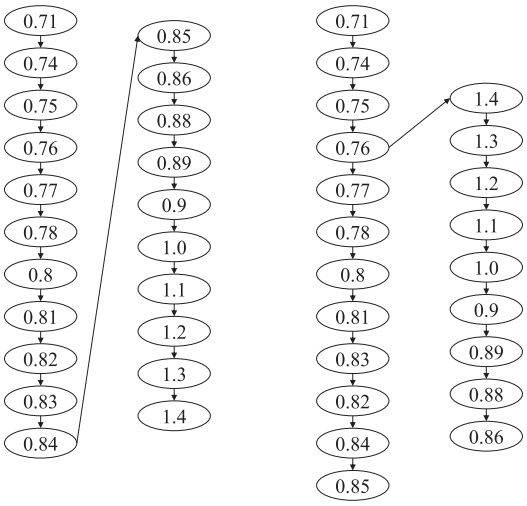

Fig. 3 The phylogenetic graphs of NetworkMiner

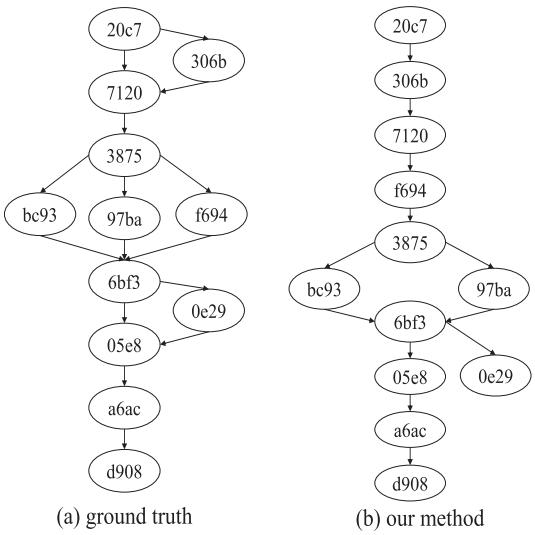

Fig. 4 The phylogenetic graphs of mineserver

wrong edge $(0.76, 1.4)$ will add edge $(1.4, 1.3)$ with opposite direction sequentially. If the distance is asymmetric, $d(1.3, 1.4) < d(1.4, 1.3)$ will avoid this mistake, that can effectively increase the recall. In next work, we will take this problem into account.

Figure 4 shows the phylogenetic graphs of Mineserver. As the figure shows, we recovered majority edges. While, reticulate edges $(20c7, 7120)$ and $(0e29, 05e8)$ are not recognized. We analysed the two trees of Mineserver, we found that the two trees have the same topology of tuple $(20c7, 306b, 7120)$ and tuple $(6bf3, 0e29, 05e8)$. There are no incompatible splits of two trees in these two tuples. Both function call graph and instruction frequency are static features, through adding dynamic feature trees may improve the performance.

4. Conclusion

In this paper, we propose a novel malware phylogenetic network reconstruction method, taking advantage of the one-to-one correspondence between reticulate events and netted components in splits graph. Our method does not require any prior knowledge about the ordering of instances, which avoids time consuming manual works and makes it more suitable for automatic analysis. Moreover, results demonstrate that our method outperforms other algorithms in both overall accuracy and precision.

Acknowledgments

This work is supported by NSFC (No.61472439, No. 61379052, No.61271252), National Natural Science Foundation of China under Grant.

References

[1] M.E. Karim, A. Walenstein, A. Lakhotia, and L. Parida, "Malware phylogeny generation using permutations of code," Journal in Computer Virology, vol.1, no.1-2, pp.13–23, 2005.
[2] J.D. Seideman, B. Khan, and A.C. Vargas, "Malware biodiversity using static analysis," International Conference on Future Network Systems and Security, pp.139–155, Springer, 2015.
[3] J.D. Seideman, B. Khan, and A.C. Vargas, "Identifying malware genera using the Jensen-Shannon distance between system call traces," 2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE), pp.1–7, IEEE, 2014.
[4] A. Pfeffer, C. Call, J. Chamberlain, L. Kellogg, J. Ouellette, T. Patten, G. Zacharias, A. Lakhotia, S. Golconda, J. Bay, R. Hall, and D. Scofield, "Malware analysis and attribution using genetic information," 2012 7th International Conference on Malicious and Unwanted Software (MALWARE), pp.39–45, IEEE, 2012.
[5] M.E. Karim, A. Walenstein, A. Lakhotia, and L. Parida, "Malware phylogeny using maximal pi-patterns," EICAR 2005 Conference: Best Paper Proceedings, pp.156–174, 2005.
[6] A. Gupta, P. Kuppili, A. Akella, and P. Barford, "An empirical study of malware evolution," 2009 First International Communication Systems and Networks and Workshops, pp.1–10, IEEE, 2009.
[7] J. Jang, M. Woo, and D. Brumley, "Towards automatic software lineage inference," Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13), pp.81–96, 2013.
[8] B. Anderson, T. Lane, and C. Hash, "Malware phylogenetics based on the multiview graphical lasso," International Symposium on Intelligent Data Analysis, pp.1–12, Springer, 2014.
[9] D. Oyen, B. Anderson, and C. Anderson-Cook, "Bayesian networks with prior knowledge for malware phylogenetics," Workshops at the Thirtieth AAAI Conference on Artificial Intelligence, 2016.
[10] G. Dan, V. Bansal, V. Bafna, and Y.S. Song, "A decomposition theory for phylogenetic networks and incompatible characters," Journal of Computational Biology, vol.14, no.10, pp.1247–1272, 2007.
[11] H.W. Kuhn, "The hungarian method for the assignment problem," Naval Research Logistics, vol.52, no.1, p.7–21, 2005.
[12] B. Anderson, "Integrating multiple data views for improved malware analysis," 2014.
[13] http://vxheaven.org/
[14] https://sourceforge.net/projects/networkminer/
[15] https://github.com/fador/mineserver