LETTER

Having an Insight into Malware Phylogeny: Building Persistent Phylogeny Tree of Families

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

SUMMARY Malware phylogeny refers to inferring evolutionary relationships between instances of families. It has gained a lot of attention over the past several years, due to its efficiency in accelerating reverse engineering of new variants within families. Previous researches mainly focused on tree-based models. However, those approaches merely demonstrate lineage of families using dendrograms or directed trees with rough evolution information. In this paper, we propose a novel malware phylogeny construction method taking advantage of persistent phylogeny tree model, whose nodes correspond to input instances and edges represent the gain or lost of functional characters. It can not only depict directed ancestor-descendant relationships between malware instances, but also show concrete function inheritance and variation between ancestor and descendant, which is significant in variants defense. We evaluate our algorithm on three malware families and one benign family whose ground truth are known, and compare with competing algorithms. Experiments demonstrate that our method achieves a higher mean accuracy of 61.4%.

key words: *malware phylogeny, persistent phylogeny tree, evolutionary relationship*

1. Introduction

Malware, short for malicious software, is a pervasive problem confronted by network security. Nowadays, new malware variants have been increasing rapidly with polymorphic and metamorphic engines. They share similar functionalities with encountered instances and exhibit characteristics of families. Therefore, malware phylogeny construction approaches have been put forward to thwart the enormous variants.

Phylogeny model inference of malware aims at reconstructing evolutionary relationships between instances within families. It helps analysts to quickly understand a new, unseen variant which is related to previously analysed samples along evolution path. Besides, malware phylogeny offers analysts a better understanding of how malware has evolved and adapted to deal with new defensive over time. It is also beneficial to forecast the evolution trend of families.

Tree-based models are often used in previous researches of malware phylogeny inference [1]–[6]. Phylogeny trees are branching diagrams that represent the relationships among instances. Karim et al. [1] used the UPGMA algorithm to generate unroot phylogeny trees. Gupta et al. [6] proposed the graph pruning technique to construct 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, those tree models are either dendrogram or directed trees with simple ancestor-descendant relations, as shown in Fig. 1 (a) and (b). They cannot depict what characteristics are descendant inherited from ancestor and what functionalities do ancestor vary from descendant, which is invaluable for understanding new variants within families.

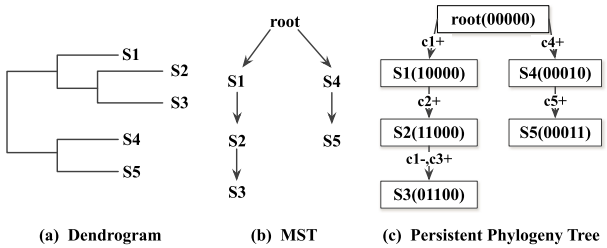


Fig. 1 Different phylogeny tree models

In this paper, we propose a novel malware phylogeny construction method taking advantage of persistent phylogeny tree model to explicitly state the functionality derivations in evolution. Figure 1 (c) exhibits a plain output of our method, in which each node corresponds to an input instance and edges represent the gain or lost of functional characters. Persistent phylogeny trees are constructed from binary character matrices under the restriction that each character can be acquired and lost only once, which is meaningful on the actual evolution history. In addition, the characters in our work are behavior patterns extracted from dynamic system call traces. After generating binary matrix for a given collection of malware instances, we employ the red-black graph algorithm to construct the persistent phylogeny tree. Meanwhile, the time complexity of our algorithm is polynomial.

We conduct experiments on three malware families and one benign family whose ground truth are known and compare with Gupta [6] and MST [2]. Results demonstrate that our method performs better in overall accuracy, precision and recall. It achieves a mean accuracy of 61.4%.

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 August 8, 2017.
Manuscript revised October 19, 2017.
Manuscript publicized January 9, 2018.

[†]The authors are with the College of Computer, National University of Defense Technology, Changsha, Hunan, China.
[††]The author is with the School of Information Science & Engineering, Lanzhou University, Lanzhou, 730107, China.
a) E-mail: wwyyjj1971@126.com

DOI: 10.1587/transinf.2017EDL8172

Table 1 Examples of behavior patterns

Behavior Pattern	System Call Sequence
Download malicious files	(getaddrinfo, socket, bind, URLDownloadToFile)
Close security software	(RegCreateKey, RegDeleteKey, RegClostKey)
Copy itself to system folder	(NtOpenFile, CopyFileA, NtSetInformationFile)

2. Proposed Method

Given a collection of malware variants within families, phylogeny inference is to construct graphs depicting evolutionary relationships between the instances. Persistent phylogeny tree is one kind of tree models. It is constructed from binary character matrices under the restriction that each character can be acquired and lost only once. Each instance is labeled by a tree node and edges represent character variations between ancestors and descendants.

In our work, we first take the behavior patterns extracted from dynamic system call sequences as functional characters. Then we generate a binary matrix to map instances and characters. After that, we employ the red-black graph approach described in [7] to construct the persistent phylogeny tree of families.

2.1 Behavior Pattern Generation

System calls are the interface of programs to interact with operating system and access system resources. They are vital in revealing functionality and behavior of programs. For example, if a program invokes function *URLDownloadToFile*, we can infer that the program may contain malicious intent to download remote files. As a consequence, system calls are widely used in malware analysis.

In order to improve the effectiveness of system calls representing the behavior of malware, we extract behavior patterns from system calls on behalf of functional characters. There exist some specific system call sequences that match specific malicious behavior patterns. For example, one system call sequence (RegCreateKey, RegDeleteKey, RegCloseKey) with parameter "SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Kaspersky" indicates the behavior of closing security software. Table 1 lists some examples of behavior patterns found in our method.

We then generate a binary matrix M, where rows correspond to malware instances and columns correspond to behavior pattern characters. And each entry M_{ij} denotes whether instance i has character j, as shown in Fig. 2.

In our work, we utilize a free online dynamic analysis sandbox service Malwr [8], to collect system calls of malware instances. On one hand, dynamic analysis has the advantage that it is resilient to low-level obfuscation techniques, such as packers or mutations. On the other hand, online analysis effectively avoids time-consuming and resource-intensive local sandbox deployment. Malwr service is based on Cuckoo sandbox. It automatically runs submitted files and returns analysis results through web pages. Those pages are dynamic loaded via Javascript. Therefore, we utilize *Selenium* Webdriver with *PhantomJS* to simulate browser operations and reload html pages automatically. Then, we use *Beautiful Soup*, a python library to parse static html files and extract system call sequences.

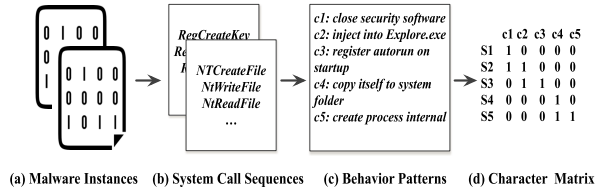


Fig. 2 Behavior pattern matrix generation

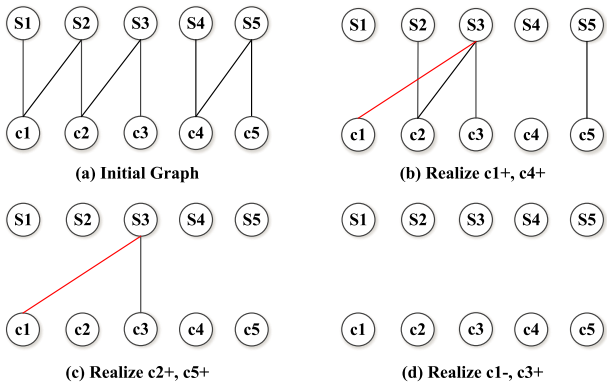


Fig. 3 Red-black graph approach

2.2 Persistent Phylogeny Construction

For a given character matrix M, with instances set $S = \{S_1, S_2, \cdots, S_n\}$ and characters set $C = \{c_1, c_2, \cdots, c_m\}$, persistent phylogeny construction aims to depict evolutionary relationships among S by a tree, under the restriction that each character can be lost and gain only once. Meanwhile, each instance is labeled by a node in the tree and edges denote character transformations during the evolution of ancestors and descendants.

The construction of a persistent phylogeny tree of a matrix M, is equivalent to a sequences of specific graph operations on the corresponding red-black graph of M [7]. A red-black graph G_{RB} associated with M is a bipartite graph, with vertex set $V = S \bigcup C$ and black edge set $E = \{(u, v) \,|\, u \in S, v \in C, M_{uv} = 1\}$. The G_{RB} of matrix in Fig. 2 (d) is shown in Fig. 3 (a).

A connected component of G_{RB} is a subgraph where any two vertexes are connected by paths. In Fig. 3 (a), it comprises two connected components and the corresponding vertex sets are: $\{c_1, c_2, c_3, S_1, S_2, S_3\}$ and $\{c_4, c_5, S_4, S_5\}$. Let $D(c)$ denote the set of instances in the connected component of G_{RB} that contains character c, and $N(c)$ denote

the set of instances that are adjacent to c in G_{RB}, that is $N(c) = \{S \mid M_{Sc} = 1\}$. For example, $D(c_1) = \{S_1, S_2, S_3\}$ and $N(c_1) = \{S_1, S_2\}$.

The realization of $c+$ denotes adding an edge (x, y) in the tree where the state of c goes from 0 of x to 1 of y. And the corresponding G_{RB} operation is that deleting all black edges incident to c and adding red edges between c and instances in $D(c)\backslash N(c)$. For example, Fig. 3 (b) is realization of c_1+ and c_4+ after Fig. 3 (a). While, the realization of $c-$ denotes adding an edge (x, y) where the state of c goes from 1 to 0 and deleting all red edges incident to c. A character c is universal or free means that it connects to all instances in $D(c)$ with black edges or red edges. For instance, c_4 is universal in Fig. 3 (a) and c_1 is free in Fig. 3 (c).

Where, a character c of G_{RB} is maximal if $N(c) \subsetneq N(c^{'})$ for any character $c^{'}$ in G_{RB}. Let C_i denote the characters that have not been realized. Then the process of constructing persistent phylogeny tree is as follows:

1. Realizing all universal characters $c+$ of G_{RB}, and updating G_{RB};
2. Realizing all free characters $c-$ of G_{RB}, and updating G_{RB};
3. For each connected component of G_{RB}, finding the maximal character c in C_i, realizing character $c+$ and updating graph G_{RB};
4. If graph G_{RB} is empty, then stop. Otherwise, go 1-3;
5. Removing tree nodes that have no corresponding instances, and merging edge characters.

The persistent phylogeny tree of matrix in Fig. 2 (d) is shown in Fig. 1 (c). We use *graphviz*, a python module to visualize the phylogeny tree.

3. Experiments

To evaluate the algorithm we proposed, we conduct experiments on four families, of which true phylogenetic graphs are described in [9]. 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 [10]. The other benign family is NetworkMiner [11], which is gathered on the open-source repository.

NetworkMiner is a network forensic analysis tool. It is used to detect OS, hostname, sessions through packet sniffing [11]. The lineage of NetworkMiner is a straight line sorted in increasing order of released versions. The ground truth of NetworkMiner is described in our previous work [12]. Figure 4 shows the persistent phylogeny tree of NetworkMiner. We correctly recovered a large proportion of ancestor-descendant relationships among the family. And each edge explicitly depicts the function character variation along evolution path.

Some instances may have the same behavior patterns, such as version 1.0, 1.1, 1.2, 1.3 and 1.4 of NetworkMiner.

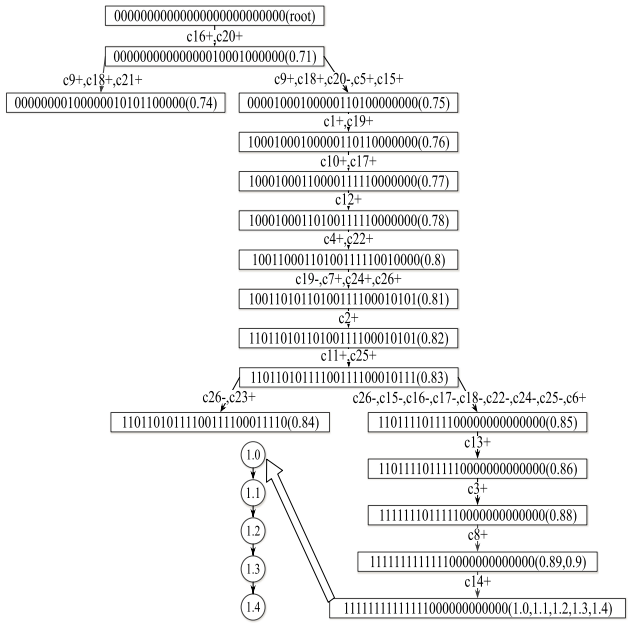


Fig. 4 Persistent phylogeny tree of networkminer

As a consequence, we construct an additional minimal spanning tree merging into persistent phylogeny tree, as shown in Fig. 4. Let S_e denote instances that share same behavior patterns, p denote the common parent of S_e. We first calculate the Euclidean distance between each pair of $\{S_e \cup p\}$. It is computed based on the frequency of n-gram system call sequences. Each instance comprises a vector $f = (f_1, f_2, \ldots, f_n)$, where f_i denotes frequency of a specific n-gram system call sequence. Then we set p as the root of MST and employ *Prim* algorithm to construct the tree. Experiments demonstrate that when $n = 3$, our method achieves best accuracy.

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:

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

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

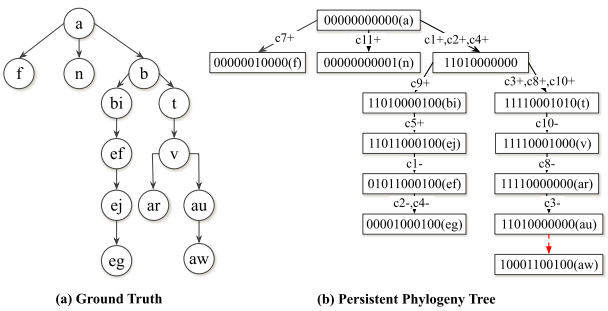
$$\|A - B\|_{F-norm} = \sqrt{\sum_i \sum_j (A_{ij} - B_{ij})^2}$$

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 Gupta [6] and MST in [2], which are representative directed tree methods in malware phylogeny reconstruction. Table 2 shows the results of the three algorithms. As the table in bold demonstrates, our algorithm outperforms the other two algorithms in F-norm,

Table 2 Results comparison

Family	Method	F-norm	Precision	Recall
Mytob	PPT	**3.3166**	**0.50**	**0.4545**
	Gupta	6.0828	0.05	0.0526
	MST	7.2801	0.0526	0.1053
Koobface	PPT	**4.0**	**0.50**	**0.50**
	Gupta	5.9161	0.3158	0.3333
	MST	7.2111	0.0278	0.0556
Bagle	PPT	**3.8729**	**0.5882**	**0.5294**
	Gupta	6.5574	0.12	0.125
	MST	8.3667	0.0208	0.0417
NetworkMiner	PPT	**2.0**	**0.8667**	**0.9**
	Gupta	5.0	0.381	0.40
	MST	5.6569	0.35	0.70



(a) Ground Truth (b) Persistent Phylogeny Tree

Fig. 5 Phylogeny trees of mytob

precision and recall of all families. Edge direction inference is the main difficulty of malware phylogeny construction. Our algorithm casts edge inference into function transformation relationship estimation. Thus effectively avoids distance based edge inference, which is error prone and local optimal. Moreover, we explicitly depict function variations over evolution on the basis of correctly recovered edges.

While, there exist some cases that character matrices have no solution for persistent phylogeny tree. Figure 5 shows the persistent phylogeny tree of family NetWorm.Win32.Mytob. Mytob is a family of IRC bots and spreads mainly through e-mail. The edge (au, aw) in the persistent phylogeny tree is red dashed because the instance *aw* has not been realized in the final red-black graph. It has been proved [7] when red edges have conflict triple, that is there exist pairs of instances and characters contain three binary pairs $(0,1)$, $(1,0)$, $(1,1)$, then the persistent phylogeny problem is unsolvable, and the final graph will not be empty. In next work, we will consider relaxing restrictions of persistent phylogeny tree, such as conflict characters can be gained or lost twice.

4. Conclusion

In this paper, we propose a novel malware phylogeny construction method based on persistent phylogeny tree model. It can not only demonstrate directed ancestor-descendant relationships between instances within families, but also depict function inheritance and variation between ancestor and descendant, which is invaluable in reverser engineering of malware variants. In our work, we take behavior patterns extracted from dynamic system call traces as input for our method. Then we employ the red-black graph algorithm to construct persistent phylogeny tree. Finally, we conduct experiments and results show that our method outperforms other algorithms in overall accuracy, precision and recall.

Acknowledgements

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, vol.523, 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," Malicious and Unwanted Software: The Americas (MALWARE), 2014 9th International Conference on, 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," Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on, 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] P. Bonizzoni, G.D. Vedova, and G. Trucco, "Solving the persistent phylogeny problem in polynomial time," 2016.
[8] https://malwr.com/.
[9] B. Anderson, "Integrating multiple data views for improved malware analysis," 2014.
[10] http://vxheaven.org/.
[11] https://sourceforge.net/projects/networkminer/.
[12] J. Liu, Y. Wang, P.D. Xie, and Y.J. Wang, "Inferring phylogenetic network of malware families based on splits graph," IEICE Trans. Inf. & Syst., vol.E100-D, no.6, pp.1368–1371, 2017.