MergeTree: a Tree Model with Merged Nodes for Threat Induction

[1]PING CHEN, [1]JINGJING HU, [1]ZHITAO WU, [2,3]RUOTING XIONG, [2,4]WEI REN

[1]China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou, CHINA
[2]School of Computer Science, China University of Geosciences, Wuhan, CHINA
[3]School of Computer Science, University of East Anglia, Norwich, UK
[4]Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan, 430078, CHINA

Abstract:- Threat tree model can clearly or- ganize threat induction information and thus is widely used for risk analysis in software assur- ance. Threat tree will grow to complicated struc- tures, e.g., the number of nodes and branches, when the threat information grows to a huge vol- ume. To extend the scalability of the threat tree model, we propose a tree model with merged n- odes so as to largely decrease the number of nodes and branches. The formal model and dedicated algorithms are proposed in details. The exper- imental results show the practicality of Merge- Tree. We also formally analyze the soundness and completeness of the proposed model.

Keywords:- Threat Tree, Semantics, Risk Analysis, Software Assurance.

Received: July 28, 2021. Revised: October 19, 2022. Accepted: November 24, 2022. Published: December 30, 2022.

1. Introduction

Threat tree model is a modeling method for threat representation and risks, which is widely used in software assurance [1]. The conditions for a threat are organized as a branch in the tree, which consists of several brothers with logical relations either "AND" or "OR" and a father who is the result of the conditions. This branch representation is iterative so that can represent complicated threat conditions like a tree [2].

The tree model presents simplicity in terms of semantics as threat information is always provided as a list of inductions, each of which is composed of certain conditions and one conclusion. Thus, tree model can smoothly represent the induction list, and afterward tree structure can be operated by available tree algorithms.

With the growing volume of induction list, however, tree structure will be complicated to due the increasing of the number of nodes [3, 4]. Simply speaking, the number of nodes is proportional to the number of inductions. Hence, how to decrease the number of nodes yet maintaining the semantics of threat tree is of great importance.

The problem seems to be straightforward, intuitively, because the number of nodes can only be decreased by merging. However, if the merge is not proper, the overhead of merging will be great. The challenge of the problem is that merging must not damage the semantics of the threat and merging should maintain the tree structure, so that legacy algorithms for tree processing can be remained.

The contribution of the paper is as follows:

1) We formally model the semantics of the threat tree with merged nodes.
2) We propose several algorithms for threat tree construction from the threat database.
3) We formally prove the equivalence in terms of semantics between induction set and threat tree.

The rest of the paper is organized as follows: Related work is reviewed in Section II.. Section III. describes the proposed model. We propose key algorithms for threat tree in Section IV.. The experimental results are shown in V.. Then analysis is given in VI.. Finally, Section VII. concludes the paper.

2. Related Work

Attack tree or their variation trees are good tools for security assessment and threat defense. Many scholars also focus on the studies for attack tree synthesis and refinement [5]. Attack trees have been applied in many fields, such as e-mail systems [6] and Cyber Physical Systems (CPS) security [7]. Other applications can be seen in [8–10]. Generally, the studies for attack trees divide into these three categories: security assessment and threat defense, tree synthesis, and large-scale nodes of attack tree management.

2.1 Attack Trees for Security Assessment and Threat Defense

The main functions of attack trees can be security assessment and threat defense. For example, Approx-Tree+ [11] analysis tool is proposed for attacker profiling

and enhanced it by incorporating the attacker's capabilities into it. Wouter et al. [12] introduced a methodology to evaluate the security of CSP system, which generates attack trees based on the system architecture automatically. The generated attack trees can provide both technical and non-technical feedback. Nishihara et al. [13] focused on refinement scenarios for attack trees which enables the assessment of the validity of attack decomposition systematically. That is, the attributes that subattacks refine an attack are described by the relationship among their effects. The proposed ideas are applied to the case study of a vehicular network system that is well-behaved.

2.2 Attack Tree Synthesis

Sophie et al. put forward ATSyRa [14], which provides a user-friendly environment for attack tree synthesis. In the ATSyRa system, users can define a structured attack tree by high-level description, and refine the synthesis interactively with the system. The relations of the nodes can be three types, namely AND, OR, and SAND. Nonetheless, the refinement analysis is still done by humans, not automatically and intelligently compared with [16]. In 2020, a Library-Based Attack Tree Synthesis scheme is proposed [17], where the inputs of the system are a library and a trace. Whereas, the proposed algorithm is only polynomial in the size of the trace. Olga et al. [15] proposed a refinement-aware method for attack tree generation by labeling technique. It solves the problem of constructing a correct attack tree while maintaining a predefined refinement relationship. Nevertheless, it has not addressed the challenge of the growing volume of nodes.

2.3 Large-scale Nodes of Attack Tree Management

With the increasing number of attacks and the complex operations between the nodes, the management of attack trees with complex structures has become a challenge. Paul et al. [18] suggested that traditional risk assessment schemes are now reaching their limit and they proposed graphical extensions to deal with scalability issues, like chain diagrams. The method has been applied in the Galileo risk management program and the results show that the system can deal with both software-intensive situations and a large number of small problems. Fila et al. [19] focused on finding the best series of Attack Defense Trees (ADTrees) from the directed acyclic graphs. Experiment results show that the countermeasures can block numerical ways of attacking and a wide class of optimization problems can be solved despite the growing volume of nodes. Vigo et al. [20] put forward a static analysis method implemented by Java to avoid the exponential explosion of analysis for attack trees, where they are automatically deduced from an algebraic specification in a syntax-directed manner. The study of a national-scale authentication system has proved the flexibility and effectiveness of the scheme.

As we can see, there is no research on the induction method to manage the attack trees with large-scale nodes. Therefore, it is of great importance to reduce the complexity of the tree model when the threat information grows to a huge volume while maintaining the semantics.

3. Proposed Model

3.1 Induction Model

To explore the atom induction, it can be classified into four types as follows:

$F1:\ A_1 \wedge A_2 \Rightarrow B$. We denote it as $A_1, A_2 \Rightarrow B$.

$F2:\ A_1 \vee A_2 \Rightarrow B$. We rephrase it as two inductions: $A_1 \Rightarrow B$; $A_2 \Rightarrow B$.

$F3:\ (A_1 \wedge A_2) \vee A_3 \Rightarrow B$. It can be denoted as two inductions: $A_1, A_2 \Rightarrow B$; $A_3 \Rightarrow B$.

$F4:\ (A_1 \vee A_2) \wedge A_3 \Rightarrow B$. It can be denoted as two inductions: $A_1, A_3 \Rightarrow B$; $A_2, A_3 \Rightarrow B$.

Definition III..1. *Simplex Form. Induction $A_1, ..., A_n \Rightarrow B$ is called simplex form where all conditions (namely, A_i, $i = 1, ..., n$) have one relation "AND" for a conclusion (namely, B).*

Proposition III..2. *All inductions can be regulated into simplex form.*

Proof. Given any inductions, a method can change it into simplex form as follows:

(1) Split all $\vee$ at the outlier layer by $F2$ and $F3$, and then obtain an induction set.

(2) For any induction in the set, it must be in the form either $F1$ or $F4$. Both can be changed into simplex form.

(3) If any induction is not simplex form, go to (1).

Later, we will propose a tuple model for representing a simplex form, and further a tree model for representing all simplex form.

3.2 Tree Model

Definition III..3. *SimplexInduction* $::=< \{condition \in Label\}, conclusion \in Label >$, *where condition are one label or multiple labels; conclusion is one label; Label is a set of strings in context, e.g.,* $A_1, ..., A_n, B$.

For example, simplex induction for $A_1 \Rightarrow B$ is $< \{A_1\}, B >$, induction for $A_1, A_2 \Rightarrow B$ is $< \{A_1, A_2\}, B >$.

As an assumption, we suppose $\forall$ $< condition, conclusion >\in SimplexInduction$, $condition \times conclusion$ is a one-to-one mapping and onto.

Definition III..4. $NODE ::=< \{label \in Label\}, to \in Label >$, *where label denotes one condition or multiple conditions; to denotes a conclusion.*

For example, the node for $A_1 \Rightarrow B$ is $< \{A_1\}, B >$, induction for $A_1, A_2 \Rightarrow B$ is $< \{A_1, A_2\}, B >$. For visualization, *label* can be looked as a node, and *to* can be looked as an edge that starts from the node and with a label *to* at the end.

Proposition III..5. *SimplexInduction is equivalent to Node.*

Proof. Straightforward. The definition is identical for the type of the components, although the names of the components are distinct.

Definition III..6. $EDGE ::=< from \in NODE, to \in NODE >$.

Proposition III..7. *If* $\exists node_a, node_b \in NODE$, $node_a.to \in node_b.label$, *then* $\exists edge \in EDGE$, $edge.from = node_a$ *and* $edge.to = node_b$.

Proof. Straightforward.

Definition III..8. $ROOT ::= \{node | node \in NODE, \nexists m \in EDGE, m.from = node\}$.

Definition III..9. $LEAF ::= \{node | node \in NODE, \nexists m \in EDGE, m.to = node\}$.

Definition III..10. $Tree ::=< \{node \in NODE\}, \{edge \in EDGE\} >$.

For example, suppose an induction set consisting of simplex forms is as follows:

1) $A_1 \wedge A_2 \wedge A_3 \Rightarrow B_1$;

2) $A_2 \wedge A_3 \Rightarrow B_2$;

3) $A_3 \wedge A_4 \Rightarrow B_4$;

4) $B_1 \wedge B_2 \wedge B_3 \Rightarrow Root$;

5) $B_3 \wedge B_4 \wedge B_5 \Rightarrow Root$.

Above induction set can be converted into a threat tree with merged nodes in Fig. 1. Note that, at the end of the edge there exists a label to denote the conclusion. Besides, if the union of edge labels is not equal to the label of the "father" node, that "father" will not be reached indeed. In that words, the threat path to the root is broken at this "father".

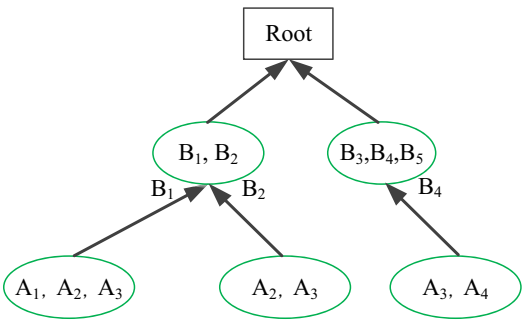

Fig. 1: A threat tree with merged nodes.

4. Proposed Algorithms

Proposition IV..1. *There exists an algorithm that can convert a simplex form to a threat branch.*

Proof. Given any induction (e.g., I) in the set, if there does not exist any node whose label is equal to the condition of the induction (i.e., $\nexists n \in NODE, n.label = I.condition$), then create a node $node$ and set the label of node as that condition (i.e., $node.label = I.condition$). Create an edge e who starts from the node ($e.from = node$). The conclusion of the induction is denoted it at the end of the edge (i.e., $node.to = conclusion$). If there exists a node (e.g., m) whose label includes the label of this edge (i.e. $node.to \in m.label$,), then set the edge point to this node (i.e., $e.to = m$).

Proposition IV..2. *There exists an algorithm that can convert an induction set with simplex forms into a threat tree with merged nodes.*

Proof. It is accomplished by conducting all inductions in the set, due to Proposition IV..1.

The algorithm that can convert an induction set with simplex forms into a tree with merged nodes, can be proposed as follows (see Algorithm 1):

Data: A set of *SimplexInduction* - *set*.
Result: A threat tree T.
while ($set! = Null$) **do**
  Fetch a $SimplexInduction \in set$;
  if ($\forall node \in T$, $SimplexInduction.condition! = node.label$) **then**
    Add *node* to T;
    $node.label \Leftarrow condition$,
    $node.to \Leftarrow conclusion$;
    Add *edge* to T;
    $edge.from \Leftarrow node$;
    Find $n \in T$ such that $conclusion \in n.label$;
    $edge.to \Leftarrow n$;
  end
  Delete *SimplexInduction* from *set*;
end
return 1;

Algorithm 1: Threat tree construction algorithm.

Proposition IV..3. *There exists an algorithm that can convert a threat branch into a simplex form.*

Proof. Given a branch, e.g., $node_1$ and $node_2$ where $edge.from = node_1$ and $edge.to = node_2$. The simplex form is as follows: $node_1.label \Rightarrow node_1.to \in node_2.label$.

Proposition IV..4. *There exists an algorithm that can convert a threat tree with merged nodes into an induction set with simplex forms*

Proof. It is accomplished by conducting all tree branches into inductions, due to Proposition IV..3.

The algorithm that can convert a tree with merged nodes into an induction set with simplex forms, can be proposed as follows (see Algorithm 2):

Data: A threat tree T.
Result: A set of *SimplexInduction* - *set*.
while $(T! = Null)$ **do**
  Fetch a $node \in T$;
  Find $edge \in T$ such that $edge.from = node$;
  if $(\forall SimplexInduction \in set,$
   $SimplexInduction.condition! = node.label)$
   then
   Add $node.label \Rightarrow node.to$ to *set*.
  end
  Delete *node* and *edge* from T;
end
return 1;

Algorithm 2: Induction construction algorithm.

5. Performance Evaluation

We set up an experiment to calculate the time consumption of these two fucntions: induction sets convert to threat tree, and threat tree convert to induction set. The number of induction sets are 50, 100, and 150. The results are shown in Fig. 2.

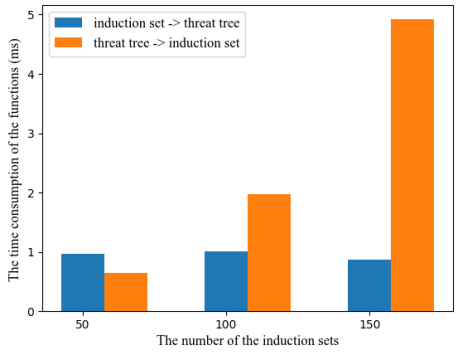

Fig. 2: The time consumption of the algorithms.

As we can see, when the number of induction sets are 50, the time of converting them to threat tree is 0.96ms, while threat tree to induction set is 0.65ms. When the number of induction set reaches to 100, the time of these two functions are 0.997ms and 1.993ms, seperately. When the number of induction set is 150, which means the tree is large enough, the time of these two functions are only 1.06ms and 4.92ms, respectively. Therefore, the time consumptions of the algorithms are within 'ms' level, which means MergeTree is practical to merge the nodes to build a threat tree from induction sets.

Besides, we calculate the edge decrease ratio for the MergeTree. We find that when the average number of nodes merged in one Node is 2 to 3, the edge decrease ratio is about 50% to 60%. When the number of nodes that merge in one Node increases, the edge decrease ratio ratio is going up, which means the MergeTree can load a large induction set with low edge degree. To notice, the MergeTree can build the tree of induction sets without much limit on node and edge degree, paving a practical and flexible way for building attack trees.

6. Analysis

Proposition VI..1. *The threat tree with merged nodes is equivalent to a set of simplex induction in terms of semantics.*

Proof. It is due to Proposition IV..2 and Proposition IV..4. We need to prove that the induction set can be transformed into threat tree. In the threat tree, there are multiple nodes (node.label, node.to) connected by edges (edge.from, edge.to). Given a simplex induction set, the context is a list of <conditions, conclusion>. Each condition and conclusion can be considered as nodes and they are connected by edges. That is, the value of node.label and edge.from is condition, and the value of node.to and edge.to is conclusion. Besides, we need to prove that the threat tree can be converted to simplex induction set. Obviously, in a threat tree, each edge is linked with two nodes, which represent the edge.from and edge.to. In simplex induction set, the conditions are edge.from and the conclusions are edge.to.

Remark VI..2.

A simplex induction is equivalent to a node and an edge indeed. The visualization in the tree provides a better understanding for induction.

A tree model will facilitate the manipulation of threat information processing, e.g., searching a threat route, detecting the existence of a threat, listing possible threat methods, and so on, by off-the-shelf tree algorithms. That is the reason of tree representation of threat induction information.

The simplex form simplifies the structure of induction, so as to simplify the structure of threat tree, with manageable overhead in the number of induction numbers.

7. Conclusion

In this paper, we study how to decrease the number of nodes (also branches) for threat model when the number of threat information keeps on increasing. The threat model with merged nodes is proposed and respective critical algorithms are provided. The presentation is formal, e.g., the equivalence between simplex induction and tree branch, as well as between threat tree and induction set, so as to derive the respective key conversion algorithms. The experiment results and analysis justifies the soundness and completeness of the proposed model. Besides, the tree structure has almost remained so that original tree algorithms for threat analysis can still work.

Acknowledgment

The research was financially supported by the Science and Technology Program of Guangzhou, China (No. 202102021216), the Foundation of Hubei Key Laboratory of Intelligent Geo-Information Processing (No. KLIGIP-2021B06), and the Foundation of National Natural Science Foundation of China (No. 61972366).

References

[1] B. Schneier, "Attack Trees: Modeling Security Threats", *Dr.Dobbs Journal*, vol.24, no.12, pp. 21-29, 1999.

[2] A.T. Ali, D.P. Gruska, "Attack Trees with Time Constraints", in *Proc. of the 28th International Workshop on Concurrency, Specification and Programming (CS&P2021)*, 2021, pp. 27-28.

[3] Asif, Waqar, Indranil Ghosh Ray, and Muttukrishnan Rajarajan. "An attack tree based risk evaluation approach for the internet of things," in *Proc. of the 8th International Conference on the Internet of Things*, 2018, pp. 1-8.

[4] Schiele, Nathan Daniel, and Olga Gadyatskaya. "A Novel Approach for Attack Tree to Attack Graph Transformation," *International Conference on Risks and Security of Internet and Systems*. Springer, 2022, pp. 1-8.

[5] H. Mantel, C. W. Probst, "On the Meaning and Purpose of Attack Trees", in *Proc. of 2019 IEEE 32nd Computer Security Foundations Symposium (CSF2019)*, 2019, pp. 184-18415.

[6] Scala, Natalie M., et al. "Evaluating mail-based security for electoral processes using attack trees." Risk Analysis (2022).

[7] Ji, Xiang, et al. "Attack-defense trees based cyber security analysis for CPSs." in *Proc. of 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2016, pp. 693-698.

[8] Kammueller, Florian. "Attack trees in Isabelle extended with probabilities for quantum cryptography," *Computers and Security*, vol. 87, pp: 101572, 2019.

[9] PETRICa, Gabriel. "Cybersecurity of WordPress Platforms. An Analysis Using Attack-Defense Trees Method." *International Conference on Cybersecurity and Cybercrime*, vol. 9, pp. 69-76, 2022.

[10] A.T. Ali, D.P. Gruska, "Attack Protection Tree", in *Proc. of the 28th International Workshop on Concurrency, Specification and Programming (CS&P2019)*, 2019, pp. 1-6.

[11] Lenin, A., Willemson, J., Sari, D.P., "Attacker Profiling in Quantitative Security Assessment Based on Attack Trees", *Bernsmed, K., Fischer-HÃŒbner, S. (eds) Secure IT Systems. NordSec 2014. Lecture Notes in Computer Science()*, vol 8788, pp. 199-212, 2014.

[12] Depamelaere, Wouter, et al. "CPS security assessment using automatically generated attack trees," in *Proc. of the 5th international symposium for ICS & SCADA cyber security research 2018. British Computer Society (BCS)*, 2018, pp. 1-10.

[13] Nishihara, Hideaki, et al. "On Validating Attack Trees with Attack Effects: An Approach from Barwise-Seligman's Channel Theory," arXiv preprint arXiv:2204.06223 (2022).

[14] Pinchinat, Sophie, Mathieu Acher, and Didier Vojtisek. "ATSyRa: an integrated environment for synthesizing attack trees," *International Workshop on Graphical Models for Security*, 2015, pp. 97-101.

[15] Gadyatskaya, Olga, et al. "Refinement-aware generation of attack trees." in *Proc. of International Workshop on Security and Trust Management*, 2017, pp. 164-179.

[16] Ali, Aliyu Tanko, and Damas Gruska. "Dynamic Attack Trees Methodology," in *Proc. of 2022 Interdisciplinary Research in Technology and Management (IRTM)*, 2022, pp. 1-9.

[17] Pinchinat, Sophie, Francois Schwarzentruber, and Sebastien Le Cong. "Library-Based Attack Tree Synthesis," in *Proc. of International Workshop on Graphical Models for Security*, 2020, pp. 24-44.

[18] Paul, Stephane, and Raphael Vignon-Davillier. "Unifying traditional risk assessment approaches with attack trees," *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 165-181, 2014.

[19] Fila, Barbara, and Wojciech Wide. "Exploiting attack defense trees to find an optimal set of countermeasures," in *Proc. of 2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, 2020, pp. 395-410.

[20] Vigo, Roberto, Flemming Nielson, and Hanne Riis Nielson. "Automated generation of attack trees." in *Proc. of 2014 IEEE 27th computer security foundations symposium*, 2014, pp. 337-350.