Log-based Anomalies Detection of MANETs Routing with Reasoning and Verification

Teng Li*, Jianfeng Ma*, Qingqi Pei*, Yulong Shen§ and Cong Sun*
§School of Computer Science
*School of Cyber Engineering
Xidian University, Shaanxi, China
Email: litengxidian@gmail.com

***Abstract*—Routing security plays an important role in Mobile Ad hoc Networks (MANETs). Despite many attempts to improve its security, the routing procedure of MANETs remains vulnerable to attacks. Existing approaches offer support for detecting attacks or debugging in different routing phases, but many of them have not considered the privacy of the nodes during the anomalies detection, which depend on the central control program or a third party to supervise the whole network. In this paper, we present an approach called LAD which uses the raw logs of routers to construct control a flow graph and find the existing communication rules in MANETs. With the reasoning rules, LAD can detect both active and passive attacks launched during the routing phase. LAD can also protect the privacy of the nodes in the verification phase with the specific Merkle hash tree. Without deploying any special nodes to assist the verification, LAD can detect multiple malicious nodes by itself. To show that our approach can be used to guarantee the security of the MANETs, we deploy our experiment in NS3 as well as the practical router environment. LAD can improve the accuracy rate from 2.28% to 29.22%. The results show that LAD performs limited time and memory usages, high detection and low false positives.**

***Keywords*—MANETs; Syslog; Verification; Privacy; Diagnostics**

I. Introduction

The Mobile Ad hoc Networks (MANETs) [1] are continuously self-configuring, infrastructure-less networks of mobile devices connected without wires. In such a network, all the mobile nodes collaborate with each other and establish routing in a self-organized way. The primary goal of routing in MANETs is to establish a correct and efficient path where data can be efficiently and securely transmitted [2]. One particular challenge for MANETs is problem diagnosis. That is, when problems occur, quickly detecting the anomalies and identifying the root causes is of utmost importance. Recent efforts in diagnosing MANETs problems have focused on using problem tickets [3] and network provenance [4]. Besides, others [5] [6] have used packets trace to diagnose problems. Although they can detect the problems, they will use all of the information of the nodes which can be difficult to use in a distributed network environment. However, network device logs are straightforward and a common source of information for anomalies detection. Log data are printed by various applications to record the conditions or events on the equipment, which are initially used for code debugging for the developers. As the logs can record the past events and attack clues, they have become the first choice of the developers to diagnose the systems or devices.

Despite the potential benefits, syslogs, unlike most other readily available troubleshooting data (e.g. problem tickets, traceroutes), are not embedded with sufficient information to not only diagnose anomalies but also point out the root cause of the problems. The raw syslog messages are originally used for debugging purposes and pose many challenges in equipment diagnosis. First, syslogs are massive in volume which contain complex kinds of messages [7], posing as a computational challenge. Second, they are unstructured data and lack homogeneity posing as a semantic challenge for analysis [8]. Third, log data are too low-level [9] and rarely contains explicit information for anomaly detection which cannot be used directly.

There are also a lot of works using syslogs to diagnose the problems of network devices [10] [11]. However, they all use raw syslogs to detect the anomalies on a sole network device. In MANETs, devices (e.g. routers) connect and communicate with each other using certain protocols. To diagnose the MANETs, we should not only consider one network device but the relationship among them. For example, when the network suffers a black hole attack [12], every node behaves well and the logs on the devices seem benign individually. However, when we diagnose all nodes involved in the event with the communication rules then we can find out the fabricated behaviors [13]. Thus, diagnosing sole devices is not adequate enough and we should consider all of the devices in the network.

In the MANETs environment, mobile devices communicate with each other in a self-organized way. Therefore, introducing a third party to diagnose the network is inadequate as this will violate the intrinsic distribution characteristic of the MANETs and bring extra security problems. During the self-diagnosis process, the devices will exchange information of the devices, which is prone to leaking privacy data (e.g. disclosing all the plain text of the syslogs). Thus, the approach should also protect the privacy of the devices during the diagnosing phase in MANETs, otherwise the diagnosis itself is not secure and will bring in new problems to the network devices.

In this work, we concentrate on two main goals: (1) detect anomalies in MANETs (such as active and passive attacks [14]) based on syslogs of the network devices; (2) protect the network devices' privacy (e.g. conceal the plain-text syslogs) during the diagnosis phase. However, there is an inherent tension between these two goals because the diagnosis usually requires revealing private log entries or packets of nodes [15]. To mitigate the above problems, we present LAD (Log-based Anomalies Detection) which uses syslogs of the network devices to diagnose the anomalies in MANETs. First, LAD transfers the raw unstructured syslogs to high level structured log entries. Second, LAD uses a Control Flow Graph [16] [17] combined with association rules to determine the relationship of the communication rules among the devices. Then, with the rules, LAD deduces the expected logs on the devices. Finally, with the help of a specific Merkle Hash Tree, LAD can achieve the privacy preserving goal while verifying the real logs on the devices and the expected ones. We have applied LAD in MANETs anomalies detection and it can detect both active and passive attacks. LAD can also detect multiple attacks in practical routers and it can achieve more stable results.

In summary, our key contributions are:
1. A novel log template extraction method is proposed combining regular expression and Jaccard similarity (Section II-A).
2. We construct a Control Flow Graph and find three different models among the templates. In this way, we can find the relations among the log templates instead of making them (Section II-B).
3. With the communication rules coded in NDlog, the reasoning process can be accomplished in a distributed way without introducing a third party (Section II-D).
4. The specific Merkle Hash Tree is constructed containing the syslogs to do confidential verification without leaking the privacy of the routers (Section II-E).

II. Methodology of LAD

In this section we elaborate LAD in detail. We explain how to transfer the raw logs to the reasoning rules with CFG and how to do the reasoning and verification.

A. Extracting the Log Templates

There are huge amounts of logs and various forms of unstructured logs with a large number of parameters in the syslogs. With the template-extracting, we can transfer the raw unstructured syslogs to structured high-level information. Then, finding the relationship among the logs can be launched.

To extract the template correctly, we propose a novel method. According to the practical experiment, a large number of parameters in log entries are IP, MAC and some explicit combination of numbers and words. To accelerate the processing time and efficiency, we use the regular expression to first filter out the numbers, IP addresses, MAC addresses, the mixture of numbers and letters (such as $eth1$, $vlan3$), and the content between some symbols (such as [], (), ""). With the process of regular expression, the primary templates are shown in Table II.

TABLE I: Examples of Regular Expression

Text Example	Regular Expression
192.168.1.1	[1-9]|[1-9]\\d|1\\d2|2[0-4]\\d| 25[0-5])(\\.(\\d|[1-9]\\d|1\\d2 |2[0-4]\\d|25[0-5]))3|[a-fA-F0-9]8
70:48:0F:36:79:0F	[a-fA-F0-9]2+:[a-fA-F0-9]2+:[a-fA-F0-9]2+:[a -fA-F0-9]2+:[a-fA-F0-9]2+:[a-fA-F0-9]2|[a-fA -F0-9]1,2+:[a-fA-F0-9]1,2+:[a-fA-F0-9]1,2+:[a -fA-F0-9]1,2+:[a-fA-F0-9]1,2+:[a-fA-F0-9]1,2
eth1	[a-zA-Z]+\\d+|\\d+[a-zA-Z] +|\\d+|(-)?[1-9][0-9]*$
udhcpd[725]	.*[\\d+].*

TABLE II: Primary logs by regular expression

Message
`udhcpd[*]: sending ACK to *`
`udhcpd[*]: received REQUEST from *`
`udhcpd[*]: sending OFFER to *`
`udhcpd[*]: received DISCOVER from *`
`* add * mcast address to master interface`
`* Setting MAC address to *`

Then, we also need to extract the template from some lexical variables without numeric ones. As shown in Table II, the words, ACK, $REQUEST$, $OFFER$ and $DISCOVER$, are also parameters. As is done in work [11], we first pick the messages with the same beginning words with the same length and the two same primary templates will only be used once. First, we use Jaccard similarity [18] to calculate the two templates' similarity as shown in Equation 1:

$$SIM(S,T) = \frac{|S \cap T|}{|S \cup T|} \quad (1)$$

We use the Jaccard similarity (>0.9) to cluster the log templates together. Then, LAD finds the other words by eliminating the same ones in every template.

$$l \epsilon \{L_i - L_{same}\}, i \; is \; the \; sequence \; of \; the \; template. \quad (2)$$

For each l in every template, LAD finds the position of l in the template and creates a position vector. LAD compares the position vector. If the vectors are the same, it can tell that the different words appear in the same position and we can know that the parameters are in these places. Due to the above approach, LAD can find out the templates of the log correctly.

B. Finding the Rules in MANETs

In MANETs, the equipment links and communicates with each other in certain rules which have been set before they send the messages. We need to find these communication rules instead of making them. Many approaches [4] [19] relied on rule-based processing which has improved the routers' debugging ability, but it requires the operator to have domain knowledge and involves a human operator to make the rules for the detection. For our black-box approach for anomaly diagnosis, we cannot know the communication rules except for finding them.

We use the Apriori algorithm to find the relations between the two log templates and construct these relations into CFG (control flow graph) [20] [21]. A control flow graph of a program flow P is a directed graph $G = (N, E, n_s, n_e)$. N is the node set that contains all of the basic blocks of the flow. E is the edge set among the nodes. n_s is the start node of the flow and n_e is the end point ($n_s \epsilon N$ and $n_e \epsilon N$). Each time the first block is executed, the following instructions must also be executed in order. Thus, with CFG, we can get the relationships and sequence order of the logs. As is shown in Figure 1, we need to convert the log templates into CFG.

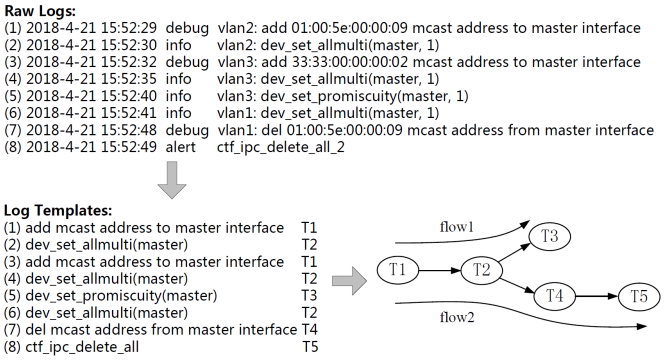


Fig. 1: Logs and CFG

We denote FS as the Following Sequence and PS as the Prior Sequence. $FS_{(T)}$ of the template T means that the following templates emerge when T shows up. $PS_{(T)}$ of the template T means the prior templates of T. Then, we can calculate the possibility of $FS_{(T)}$ and $PS_{(T)}$.

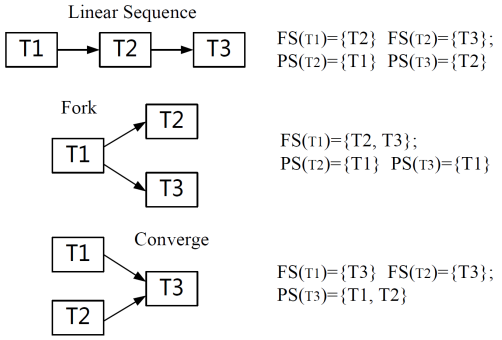


Fig. 2: Three types of sub-structures of CFG

C. The Deduction Rules for LAD

We transfer the raw syslogs to the structured log entries. For example, when a sender wants to find a route, it first broadcasts a routing request, `sendRequest(@C,S,D,SEQ)`, to its neighbors. The request means `S` wants to find a routing path to `D` whose sequence number is in `SEQ` and this request is logged at `C`. `C` and `R` represent the sender's and receiver's storage place respectively. `S` represents the sender, `D` is the receiver and `SEQ` represents the destination sequence number which is used to determine the freshness of the routing information. `IP` means the destination's plain IP address. `R` is the intermediate router, and it stores its messages in the place of `M`. `MSG` is the message that the source node wants to send to the destination host.

With the help of CFG, we can get the relations of the syslogs and LAD can use these rules for the reasoning. In the routing environment, we only trust the sender and we think the destination has the possibility to deceive the source node by launching attacks such as a black hole attack [22] or worm hole attack [23]. This attack scenario is more common in MANETs.

First, LAD uses syslogs on the trusted source node S to deduce the information that should be held by destination node D. The sender is not sure whether D is benign or it received the messages the source node sent. LAD verifies D's real log to check whether it can provide evidence to prove its correctness by using the Merkle Hash Tree. If the result is true, we can know that the intermediated nodes forwarded the messages correctly and there is no active attack.

Then, LAD uses the information from the source and destination to verify whether the intermediated nodes are well-behaved to check if there are passive attacks in the route. If the result is false, it means that some nodes must launch the active attacks and LAD just uses the logs of the source to deduce the logs of the intermediated routers. In this way, we can finally guarantee the security of the intermediated routers along the routing path and make sure the whole path is healthy. We use MHT for the verification and this can guarantee privacy during the verification phase. Next, we elaborate the details of the reasoning and verification.

D. Reasoning Approaches

First we use the sender's information to infer the messages on the receiver to identify the destination node. For convenience, this phase is denoted by `S`→`D` (source node deduces the destination node), which consists of the reasoning of the following propositions:

```
getReply(@C,S,D,IP,SEQ) ∧ sendRequest(@C,S,D,IP,SEQ)
    → sendReply(@R,S,D,IP,SEQ)                          (3)
findDest(@C,S,D,IP,SEQ) ∧ getReply(@C,S,D,IP,SEQ)
    → getRequest(@R,S,D,SEQ)                            (4)
dataLink(@C,S,D,IP,SEQ,'YES')
    ∧ msgRequest(@C,S,D,IP,SEQ,MSG)
    → authSend(@R,S,D,IP,SEQ,MSG)                       (5)
```

We use the messages of S stored at the place of C to infer the information of the destination stored at R. Combined with the basic rules and NDlog language [24], the expected information of the receiver can be obtained. These messages should be on the destination node according to the transmission mechanism. Then, a significant problem is how to determine whether or not the receiver has such information. Next, we will introduce the Merkle Hash Tree into our method to realize the goal of privacy protection.

E. Privacy-Preserving Verification

In this section, we introduce the mechanism of privacy-preserving verification. We use the Merkle Hash Tree [25] (MHT) to preserve privacy during verification. MHT is a tree in which every non-leaf node is labeled with the hash value of the labels of its children nodes, and every leaf node is labeled with the hash value of real data. As a kind of binary tree, the edges of MHT from every parent's node to its two children are tagged with 0 and 1 respectively; see Figure 3.

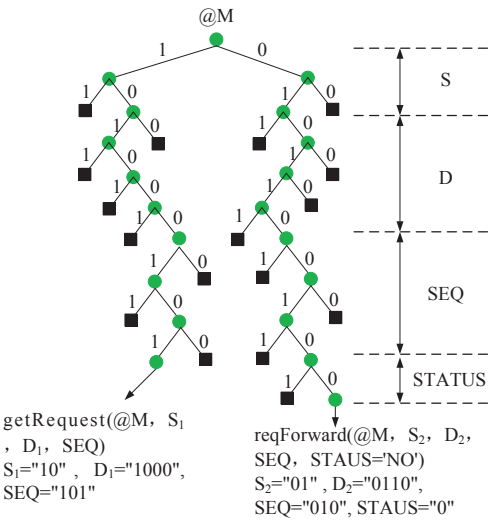


Fig. 3: Merkle Hash Tree

Before building the tree, we should first encode the information. Take the following message encoding for example. `getRequest(@M,S1,D1,SEQ)` can be encoded by specifying S_1="10", D_1="1000", SEQ="101". The message `reqForward(@M,S2,D2,SEQ,STAUS='NO')` can be encoded by specifying S_2="01", D_2="0110", SEQ="010", STAUS(NO)="0". How many bits the variables, such as S_1 and S_2, cost for the message encoding will depend on the number of nodes and the number of variables. Our goal is to distinguish each message on different nodes by encoding the messages.

Next, it is the tree building. We built MHT with the same approach as in our previous work [13]. After we encode the message `reqForward(@M,S2,D2,SEQ,STAUS='NO')` as a string "0101100100", we use it to build the tree. Each router will build a Merkle Hash Tree using its log information. We label the node's left child as 1 and right child as 0. With the string, we can build up a tree, which is stored as an array.

Then, we can calculate the hash value of each node: H_i=H(node_num‖bit_data‖parent_num‖string‖parent_bit_data)‖ H_{left_child}‖ H_{right_child}. As an index of the node array, node_num specifies the array element w.r.t the current node. Similarly, parent_num specifies the array element w.r.t the parent of the current node. string represents the IP address string or the value of DesIPhash. Also bit_data and parent_bit_data are the value of current node and parent node respectively, which can be either 0 or 1.

When a node is the root of Merkle Hash Tree, we define parent_bit_data of this node as 'X'. H_{left_child} and H_{right_child} are the hash value of the left child and right child respectively. They will be empty if the current node is a leaf node. We can calculate the hash value from the leaf node to the root, and this value will be published. That means every node may know the root hash value of any other nodes. In the verification phase, the sender has the destination's published hash value in advance and then calculates the root hash by itself with the parameters that the destination provides. If the latter hash value equals the published value, the verification result is true.

After we have ensured that the destination node is the legal host, we move to the intermediate router verification phase. To make sure the whole routing path is healthy without malicious nodes, we need to verify the forwarding nodes, ensure they transmit the data according to the rules and eliminate the possibility of launching attacks such as passive or active attacks. We have proven that the information on the destination node is correct, and we can use the information on both S and D to conduct the following deduction. We call this process as S+D→M (source node and destination node deduce the intermediate routers). In this process we should reason the following propositions:

```
sendRequest(@C,S,D,IP,SEQ)
    → getRequest(@M,S,D,IP,SEQ)                                  (6)
getReply(@C,S,D,IP,SEQ) ∧ sendRequest(@R,S,D,IP,SEQ)
    → reqForward(@M,S,D,IP,SEQ,'NO')                             (7)
getReply(@C,S,D,IP,SEQ) ∧ getRequest(@R,S,D,IP,SEQ)
    → findDest'(@M,S,D,IP,SEQ)                                   (8)
sendMsg(@C,S,D,IP,SEQ,MSG)
    ∧ authSend(@R,S,D,IP,SEQ,MSG)
    → msgForward(@M,S,D,IP,SEQ,MSG)                              (9)
```

Then we use the Merkle Hash Tree to verify that the real log of the intermediate routers comply with the expected log information, e.g. `reqForward(@M,S,D,IP,SEQ,STAUS)`.

If the logs from the destination are false, we should find out which intermediated routers influenced it and identify the passive attacker. We call this process as S→M (source node deduces the intermediate routers). In this process we should reason the following propositions:

```
sendRequest(@C,S,D,IP,SEQ)
    → reqForward(@M,S,D,IP,SEQ)                    (10)
sendRequest(@C,S,D,IP,SEQ)
    → getRequest(@M,S,D,IP,SEQ,STAUS)              (11)
findDest(@C,S,D,IP,SEQ)
    → findDest'(@M,S,D,IP,SEQ)                     (12)
dataLink(@C,S,D,IP,SEQ,'YES')
    ∧ sendMsg(@C,S,D,IP,SEQ,MSG)
    ∧ msgRequest(@C,S,D,IP,SEQ,MSG)
    → authSend(@R,S,D,IP,SEQ,MSG)                  (13)
```

III. Evaluation

In this section, we evaluate LAD through our experimental results. Specifically, we show the performances of LAD in log templates extraction, CFG construction, and attack detection. We did our experiments on NS-3 and a real router environment respectively.

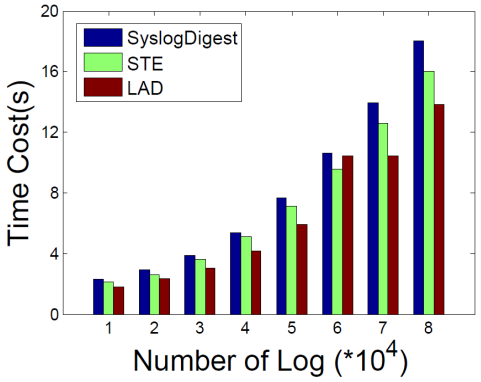
A. Experiment Setup

In NS-3, we configured the nodes that ran the AODV protocols and we injected few malicious nodes that dropped or tampered with the logs of the nodes. As for the real routers, we launched attacks towards the routers on the link. To evaluate LAD's performance, we set the former work CRVad [24], PVad [26] and SyslogDigest [3] as the benchmark which does not use the algorithm to find the rules and is handled by the operators.

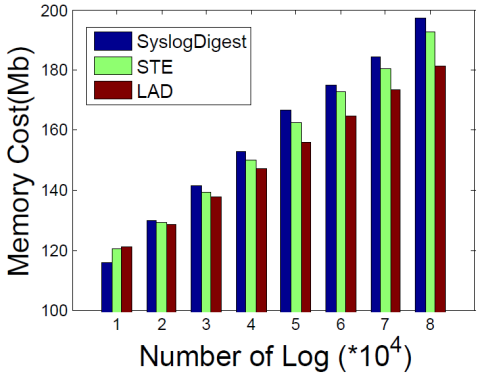
B. The Performance of Log Templates Extracting

We compare LAD's templates extracting approach with former methods: SyslogDigest [3] and STE [27]. SyslogDigest first clusters the same beginning logs together and constructs them as a sub-type tree. For the parameters in the tree, they will present as the branches or children of a node. The key process of SyslogDigest is pruning the tree until it has the desired degree properties. If a parent node has more than k children, it discards all of the children to make the parent a leaf itself. STE takes another method using the following features of log messages: parameters appear less frequently than template words; and messages have similar structures with the positions of the words. The approach gives each word in the log a score but the word score may be inaccurate for repeated words in several formats. The time and memory cost comparisons are shown in Figure 4.

For using the regular expression to process the raw syslogs, we can filter out most parameters in the raw logs, such as IP addresses, MAC addresses and a mixture of numbers and words. As for the STE approach, it is a time-consuming work to give the score for each word and DBSCAN can cost lots of memory as the number of logs increases. According to Figure 4(a) and Figure 4(b), LAD has less time and memory cost than SyslogDigest and STE in the template extraction process. Because SyslogDigest and LAD both need to construct the tree



(a) Time Cost



(b) Memory Cost

Fig. 4: Cost of Templates Extracting

and traverse each node's sub-tree, both of them finally have $O(n^2)$ in time complexity. Besides, the time complexity of DBSCAN is also $nlog(O(n^2))$.

C. Detection of the Active and Passive Attacks

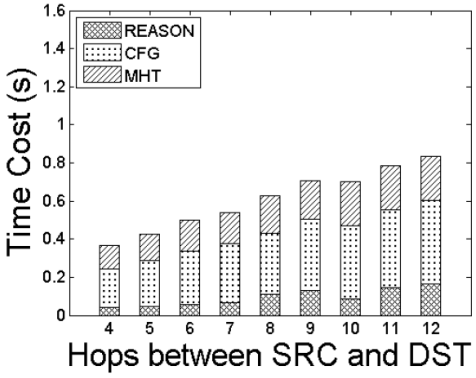
We inject two kinds of errors into the MANETs which are shown in Table III. We set the hops between the source and destination from 4 to 12, and the malicious nodes are random nodes of the intermediated routers in the link. $F1$is an active attack which discards the packets during the transmission and only one malicious node launches this attack. $F2$ is the passive attack which does not influence the packets of the network but tampers its own logs and there are two suspicious nodes doing this.

Faults	Attack	Malicious Behavior	Attacker
F1	Active	Discard packets	Single
F2	Passive	Tamper syslogs	Double

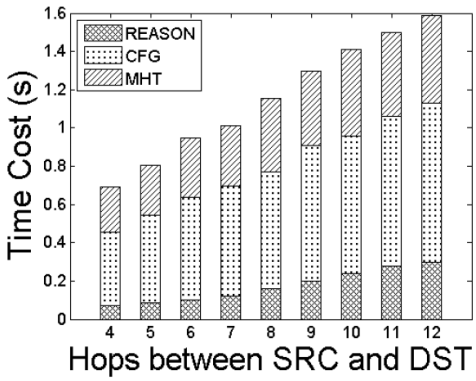
TABLE III: Injected faults in our experiments

In Figure 5, $F2$ is more time-consuming than $F1$. In passive attack detection, LAD has to check all of the intermediated routers and finishes all of the processes of reasoning, CFG construction and MHT verification. Besides, in $F2$ there are two malicious attackers and this will also cost more time than

detecting just one attacker in $F1$. From Figure 5(a) - 5(b), CFG construction occupies the most time in the whole turnaround time cost of the detection. Because LAD has to analyze the three substructures and calculates the relations of each one, this can cost a lot of time during this process. However, CFG construction mitigates the burden of the reasoning, because it provides the relations among the templates and LAD can use these results for direct deduction. Comparing Figure 5(a) and Figure 5(b), the reasoning cost of $F2$ is nearly double that of $F1$, because $F2$ has two attackers while the single reasoning time cost in these two faults detection phases is almost the same. As for the Merkle Hash Tree verification, $F2$ has more intermediated nodes to verify. Thus, it will cost more time in the MHT construction and root hash value calculation processes, which is also shown in Figure 5.



(a) F1



(b) F2

Fig. 5: Time Cost of Detection Injected Faults

We also launched an experiment on the real router environment, which consists of Cisco, Huawei, and Dlink. We launched the attacks on these routers during three months and the attacks could influence the sequences of the log orders. We launched the attacks in the platform of Windows 7 and Ubuntu 12.07, which contain Violence Crack Login, IP Spoofing Attack, SSL Attack, DOS Attack and ARP Spoofing Attack. We used different tools and our own codes to realize these attacks, which are shown in Table IV. We compare the results of LAD with CRVad [24], PVad [26] and SyslogDigest [3], which can also verify the correctness of the nodes in MANETs. CRVad uses the reasoning and verification to check the nodes but it does not use the algorithm to find the rules and the rules are made by the operators. PVad uses the Apriori algorithm to do the rule-mining among the routers and SyslogDigest focuses on the syslog analyses to do the anomaly detection.

TABLE IV: Attacks and tools

No.	Attack Name	Tool or Method	Platform
1	Violence Crack Login	Java code	Windows 7
2	IP Spoofing Attack	Nmap	Windows 7
3	SSL Attack	THC-SSL	Windows 7
4	DOS Attack	HULK	Ubuntu 12.07
5	ARP Spoofing Attack	WinArpAttacker	Windows 7

$$Precision = \frac{Correctly\ detected\ attacks}{Total\ inserted\ attacks} \tag{14}$$

$$Recall = \frac{Correctly\ detected\ attacks}{Total\ detected\ attacks} \tag{15}$$

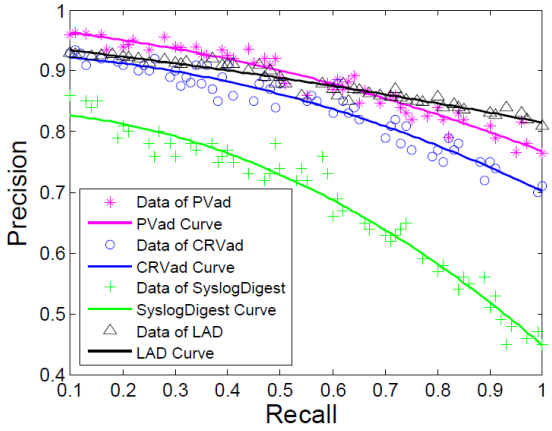


Fig. 6: Comparison among LAD, PVad, CRVad and SyslogDigeston PR

We denote the precision and recall with Equation 14 and Equation 15. We collected the key-value pairs of recall and precision. Then, we select the points which range from 0.1 to 1 in the recall and draw the fitted curve in Figure 6. The error sum of the squares of LAD, PVad, CRVad and SyslogDigest are 0.0031, 0.0091, 0.0144, and 0.0257 respectively, which are acceptable in our experiments. LAD and PVad can achieve better results than CRVad and SyslogDigest. As the results show, LAD can have more stable detection results than PVad.

IV. Conclusion

In this paper, we proposed an approach, LAD, to detect anomalies of MANETs routing with reasoning and verification. We transferred the raw syslogs to structured rules with the control flow graph construction. We avoided just

analyzing the single router device with the raw log on the router but combined the communication rules in the MANETs. Thus, we achieved the goal of detecting anomalies of the devices connected in the network. LAD uses the rules to do the reasoning without introducing a third party. With the construction of a specific Merkle Hash Tree, we can verify the truth of the syslogs without disclosing privacy of the routers. According to our experiment, LAD can detect the active and passive attack in the MANETs and the approach is scalable and practical for use in real MANETs.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China (61502368, 61602357 and U1405255), the National High Technology Research and Development Program (863 Program) of China (No.2015AA017203, No.2015AA016007), Natural Science Basis Research Plan in Shaanxi Province of China (Grant No. 2017JM6047 and 2016JM6034), the Fundamental Research Funds for the Central Universities (XJS17077, JBX171507, JB170303), China Postdoctoral Science Foundation Funded Project(2016M592762).

REFERENCES

[1] Z. Wei, H. Tang, F. R. Yu, M. Wang, and P. Mason, "Security enhancements for mobile ad hoc networks with trust management using uncertain reasoning," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4647–4658, 2014.

[2] M. Conti and S. Giordano, "Mobile ad hoc networking: milestones, challenges, and new research directions," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 85–96, 2014.

[3] T. Qiu, Z. Ge, D. Pei, J. Wang, and J. Xu, "What happened in my network: mining network events from router syslogs," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 472–484.

[4] Y. Wu, M. Zhao, A. Haeberlen, W. Zhou, and B. T. Loo, "Diagnosing missing events in distributed systems with negative provenance," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 383–394.

[5] M. Attariyan, M. Chow, and J. Flinn, "X-ray: Automating root-cause diagnosis of performance anomalies in production software." in *OSDI*, vol. 12, 2012, pp. 307–320.

[6] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica, "X-trace: A pervasive network tracing framework," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*. USENIX Association, 2007, pp. 20–20.

[7] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference on Software Engineering Companion*. ACM, 2016, pp. 102–111.

[8] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection." in *USENIX Annual Technical Conference*, 2010.

[9] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California fault lines: understanding the causes and impact of network failures," in *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4. ACM, 2010, pp. 315–326.

[10] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1285–1298.

[11] T. Li, J. Ma, and C. Sun, "Dlog: diagnosing router events with syslogs for anomaly detection," *The Journal of Supercomputing*, vol. 74, no. 2, pp. 845–867, 2018.

[12] S. Misra, K. Bhattarai, and G. Xue, "Bambi: Blackhole attacks mitigation with multiple base stations in wireless sensor networks," in *Communications (ICC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–5.

[13] T. Li, J. Ma, and C. Sun, "Srdpv: secure route discovery and privacy-preserving verification in manets," *Wireless Networks*, pp. 1–17, 2017.

[14] D. Kapetanovic, G. Zheng, and F. Rusek, "Physical layer security for massive mimo: An overview on passive eavesdropping and active attacks," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 21–27, 2015.

[15] A. J. Gurney, A. Haeberlen, W. Zhou, M. Sherr, and B. T. Loo, "Having your cake and eating it too: Routing security with privacy protections," in *Proceedings of the 10th ACM workshop on hot topics in networks*. ACM, 2011, p. 15.

[16] M. H. Nguyen, T. B. Nguyen, T. T. Quan, and M. Ogawa, "A hybrid approach for control flow graph construction from binary code," in *Software Engineering Conference (APSEC), 2013 20th Asia-Pacific*, vol. 2. IEEE, 2013, pp. 159–164.

[17] T. Jia, L. Yang, P. Chen, Y. Li, F. Meng, and J. Xu, "Logsed: Anomaly diagnosis through mining time-weighted control flow graph in logs," in *Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on*. IEEE, 2017, pp. 447–455.

[18] J. Ji, J. Li, S. Yan, Q. Tian, and B. Zhang, "Min-max hash for jaccard similarity," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 301–309.

[19] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. T. Loo, and M. Sherr, "Secure network provenance," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 295–310.

[20] E. J. Schwartz, J. Lee, M. Woo, and D. Brumley, "Native x86 decompilation using semantics-preserving structural analysis and iterative control-flow structuring," in *Proceedings of the USENIX Security Symposium*, vol. 16, 2013.

[21] C. Dietrich, M. Hoffmann, and D. Lohmann, "Cross-kernel control-flow–graph analysis for event-driven real-time systems," in *ACM SIGPLAN Notices*, vol. 50, no. 5. ACM, 2015, p. 6.

[22] H. Deng, W. Li, and D. P. Agrawal, "Routing security in wireless ad hoc networks," *IEEE Communications magazine*, vol. 40, no. 10, pp. 70–75, 2002.

[23] T. Hayajneh, P. Krishnamurthy, D. Tipper, and A. Le, "Secure neighborhood creation in wireless ad hoc networks using hop count discrepancies," *Mobile Networks and Applications*, vol. 17, no. 3, pp. 415–430, 2012.

[24] T. Li, J. Ma, and C. Sun, "Crvad: Confidential reasoning and verification towards secure routing in ad hoc networks," in *International conference on algorithms and architectures for parallel processing*. Springer, 2015, pp. 449–462.

[25] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, "Mur-dpa: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2609–2622, 2015.

[26] T. Li, J. Ma, C. Sun, D. Wei, and N. Xi, "Pvad: Privacy-preserving verification for secure routing in ad hoc networks," in *Networking and Network Applications (NaNA), 2017 International Conference on*. IEEE, 2017, pp. 5–10.

[27] T. Kimura, K. Ishibashi, T. Mori, H. Sawada, T. Toyono, K. Nishimatsu, A. Watanabe, A. Shimoda, and K. Shiomoto, "Spatio-temporal factorization of log data for understanding network events," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 610–618.