SCIENCE CHINA
Information Sciences

• HIGHLIGHT •

November 2017, Vol. 60 118101:1–118101:3
doi: 10.1007/s11432-016-0370-y

NetPro: detecting attacks in MANET routing with provenance and verification

Teng LI[1,2*], Jianfeng MA[1,2] & Cong SUN[2]

[1]*School of Computer Science and Technology, Xidian University, Xi'an* 710071*, China;*
[2]*School of Cyber Engineering, Xidian University, Xi'an* 710071*, China*

Received July 21, 2016; accepted September 12, 2016; published online December 9, 2016

Citation Li T, Ma J F, Sun C. NetPro: detecting attacks in MANET routing with provenance and verification. Sci China Inf Sci, 2017, 60(11): 118101, doi: 10.1007/s11432-016-0370-y

The Mobile Ad hoc Network (MANET) is a continuously self-configuring network of a collection of mobile devices with neither a defined infrastructure nor a central administration facility [1]. The primary goal of MANET routing is to provide a secure, correct, and efficient path to the target node such that the data are transmitted quickly and are not corrupted. Besides a passive attack, in which the network intruder attempts to eavesdrop on the communication, there exist many types of active attacks that tend to damage the network by changing the data stored within the MANET.

In a direct attack, e.g., black hole attack or worm hole attack, the adversaries can modify or drop packets and this will usually disable the network. In an indirect attack, an adversary deliberately modifies its log without modifying the data that is forwarded in order to confuse the network debugging system. Examples are PeerReview [2] and NetReview [3], which depend on the node's log to detect faults within the network. The capability of identifying malicious adversaries is also essential for diverse network management tasks such as performing network accounting [4], identification of malicious and misbehaving nodes, and enforcing trust management policies within distributed systems. To identify malicious nodes engaged in direct or indirect attack and explain the origins of these misbehaviors in the network without disclosing the node's privacy, we present NetPro, an automated system that detects routing faults occurring in MANET. Our approach uses provenance reasoning to trace back the faults that have occurred in the MANET until we spot the fault causes.

In this article, we first spot the influenced nodes automatically without the use of human inspectors and then trace back to find out which node originally triggers such faults by using provenance reasoning. The provenance of an observed event is represented as a chain of events that constitutes the path from the node that generated the fault to the current device. This backward trace links the event to its original cause(s). The chain is referred to as the provenance of the event. If a request log entry is damaged at the destination node, NetPro can then trace the request log back to the previous-hop node, where another damaged log entry may also be found. We continue this process in an iterative manner to track the fault to prior nodes until we finally deduce that the damaged log entry was caused by a specific malicious behavior of the tracked node, e.g., package tampering or truncating. The provenance of NetPro can help us explain why such an event was abnormal or why such a log entry was absent or damaged.

Using our algorithm, we will verify the data packets and log entries to identify an invalid node state and then retrieve the data from the prove-

* Corresponding author (email: liteng@stu.xidian.edu.cn)
The authors declare that they have no conflict of interest.

nance. This kind of attack detection usually requires that the participating nodes disclose details of routing policy or log data. Because of privacy concerns among a set of heterogeneous nodes, the hosts of MANET are reluctant to disclose this data. Recent work, e.g., PeerReview [2] and Y! [5], can detect faults without consideration of the issue of privacy. To deal with this conflict, we use the Merkle Hash Tree (MHT) and digital signature algorithm to protect the privacy of the nodes.

Provenance verification. Merkle Hash Tree (MHT) is a binary hash tree, each of whose nonleaf nodes are labeled with the hash of the labels or values of their child nodes, and each leaf node is labeled with the hash of valid node data. A MHT is commonly used to provide efficient and secure verification of the contents of large data structures. We design a variation of a MHT that constructs the node's log which is used to verify that the privacy is preserved. Beginning at the root of the MHT, the edges from each parent to its two children are tagged respectively with 0 and 1.

The purpose of verification is to ensure the expected actions or messages derived in the reasoning phase are satisfied by the real logs of related nodes. The process consists of three steps: predicate encoding, MHT building, and verification. Before building the tree, it is necessary to encode the log entries. Take the following predicates encoding for example. reqForward(@M, S_1, D_1, SEQ, STAUS = "NO", T) can be encoded by specifying S_1="01", D_1="0110", SEQ="010", STAUS(NO)="0". reachDest(@M, $S_2, D_2, \text{SEQ}, \text{STAUS}$ = "YES", T) can be encoded by specifying S_2 ="10", D_2 = "1000", SEQ ="101", STAUS(YES) ="1". How many bits the variables, such as S_1 and S_2, cost for the predicate encoding depends on the count of nodes and variables. Our goal is to distinguish each predicate on different nodes.

After we encode the predicate, e.g., reqForward(@$M, S_1, D_1, \text{SEQ}, \text{STAUS}$ = "NO", T) as a string, e.g., "0101100100", we use it to build the MHT. Each node will search its local log to locate the predicating log, which is used to construct the MHT. We label the node's left child as 1 and right child as 0. Then, we can calculate the hash value of each node i:

$$H_i = H((\text{bit\_data}_{(i)} \parallel \text{parent\_bit\_data}_{(i)}) \parallel H_{\text{left\_child}(i)} \parallel H_{\text{right\_child}(i)}). \quad (1)$$

$\text{bit\_data}_{(i)}$ and $\text{parent\_bit\_data}_{(i)}$ are respectively the value of current node and parent node, which can be either 0 or 1. $H_{\text{left\_child}(i)}$ and $H_{\text{right\_child}(i)}$ are respectively the hash value of the node(i)'s left child and right child. They will be empty if the current node is 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 node.

Provenance reasoning. To detect both direct and indirect attack, we first perform the detection as we move toward the destination. During the initial detection, we use the log entry of the trusted sender to deduce the expected log entry in the destination. After verifying the real log with the expected log on the destination, we can determine the nature of the attack against the MANET. If the log at the destination is not correct, NetPro performs the direct attack detection. We assume that the integrity of log located at the destination and the intermediate router's log are valid during the initial attack detection and also during the direct attack detection. If the log at the destination is correct, NetPro performs indirect attack detection to check if the intermediate routers have changed the content of their log. During this phase, we assume that the integrity of the log at the destination has not been violated.

If invalid log content or absent log events are discovered in the routing phase, we classify these events into positive and negative events. In the case of a positive event, such as an invalid log entry, we can apply the positive provenance to recursively derive a backward trace terminating the recursion when the original attacker is identified. The goal of positive provenance is to detect inconsistent or incorrect states within the faulty component. Negative provenance is used to explain why such an event is missing or absent, e.g., why such a routing request is not arrived on the destination or why the acknowledgement message has not appeared on the source node. Although these events cannot be directly explained by the positive provenance, we can also form a similar back trace that explains where the missing log or packets should have come according to the basic rules. The goal of negative provenance is to find out the ways that a missing event should have occurred and show the reason why it did not come to pass.

Evaluation. We apply NetPro to the multi-black hole occurrences detection. NetPro can automatically detect the invalid behavior and identify the malicious nodes. First, NetPro extracts the log from the nodes with the logging module as the files with a suffix of .pcap. Second, at the source node, NerPro uses its log with the corresponding rules to deduce the expected log on the target nodes (with the open source tool IRIS). Third, combining with the MHT, the source node conducts the provenance verification to confirm if the real log

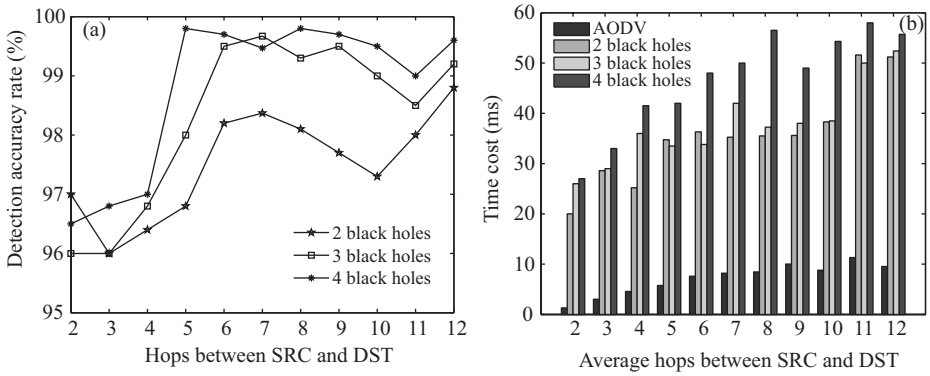

Figure 1 NetPro performance on multi-black holes detection. (a) Detection accuracy rate on multi-black holes; (b) time cost of multi-black holes detection.

entries match with the expected ones. Finally, according to our algorithms, NetPro returns the provenance graph for the whole deducing and verification phase which shows the exact malicious nodes and their attacking behaviors.

We inject multiple black holes into the MANET such that many of the nodes cooperate with other nodes to launch the attack. We compare the results generated by NetPro from two black hole environments with results generated by four black hole environments and we conclude that our method produces better results in handling multiple black holes. The coordinating malicious nodes cooperate with each other and each node assists the other nodes to deceive the source node. When the benignant nodes locate a malicious node, it is easy to identify the nodes that assisted the malicious node. The results in Figure 1(a) show that we gain good results when black holes arise. Figure 1(b) shows the time cost of detecting multiple black holes. An increase in the number of black holes will result in the increase in the detection time cost. We also notice that when detecting multi-black holes the detection accuracy rate drops due to the increase in the number of hops, from 5 hops to 12 hops. We believe that this phenomenon is related to the original AODV protocol's packet loss rate, because we know that the packet loss rate increased from 5 hops.

Conclusion. In this article, we propose an approach, NetPro, to automatically detect the direct and indirect attacks by using provenance in MANET. First, we use NDlog in reasoning expected log and then we check whether the destination has been influenced. Next, we conduct the direct attack detection or the indirect attack detection with the assist of MHT to preserve the privacy. Finally, NetPro returns a provenance graph to show the tracking trace to the malicious node. NetPro can explain why an event is faulty and why an expected event did not occur with the concept of the provenance. According to the algorithm of NetPro, we can detect which router is malicious and find the type of the attack without revealing any privacy of each node. Due to our experiment, NetPro is scalable and practical for use on real MANET routing security.

Acknowledgements This work was supported by National High Technology Research and Development Program of China (863) (Grant Nos. 2015AA017203, 2015AA016007), Key Program of National Natural Science Foundation of China (Grant No. U1405255), National Natural Science Foundation of China (Grant No. 61303033), Natural Science Basis Research Plan in Shaanxi Province of China (Grant No. 2016JM6034), Aviation Science Foundation of China (Grant Nos. 2013ZC31003, 20141931001), and Special Research Foundation of MIIT (Grant No. MJ-2014-S-37).

References

1 Nadeem A, Howarth M P. A survey of manet intrusion detection & prevention approaches for network layer attacks. IEEE Commun Surv Tut, 2013, 15: 2027–2045
2 Haeberlen A, Kouznetsov P, Druschel P. Peerreview: practical accountability for distributed systems. In: Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles, Stevenson, 2007. 175–188
3 Haeberlen A, Avramopoulos I C, Rexford J, et al. Netreview: detecting when interdomain routing goes wrong. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, Boston, 2009. 437–452
4 Jiang J, Li W, Luo J, et al. A network accountability based verification mechanism for detecting interdomain routing path inconsistency. J Netw Comput Appl, 2013, 36: 1671–1683
5 Wu Y, Zhao M, Haeberlen A, et al. Diagnosing missing events in distributed systems with negative provenance. ACM SIGCOMM Comput Commun Rev, 2015, 44: 383–394