Volume 5, Issue 4, 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Malware Detection and Mitigation Using Probabilistic Threat Propagation

B. Ranjani
P.G.Student ,
Department of Computer Science and Engineering,
A.V.C. College of Engineering, Mayiladuthurai.
Tamil, Nadu, India

J. Sudha
Associate Professor,
Department of Computer Science and Engineering,
A.V.C. College of Engineering, Mayiladuthurai.
Tamil, Nadu, India

Abstract— Security is important for many network applications. Gradually, new malwares become a great challenge for malware detectors. Malicious nodes are modelled as faulty nodes behaving intelligently to lead to an incorrect decision or energy depletion without being easily detected. This paper proposes PTP methodology to propagate trust across network. PTP method is used to detect malicious and infected nodes in both monitors and external internet. Given an initial set of known malicious node, prior detection for community is leveraged by propagating threat probabilities across network nodes. This work provides a way out to identify and mitigate malicious nodes in wireless and ad-hoc networks through message transmission between nodes/users in a network.

Keywords— Attacks, Community Detection, Malicious Node, Malware Detection, Network Security, Threat Propagation .

I. INTRODUCTION

Detection of malicious activity is an important distress for network defenders in the sphere of network security analysis. Prior methods of network anomaly detection have focused on defining a temporal model of what is "normal" and deteriorating the "abnormal" activity that does not fit into the construct. When monitoring traffic to and from IP addresses on a large network, this become computationally complex, and potentially inflexible, as a state model must be maintained for each address. Noise, faults and malicious nodes in the network are the potential sources for incorrect readings and reports. Contrasting noise and faults, malicious nodes can randomly modify the sensed data and purposely generate wrong reports. A reliable result detection of such wrong data and reports is ensured by detecting and isolating malicious nodes [1],[2] that greatly reduce the impact on decision-making.

In many situation, network defenders have been able to discover nodes of known maliciousness [6], either determined from previous detection mechanisms for internal nodes or from provided blacklists [11] for external nodes. Using these as tips, research can be performed on network traffic [5] to identify hosts communicating with these known malicious nodes. Malicious activity is highly localized in network spaces forming communities of maliciousness. It can be leveraged for the detection of malicious network nodes by defining locality as hosts that frequently communicate with one another or share a common set of resources.

In this paper, a method is presented for detecting malicious nodes on a network, independent of their individual activities, by leveraging the fact that malicious activity tends to be localized. Given a tip node of known maliciousness, a graph analysis is performed to calculate the threat probability of neighboring nodes, iterating until a statistical probability of threat has been computed and converged for each graph node. Simply say, the probability of a node being malicious is proportional to the level of maliciousness of its neighborhood.

II. RELATED WORK

Identifying members of a group from a network topology has been proposed in several community detection algorithms. The structure in the network is spotted in most network. Given the neighborhood of either node, the random variables in Markov Random Fields (MRF) model, as an undirected graph in which two non-neighboring nodes are conditionally independent from one another. This problem becomes a Conditional Random Field (CRF) when given an observation of a random variable or a tip in the graph. The flow direction should affect the beliefs about nodes in the graph in applications where 'threat' is being passed between nodes. The iterative approaches to the inference problem is taken in our current work when applied to threat networks, by propagating belief to/from a node based on the degree of those nodes. This method does not suffer from the adverse effect as it yields a true statistical probability output. To our knowledge, no other method is currently able to do this while maintaining asymmetric factors.

III. PROBLEM DEFINITION

Techniques for network security analysis have traditionally focused on the events of the network hosts. Strictly based on their association with other known malicious nodes, few works has been done to detect and predict malicious or infected nodes. This is referred to as community detection method as it is highly prevalent in the graph analytics world.

A .Threat Propagation On Graphs

Formally, we define the graph G = (X, E), where X represents the set of nodes on the graph and E is the set of edges between nodes. If there exists an edge $e_{ij} \in E$, then there exists some quantifiable direct relationship between nodes x_i and x_j in G. Additionally, the relationship power can be represented by weighing w_{ij} on e_{ij}. Here, the two communities of interest have been defined for the detection problem: malicious and benign. The probability of being in the malicious community as P(x) and the probability of being in the benign community is 1 − P(x). This probability is defined only with the observed relationships in the network (e.g. edges) and the priori known malicious entities.

B .Threat Provenance

Previous methods of iterative threat propagation on graphs do not track the provenance of the propagated threat thus leads to an inaccurate inference. Dampening factor or edge weights less than unity have been employed in previous works which reduce the amount of threat propagating between nodes. These methods have the outcome of exponentially decreasing the propagated threat level of any given node such that all connected nodes will not converge to the same value (typically 1). As a byproduct of this dampening, a probability can be used as a statistical prior for downstream analytics.

IV. PROPOSED WORK

A. Probabilistic Threat Propagation

Probabilistic Threat Propagation (PTP) method is proposed in order to accurately calculate the threat level at each node as well as to track the provenance of the propagated threat. For this reason, the level of threat at each node is equal to the weighted sum of the threat of neighboring nodes, discounting the level of threat those nodes received from the node of interest. Eq. (1) calculates the threat on node x_i as the probability of maliciousness mathematically as follow:

$$P(x_i; G) = \sum_{j \in N(x_i)} w_{ij} P(x_j | x_i = 0; G) \qquad (1)$$

PTP method is able to find the exact place of malicious node accurately and remove it. This method has been checked by passing messages between neighboring nodes in the system. The malicious node fabricates the message passed between nodes. Hence, the malicious node is detected and removed or make them as a honest/Benign node by giving correct key value to the administrator as shown in Fig. 1.

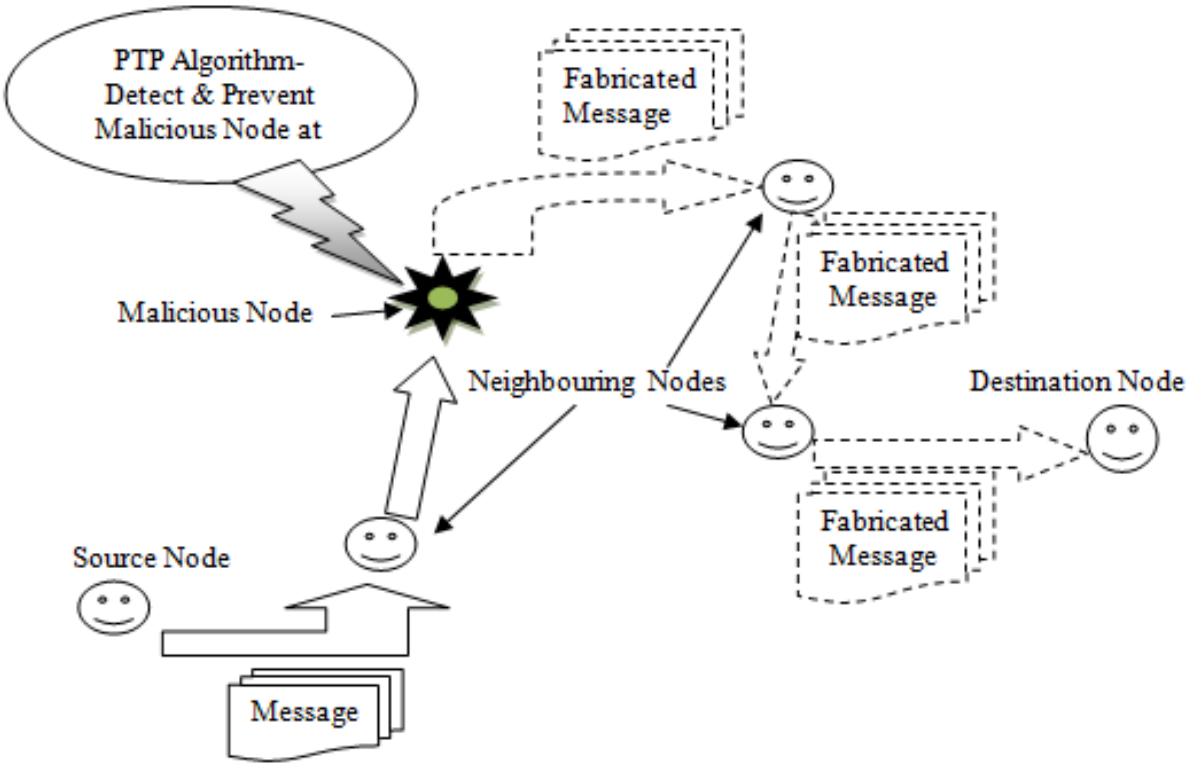

Fig. 1 System Architecture for PTP Method to Detect Malicious Node

The conditional probability of neighboring nodes in the absence of the node of interest is computed by conditioning on the current node being equal to 0. Distinctively, PTP is initialized with the set {tips}, those nodes which are known to be malicious, and assign them a priori probabilities $P(x \in \{tips\}) = \gamma$, where $\gamma \in [0, 1]$ relays the confidence in the tip. All other nodes are initialized to priors of $P(x \notin \{ti\ ps\}) = \alpha$, $0 \leq \alpha < \gamma$. At each iteration, the new probabilities using Eq. (1),is computed and $P(x \in \{tips\}) = \gamma$, is reassigned such that the tip nodes remain constant. This process continues until convergence of $P(x_i)$, $\forall i$.

B. Inference Methods

1) Exact Inference: Exact inference of Eq. (1) is possible for graphs with a single tip and no cycles. Any graph formulated beneath those conditions can be mapped as a tree graph with the tip acting as the root node. Under this mapping, threat propagates downward from the root node, through each node in the graph, stopping at the leaf nodes. Eq. (1) guarantees that threat never propagates back upwards by tracking the provenance.

2) Approximate Inference: Exact inference of Eq. (1) requires $P(x_j|x_i = 0)$ be computed for all pairs of nodes in the graph. Hence, an efficient approximation to Eq. (1) is developed which leverages our recursively defined probabilities. Mathematically, we solve iteratively over *k* as in Eq. (2):

$$P^k(x_i) = w_{ij}\left(P^{k-1}(x_j) - C^{k-1}(x_i, x_j)\right) \qquad (2)$$

where $P^k(x_i)$ is the probability of x_i at iteration *k*, and $C^{k-1}(x_i, x_j)$ is the portion of $P^{k-1}(x_j)$ which was directly computed from x_i in the previous iteration. This achieves our desire of removing the direct influence x_i has on x_j from feeding back to x_i and enables arbitrary and asymmetric weights w_{ij}.

V. PTP ALGORITHM

The prior belief that any node is malicious has been initialized to be $P^0(\{x_i\}) = \alpha$, $\forall i$. Here we set $\alpha = 0$. Eq. (2) is solved efficiently using linear algebra. Given a set of *N* nodes, the node threat probability vector $P \in R^N$ such that $P(i) = P(x_i)$, the weight matrix $W \in R^{N \times N}$ such that $W(i, j) = W_{ij}$, the transfer matrix $T \in R^{N \times N}$ such that $T(i, j) = W(i, j) * P(j)$, and the contribution matrix $C \in R^{N \times N}$, which we initialize to zeros. At each iteration, we assign $C = T - W \circ C^T$, where $A \circ B$ is the Hadamard product (e.g. $A(i, j) * B(i, j)$). This equation is the interior of the sum in Eq. (3), which computes each node's contribution to each other node, subtracting off the contribution from the previous iteration. To compute the node probabilities, we simply sum the elements of the newly computed C, $P = \langle C, \bar{1} \rangle$ where $\bar{1}$ is the *N*-element vector of ones and $< \cdot, \cdot >$ represents the inner product. Algorithm 1 details the full computation performed by PTP.

Algorithm 1 Probabilistic Threat Propagation

Require: W, {tips}, α
P ← , P({tips}) ←
C ← 0^N
repeat
T − W ⊗
C ← T − W ∘
P ← ⟨C,
C({tips}, ·) ←
P ← ({tips}) ←
until P has converged
return P

PTP on a tree graph performs exact inference and is equivalent to the belief propagation algorithm. This algorithm is often referred to as loopy belief propagation when it has been successfully used on general graphs. However, PTP in the general case is not equivalent to loopy belief propagation since converting a general graph to a factor graph for PTP is not straight forward as the edge weights are not necessarily symmetric. The weight matrix W is one of the necessary input to PTP. This is computed generically via the function $W_{ij} = f(x_i, x_j)$.

VI. IMPLEMENTATION

A. Flow Information Monitoring and Recording

The main objective of this module is to monitor and record the flow of packet information. The packets are passed through a router as flows. A flow is defined by pair information for the upstream router where the packet came from and the destination address of the packet. This module monitors the incoming packet flow and outgoing packet flow information. The flow of packet information is recorded to the temporary routing table in non attack duration time. Fig. 1. illustrates the above description.

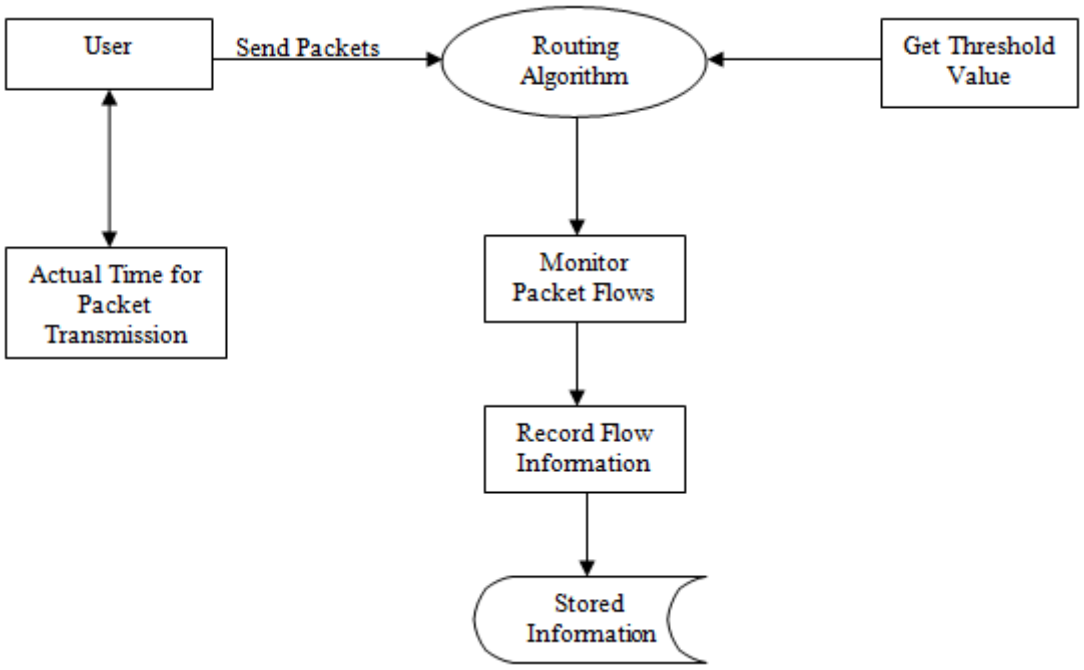

Fig. 2 Local Flow Monitoring and Recording

B. Analyzing the Flow Variation

The main objective of this module is to analyze the flow variation for each and every packet from the temporary routing table. A flow is defined by a pair the upstream router where the packet came from and the destination address of the packet. Entropy is a measure of randomness. Entropy variation is employed here to measure the changes of randomness of flows at a router for a given time interval. To identify the abnormality of Distributed Denial of Service (DDoS) attacks, change point of flows. Some flows increase significantly in a very short time period in DDoS attack cases at the time measure flow variation using entropy. The network traffic for a router may dynamically change a lot from peak to off-peak service times. This type of change lasts for a relatively long time interval. The change of traffic is quite smooth in our context if we break down these changes into seconds.

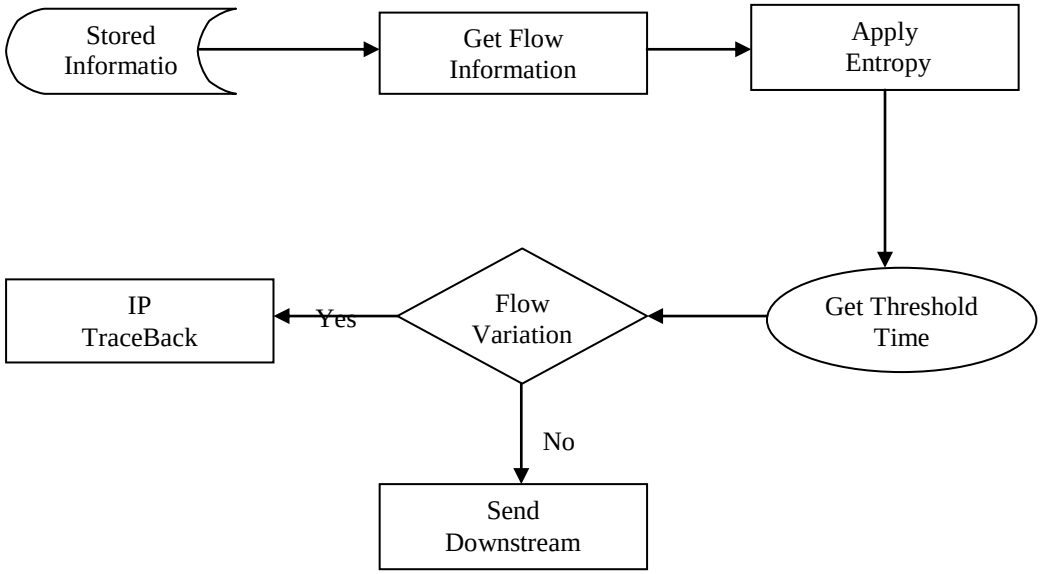

Fig. 3 Analyzing Flow Variation

C. Workload Distribution (IP Pushback Tracing)

PTP and IP TraceBack algorithm is used here to trace the actual source of attacker. The effectiveness and efficiency of the proposed entropy variation is based on IP traceback mechanism. The first task is to show that the flow entropy variation is stable for non attack cases and find out the fluctuations for normal situations. The second task is to demonstrate the relationship between the drop of flow entropy variation and the increase of attack strength so that we can identify the threshold for identifying attack sources. The whole attack tree for traceback is simulated and evaluation id done to calculate the total traceback time. At the non attack period, the local flow monitoring algorithm accumulates information from normal network flows and progress the mean and the standard variation of flows. The progressing suspends when a DDoS attack is ongoing.

Once a DDoS attack has been confirmed by any of the existing DDoS detection algorithms, then IP traceback algorithm installed at routers get started by the victim. At the upstream routers it is triggered by the IP traceback requests from the victim or the downstream routers which are on the attack path. As a result, we do not need to change the current routing software.

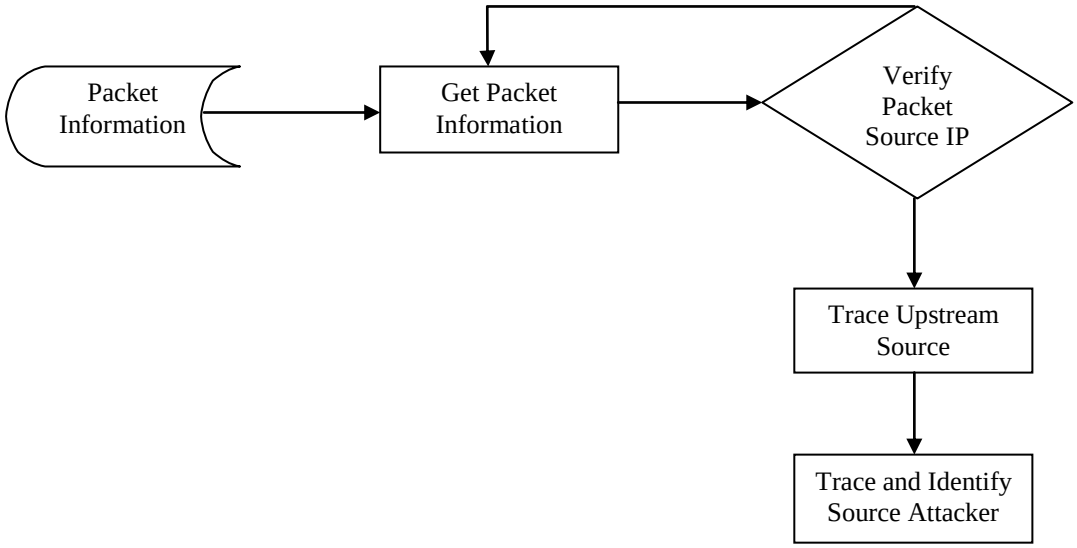

Fig. 4 IP PushBack Tracing

D. Identifying and Blocking the Attack Source

The main objective of this module is to identify the attack sources and also to block the source of an attack. The flow entropy variations information are observed and stored at routers. On identifying the attack by the victim via detection algorithms, the pushback tracing procedure is initiated. First it identifies the upstream routers where the attack flows came from, and then submits the traceback requests to the related upstream routers. This process is continued until it reaches the discrimination limitation of DDoS attack flows. Extensive experiments and simulations have been conducted, and the results demonstrate that the proposed mechanism works very well in terms of effectiveness and efficiency. Finally identified a source attacker and blocking the source attacker from the current network access.

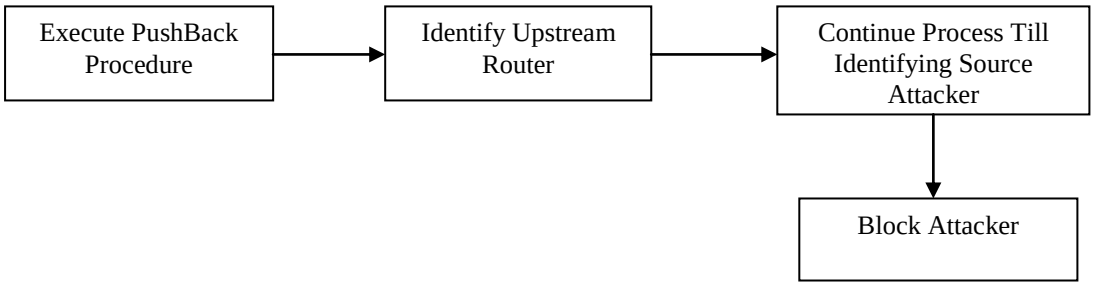

Fig. 5 Identify and Block Source Attacker

E. Evaluate the Total TraceBack Time

This module is used to find the total trace back time for identifying the source of an attacker. The effectiveness and efficiency of the proposed entropy variation based on IP trackback mechanism. Our first task is to show that the flow entropy variation is stable for non attack cases, and find out the fluctuations for normal situations; the second task is to demonstrate the relationship between the drop of flow entropy variation and the increase of attack strength so that we can identify the threshold for identifying attack sources. We further simulate the whole attack tree for traceback and evaluate the total traceback time. Consequently, in order to evaluate our scheme, we have carefully conducted extensive simulations and real case observations. The simulation settings are arranged according. We set the attack tree as a binary tree or three-branch tree, respectively, and zombies are distributed in the attack tree uniformly. We note that, our entropy variation traceback mechanism is independent from the topology of attack network and it is also independent from the network topology of victims, such as 5-10 minutes attack duration, 10,000 packets per second of attack flows.

The performance evaluation included two parts the first one focused on the entropy variation monitoring at a local router; and the second part was to demonstrate the effectiveness of DDoS attacker traceback and the overall traceback time. We conducted two simulations for this purpose: we make the mean of flows as 100 packets per time unit, and the standard variations of the flows are 25 and 50 respectively. We observe the changes of the entropy variation against the number of flows. Moreover, the standard variation of entropy variation decreases when the number of flows increases.

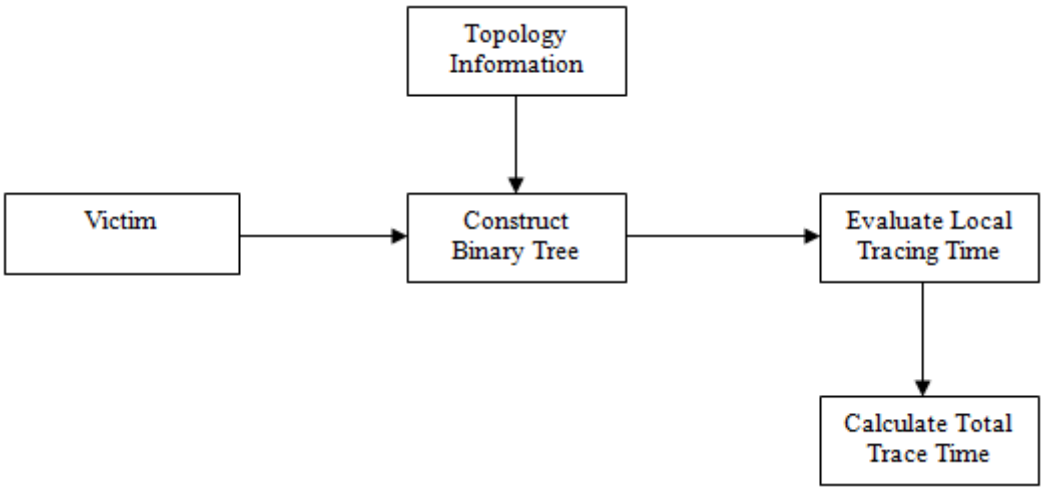

Fig. 6 Evaluate Total TraceBack Time

VII. CONCLUSIONS

Probabilistic Threat Propagation, an iterative graph analytic determines the statistical probability that a node in an observed network is malicious conditioned on the knowledge of some malicious entities. Tracking threat provenance to avoid node threat levels is critical. PTP performs exact inference and is equivalent to the belief propagation algorithm in case of acyclic graph. In general, PTP outputs approximations of true statistical probabilities that are easily interpretable by an analyst. PTP is used in network security applications of botnet detection and prediction of malicious domains. PTP generalizes for detection of insurgent cells and malicious email communities. In future, work is done to propagate trust across network components to measure the trustworthiness of an incoming connection request. Different edge weighting functions are analyzed, which becomes critical for temporal analyses. At-last, during symmetric weights, the relationship between loopy belief propagation and PTP is determined.

REFERENCES

[1] M. Roesch, "*SNORT—Lightweight intrusion detection for networks*," in *Proc.13th LISA Conf.*, 1999, pp. 229–238.

[2] M. P.Collins and M. K. Reiter, "*On the limits of payload-oblivious network attack detection*," in *Proc. 11th Int. Symp. Recent Adv. Intrusion Detection (RAID)*, 2008, pp. 251–270.

[3] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "*BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection*," in *Proc. 17th USENIX Security Symp.*, 2008, pp. 139–154.

[4] G. Gu, J. Zhang, and W. Lee, "*BotSniffer: Detecting botnet command and control channels in network traffic*" in *Proc. 15th Annu. Netw. Distrib. Syst. Security Symp. (NDSS)*, Feb. 2008.

[5] M. P. Collins *et al.*, "*Using uncleanliness to predict future botnet addresses*," in *Proc. 7th ACM SIGCOMM Conf. Internet Meas.*, 2007, pp. 93–104.

[6] T. Yu, R. Lippmann, J. Riordan, and S. Boyer, "*EMBER: A global perspective on extreme malicious behavior*," in *Proc. 7th Int. Symp. Vis. Cyber Security*, 2010, pp. 1–12.

[7] R. Kindermann and J. Snell, *Markov Random Fields and Their Applications*, Providence, RI, USA: AMS, 1980.

[8] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "*Identifying suspicious URLs: An application of large-scale online learning*," in *Proc. 26th Int. Conf. Mach. Learn. (ICML)*, 2009, pp. 681–688.

[9] A. S. Yoshiro Fukushimayz, Y. Horiyz, and K. Sakuraiyz, "*A Behavior Based Malware Detection Scheme for avoiding False Positive*", IEEE, (2010), pp. 6.

[10] A. A. E. Elhadi, M. A. Maarof and A. H. Osman, "*Malware detection based on hybrid signature behavior application programming interface call graph*", American Journal of Applied Sciences, vol. 9, (2012), pp. 283-288.

[11] J. Zhang, P. Porras, and J. Ullrich, "*Highly predictive blacklisting,"* in *Proc. 17th Conf. Security Symp.*, 2008

