International Journal of Advanced Computational Engineering and Networking, ISSN: 2320-2106, Volume-2, Issue-12, Nov.-2014

NETWORK INTRUSION NODE CLONE DETECTION IN VIRTUAL NETWORK SYSTEMS

[1]R.ASHOK, [2]S.KOKILA, [3]M.GIRI

[1]Student, Dept of Software Engineering, Sreenivasa Institute of Technology And Management Studies, Chittoor, India
[2]Assistant Professor, Dept of Computer Science & Engineering, Sreenivasa Institute of Technology And Management Studies, Chittoor, India
[3]HOD, Professor, Dept of Computer Science & Engineering, Sreenivasa Institute of Technology And Management Studies, Chittoor, India
E-mail: ashokrreddy1205@gmail.com, kokila.s521@gmail.com, prof.m.giri@gmail.com

Abstract- This is because cloud users may install in danger applications on their virtual equipment. To prevent vulnerable virtual machines from being compromise in the cloud, a multi-phase scattered vulnerability exposure, measurement, and counter measure selection mechanism called NICE, which is built on attack graph based analytical models and reconfigurable virtual network-based counter measures. The proposed framework leverages Open Flow network programming APIs to build monitor and control plane over distributed programmable virtual switches in order to significantly improve attack detection and mitigate attack consequences. The system and security evaluations exhibit the efficiency and effectiveness of the proposed Solution. In this paper, we propose two novel node clone detection protocols with different tradeoffs on network conditions and performance. The first one is based on a distributed hash table (DHT), by which a fully decentralized, key-based caching and checking system is raise to catch cloned nodes effectively. The protocol performance on efficient storage consumption and high security level is theoretically deducted through a probability model, and the resulting equations, with necessary adjustments for real application, are supported by the simulations. Although the DHT-based protocol incurs similar communication cost as preceding approaches, it may be considered a little high for some scenarios. To address this concern, our second distributed detection protocol, named randomly directed exploration, presents good communication performance for dense sensor networks, by a probabilistic directed forwarding technique along with random initial direction and border determination. The simulation results uphold the protocol design and show its efficiency on communication overhead and satisfactory detection probability.

Keywords- NICE, DTH, Sensor Networks

I. INTRODUCTION

In this article, we propose NICE (Network Intrusion detection and Counter measures Election in virtual network systems) to establish a defense-in-depth intrusion detection framework. For better attack detection, NICE incorporates attack graph analytical procedures into the intrusion detection processes. We must note that the design of NICE does not intend to improve any of the existing intrusion detection algorithms; indeed, NICE employs a reconfigurable virtual networking approach to detect and counter the attempts to compromise VMs, thus preventing zombie VMs. In general, NICE includes two main phases: deploy lightweight mirroring-based network intrusion detection agent (NICE-A) on each cloud server to capture and analyze cloud traffic. A NICE-A periodically scans the virtual system vulnerabilities within a cloud server to establish Scenario Attack Graph (SAGs), and then based on the severity of identified vulnerability towards the collaborative attack goals, NICE will decide whether or not to put a VM in network inspection state. (2) Once a VM enters assessment state, Deep Packet Inspection (DPI) is applied, and/or virtual network reconfigurations can be deployed to the inspecting VM to make the potential attack behaviors prominent. In this paper, we present two novel, practical node clone detection protocols with different tradeoffs on network conditions and performance. The first proposal is based on a distributed hash table (DHT), by which a fully decentralized, key-based caching and checking system is constructed to catch cloned nodes. The protocol's performance on memory consumption and a critical security metric are theoretically deducted through a probability model, and the resulting equations, with necessary adjustment for real application, are supported by the simulations. In accordance with our analysis, the comprehensive simulation results show that the DHT-based protocol can detect node clone with high security level and holds strong resistance against adversary's attacks. Our second protocol, named randomly directed exploration, is intended to provide highly efficient communication performance with adequate detection probability for dense sensor networks. In the protocol, initially nodes send claiming messages containing a neighbor-list along with a maximum hop limit to randomly selected neighbors; then, the subsequent message transmission is regulated by a probabilistic directed technique to approximately maintain a line property through the network as well as to incur sufficient randomness for better performance on communication and resilience against adversary. In count border determination machinery is employed to further reduce communication payload. During forwarding, intermediate nodes explore claiming messages for node clone detection.

By design, this protocol consumes almost minimal memory, and the simulations show that it outperforms all other detection protocols in terms of communication cost, while the detection probability is satisfactory.

II. NICE MODELS

In this section, we describe how to utilize attack graphs to model security threats and vulnerabilities in a virtual networked system, and propose a VM protection model based on virtual network reconfiguration approaches to prevent VMs from being exploited.

Threat Model

In our attack model, we assume that an attacker can be located either outside or inside of the virtual networking system. The attacker's primary goal is to exploit vulnerable VMs and compromise them as zombies. Our protection model focuses on virtual-network-based attack detection and reconfiguration solutions to improve the resiliency to zombie explorations. Our work does not involve host-based IDS and does not address how to handle encrypted traffic for attack detections. Our proposed solution can be deployed in an Infrastructure-as-a-Service (IaaS) cloud networking system and we assume that the Cloud Service Provider (CSP) is benign. We also assume that cloud service users are free to install whatever operating systems or applications they want, even if such action may introduce vulnerabilities to their controlled VMs. Physical security of cloud server is out of scope of this paper.

Attack Graph Model

An attack graph is a modeling tool to illustrate all possible multi-stage, multi-host attack paths that are crucial to understand threats and then to decide appropriate Counter measures. In an attack graph, each node represents either precondition or consequence of an exploit. The actions are not necessarily an active attack since normal protocol interactions can also be used for attacks.

Attack graph is helpful in identifying potential threats, possible attacks and known vulnerabilities in a cloud system.

Since the attack graph provides details of all known vulnerabilities in the system and the connectivity information, we get a whole picture of current security situation of the system where we can predict the possible threats and attacks by correlating detected events or activities. If an event is recognized as a potential attack, we can apply specific countermeasures to mitigate its impact or take actions to prevent it from contaminating the cloud system. To represent the attack and the result of such actions, we extend the notation of MulVAL logic attack graph as presented by X. Ou et al. and define as Scenario Attack Graph (SAG).

Definition (Alert Correlation Graph). An ACG is a three tuple ACG =(A, E, P), where
1. A is a set of aggregated alerts. An alert a ∈ A is a data structure (src, dst, cls, ts) representing source IP address, destination IP address, type of the alert, and timestamp of the alert respectively.
2. Each alert a maps to a pair of vertices (v) in SAG using map(a) function, i.e., map(a):a→{(vc,vd)|(a.src ∈ vc.Hosts) ∧ (a.dst ∈ v.Hosts) ∧(a.cls = v.vul)}.
3. E is a set of directed edges representing correlation c between two alerts (a, a) if criteria below satisfied:
i. (a.ts<a.ts) ∧ (a.ts - a.ts < threshold)
ii. ∃(vd,vc) ∈ Epre:(a.dst ∈ vdd.Hosts ∧ a.src∈v.Hosts)
4. P is set of paths in ACG. A path Sc⊂ P is a set of related alerts in chronological order.

We assume that A contains aggregated alerts rather than raw alerts. Raw alerts having same source and destination IP addresses, attack type and timestamp within a specified window are aggregated as Meta Alerts. Each ordered pair (a, a) in ACG maps to two neighbor vertices in SAG with timestamp difference of two alerts within a pre defined threshold. ACG shows dependency alerts in chronological order and we can find related alerts in the same attack scenario by searching the alert path in ACG. A set P is used to store all paths from root alert to the target alert in the SAG, and each path ⊂ P represents alerts that belong to the same attack scenario.

Algorithm 1 Alert_Correlation

Require: alert a_c, SAG, ACG
1: **if** (a_c is a new alert) **then**
2: create node a_c in ACG
3: $n_1 \leftarrow v_c \in map(a_c)$
4: **for all** $n_2 \in parent(n_1)$ **do**
5: create edge $(n_2.alert, a_c)$
6: **for all** S_i containing a **do**
7: **if** a is the last element in S_i **then**
8: append a_c to S_i
9: **else**
10: create path $S_{i+1} = \{subset(S_i, a), a_c\}$
11: **end if**
12: **end for**
13: add a_c to $n_1.alert$
14: **end for**
15: **end if**
16: **return** S

III. PROPOSED MODEL

DHT-BASED DETECTION PROTOCOL

The principle of our first distributed detection protocol is to make use of the DHT mechanism to form a decentralized caching and checking system that can effectively detect cloned nodes. Essentially,

DHT enables sensor nodes to distributive construct an overlay network upon a physical sensor network and provides an efficient key-based routing within the spread over the surface network. A message associated with a key will be transmitted through the overlay network to reach a destination node that is solely determined by the key; the source node does not need to specify or know which node a message's destination is the DHT key-based routing takes care of transportation details by the message's key. More significantly, messages with a same key will be stored in one destination node. Those facts build the foundation for our first exposure protocol. As a beginning of a round of DHT-based clone detection, the initiator broadcasts the action message including a random seed. Then, every witness constructs a claiming message for each neighbor node, which is referred to as an examinee of the observer and the message, and sends the message with probability independently. The introduction of the claiming probability is intended to reduce the communication overwork in case of a high-node-degree network. In the protocol, a message's DHT key that determines its routing and destination is the hash value of concatenation of the seed and the examinee ID. By means of the DHT mechanism, a claiming message will eventually be transmitted to a deterministic destination node, which will cache the ID-location pair and check for node clone detection, acting as an inspector. In addition, some intermediate nodes also behave as inspectors to improve resilience against the adversary in an efficient way.

Protocol Details

As a requirement, all nodes willingly build a Chord overlay network over the sensor network. Cloned node may not participate in this procedure, but it does not give them any advantage of avoiding detection. The structure of the cover network is independent of node clone detection. As a result, nodes possess the information of their direct predecessor and successor in the Chord ring. In addition, each node caches information of its g consecutive successors in its successors table. Many harmony systems develop this kind of cache machinery to reduce the communiqué cost and increase systems strength. More importantly in our protocol, the facility of the successors table contributes to the economical selection of inspectors. One detection round consists of three stages.

Stage 1: Initialization

To activate all nodes starting a new round of node clone detection, the architect uses a broadcast authentication scheme to release an action message including a monotonously increasing nonce, a random round seed, and an action time. The nonce is intended to prevent adversaries from launching a DoS attack by repeating broadcasting action messages.

The action message is defined by

$$M_{\mathrm{ACT}} = \mathrm{nonce}, \mathrm{seed}, \mathrm{time}, \{\mathrm{nonce} \parallel \mathrm{seed} \parallel \mathrm{time}\}_{K^{-1}_{\mathrm{initiator}}}$$

Stage 2: Claiming neighbors information

Upon receiving an action message, a node verifies if the message nonce is greater than last nonce and if the message signature is valid. If both pass, the node updates the nonce and stores the seed. At the designated action time, the nodule operates as an observer that generates a claiming message for each neighbor (examinee) and transmits the message through the overlay network with respect to the claiming probability P_c. The claiming message by observer for examinee is constructed by

$$M_{\alpha\beta} = \mathrm{id}_\beta, L_\beta, \mathrm{id}_\alpha, L_\alpha, \{\mathrm{id}_\beta \parallel L_\beta \parallel \mathrm{id}_\alpha \parallel L_\alpha \parallel \mathrm{nonce}\}_{K_\alpha^{-1}}$$

where are locations of and , respectively. Nodes can start transmitting claiming messages at the same time, but then huge traffic may cause serious interference and degrade the network capacity. To mitigate this predicament, we may specify a sending period, during which nodes randomly pick up a conduction time for every claim message.

Stage 3: Processing claiming messages

A claiming message will be forwarded to its destination node via several Chord intermediate nodes. Only those nodes in the overlay network layer (i.e., the source node, Chord halfway nodes, and the destination node) need to process a message, whereas other nodes along the path simply route the message to temporary targets. Algorithm 1 for handling a message is the kernel of our DHT-based detection protocol. If the algorithm returns NIL, then the message has arrived at its destination. Otherwise, the message will be subsequently forwarded to the next node with the ID that is returned by Algorithm 1.

Criteria of determining inspectors: During handling a message in Algorithm 1, the node acts as an inspector if one of the following conditions is satisfied.

Algorithm 1: dht_handlemessage($M_{\alpha\beta}$): handle a message n the DHT-based detection, where y is the current node's Chord coordinate, finger$[i]$ is the first node on the ring that succeeds key $((y + 2^{h-i}) \bmod 2^h), i \in [1, t]$, successors$[j]$ is he next jth successor, $j \in [1, g]$

Output: NIL if the message arrives at its destination; otherwise, it is the ID of the next node that receives the message in the Chord overlay network

```
1: key ⇐ H(seed || id_β)
2: if key ∈ (predecessor, y] then {has reached destination }
3:     inspect(M_αβ) {act as an inspector, see Algorithm 2}
4:     return NIL
5: for i = 1 to g do
6:     if key ∈ (y, successors[i]] then {destination is in the
       next Chord hop}
7:         inspect(M_αβ) {act as an inspector, see
           Algorithm 2}
8:         return successors[i]
9: for j = 1 to t do {for normal DHT routing process}
10:    if key ∈ [(y + 2^{h-i}) mod 2^h, y) then
11:        return finger[j]
12: return successors[g]
```

Algorithm 2: inspect($M_{\alpha 4\beta}$): Inspect a message to check for clone detection in the DHT-based detection protocol

```
1: verify the signature of M_{α4β}
2: If id_β found in cache table then
3:     If id_β has two distinct locations {found clone,
       become a witness}
4:         broadcast the evidence
5: else
6:     buffer M_{α4β} into cache table
```

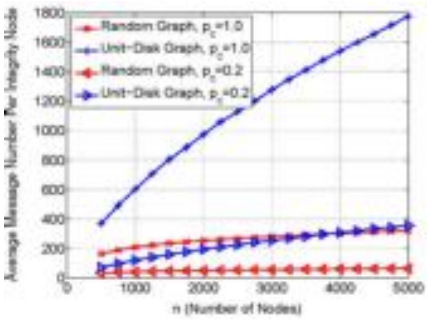
1) This node is the destination node of the claiming message.
2) The destination node is one of the successors of the node.
In other words, the destination node will be reached in the next Chord hop.

While the first criterion is intuitive, the second one is subtle and critical for the protocol performance. By Algorithm 1, roughly g/g+1of all claiming messages related to a same examinee's ID will pass through one of the predecessor of the destination. Thus, those g nodes are much more likely to be able to detect a clone than randomly selected inspectors. As a result, this criterion to decide inspectors can increase the average number of witnesses at a little extra memory cost. We will theoretically quantify those performance measurements later. In Algorithm 1, to examine a message for node clone detection, an inspector will invoke Algorithm 2, which compares the message with previous inspected messages that are buffered in the cache table. Naturally, all records in the cache table should have dissimilar examinee IDs, as implied in Algorithm 2. If detecting a clone, which means that there exist two messages $M_{\alpha 4\beta}$ and $M_{\alpha' 4\beta'}$ satisfying $\mathrm{id}_\beta = \mathrm{id}_{\beta'}$ and, $L_\beta \neq L_{\beta'}$ the witness node then broadcasts the evidence $M_{\text{evidence}} = (M_{\alpha 4\beta}, M_{\alpha' 4\beta'})$ to notify the whole network. All integrity nodes verify the evidence message and stop communicating with the cloned nodes. To avert cloned nodes from joining the arrangement in the future, a revocation list of compromised nodes IDs may be maintained by nodes individually.

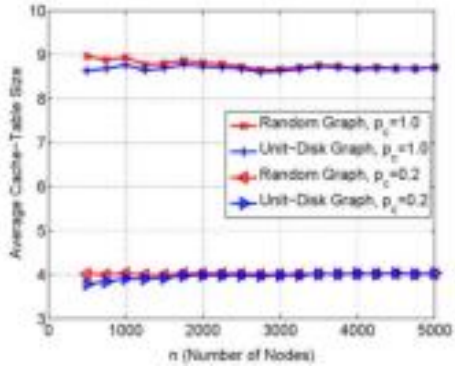
It is worth noting that messages $M_{\alpha 4\beta}$ and $M_{\alpha' 4\beta'}$ are legitimate by observers and, respectively. Therefore, the witness does not need to sign the evidence message. If a malicious node tries to launch a DoS attack by broadcasting a bogus evidence message, the next integrity node receiving it can immediately detect the bad performance by verifying the signatures of $M_{\alpha 4\beta}$ and $M_{\alpha' 4\beta'}$ before forwarding to other nodes.

Performance Analysis of DTH-Based Protocol
For the DHT-based detection protocol, we use the following specific measurements to evaluate its concert:

• Average number of transmitted messages, representing the protocol's communication cost;
• Average number of witnesses, serving as the protocol's security level because the detection protocol is deterministic and symmetric;
• Average size of node cache tables, standing for the protocol's storage consumption;



Communication cost



Storage consumption

Remarks: In the reality of the DHT-based protocol, each examinee, on average, has claiming messages. Thus, we can use $m = dp_c$. However, the transmissions of claiming messages associated with a same examinee's ID actually converge over the Chord key-based routing system. For a specific status of a Chord system, each segment has a different length, which would affect the hitting probability. In the ideal case, we take into account one level of such unbalanced segment.

Therefore, if all messages transmissions can be completed in two hops(that is, the first hop can reach the destination or one of its predecessors), the ideal case analyze is directly applicable. However, most messages shall go through more than two hops; then, each preceding hop transmission will introduce correlation on the next hop transmission. Accordingly they are not independent, and it is very difficult to directly model and analyze performance for the multilevel effect. The average size of cache tables for integrity nodes and the average witness number are illuminated in Fig. 2(c) and (d), respectively,

Which obviously indicates that those two performance

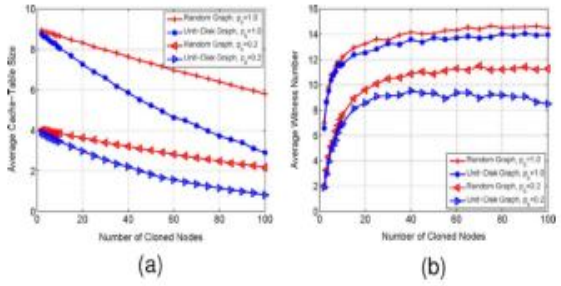
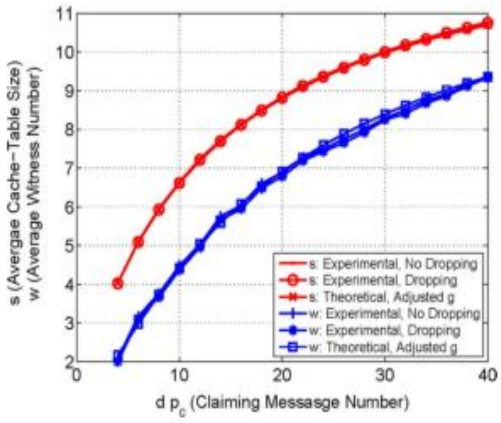


Fig. 3. Simulation results of the DHT-based detection on different numbers of cloned nodes, which discard all messages passing through them; $n = 1000$. (a) Storage consumption. (b) Security level.

Verification of Performance Analysis

To evaluate its applicability of the theoretical analysis on cache-table size and witness number in Section V-B, we carry out a third simulation, in which $n = 2000$, $d = 40$, there are two



ig. 4. Simulation results for verifying performance analysis of the DHT-based etection, where g is adjusted by 80% in the theoretical calculation.

cloned nodes, and the claiming probability increases from 10% to 100%. Since network scenarios do not affect the results on the two performance metrics, we run the simulation only on random graph. For the purpose of comparison, we test the performance for both cases that cloned nodes drop messages and comply with protocol (no-dropping).

CONCLUSION

Sensor nodes lack tamper-resistant hardware and are subject to the node clone attack. In this paper, we present two distributed detection protocols: One is based on a distributed hash table, which forms a Chord overlay network and provides the key-based routing, caching, and checking services for clone discovery and the other uses probabilistic is directed technique to achieve well-organized communication overhead for satisfactory detection probability. While the DHT-based protocol provides high security level for all kinds of sensor networks by one Deterministic witness and additional recollection-efficient, probabilistic witness, the randomly directed exploration presents outstanding communication performance and minimal storage consumption for dense sensor networks.

REFERENCES

[1] B. Parno, A. Perrig, and V. Gligor, “Distributed detection of node replication attacks in sensor networks,” in Proc. IEEE Symp. Security Privacy, 2005, pp. 49–63.

[2] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “LookingupdatainP2Psystems,”Commun. ACM,vol.46,no.2,pp. 43–48, 2003.

[3] Y. Zhang, W. Liu, W. Lou, and Y. Fang, “Location-based compromisetolerant security mechanisms for wireless sensor networks,” IEEE J. Sel. Areas Commun., vol. 24, no. 2, pp. 247–260, Feb. 2006.

[4] S.Zhu, S.Setia, and S.Jajodia, “LEAP: Efficient security mechanisms for large-scale distributed sensor networks,” in Proc.10thACMCCS, Washington, DC, 2003, pp. 62–72.

[5] R. Anderson, H. Chan, and A. Perrig, “Key infection: Smart trust for smart dust,” in Proc. 12th IEEE ICNP, 2004, pp. 206–215.

[6] M. Conti, R. D. Pietro, L. V. Mancini, and A. Mei, “A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks,” in Proc. 8th ACM MobiHoc, Montreal, QC, Canada, 2007, pp. 80–89.

[7] B. Zhu, V. G. K. Addada, S. Setia, S. Jajodia, and S. Roy, “Efficient distributed detection of node replication attacks in sensor networks,” in Proc.23rdACSAC, 2007, pp. 257–267.

[8] H.Choi,S.Zhu,andT.F.LaPorta,“SET:Detectingnodeclonesin sensor networks,” in Proc. 3rd SecureComm, 2007, pp. 341–350.

[9] R. Brooks, P. Y. Govindaraju, M. Pirretti, N. Vijaykrishnan, and M. T.Kandemir, “On the detection of clones in sensor networks using random key predistribution,” IEEE Trans. Syst.s, Man, Cybern. C,Appl. Rev., vol. 37, no. 6, pp. 1246–1258, Nov. 2007.

[10] L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks,” in Proc. 9th ACM Conf. Comput. Commun. Security, Washington, DC, 2002, pp. 41–47.

[11] A. Shamir, “Identity-based cryptosystems and signature schemes,” in Proc. CRYPTO, 1984, LNCS 196, pp. 47-53.

[12] R. Poovendran, C. Wang, and S. Roy, Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks.New York: Springer-Verlag, 2007.

[13] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” IEEE Commun. Mag., vol. 40, no. 8, pp. 102–114, Aug. 2002.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, “A scalable content-addressable network,” in Proc. SIGCOMM,San Diego, CA, 2001, pp. 161–172.

[15] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F.Dabek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup protocol for internet applications,” IEEE/ACM Trans. Netw., vol. 11,no. 1, pp. 17–32, Feb. 2003.

[16] A. I. T. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” in Proc. IFIP/ACM Int. Conf. Distrib. Syst. Platforms Heidelberg, 2001,

pp. 329–350.

[17] A. Varga and R. Hornig, “An overview of the OMNeT++ simulation environment,” in Proc. 1st Int. Conf. Simulation Tools Tech. Commun., Netw. Syst. Workshops, Marseille, France, 2008, pp. 1–10.

[18] A. Awad, C. Sommer, R. German, and F. Dressler, “Virtual cord protocol (VCP): A flexible DHT-like routing service for sensor networks,”in Proc. 5th IEEE MASS, 2008, pp. 133–142.

[19] R. Diestel, Graph Theory, 3rd ed. New York: Springer, 2006.

★★★