Real-Time Lateral Movement Detection Based on Evidence Reasoning Network for Edge Computing Environment

Zhihong Tian, Wei Shi, Yuhang Wang, Chunsheng Zhu, Xiaojiang Du, Shen Su, Yanbin Sun, and Nadra Guizani

***Abstract*—Edge computing provides high-class intelligent services and computing capabilities at the edge of the networks. The aim is to ease the backhaul impacts and offer an improved user experience. However, the edge artificial intelligence exacerbates the security of the cloud computing environment due to the dissociation of data, access control, and service stages. In order to prevent users from carrying out lateral movement attacks in an edge-cloud computing environment, in this paper we propose a real-time lateral movement detection method, named CloudSEC, based on an evidence reasoning network for the edge-cloud environment. First, the concept of *vulnerability correlation* is introduced. Based on the vulnerability knowledge and environmental information of the network system, the evidence reasoning network is constructed, and the lateral movement reasoning ability provided by the evidence reasoning network is then used. The experiment results show that CloudSEC provides a strong guarantee for the rapid and effective evidence investigation, as well as real-time attack detection.**

***Index Terms*—Cloud computing, correlation, edge artificial intelligence, lateral movement, network security.**

Manuscript received February 13, 2019; revised March 14, 2019; accepted March 18, 2019. Date of publication March 27, 2019; date of current version July 3, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61871140, Grant 61572153, Grant 61702220, Grant 61702223, and Grant U1636215, and in part by the National Key Research and Development Plan under Grant 2018YFB0803504. Paper no. TII-19-0462. *(Corresponding author: Shen Su.)*

Z. Tian, Y. Wang, S. Su, and Y. Sun are with the Cyber Space Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China (e-mail: tianzhihong@gzhu.edu.cn; apple125110@126.com; johnsuhit@gmail.com; sunyanbin@gzhu.edu.cn).

W. Shi is with the School of Information Technology Faculty of Engineering and Design, Carleton University, Ottawa, ON K1S 5B6, Canada, and also with the University of Ontario Institute of Technology, Oshawa, ON L1G 0C5, Canada (e-mail: wei.shi@carleton.ca).

C. Zhu is with the Department of Mathematics, Statistics, and Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada (e-mail: chunsheng.tom.zhu@gmail.com).

X. Du is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: dxj2005@gmail.com).

N. Guizani is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: guizani@gonzaga.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2907754

I. Introduction

CLOUD computing services are critical information platform, application, and infrastructure resources that users access via Internet [1]–[3]. These services, offered by companies such as Google and Amazon, enable customers to leverage powerful computing resources that would otherwise be beyond their means to purchase and maintenance. The success of the cloud computing has already changed the appearance of the world information infrastructure.

In order to ease the backhaul impacts and offer an improved user experience, edge computing is designed to share the load of cloud center and provides an IT service environment and computing capabilities at the edge of the network [4], [5]. The environment of edge computing is characterized by low latency, proximity, and high bandwidth. Combined with the traditional cloud computing environment, the edge computing enables innovative services such as e-Health, augmented reality, smart camera, gaming, and industry automation [6].

The "edge-cloud" model extends the complexity of cloud services [7]–[11]. More authorities have to be transferred to the edge participants, and more stages and data interactions will be involved during the procedure of services [12]–[15]. As a result, it turns out to be impractical to prevent all intrusions. Thus, monitoring the operation of a system is essential to its security. To that end, researchers collect monitoring information to estimate the current state of the system based on the usage patterns, detect whether there exist intrusions, and drive response actions.

Lateral movement techniques are frequently used to launch cyber-attack, especially in the hierarchical architecture system. In the edge-cloud environment, two main challenges are involved in the lateral movement detection.

1) Data persistence in edge computing may be different with the traditional scenario; there is less certainty in where data originated from, whether the data would be persistent, and where it will be stored. Since the edge nodes are often the memory limited and computationally constrained devices. The traditional lateral movement detection methods that need a significant manual effort and business correlated knowledge lack the opportunity to flourish in such a scenario.

2) The underlying architecture of the edge-computing environment is often dynamic, for example, the vehicular frog computing system based on the vehicle ad hoc networks. As a result, the lateral movement detection methods that rely on detecting changes in node behaviors may not be applied successfully on the edge environment.

Existing solutions are with few considerations for such a scenario. When a host is discovered to be compromised, there are a couple of fundamental questions that should always be asked: What was the route this attack traverse? How was the movement possible? What was the end target? What controls were executed to make the threat persistent? In order to meet these forensic requirements, the lateral movement detection method should be applied to the edge-cloud computing environments.

In this paper, we present a lateral movement detection method based on evidence reasoning network (ERN) for the edge-cloud computing environment, referred to as CloudSEC. CloudSEC introduced the concept of vulnerability correlation and built an ERN based on network system vulnerabilities and environmental information. Experimental results show that CloudSEC supplies a complete and credible evidence chain, and it also has the capacity of real-time lateral movement reasoning. These provide a strong guarantee for rapid and effective evidence investigation. The contributions of CloudSEC are threefold: First, it offers a list of concrete evidence on the detected attack, which gives more confidence to the cloud service provider. Second, it enables the cloud service provider to be fully aware of the consequences of an attack. Finally, it enables the cloud service provider to better determine an appropriate course of actions that need to be taken.

The remainder of this paper is organized as follows. In Section II, we present related previous works, such as event correlation in a virtual machine (VM). In Section III, we introduce the overall structure of our method and present the details of our proposed lateral movement detection method and its correlation algorithms. The experiments conducted to evaluate our method are discussed in Section IV. Section V concludes this paper with a summary and discussion of future research directions.

II. Related Works

Up to now, there have been several proposed techniques of analyzing android malware cloud data [16]–[20]. In this paper, we focus exclusively on the related works using forensic techniques on monitoring VMs in a cloud computing environment.

Most mature forensic investigation tools such as EnCase [21] and Safeback [22] focus on capturing and analyzing evidence from media stores on a single host. Mnemosyne is a dynamically configurable advanced packet capturing application that supports multistream capturing, sliding-window-based logging to conserve space, and query support on collected packets [23]. The evidence graph network forensic analysis mechanism [24], [25] includes effective evidence presentation, manipulation, and automated reasoning. Although it is nice to present evidence correlation in a graphic mode, this system is still a prototype and lacks the effective capability of inference. Tian *et al.* [26], [27] developed a network intrusion forensic system based on a transductive scheme and Dempster–Shafer theory that can efficiently detect and analyze computer crimes in networked environments and extract digital evidence automatically. ForNet [28] is a novel distributed logging mechanism that focuses on network forensic evidence collection rather than evidence analysis. These approaches rely on the long-term log data collection or on the statistical-based methods that are not suitable for the poor data persistence environment.

Different from the above-mentioned research, Walls *et al.* [29] developed DEC0DE, a system for recovering information from phones with unknown storage formats, which was a critical hurdle in forensic triage. Because phones have myriad of custom hardware and software, all stored data must be examined. Via flexible descriptions of typical data structures, and using a classic dynamic programming algorithm, the DEC0DE system is able to identify call logs and address book entries in phones across a wide range of models and manufacturers. In [30], Li *et al.* perform extensive study on existing fuzzy hashing algorithms with the goal of understanding their applicability in clustering similar malware. They developed a memory triage tool that uses fuzzy hashing to intuitively identify malware by detecting common pieces of malicious code found within a process. In [31], a host-based intrusion detection system gained a high degree of visibility, as it is integrated into the host's monitoring process. A new approach, based on the k-nearest neighbor classifier, is used to classify program behavior as normal or intrusive. Ko *et al.* [32] introduced an approach that integrates intrusion detection techniques with software wrapping technology to enhance a system's ability to defend against intrusions. In particular, they employ the NAI Labs Generic Software Wrapper Toolkit to implement all or part of an intrusion detection system as ID wrappers.

Because the above host-based methods operating at a user level, unfortunately, these systems are quite susceptible to attacks once an attacker has gained access privilege to a host. Besides, an operating system crash will generally cause the system to fail to open. Since the host-based method runs in the same fault domain as the rest of the kernel, this will often cause the entire system to crash or allow the attacker to compromise the kernel [33]–[35].

III. Proposed CloudSEC Framework

The overview of the proposed CloudSEC architecture is shown in Fig. 1. CloudSEC consists of two components: EventTracker and AlertCorrelator. At the ubiquitous architecture of the edge-cloud infrastructure, it can be considered as the hosts of the known operating system; more specific, in the current solutions, these hosts are the VM and the containers. Each VM or the container has a built-in EventTracker that is used to monitor user activities and detect intrusions by auditing the event log on this system and analyzing commands and tracing system calls made by the users. AlertCorrelator is located at the edge of the cloud computing environment. It correlates and analyzes alerts generated by multiple distributed network intrusion detection sensors (NIDS) deployed in a specific cloud computing environment. A central management unit is responsible for exchanging

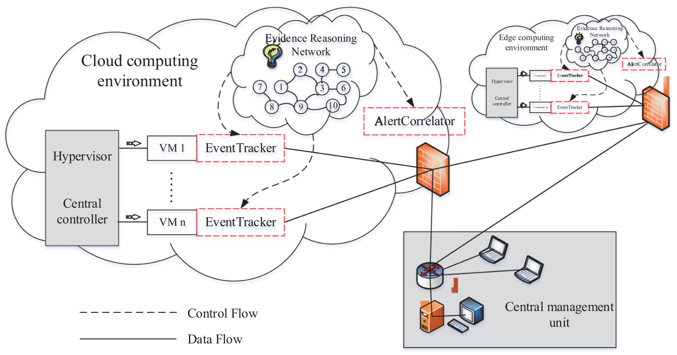


Fig. 1. Architecture of CloudSEC.

information and put in place a set of evaluation criteria used to evaluate the trustworthiness of the results. In the following sections, we describe these two above-mentioned components in detail.

The main idea behind CloudSEC is the modeling of the ERN. It is known that a lateral movement is a sequence of small attack units/steps happened in different times and network locations following a certain logic. Each step of an attack can be considered as a preparation step for the next one. In the ERN model, a detected attack step is referred to as evidence. If we can discover all hidden correlation in an evidence chain, then a lateral movement is detected. Based on the concept of vulnerability and vulnerability correlation, an ERN is constructed to correlate and reasoning out evidence chains and eventually achieve the goal of lateral movement detection. CloudSEC enables lateral movement detection on an unstable underlying structure, with the low persistency of data. To ease the understanding of the ERN, we first describe the concept of vulnerability and vulnerability correlation.

A. Vulnerability and Vulnerability Correlation

The concept of vulnerability refers to the defects of the computer system following a certain security strategy. Since vulnerability is an intrinsic factor of security incidents, we define vulnerability and vulnerability correlation as follows.

Definition 1 (Vulnerability): A vulnerability is a defect existing in a software system or a software component. The exploitation and utilization of such a defect would violate one or more security policies and adversely affect the confidentiality, identifiability, and usability of the software system. We denote vulnerability as v in the rest of this paper.

Definition 2 (Vulnerability correlation): Suppose that a software system S has n vulnerabilities $V[S] = \{v_1, v_2, \ldots, v_n\}$, if $\exists V_{li}(i = 1 \sim k \wedge k \leq n) \in V[S]$, so that the attacker could launch a multiple-stage attack with $v_{l1}, \ldots, v_{lk}$; then, we say $v_{l1}, \ldots, v_{lk}$ are correlated, or we say there exists a correlation among $v_{l1}, \ldots, v_{lk}$.

Vulnerability correlation can be represented either as an AND or an OR structure (as shown in Fig. 2).

In an AND structure, the premise of utilizing vulnerability v to attack the target system is to utilize all the vulnerabilities of $v_{li}(1 \leq i \leq n, n \geq 2)$. In an OR structure, the premise of utilizing vulnerability v to attack the target system is to utilize any one of the vulnerabilities in $v_{li}(1 \leq i \leq n, n \geq 1)$.

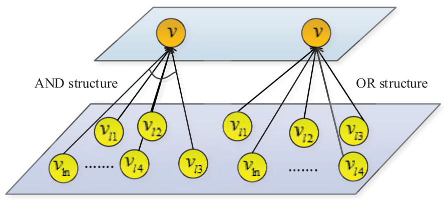


Fig. 2. Two basic structures of vulnerability correlation.

Given that the nature of an attack is to exploit one or more existing vulnerabilities on a computer system, we need to associate the attacks to the vulnerabilities. To do so, we map the attack feature space into the vulnerability space, so that we could describe the attack correlations in the perspective of vulnerability correlations. Therefore, in order to form a credible evidence chain, we construct an ERN to correlate discrete attack evidences based on vulnerability correlation.

B. ERN Model

We use a directed graph to describe the connectivity and vulnerability correlations of an information system network and denote it as an ERN.

Definition 3 (ERN): The ERN is a directed graph G that is described using a set of five-tuples: $\{N, E, L, W, D\}$.

1) N is a set of n vertices, each of which carries some vulnerabilities information. A tuple (n_i, v_i) together represents a network node, where n_i is the ID of the network node and v_i is a set of vulnerabilities on n_i and $0 \leq i \leq n, n \geq 0$.
2) E is a set of e directed links, each of which is an ordered pair: $e_j(e_j \in E) = ((n_x, v_x), (n_y, v_y))$, where n_x contains v_x and n_y contains v_y, $0 \leq j \leq e$, $0 \leq x \leq n$, $0 \leq y \leq n$. Link e_j indicates that (n_x, v_x) interconnects with (n_y, v_y), and v_x and v_y have vulnerability correlation. We say that (n_x, v_x) is a parent node, and (n_y, v_y) is a child node. For simplicity, we denote each such a directed link e_j as $n_x \rightarrow n_y$. $\forall (n_i, v_i) \in N$, we define $IN(n_i, v_i) = \{E_I | E_I = ((n_l, v_l), (n_i, v_i)), E_I \subseteq E, (n_l, v_l) \in N, 0 \leq l \leq n\}$ as a set of incoming link with node (n_i, v_i) as a child node. Similarly, we define $\mathrm{OUT}(n_i, v_i) = \{E_J | E_J = ((n_i, v_i), (n_m, v_m)), E_J \subseteq E, (n_m, v_m) \in N, 0 \leq m \leq n\}$ as a set of directed links that have node n_i as a parent node.
3) L is a set of logical expressions representing the relationships among the directed links using AND and OR logical expression operators along with brackets. L is one-to-one mapped to N, expressing the relationships among the directed links going in or out each specific node. $\forall (n_i, v_i)$, if $\mathrm{IN}(n_i, v_i) = \{e_1, e_2, e_3 | e_1 = ((n_1, v_1), (n_i, v_i)), e_2 = ((n_2, v_2), (n_i, v_i)), e_3 = ((n_3, v_3), (n_i, v_i))\}$, and $l_i = (e_1 \wedge e_2) \vee e_3$, where $l_i \subseteq L, n_l \in N, 0 \leq l \leq n\}$, then parent nodes (n_1, v_1) and (n_2, v_2) both fall into an AND relationship with (n_i, v_i). On the other hand, (n_3, v_3) and (n_i, v_i) fall into an OR relationship.

4) W is defined as a set of risk weights for each vertex in N. W has an one-to-one mapping to N. $\forall n_i \in N$, we define $w_i = (f_i + p_i + r_i)/3$, where $f_i \in [0, 1]$ denotes the functional value of vertex (n_i, v_i), including information servers, database servers, work stations, etc.; $p_i \in [0, 1]$ denotes the probability of successful exploitation of the vulnerability v_i on vertex (n_i, v_i); and $r_i \in [0, 1]$ denotes the security impact of vulnerability v_i. W is defined to set up standard for evaluating the generated evidence chain.
5) $D \in \{qn, pptr\}$ contains a set of data structures for each vertex in N. It has a one-to-one mapping to N. Here, qn is a circular queue of length k for evidence storage. Each element of the queue may be represented as $qn = \{ts, cptr, wt\}$, where ts is the timestamp; $cptr$ is the pointer pointing to the child node used to reconstruct an attacking process flow; s is used to show the state of this node. Its value ranges from 0 to 3, representing, respectively, a start node, both a start node and a virtual node, an intermediate node, or a virtual node; and wt is the risk weight. $pptr$ stores a set of pointers pointing to the parent nodes of this vertex. Such reversed pointers (i.e., pointers pointing to the parent nodes) are used to speed up the evidence reasoning process. For simplicity, we use "." to index each specific element, for example, $d_i.pptr$ or $d_i.qn_i.ts$.

The following steps describe briefly the procedures to generate an ERN.

1) Report the topology of the target network by device users such as the administrators or the node hosts of the target network to be protected.
2) Use existing vulnerability scanning tools such as Nmap and Nessus to probe the target network from different locations within and outside the managed network domain.
3) Construct the vertex set.
4) Traverse all vertices in the target network and construct recursively a set of directed links E and the logical expression set L based on node connectivity and vulnerability correlations.
5) Compute W using the predetermined risk weights and initialize data structure set D.

Obviously, the execution of steps 1 and 2 depends on the characteristics and scale of the target network topology. However, the efficiency and executability of the forensics can be solved by precomputing the ERN *a priori*.

Fig. 3 illustrates an example of a network information system including five linux systems (A, B, C, D, and E). Server A has vulnerability v_1 that threatens the root users and vulnerability v_2 that threatens regular users. Service *rshd* is running on B, which permits root users of A and D to gain remote access on B executing shell command. The VM C runs *telnetd* and *rshd* services. *rshd* allows only root user of VM D to execute shell command remotely on C. C also has vulnerability v_4 that threatens all root users and vulnerability v_5 that threatens regular users. D runs *sshd* service. It has vulnerability v_3 that threatens root users. The edge container E has vulnerability v_6 that threatens all root users. All regular users of A and the root user of C could gain remote access to D as root users via *ssh*.

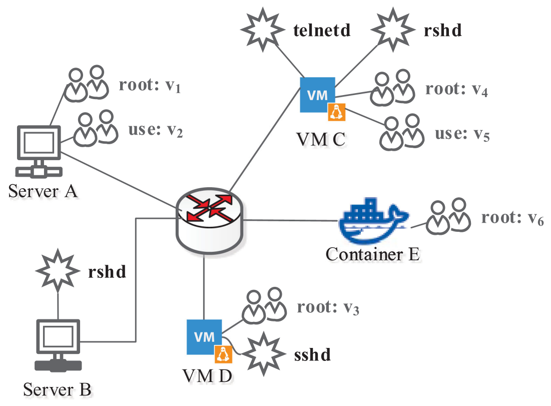


Fig. 3. Example of an information system network.

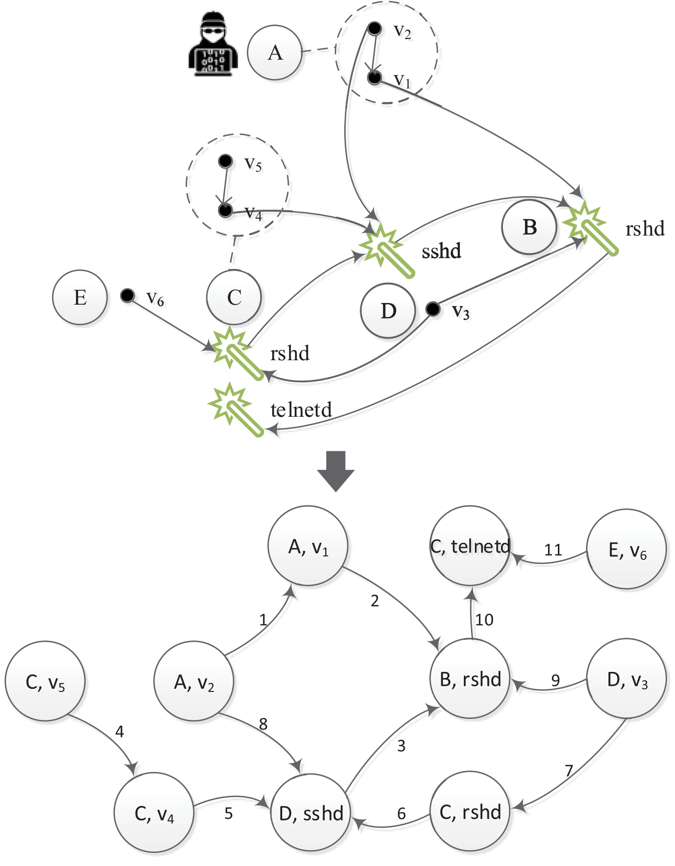


Fig. 4. Example of an ERN.

The root users of B and E could gain remote access to C as a root user through *telnet*.

The constructed ERN of the above information system network is shown in Fig. 4, including ten network nodes and 11 directed links.

C. Evidence Chain Reasoning

ERN provides a framework for the reconstruction of attacking flow and evidence chain reasoning. Each evidence chain can be represented using a subgraph of the ERN. The main purpose of evidence preprocessing is to map the extracted evidence out

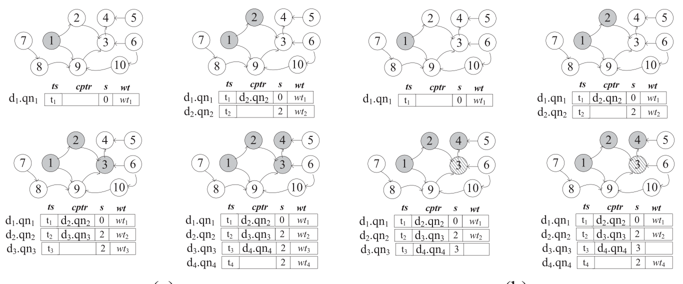


Fig. 5. Evidence chain reasoning process examples. (a) Evidence chain reasoning process. (b) Timing-independent evidence chain reasoning process.

of the attack flows to the vertex of the ERN, noted as function map().

Evidence chain reasoning could be abstracted as follows: assume that there is a time-sequenced evidence list: $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, $\forall \varepsilon_i = 1, 2, \ldots, n$, searching for the evidence set within the range of $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{i-1}$, such that there exists a directed link from map(ε_h) to map(ε_i) in the ERN, noted as map$(\varepsilon_h) \rightarrow$ map(ε_i), $0 \leq h \leq n$.

The procedure of evidence chain reasoning includes the following three steps.

Step 1: Initialization: Fetch an evidence ε_i and map it to a vertex in the ERN of a given system, e.g., map$(\varepsilon_i) = n_i$. Then, set $d_i.qn_i.ts$ as the timestamp t_i.

Step 2: Association analysis: The analysis methods depend on the number of links pointing to a vertex n_i in the ERN.

1) When IN$(n_i) = 0$, ε_i is the start point of this ERN. Then, set $d_i.qn_i.s = 0$ and $d_i.qn_i.wt = w_i$.
2) When IN$(n_i) > 0$, use reverse index data structure $d_i.pptr$ to find all the parent nodes of n_i. If a parent node n_j's $d_j.qn$ is nonempty, we say (n_j, n_i) is a truth-value link.
   a) If the logical expression l_i returns a true value with all the above-mentioned truth-value links, then we say ε_i is an intermedia node. Thus, we set $cptr = d_i.qn_i$ for all parent nodes that make l_i true, and set $d_i, qn_i.s = 2$, $d_i.qn_i.wt = w_i$;
   b) If the logical expression l_i returns a false value with all the above-mentioned truth-value links, then we know that ε_i has no correlation with the other evidences. Thus, we consider ε_i as a start node and set $d_i.qn_i.s = 0$ and $d_i.qn_i.wt = w_i$ accordingly.

Step 3: Evidence chain generation: Conduct breadth-first search (BFS) from a vertex n_i in the ERN, of which the in-degree is 0 (i.e., IN$(n_i) = 0$) and generate evidence chain base on the *cptr* of the traversed vertices.

Fig. 5(a) illustrated the above-mentioned procedure. The elements in the data structure D change together with the evidence, as shown in Fig. 4.

D. Timing-Independent Evidence Chain Reasoning

Whether the evidence chain reasoning algorithm can draw a correct conclusion depends on the correct time stamping of the evidences. In order to solve the inconsistent time-stamping problem, which is often a challenge, we propose a timing-independent evidence chain reasoning algorithm. The process is as follows.

Step 1: Initialization: Fetch the next evidence ε_i and map it to a vertex of the ERN of the given system, i.e., map$(\varepsilon_i) = n_i$. Then, set $d_i.qn_i.ts$ as the timestamp t_i;

Step 2: Vertex checking: if $d_i.qn_i.s = 1$ or $d_i.qn_i.s = 3$, then $(d_i.qn_i.s) --$, $d_i.qn_i, wt = w_i$, $d_i.qn_i.ts = t_i$; then, go back to step 1; otherwise, go to step 3.

Step 3: Association analysis: The analysis methods vary based on different types of vertices of the ERN.

1) *Case A:* When IN$(n_i) = 0$, ε_i is the start point; set $d_i.qn_i.s = 0$ and $d_i.qn_i.wt = w_i$.
2) *Case B:* When IN$(n_i) > 0$, if we substitute all truth-value links into l_i, and l_i is true, then ε_i is an intermedia node. Thus, we set $cptr = d_i.qn_i$ for all parent nodes that make l_i true, and set $d_i.qn_i.s = 2$, $d_i.qn_i.wt = w_i$.
3) *Case C:* When IN$(n_i) > 0$, if we substitute all truth-value links into l_i, and l_i is false, then traverse all parent nodes of n_i to search for one of the parent nodes n_j, which makes l_i true when we increase a virtual record to n_j. The virtual record generated in the reasoning process indicates that the evidence is not acquired by EventTracker, so the risk weights are not taken into account (see further explanation later in Section III-F). We further conduct association analysis on n_j; if case A is met, then set $d_j.qn_j.s = 1$, $d_j.qn_j.cptr = d_i.qn_i$, $d_i.qn_i.s = 2$, $d_i.qn_i.wt = w_i$; if case B is met, then set $d_j.qn_j.s = 3$, $d_j.qn_j.cptr = d_i.qn_i$, $d_i.qn_i.s = 2$, $d_i.qn_i.wt = w_i$; if case C is met, then ε_i has no correlation with the other evidences. Thus, we take ε_i as a start node, and set $d_i.qn_i.s = 0$, $d_i.qn_i.wt = w_i$.

Step 4: Evidence chain generation: Conduct BFS from an ERN vertex, of which the in-degree is 0, and generate evidence chain based on the *cptr* of the traversed vertices.

Fig. 5(b) shows the approximate reasoning process, where vertex 3 is a virtual record.

E. Time Complexity Analysis

Let N and E be the number of vertices and edges/links in a given ERN. The first step of our evidence chain reasoning algorithm finishes in constant time. The time complexity of the second step is $O(E/N)$. And the time complexity of the third step is $O(N + E)$. Therefore, the overall time complexity of the evidence chain reasoning algorithm is $O(N + E)$.

For the time complexity of the "timing-independent evidence chain reasoning" algorithm, the first and second steps finish in constant time. Since the algorithm needs to determine the status of one of the parents node of a virtual node, the average time complexity of step 3 is $O((E/N)^2)$, and BFS time complexity is $O(N + E)$. For a directed complete graph, $E = N^2 - N$; thus, $N + E \geq (E/N)^2$. Hence, the time complexity of the "timing-independent evidence chain reasoning" algorithm is $O(N + E)$.

F. Power of the Evidence Chain

The power of electronic evidence refers to the persuasive power of an evidence in proving the case. Given that W of an

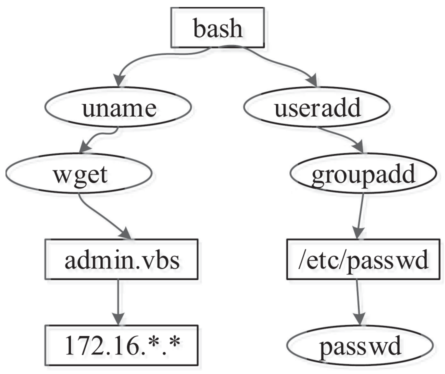


Fig. 6. Example of an ERN for system calls.

ERN defines the risk weight of every vertex, EventTracker adds up the risk weight of all evidence on an evidence chain, and it will be used to evaluate the power of an evidence chain.

As mentioned earlier, the virtual record generated in the reasoning process indicates that the evidence is not acquired by the EventTracker. Therefore, its risk weight is not added into the overall weights W. Assume that the sum of risk weights of all evidence in an evidence chain is w, and the sum of risk weights of all the virtual records on this evidence chain is w'; the weight of the evidence chain should be calculated, as shown in formula (1). The bigger the value of $confidence$, the bigger the evidence chain power

$$confidence = \frac{w}{w + w'}. \tag{1}$$

G. Application of an ERN in EventTracker

System calls are generally used to change the states of an operation system, such as creating files, forking processes, changing registries, etc. Monitoring the changes of system calls can get many essential characteristics of program operations and, thus, form the basis of dynamic security analysis. In an EventTracker, Virtual Machine Introspection technology is used to monitor system calls in cloud computing hosts.

There are parameter dependencies among sequences of system calls generated by program operation. EventTracker maps system calls and their dependencies into an ERN model as the nodes and the links, respectively. For example, Fig. 6 illustrates an example of an ERN for system calls. First, an attacker uses *useradd* and *groupadd* commands to add system users and set permissions. This attacker then sets up passwords through *passwd* commands. After checking the kernel version information with command *uname*, “admin.vbs” script is downloaded from the remote host 172.16.*.* through FTP.

The goal of an EventTracker is to identify complex lateral movement attacks inside the cloud hosts. In the intrusion detection literature, an attack scenario (or attack pattern) is a sequence of explicit attack steps, which are logically linked and lead to an objective. When a set of correlation is received, if an attack scenario is detected, it will raise an intermediate attack alarm, which will prompt the system to capture the causal relationships among the evidences.

H. Design and Implementation of AlertCorrelator

AlertCorrelator is located at the edge of a cloud computing environment. It correlates and analyzes alerts generated by multiple NIDSs deployed in the cloud computing environment. The NIDS is responsible for monitoring network traffic in real time, finding intrusion events and storing corresponding IP packets in a database. According to the design of the ERN, we can see that AlertCorrelator does not tolerate false positives in alerts, but it can accept a certain degree of false negatives.

AlertCorrelator collects and preprocesses the alerts first. Every collected alert is coded and normalized into a standardized format. Additional information, such as timestamps and source address of the attacker, are also added into the database. Events generated by different NIDSs and related to the same attack are merged into a single alarm. Then, these preprocessed alarms are added into the ERN as evidence in a chronological order. At last, timing-independent evidence chain reasoning algorithm is called. In this phase, the ERN model determines whether the attack is either a successful attack or a nonrelevant attack, i.e., an attack that does not lead to lateral movement attack.

For large-scale networks, when new NIDSs are added to the cloud boundary, augmenting an existing ERN can be done easily in an iterative fashion. This demonstrates the high scalability and flexibility of AlertCorrelator.

IV. Experiment Evaluation

To further verify the effectiveness of our proposed method, we developed our EventTracker prototype system using C language on Linux RedHat 7.3. We use the intrusion detection system Snort 2.4.3 as the attack detection component and Graphviz to visualize the evidence chain. We conduct our experiments on Shuguang servers (CPU 2.4G, MEM 8G). We report in this section two sets of experiments.

A. Parameter Setup

In order to evaluate the power of the evidence chain, we need to set an appropriate risk weights of ERN. Since the difficulty of utilizing each vulnerability in different situation is different, the probability of successful exploitation of each vulnerability is different. In our experiments, we assume that all the probabilities of successful exploitation of each vulnerability are equal, and the functional attributes of network nodes are also equal. As for the security impact of vulnerabilities, we classify the vulnerabilities with the impact scope into nine categories and set each category with a certain weight, as shown in Table II. Note that this weight value is dynamic, it could be adjusted in the system to reflect the scenario features better.

B. Reasoning Result of the Lincoln Dataset

Experiment 1 uses the LLDOS1.0 and LLDOS2.0.2 datasets from MIT Lincoln Laboratories. The test bed producing this dataset includes an external Internet environment simulated by 14 hosts, an internal network of 39 hosts, and a DMZ area consisting of six hosts, covering operating systems including Windows, Linux RedHat 5.0, SunOS 4.1.4, and Solaris 2.7.

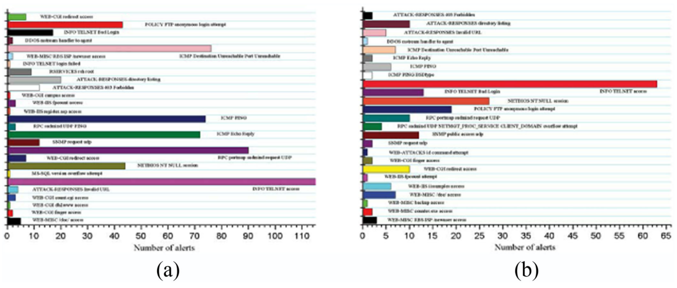


Fig. 7. Categories and quantity of the LLDOS dataset. (a) Types and quantity of evidences detected from the LLDOS1.0 Intranet Dataset. (b) Types and quantity of evidences detected from the LLDOS2.0.2 Intranet Dataset.

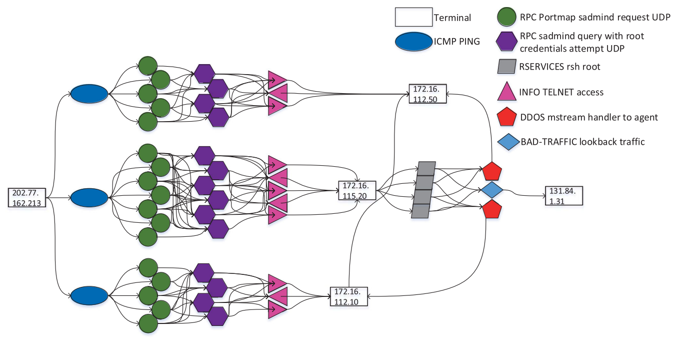

Terminal
ICMP PING
RPC Portmap sadmind request UDP
RPC sadmind query with root credentials attempt UDP
RSERVICES rsh root
INFO TELNET access
DDOS mstream handler to agent
BAD-TRAFFIC lookback traffic
202.77.162.213
172.16.112.50
172.16.115.20
172.16.112.10
131.84.1.31

Fig. 8. Evidence chain of the LLDOS1.0 Intranet Dataset.

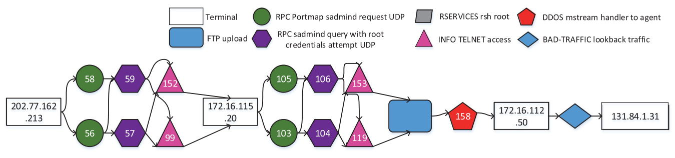

Terminal
FTP upload
RPC Portmap sadmind request UDP
RPC sadmind query with root credentials attempt UDP
RSERVICES rsh root
INFO TELNET access
DDOS mstream handler to agent
BAD-TRAFFIC lookback traffic
202.77.162.213
58
59
152
56
57
99
172.16.115.20
105
106
153
103
104
119
158
172.16.112.50
131.84.1.31

Fig. 9. Evidence chain of the LLDOS2.0 dataset.

TABLE I
WEIGHT OF VULNERABILITY IMPACT

Vulnerability category	Impact scope	Weight
1	System administrators, managing system resources, system files, system processes, and other resources	1.0
2	System administrator with partial permissions	0.8
3	Permissions of any number of system ordinary users with more independent and private resources	0.6
4	Permissions of a system ordinary user and partial permissions of other ordinary users	0.5
5	Permissions of a system ordinary user created by the system initialization or created by the system administrator with its own private resources	0.4
6	Partial permission of a ordinary user	0.2
7	Remote visitors who can access network services, usually trusted visitors, who can interact with network service processes, scan system information, and so on	0.1
8	Remote visitors who are connected to the target system at the physical layer, usually untrusted or firewalled visitors.	0.0

Attack detection component is used to produce an intranet dataset of size 179 MB from each of the two LLDOS datasets. The diversity of both terminals and data ensures the simulation of the edge-computing scenario. The type and amount of evidence generated are shown in Fig. 7. Fig. 8 shows the evidence chain deduced by EventTracker on LLDOS1.0. Here, we identify each entity in the traffic by the MAC of each host.

This evidence chain indicates the attacker's five attack phases: the attacker first scans the entire network from host 202.77.162.213 to learn the target host's IP address range; then, it executes Sadmind Overflow program with the ping option checked in order to verify on each host whether the Sadmind service is running. It then determines the final destination host [e.g., 172.16.115.20 in Fig. 5(b)]; next, the attacker uses the Solaris operating system's Sadmind vulnerability to implement a buffer overflow attack; after gaining the root privileges, the attacker installs the Mstream backdoor program through *telnet* and *rpc*; finally, the attacker uses this compromised slave to launch a distributed denial-of-service (DDoS) attack to host 172.16.115.20/172.16.112.10/131.84.1.131. The evidence chain of the above-mentioned attacking process is shown in Fig. 5(b). This result matches the document provided by MIT Lincoln, which proves the effectiveness of the proposed EventTracker. Since there is no virtual records, the $confidence$ is 100% in this case.

The deducted evidence chain of LLDOS2.0.2 is shown in Fig. 9. This evidence chain also contains a complete attack sequence and uses a springboard attack. Similar to LLDOS 1.0, the attacker also implemented an attack on host 202.77.162.213 and succeeded in obtaining the root privileges of hosts 172.16.115.20 and 172.16.112.50 that are using the Solaris operating system with Sadmind vulnerability. However, compared to the former one, the LLDOS2.0.2 attack process is more complex and with latency. For example, the attacker did not use the "ICMP echo reply," which could be easily shielded. Instead, it uses the legitimate "DNS HINFO" query to obtain the target host (a DNS server). It is noted that the shadow node "FTP upload" in the evidence chain indicates that the attacker used FTP to upload the evidence of the Mstream backdoor program and attacking script. However, the attack detection component missed this event. To ensure the integrity of the evidence, EventTracker generated a virtual record. According to the weights set in Table I, the $confidence$ is reduced to 86% accordingly. Then, we use the timestamp of the alert of "INFO TELNET access 119" to find the PCL. Two relations were found, which is shown in Fig. 9. This further proves the correctness of our reasoning algorithm.

The second experiment uses the Treasure Hunt dataset collected by the University of California Santa Barbara to guide students in the design of network offense and defense course. Its network topology is divided into three subnetworks: Alpha, Omega, and DMZ, which includes MySQL servers, event processing servers, file servers, and WEB servers. Although the network topology is not complex, a wide range of attack

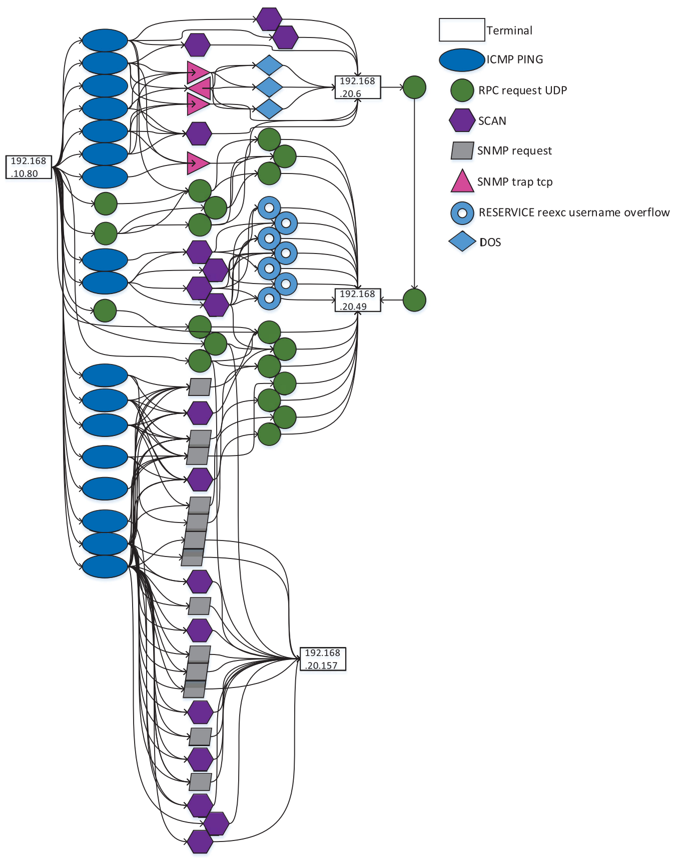


Fig. 10. Evidence chain of the alpha subnetwork.

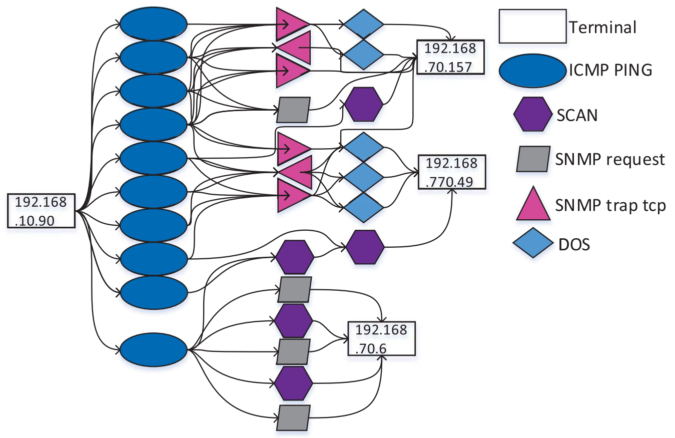


Fig. 11. Evidence chain of the omega subnetwork.

methods are adopted in this network. Therefore, we believe that this dataset is suitable for the edge-cloud scenario functional testing of the proposed EventTracker. The corresponding evidence chains are shown in Figs. 10 and 11, respectively, with $confidence$ interval of 91.7% and 33.3%.

C. Performance Analysis

Table II gives the time overhead of CloudSEC in handling the above sets of data. It is shown that the number of evidence chains

TABLE II
Time Overhead of CloudSEC

Data Sets			Intrusion Events	Process Time (sec)	Average Processing Speed (one event per secnod)
MIT/LL 2000	LLDOS 1.0	DMZ	2498	15.9806	78.8505
		Inside	905	16.7467	
	LLDOS 2.0.2	DMZ	1125	16.5234	
		Inside	624	16.0881	
Treasure Hunt	Alpha		732	15.8154	119.8598
	Omega		732	15.8154	

CloudSEC could handle per second is around 100. According to [19], about 10–20 000 security events per day on average are detected by each detection point. CloudSEC greatly exceeds the current actual processing requirements under the conditions of real-time reasoning lateral movement. As shown in the table, the average time cost per event is the millisecond level, which is on the same order of magnitude as the current network event. This frontage is suitable for deploying in the edge-cloud environment.

V. Conclusion

In this paper, we proposed a new method to track events for lateral movement detection. The concept of vulnerability correlation was introduced. Methods on how to construct an ERN based on the vulnerability knowledge and network environment information were provided. Then, two lateral movement reasoning algorithms based on the constructed ERN were presented. The proposed CloudSEC provides a strong guarantee for the rapid and effective evidence investigation, as well as real-time attack detection; this advantage is more suitable for those complex edge-cloud computing environments, for example, the services based on the collaboration between the edge artificial intelligence and the cloud computing. Experiments using various real network datasets proved the correctness of the proposed approach. Theoretical analysis concluded that both chain reasoning algorithms achieve linear time complexity. In the future, we aim at improving the performance on ERN generation, which further improves the overall CloudSEC efficiency; at the same time, multitype of lateral movement tricks will be evaluated.

References

[1] X. Du, M. Guizani, Y. Xiao, and H. H. Chen, "A routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 3, pp. 1223–1229, Mar. 2009.
[2] Y. Xiao *et al.*, "A survey of key management schemes in wireless sensor networks," *J. Comput. Commun.*, vol. 30, no. 11–12, pp. 2314–2341, 2007.
[3] X. Du, Y. Xiao, M. Guizani, and H. H. Chen, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Netw.*, vol. 5, no. 1, pp. 24–34, 2007.

[4] Y. Xiao *et al.*, "Internet protocol television (IPTV): The killer application for the next generation Internet," *IEEE Commun. Mag.*, vol. 45, no. 11, pp. 126–134, Nov. 2007.
[5] X. Du and H. H. Chen, "Security in wireless sensor networks," *IEEE Wireless Commun. Mag.*, vol. 15, no. 4, pp. 60–66, Aug. 2008.
[6] Y. C. Hu *et al.*, "Mobile edge computing—A key technology towards 5G," ETSI White Paper 11, 2015, pp. 1–16.
[7] Z. Tian *et al.*, "A data-driven method for future Internet route decision modeling," *Future Gener. Comput. Syst.*, vol. 95, pp. 212–220, 2019.
[8] Q. Tan *et al.*, "Towards a comprehensive insight into the eclipse attacks of Tor hidden services," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2846624.
[9] J. Cheng, R. Xu, X. Tang, V. S. Sheng, and C. Cai, "An abnormal network flow feature sequence prediction approach for DDoS attacks detection in big data environment," *CMC: Comput., Mater. Continua*, vol. 55, no. 1, pp. 095–119, 2018.
[10] Y. Liu, H. Peng, and J. Wang, "Verifiable diversity ranking search over encrypted outsourced data," *CMC: Comput., Mater. Continua*, vol. 55, no. 1, pp. 037–057, 2018.
[11] J. Cui, Y. Zhang, Z. Cai, A. Liu, and Y. Li, "Securing display path for security-sensitive applications on mobile devices," *CMC: Comput., Mater. Continua*, vol. 55, no. 1, pp. 017–035, 2018.
[12] F. Binxing, J. Yan, L. Aiping, and Z. Weizhe, "Cyber ranges: State-of-the-art and research challenges," *J. Cyber Secur.*, vol. 1, no. 3, pp. 1–9, 2016.
[13] G. Jia *et al.*, "Edge computing-based intelligent manhole cover management system for smart cities," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1648–1656, Jun. 2018.
[14] W. Yuan *et al.*, "Edge-dual graph preserving sign prediction for signed social networks," *IEEE Access*, vol. 5, pp. 19383–19392, 2017.
[15] C. Zhu *et al.*, "Social sensor cloud: Framework, greenness, issues, and outlook," *IEEE Netw.*, vol. 32, no. 5, pp. 100–105, Sep./Oct. 2018.
[16] C. Zhu *et al.*, "Secure multimedia big data in trust-assisted sensor-cloud for smart city," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 24–30, Dec. 2017.
[17] L. Zhu, M. Li, Z. Zhang, and Z. Qin, "ASAP: An anonymous smart-parking and payment scheme in vehicular networks," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2018.2850780.
[18] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani, "Privacy-preserving DDoS attack detection using cross-domain traffic in software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 628–643, Mar. 2018.
[19] L. Zhu *et al.*, "PRIF: A privacy-preserving interest-based forwarding scheme for social Internet of vehicles," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2457–2466, Aug. 2018.
[20] S. Su, Y. Sun, X. Gao, J. Qiu, and Z. Tian, "A correlation-change based feature selection method for IoT equipment anomaly detection," *Appl. Sci.*, vol. 9, no. 3, 2019, Art. no. 437.
[21] J. Qiu, Y. Chai, Y. Liu, Z. Gu, S. Li, and Z. Tian, "Automatic non-taxonomic relation extraction from big data in smart city," *IEEE Access*, vol. 6, pp. 74854–74864, 2018.
[22] EnCase Forensic Tool. 2018. [Online]. Available: http://www.guidancesoftware.com
[23] SafeBack Bit Stream Backup Software. 2018. [Online]. Available at: http://www.forensics-intl.com/safeback.html
[24] A. Mitchell and G. Vigna, "MNEMOSYNE: Designing and implementing network short-term memory," in *Proc. Int. Conf. Eng. Complex Comput. Syst.*, Dec. 2002, pp. 91–100.
[25] W. Wang and T. E. Daniels, "Network forensics analysis with evidence graph," in *Proc. Digit. Forensic Res. Workshop*, New Orleans, LA, USA, 2005.
[26] Z. Tian, W. Jiang, and Y. Li, "A transductive scheme based inference techniques for network forensic analysis," *China Commun.*, vol. 12, no. 2, pp. 167–176, Feb. 2015.
[27] Z. Tian, W. Jiang, Y. Li, and L. Dong, "A digital evidence fusion method in network forensics systems with Dempster-Shafer theory," *China Commun.*, vol. 11, no. 5, pp. 91–97, May 2014.
[28] K. Shanmugasundaram, N. Memon, A. Savant, and H. Bronnimann, "ForNet: A distributed forensics network," in *Proc. 2nd Int. Workshop Math. Methods, Models Archit. Comput. Netw. Secur.*, Saint Petersburg, Russia, 2003, pp. 1–16.
[29] R. J. Walls, E. Learned-Miller, and B. N. Levine, "Forensic triage for mobile phones with DEC0DE," in *Proc. 20th USENIX Conf. Secur.*, Aug. 2011, p. 7.
[30] Y. Li *et al.*, "Experimental study of fuzzy hashing in malware clustering analysis," in *Proc. 8th Workshop Cyber Secur. Exp. Test*, 2015, p. 8.
[31] Y. Liao and V. R. Vemuri, "Using text categorization techniques for intrusion detection," in *Proc. 11th USENIX Secur. Symp.*, Aug. 2002, pp. 51–59.
[32] C. Ko, T. Fraser, L. Badger, and D. Kilpatrick, "Detecting and countering system intrusions using software wrappers," in *Proc. 9th USENIX Sec. Symp.*, Aug. 2000, p. 11.
[33] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2003, pp. 191–206.
[34] N. Joseph, S. Sunny, S. Dija, and K. L. Thomas, "Volatile internet evidence extraction from windows systems," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res.*, 2014, pp. 1–5.
[35] F. Jiang *et al.*, "Deep learning based multi-channel intelligent attack detection for data security," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2018.2793284.

Zhihong Tian received the Ph.D. degree in network and information security from the Harbin Institute of Technology, Harbin, China, in 2006.

From 2003 to 2016, he was with the Harbin Institute of Technology, Harbin, China. He is currently a Professor, a Ph.D. supervisor, and the Dean of the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China. His current research interests include computer networks and network security.

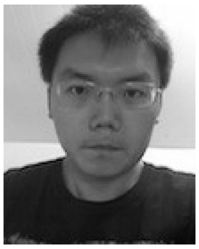
Dr. Tian is the Standing Director of the CyberSecurity Association of China and a member of the China Computer Federation.

Wei Shi received the bachelor's degree in computer engineering from the Harbin Institute of Technology, Harbin, China, in 2001, and the Ph.D. degree in computer science from Carleton University, Ottawa, ON, Canada, in 2007.

She is an Assistant Professor with the University of Ontario Institute of Technology, Oshawa, ON, Canada. She is also an Adjunct Professor with Carleton University. Prior to her academic career, as a Software Developer and Project Manager, she was closely involved in the design and development of a large-scale electronic information system for the distribution of welfare benefits in China.

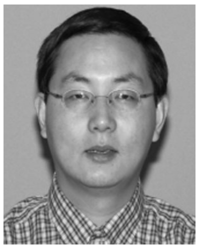
Yuhang Wang received the B.S. degree in computer science from the Heilongjiang University of Science and Technology, Harbin, China, in 2009, and the M.S. degree in computer science from the Harbin Institute of Technology, Harbin, in 2012.

He is currently an Assistant Professor with Guangzhou University, Guangzhou, China. His research interests include network and information security and location privacy.



Chunsheng Zhu received the B.E. degree in network engineering from the Dalian University of Technology, Dalian, China, in 2010. He is currently working toward the master's degree with the Department of Mathematics, Statistics, and Computer Science, St. Francis Xavier University, Antigonish, NS, Canada.

His current research interests include wireless sensor networks and security.



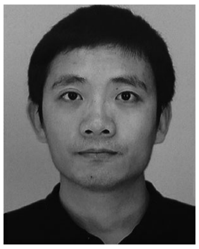
Xiaojiang Du received the Ph.D. degree in electrical engineering from the University of Maryland, College Park, MD, USA, in 2003.

He is currently a Tenured Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. He has authored more than 260 journal and conference papers in these areas, and a book (Springer). He has been awarded over \$5 million U.S. dollars in research grants from the U.S. National Science Foundation, Army Research Office, Air Force Research Laboratory, NASA, the State of Pennsylvania, and Amazon. He serves on the editorial boards of three international journals. His research interests include wireless communications, wireless networks, security, and systems.

Prof. Du is a life member of the Association for Computing Machinery.

Shen Su was born in 1985. He received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2016.

He is currently an Assistant Professor with Guangzhou University, Guangzhou, China. His current research interests include interdomain routing and security.



Yanbin Sun received the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2016.

He is currently an Assistant Professor with Guangzhou University, Guangzhou, China. His research interests include information-centric networking and scalable routing.

Nadra Guizani received the B.S. degree in computer engineering from Kuwait University, Kuwait, in 2011, and the Ph.D. degree in computer engineering from Purdue University, West Lafayette, IN, USA, in 2018.

She is a Teaching Assistant with the Electronic Devices and Design Laboratory, Purdue University, West Lafayette, IN, USA. Her research interests include networks, machine learning, and prediction techniques.