Bayes-Based ARP Attack Detection Algorithm for Cloud Centers

Huan Ma
the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China.

Hao Ding
the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China.

Yang Yang
the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China.

Zhenqiang Mi
the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China.

James Yifei Yang
the Department of Electronics and Communication Engineering, University of Illinois at Urbana-Champaign, Champaign, IL 61801, USA.

See next page for additional authors
Follow this and additional works at: https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology

Part of the Computer Sciences Commons, and the Electrical and Computer Engineering Commons

Recommended Citation
Huan Ma, Hao Ding, Yang Yang et al. Bayes-Based ARP Attack Detection Algorithm for Cloud Centers. Tsinghua Science and Technology 2016, 21(1): 17-28.

This Research Article is brought to you for free and open access by Tsinghua University Press: Journals Publishing. It has been accepted for inclusion in Tsinghua Science and Technology by an authorized editor of Tsinghua University Press: Journals Publishing.
Bayes-Based ARP Attack Detection Algorithm for Cloud Centers

Authors
Huan Ma, Hao Ding, Yang Yang, Zhenqiang Mi, James Yifei Yang, and Zenggang Xiong
Bayes-Based ARP Attack Detection Algorithm for Cloud Centers

Huan Ma, Hao Ding, Yang Yang, Zhenqiang Mi*, James Yifei Yang, and Zenggang Xiong

Abstract: To address the issue of internal network security, Software-Defined Network (SDN) technology has been introduced to large-scale cloud centers because it not only improves network performance but also deals with network attacks. To prevent man-in-the-middle and denial of service attacks caused by an address resolution protocol bug in an SDN-based cloud center, this study proposed a Bayes-based algorithm to calculate the probability of a host being an attacker and further presented a detection model based on the algorithm. Experiments were conducted to validate this method.

Key words: cloud computing; Bayes; ARP attack detection; software-defined network

1 Introduction

The blooming cloud infrastructure service industry has attracted numerous tenants to cloud data centers that internally manage infrastructure; such a setup has subsequently resulted in security problems[1,2]. Internal network attacks such as Address Resolution Protocol (ARP)[3] attacks are increasingly problematic for cloud centers, particularly when tenant networks do not operate in isolation from others and one tenant’s ARP frames can affect those of others. In addition, tenants require an internal network security service that is provided by cloud centers and not by themselves. To ensure effective Virtual Machine (VM) migration without interrupting communication in a private cloud network, all VMs work in a two-layer network[4]. For a VM to communicate with another VM with only a known Internet Protocol (IP) address, the destination VM’s Media Access Control (MAC) address must be obtained. The ARP is used to translate an IP address into a MAC address. Translated addresses are stored in the VM’s ARP cache as a temporary IP-MAC pair to reduce the translation time and increase the network-based communication speed.

To translate an IP address into a MAC address, the ARP request frame, which contains the IP address of the destination VM, is broadcasted in a private cloud network. The destination VM sends an ARP response frame, which contains its own IP and MAC addresses, to the source VM after it receives the request frame. Finally, the source VM receives the ARP response frame and inserts a new IP-MAC pair into its ARP cache. However, if a malicious attacker is snooping the ARP broadcast frame, the attacker will continuously send forged ARP response frames to corrupt the ARP cache of the source VM. If the ARP cache is corrupted, the source VM sends subsequent network packets to the wrong MAC address, which results in chaotic and erroneous network communications and leads to ARP attacks. For instance, the Man-In-The-Middle (MITM) attack is attributed to corrupted ARP cache of the source and destination VMs, and the Denial-of-Service (DoS) attack is caused by corrupting the ARP cache of the source VM[5]. Because ARP attacks result in adverse impacts that threaten the effectiveness and security of network communications, many solutions have...
been proposed. However, available solutions require a significant amount of manual operation or financial investment and do not satisfy requirements to realize an ideal solution[5]. In cloud centers, tenants pay for cloud services and do not expect to be directly involved with managing infrastructure operations. Meanwhile, cloud providers are responsible for providing security networks for tenants. To realize a win-win scenario for both cloud providers and tenants, a progressive solution must be introduced.

Flexible control of computing and storage resources implies that cloud centers can satisfy various tenants’ resource requirements. To achieve flexible network control, a new type of network management technology (i.e., Software-Defined Network (SDN))[6,7] has been applied to cloud centers. SDN decouples control and data planes, and all control functions are centralized in a controller. Consequently, SDN not only improves network performance but also addresses network security issues by deploying advanced custom control programs.

In this study, we addressed the internal ARP attack issue of cloud centers with the application of SDN technology. We proposed an algorithm using the Bayesian theorem to calculate the probability of a VM being an attacker and further presented a detection model based on the algorithm. Experiments were conducted to prove the validity and effectiveness of this method.

The rest of this paper is organized as follows. Section 2 describes related works. Section 3 addresses some obvious ARP attack features. Section 4 presents the prediction probability algorithm. In Section 5, on the basis of this algorithm, we propose an ARP attacker detection model. In Section 6, we verify the validity and effectiveness of the method with simulations. Section 7 concludes the study.

2 Related Works

Researchers have proposed many solutions to address the ARP attack problem. One of earlier methods proposed is static ARP cache[8]. A system administrator inserts static IP-MAC pairs into an ARP cache. ARP frames cannot change static IP-MAC pairs, and the static ARP cache never expires, thereby ensuring that ARP attacks will not occur. However, when a network is large, the significant amount of manual configurations required increases the level of difficulty that is needed for static ARP deployment.

Because of the stateless characteristic of ARP[9], Tripunitara and Dutta[10] proposed a stateful ARP protocol. If one host receives an ARP response frame, it only updates the ARP cache when an associated ARP request frame exists. Lootah et al.[11] also proposed a stateful ARP; it uses a local ticket agent server to associate tokens to every host, and ARP frames carry the tokens. When a host receives an ARP frame, it determines the frame’s validity. As this type of stateful ARP requires modifications to be made to the original ARP, it is not compatible with a stateless ARP, which limits its usage.

Therefore, ARP frames are plaintext to prevent a malicious host from identifying ARP broadcast frames and attacking the source host. Bruschi et al.[12] used a secret key technology to encrypt and decrypt ARP frames. Secret keys are stored in the authoritative key distribution server and are associated with the secure dynamic host configuration protocol. However, encryption and decryption require more computational resources, and the security ARP is not compatible with the original ARP. Kumar and Tapaswi[13] used a centralized ARP central server to manage ARP table entries and validate all hosts’ ARP table entries. However, the dedicated server is very expensive.

There are also studies that focus on active detection technology that is compatible with the original ARP. When a host receives an ARP response frame, it initiates a vote in all neighbor nodes[14]. Based on the voting result, it decides whether the information in the ARP response frame can be trusted. Nam et al.[15] improved the voting process to eliminate unfairness due to differential cable and wireless transmission rates. Gao and Xia[16] used Internet Control Message Protocol (ICMP) response information to verify the accuracy of the ARP response frame information. Neminath et al.[17] proposed a method that uses a TCP SYN packet response message to detect the source MAC and IP addresses of ARP frames. Pandey[18] used probe packets to verify ARP responders. In addition, previous studies[19,20] describe the verification of ARP information through a discrete-event system. The aforementioned methods are highly compatible with existing ARP protocols but require additional software to be deployed in hosts. Moreover, the validation process slows down the ARP cache update process, which leads to additional network overheads and delays. Moreover, utilizing the high-layer protocol
information to detect whether an ARP frame was valid violates the principle of network hierarchy.

However, in large-scale cloud centers, none of the aforementioned solutions can be effectively deployed because of extremely high implementation costs. Moreover, these solutions do not consider a sufficiently wide range of pertinent aspects of ARP attack. An ideal ARP attack solution should satisfy the following requirements\[5\]:

1. Should be backward compatible;
2. Should not require additional software in hosts;
3. Should consume low resources;
4. Should be capable of detecting ARP attacks;
5. Should be cost effective; and
6. Should be capable of preventing all types of ARP attacks.

SDN, as a new type of network management technology, has previously been applied in cloud centers. In SDN-based networks, devices are managed by a centralized SDN controller, which improves the functionality of the network. SDN technology can also be used to detect network attacks such as MITM and DoS by analyzing run-time network packets. To resolve an ARP attack, a global ARP cache can be constructed in the SDN controller with a view of the global network. In addition, the SDN controller should be able to manipulate the ARP cache of all VMs with custom response frames.

Because a distributed SDN controller can perform more efficiently using custom applications, an SDN-based ARP attack detection solution is considered to be more advantageous than others, as shown in Table 1. Note that although both the active and SDN-based solutions are inexpensive, the latter is distinctively advantageous because of its controllability, which allows cloud centers to be more resilient during the aforementioned types of attacks.

### 3 Features of ARP Attacks

ARP frames can be divided into two types: request and response frames. MAC and IP addresses of a requestor and the IP address of the desired respondent are stored in ARP request frames, whereas MAC and IP addresses of the requestor and desired respondent are stored in ARP response frames. In addition, ARP frames are encapsulated in a data link layer frame that contains MAC addresses corresponding to these frames. Using the global ARP cache in the SDN controller, we can obtain some obvious ARP attacker features.

In a two-layer network, if one VM receives an ARP frame and the source MAC address is different from that in the data link layer frame, we can deduce that this ARP frame is forged; the VM that sent the frame may be an ARP attacker. In a normal ARP running process, a VM sends an ARP request frame, and the rest of the VMs receive one broadcast frame. The desired respondent then sends one ARP response frame, and the requestor receives this frame. Normally, the number of ARP frames that one VM sends should not exceed the number of frames that it receives. If a VM violates this regulation when an SDN controller is used, then this may be an ARP attacker.

In addition, ARP attackers send numerous forged ARP frames, which result in the mapping of some IP addresses to wrong MAC addresses. Three schemas describe IP-MAC pair-mapping options: 1-1, 1-N, and N-1; here, N means multiple. In a two-layer network, a mapping schema should strictly be 1-1. If one ARP frame activates other mapping schemas as options, the host that sent it may be an ARP attacker. Thus, we can define four basic ARP attacker features:

- The MAC address in an ARP frame is different from that in a data link layer frame of the same network packet;
- The number of ARP frames that a host sends is greater than the number of ARP frames that a VM receives;
- An ARP frame results in a 1-N mapping schema appearing in the global ARP cache; and
- An ARP frame results in an N-1 mapping schema appearing in the global ARP cache.

When we could not decide an ARP frame’s feature, we considered it a normal feature.
4 ARP Attacker Detection Algorithm Based on Prediction Probability

In terms of an ARP attack, the global IP-MAC mapping cache can help identify the attacker. However, because its viability depends on full system cooperation of all ends, the presence of existing attacks would not be detected. This accounts for its limited consideration in traditional ARP attack detection solutions. However, with SDN technology, it is possible for SDN controllers to collect all ARP frames to establish a knowledge base about the global IP-MAC mapping cache. However, sometimes, transmission errors in a network and changes in the host configuration may lead to MAC-IP mapping cache chaos. Under these circumstances, legal hosts may be mistakenly identified as attackers. Thus, by increasing the accuracy associated with this technology, we propose an improved iterative algorithm based on the Bayesian theorem.

4.1 The probability prediction algorithm

The global MAC-IP mapping cache could reveal attack features of network hosts, such as duplicate IP addresses or mismatched MAC addresses. These can be detected by checking the MAC-IP mapping cache and by examining other information about a network. Given that the cloud center network is denoted as set \( T = \{ t_i \mid 1 \leq i \leq m \} \), if \( m \) host features are assumed to be in the network, which consists of both attacker and normal features, we can set the prior probability of the host \( C_j \) as \( P(C_j) \) and a non-attacker one as \( P(\overline{C}_j) \) based on prior knowledge. Additionally, the conditional probability can be expressed as \( P(T_x \mid C_j) \) and \( P(T_x \mid \overline{C}_j) \), \( 1 \leq x \leq m \). And \( P(C_j) \) and \( P(\overline{C}_j) \) satisfy Formula (1):

\[
P(C_j) + P(\overline{C}_j) = 1 \quad (1)
\]

When the SDN controller detects the appearance of feature \( T_x \), the posterior probability, \( P(C_j \mid T_x) \), can be calculated with the Bayesian theorem Formula (2):

\[
P(C_j \mid T_x) = \frac{P(C_j)P(T_x \mid C_j)}{P(C_j)P(T_x \mid C_j) + P(\overline{C}_j)P(T_x \mid \overline{C}_j)} \quad (2)
\]

If the SDN controller’s decision about whether \( C_j \) is an attacker is only dependent on the value of \( P(C_j \mid T_x) \), the probability of an erroneous decision being returned can be high. Consequently, the subsequent impact on related hosts would be significant. To address this limitation, we record the consecutive features of \( C_j \), denoted by set \( S \), which is defined as \( S = \{ S_i, t \leq i \leq t + a, S_i \in T \} \). Then, we utilize the conditional probability \( P(S \mid C_j) \) to calculate the posterior probability \( P(C_j \mid S) \). Determining whether \( C_j \) is an attacker with more \( T_x \) returns more accurate in set \( S \). However, more features require more corresponding conditional probabilities. This requires a large storage capacity to accommodate a larger number of features. In addition, some multiple conditional probabilities will be zero, thereby complicating computational progress. To simplify this process, we assume that the appearance of each feature is mutually independent and use an iterative algorithm to calculate the posterior probability of \( P(C_j) \) when multiple features occur in a sequence. The algorithm is derived as follows.

Firstly, we obtain Formula (3) from Bayesian theorem:

\[
P(C_j \mid S_1 \ldots S_t \ldots S_t + a) + P(\overline{C}_j) \mid S_1 \ldots S_t \ldots S_t + a) =
\]

\[
P(C_j)P(S_1 \ldots S_t \ldots S_t + a \mid C_j) + P(\overline{C}_j)P(S_1 \ldots S_t \ldots S_t + a \mid \overline{C}_j) +
\]

\[
P(\overline{C}_j)P(S_1 \ldots S_t \ldots S_t + a \mid \overline{C}_j) + P(\overline{C}_j)P(S_1 \ldots S_t \ldots S_t + a \mid \overline{C}_j) \quad (3)
\]

If the feature \( T_x \) appears in \( C_j \), we take the posterior probability \( P(C_j \mid T_x) \) as the correction probability of \( P(C_j) \). When the next feature occurs, we continue to use Formula (2) to calculate the posterior probability of \( P(C_j) \). Thus, we only retain the most current \( P(C_j) \) value and all of the single conditional probabilities, which simplifies the computation.

Secondly, our iterative prediction probability Formula (4) is expressed as follows:

\[
P(C_j^{i+1}) = P(C_j^i \mid T_x) =
\]

\[
P(C_j^i)P(T_x \mid C_j^i)P(C_j^i \mid T_x \mid C_j^i) + P(\overline{C}_j)P(T_x \mid \overline{C}_j) \quad (4)
\]

Next, we have the following descriptions of the algorithm.

**Lemma 1** If features show up independently from each other, the improved iterative algorithm is an alternative form of calculation based on the Bayesian theorem. In this case, they both return the same result.

**Proof** The features are independent from each other, so we have
\[
P(S_t \cdots S_{t+a} | C_j) = P(S_t \cdots S_{t+a-1} | C_j) P(S_{t+a} | C_j)
\]
and
\[
P(C_j^{t+a} | S_t \cdots S_{t+a-1}) = P(C_j^{t+a-1} | S_t \cdots S_{t+a-1}).
\]

Then,
\[
P(C_j^{t+a}) = P(C_j^{t+a} | S_t \cdots S_{t+a}) = \frac{P(C_j^{t+a-1} | S_t \cdots S_{t+a-1})}{P(C_j^{t+a-1}) P(S_t \cdots S_{t+a-1} | C_j^{t+a-1})} X \frac{P(C_j^{t+a-1} | S_t \cdots S_{t+a-1})}{P(S_t \cdots S_{t+a-1} | C_j^{t+a-1})} X \frac{P(S_t \cdots S_{t+a-1} | C_j^{t+a-1})}{P(S_t \cdots S_{t+a-1} | C_j^{t+a-1})}
\]

In the equation above, \( P(C_j^{t+a-1} | S_t \cdots S_{t+a-1}) \) is the posterior probability of \( P(C_j) \) when \( S_t \cdots S_{t+a-1} \) features occur. When the feature \( S_{t+a} \) occurs, the calculation for posterior probability \( P(C_j^{t+a}) \) still follows the Bayesian theorem. If we calculate the posterior probability in the order of the features appear, we can learn when multiple features appear in succession. The result that is obtained is the same as that calculated from the Bayesian theorem. Therefore, the proposition is proven.

We can use this iterative algorithm to calculate the posterior probability of \( P(C_j) \) when multiple features occur, giving us the prediction probability of whether \( C_j \) is an attacker or not.

If we apply the probability prediction algorithm in cloud centers, the SDN controller can analyze the network feature of \( C_j \) when \( C_j \) sends packets. Then the probability of \( C_j \) being an attacker \( P(C_j) \) can be updated using Formula (4).

**Lemma 2** If \( P(C_j) \) increases when a feature \( T_x \) of \( C_j \) occurs, the probability of an attacker causing this feature will be greater than that of a non-attacker, i.e., \( P(T_x | C_j) > P(T_x | \overline{C}_j) \).

**Proof** For \( P(C_j) + P(\overline{C}_j) = 1 \), we can solve the following equation by applying Formula (4),
\[
\frac{P(C_j)}{P(C_j-1)} = \frac{P(C_j^t | T_x)}{P(C_j-1)} = \frac{P(T_x | C_j^t)}{P(T_x | C_j-1)} = \frac{P(T_x | C_j^t)}{P(T_x | \overline{C}_j^t)} - 1.
\]

If \( P(T_x | C_j^t) > 1 \), then
\[
[P(T_x | C_j^t) - P(T_x | \overline{C}_j^t)] > 0.
\]

Since \( P(C_j) < 1 \), we know that
\[
P(T_x | C_j^t) > P(T_x | \overline{C}_j^t).
\]

Hence, the proposition is proven.

Assuming that a feature \( T_x \) is an obvious feature of an attacker, i.e., \( P(T_x | C_j) > P(T_x | \overline{C}_j) \), the appearance of \( T_x \) will increase the probability \( P(C_j) \). Otherwise, if a feature \( T_x \) is an obvious feature of a non-attacker, i.e., \( P(T_x | C_j) < P(T_x | \overline{C}_j) \), the appearance of \( T_x \) will decrease the probability \( P(C_j) \). ■

**Lemma 3** When the value of \( P(C_j) \) is close to 1, the value of \( P(C_j) \) rarely fluctuates significantly. Accordingly, when the probability of \( C_j \) being an ARP attacker is close to 1, \( C_j \) should be judged as an attacker.

**Proof** According to Formula (4), if we assume that \( P(C_j^t) \) is close to 1, then,
\[
\lim_{P(C_j^t) \to 1} \Delta P(C_j^t) = \lim_{P(C_j^t) \to 1} \frac{P(C_j^t) - P(C_j^{t-1})}{P(C_j^{t-1})} = 1 - 1 = 0.
\]

From the equation above, we know when \( P(C_j) \) is close to 1, \( \Delta P(C_j) \) is close to 0. The resultant numerical size of \( P(C_j) \) is stable. ■

**Lemma 4** When the probability of \( C_j \) being an attacker \( P(C_j) \) approaches 0, \( C_j \) is considered to be a non-attacker. The appearance of any new feature will increase the numerical value of \( P(C_j) \) rapidly.

**Proof** According to Formula (4), we assume that \( P(C_j^{t-1}) \) is close to 0, then the changing rate of \( P(C_j^t) \) is
\[
\lim_{P(C_j^{t-1}) \to 0} \Delta P(C_j^t) = \lim_{P(C_j^{t-1}) \to 0} \frac{P(C_j^t) - P(C_j^{t-1})}{P(C_j^{t-1})} = \frac{P(T_x | C_j^{t-1})}{P(T_x | \overline{C}_j^{t-1})} - 1.
\]

Based on the equation above, when \( P(C_j) \) is close to 0, the appearance of any feature will change \( P(C_j) \) by
\[
\frac{P(T_x | C_j^{t-1})}{P(T_x | \overline{C}_j^{t-1})} - 1 \times \text{times. The appearance of } T_x \text{ will}
\]
lead to a rapid increase in $P(C_j)$, especially if $T_x$ is an obvious feature of an attacker.

Lemmas 3 and 4 conform to this logic. If $C_j$ is identified as an attacker, the presence of any normal feature will not change the result. Otherwise, if it is uncertain as to whether $C_j$ is an attacker, the presence of any attacker feature will increase its suspicion.

4.2 The attacker detection algorithm

When the SDN controller refreshes $P(C_j)$, it is possible to decide whether $C_j$ is an attacker based on the numerical value of $P(C_j)$. We conclude that $C_j$ is an attacker if the following condition is satisfied in Formula (5).

$$P(C_j) > P_t, \quad 0 < P_t < 1$$

(5)

$P_t$ is the threshold, its size has an influence on misjudgment and the probability of omission. To decrease these adverse effects, a method to dynamically adjust the threshold is needed.

In this paper, we record the number of attackers, denoted as $A$, detected in fixed time period. We then adjust the value of $P_t$ according to Formula (6),

$$P_t = \begin{cases} 
1, & 1 \leq P_t + (\alpha - A)\mu; \\
P_t + (\alpha - A)\mu, & P_{\text{min}} < P_t + (\alpha - A)\mu < 1; \\
P_{\text{min}}, & P_t + (\alpha - A)\mu \leq P_{\text{min}} 
\end{cases}$$

(6)

In Formula (6), $\mu$ is the step size, $\alpha$ is the expected number of attackers, and $P_{\text{min}}$ is the lower limit of $P_t$.

Host configuration changes and network transmission errors in a cloud center will increase the probability of suspicion with respect to a host. Therefore, if $A$ is smaller than $\alpha$, $P_t$ needs to be increased to reduce misjudgment. In contrast, if the number of attackers $A$ is larger than $\alpha$, we need to decrease $P_t$ to detect attackers more rapidly.

When attackers are found, the SDN controller should block their network communication, and broadcast the ARP packet that contains the correct IP-MAC information.

A schematic of the described ARP attacker detection model is described in Fig. 1.

5 SDN-Based Implementation

The most widely used SDN technology interface protocol is OpenFlow[6,7]. In this study, we deploy an ARP proxy module in the SDN controllers of an Openflow-based network. The attacker detection method and attacker punishment actions are added to the module to complete the ARP attack defense mechanism. We assume that all VMs are connected to the Openflow-based virtual switch of physical machines, which ensures that packets sent by attackers can be controlled.

There are four types of global information saved in the ARP proxy module. First, information about the number of packets that the VMs receive and send is recorded in set $W$, where the element is denoted as $(C_j, Q_j, R_j)$. Then, the IP-MAC mapping information that was recorded based on ARP packets is defined as set $G$ and the element as $(C_j, IP_j, MAC_j)$. Thirdly, the predicted probability of each VM being an attacker is recorded in set $P$, where every element is denoted as $P(C_j)$. Finally, temporal information that corresponds to when a VM is considered to be an attacker is denoted by set $Z$. $C$ is the identifier set of VMs and is determined by the switch and port number. $Q$ is the set of numbers that refers to the quantity of ARP frames sent by VMs, while $R$ is the set of ARP packets received by VMs. IP and MAC represent the set of source IP and source MAC addresses in ARP packets sent by VMs, respectively.

The pseudo-code of the ARP proxy module in the SDN controller is shown in Figs. 2 – 5. The initial probability of $P(C_j)$ is $P_t/2$. The time duration of recording the attacker is $t$.

6 Simulation

We conducted our simulation using Mininet and Floodlight to simulate the SDN network environment.
To simulate an ARP attack, each host will change its network configuration and send an ARP frame. We deployed an Openflow switch and 1000 host nodes. In the SDN network simulation environment, there is a 1-N or N-1 mapping occurring. The last one is a normal feature. We initialize the prior and conditional probabilities in two scenarios as shown in Table 2. In practice, the prior probability and conditional probability can be initialized as the real network status and continually optimized.

6.1 Misjudgment probability experiment

To test the misjudgment probability associated with the detection model, the model was evaluated 200 times and an average of the different β and P_i was calculated, respectively. First, we compared the misjudgment probability obtained with the theoretical misjudgment probability when P(C) was refreshed at different times. The refresh time of the P(C) of host C is defined as how many ARP frames host C sent. The first function Main defines the main function that receives ARP frames and updates the ARP proxy function. The second function Function 1: Receive one ARP request frame receives ARP request frames and sends ARP request packets to the destination IP address in ARP packet. The third function Function 2: Receive the ARP response packet receives ARP response packets and updates the ARP proxy function. The fourth function Function 3: Refresh P(C) updates the prior probability P(C) based on the number of ARP frames sent by each host.
values of $\beta$ ranged from 1% to 10%, with a 1% increase per test value, and the values of $P_t$ were 1% and 32%. The results are shown in Figs. 6–13.

The theoretical misjudgment probability within the first five refresh times is achieved by the permutation and combination of the probability calculation using our detection algorithm. At the end of the different steps, the ARP attacker features are made to occur in a normal host and in the rest refresh time, the ARP attacker features do not occur. With this setup, the misjudgment probability at different refresh times can be determined and are summed to acquire the theoretical misjudgment probability.

From the experimental results, we know that the value of $P_t$ has some influences on the misjudgment probability. In particular, when $P_t$ is 1%, the influences are more obvious. In Figs. 6–8, the average of the differences between two kinds of values are about 2% and 0.8%. In Figs. 10–12, the average of the differences are about 1.2% and 0.5%. They are all quite small, which correspond with the validation results of
Huan Ma et al.: Bayes-Based ARP Attack Detection Algorithm for Cloud Centers

Fig. 12 The misjudgment probability under serious attack and $P_t=10\%$.

Fig. 13 The misjudgment time under serious attack and $P_t=10\%$.

our experiment. Additionally, the different values of $P_t$ were found to affect the misjudgment probability. For example, the misjudgment probability in Fig. 6 is greater than that in Fig. 8 as $P_t$ in Fig. 6 is smaller than that in Fig. 8. The same was observed in the results presented in Figs. 10 and 12. Due to the smaller $P_t$ value, our detection model does not contain enough information to definitively determine whether one host is an ARP attacker or not; this is because the prediction probability has already reached the decision threshold.

Furthermore, we find that slight and serious attack situations will also affect the misjudgment probability. When a network is under slight attack and the attack features are present, the rate at which $P(C)$ changes will be slower than that under a serious attack. Consequently, our detection model has enough information to decide whether one host is an ARP attacker or not; this is because the prediction probability has already reached the decision threshold.

Comparing the results presented in Figs. 7 and 9 to those in Figs. 11 and 13, it is observed that $P_t$ is $1\%$, the misjudgment is returned earlier than when $P_t$ is $10\%$. This proves that the aforementioned conclusion is valid. As the initialization of $P(C)$ is $P_t/2$, when host $C$ does not contain a normal network feature, it is easy to misjudge the situation. Consequently, there are some misjudgments associated with when refresh times of $P(C)$ is 1 or 2. When we set the initial $P(C)$ to $0.01\%$, $P_t$ is $1\%$ and $\beta$ is $10\%$, we obtain results under a serious attack scenario, as shown in Fig. 14. We find that the lower the initial $P(C)$, the more significantly the misjudgment occurrence time will decrease.

To locate the onset of an ARP attack rapidly in a real network with minimized instances of misjudgment, the serious attack probability setting is best and the initial $P(C)$ should be lower. However, we must evaluate the network error or VM migration probability and trade off the probability setting and real network attack status.

6.2 Omission probability experiment

In this section, we obtain the omission probability of our detection model under different situations. We set the $P(C)$ refresh time as the evaluation criteria. To reduce the misjudgment probability, we set the initial $P(C)$ value to $P_t/10$ and $\beta$ is defined as $0.1$. There are 100 ARP attackers sending forged ARP frames in the simulation network.

In Fig. 15, when the $P(C)$ refresh time reaches 4, the omission probability is almost 0 under all of the different situations evaluated. Since an ARP attack
lasts for a short period of time and the normal ARP frames’ number is precious few. Consequently, the ARP attacker can be found quite rapidly.

6.3 Dynamic threshold algorithm experiment

To test the effectiveness of the dynamic threshold algorithm, we calculated the misjudgment probability under slight attack. The value of $\beta$ was 10% and the value of $P_t$ was 32%. The values of parameter $\mu$ were 0.05 and 0.1. The values of $\alpha$ were 1 and 2. The result is shown in Fig. 16.

The misjudgment probability shows a significant decrease after using the dynamic threshold algorithm. In addition, the higher the values of $\mu$ and $\alpha$, the smaller the misjudgment probability.

In addition, we established a host that regularly sent fake ARP frames with 80% probability. Combined with the dynamic threshold algorithm, we recorded the number of ARP packets the attacker sent under slight attack and serious attack. We tested each case 200 times. The results are shown in Fig. 17.

The results show that an attacker can be detected quickly under both slight and serious attacks. However, in the case of slight attack, the probability did not change quickly, thus, it took longer time to find the attacker. Eventually, the omissions in each case were minimal.

6.4 Comparison of different ARP detection methods’ comparison

In contrast, we applied a decision tree and MR-ARP[13] to detect the attacker. In MR-ARP, a voting mechanism is used to verify the validity of information associated with an ARP frame. We install a voting agent in each host and set the voting hosts’ number to 100. When the accuracy of the voting process is less than 0.80, we consider the host that sent the particular ARP frame to be forged. The test lasts ten seconds and the ARP cache that is retained never expires. There is no network error and $\beta$ is 10%. $P_t$ is 10%. The initial $P(C)$ is $P_t/10$. The attack situation is slight. The $P(C)$ refresh times is 5. The simulation will count the number of forged poisoned hosts at different times with a range of methods. The results are shown in Fig. 18.

In Fig. 18, with the help of a global ARP cache and centralized controller, the SDN-based method and decision tree were found to be capable of preventing the ARP attack and corrected the adverse affected ARP cache. At the beginning of the test, where there was a lack of global ARP information, the SDN-based method and decision tree was unable to effectively distinguish the forged ARP frames; this resulted in a few adversely affected nodes. Since the decision tree could rely on one ARP frame to make a decision, compared with the SDN-based method that must calculate the prediction probability, the first method results a fewer number of adversely affected hosts. Nevertheless, this leads to a higher omission probability than the SDN-based method. However, the MR-ARP method just uses the voting information, and part of the global IP-MAC mapping information, to verify the accuracy of ARP frames. The use of this combination of information is associated with a certain probability of failure. At the beginning, it can prevent some forged ARP frames, but not all. Moreover, MR-ARP could not prevent the ARP attack or correct the affected ARP cache. Consequently, the number of affected hosts will continually increase.

When the number of ARP attackers is large, the failure of voting will be greater and the increase of adversely affected hosts will be more rapid. As a result, the MR-ARP will lose efficacy, while the SDN-based
method and decision tree will still work. From a long-term perspective of long-term running, the global ARP cache is useful for ARP attack detection.

6.5 Impact on network

In addition, the method we proposed has little impact on network traffic. The establishment of a global ARP cache can reduce the number of broadcast ARP request packets. This is described in Table 3. \( X \) is the number of all hosts. \( x \) is the number of voting nodes.

As shown in Table 3, the original ARP will use \( X \) frames to broadcast the ARP request and one frame to respond to the request. The method we proposed just needs one ARP request frame and one ARP response frame to obtain the IP-MAC mapping, which has little impact on network traffic and significantly reduces the ARP broadcast. Based on the original ARP, MR-ARP will send \( x \) voting requests and receive \( x \) voting responses. Furthermore, the number of packets that active ARP detection uses is two more than the original ARP. This is because it will communicate with the destination node after receiving the ARP response frame. The establishment of a global ARP cache can reduce the number of broadcast ARP request packets.

7 Conclusions and Future Work

To resolve the problem of internal network attacks within cloud centers, this study proposes a Bayes-based prediction probability algorithm and an attacker detection method. This method uses SDN technology to process ARP packets and control the communication of the entire internal network. With SDN technology, we were able to distinguish attack and normal features, thereby identifying any attacker in any SDN-based network. Experiment-based results showed that the proposed method we proposed can effectively decrease the misjudgment probability with few omissions.

If the attack frequency is relatively low and the attack feature is absorbed into the normal feature, our method may not be able to identify the attacker and consider the attack as a network error. Our future work aims to optimize the method with the consideration of this current limitation and test the attacker detection model in a real environment.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Nos. 61472033, 61370092, and 61272432).

References

[1] R. Miao, M. Yu, and N. Jain, NIMBUS: Cloud-scale attack detection and mitigation, in Proceedings of the 2014 ACM Conference on SIGCOMM, New York, USA, 2014, pp. 121–122.
[2] S. Alarifi and D. Wolthusen, Mitigation of cloud-internal denial of service attacks, in Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on, Oxford, UK, 2014, pp. 478–483.
[3] D. C. Plummer, Rfc 826: An ethernet address resolution protocol, InterNet Network Working Group, 1982.
[4] S. B. Rathod and V. K. Reddy, Secure live vm migration in cloud computing: A survey, International Journal of Computer Applications, vol. 103, no. 2, pp. 18–22, 2014.
[5] C. L. Abad and R. Bonilla, An analysis on the schemes for detecting and preventing arp cache poisoning attacks, in Distributed Computing Systems Workshops, 2007. ICDCS07. 27th International Conference on, Toronto, Canada, 2007, p. 60.
[6] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, On scalability of software-defined networking, Communications Magazine, IEEE, vol. 51, no. 2, pp. 136–141, 2013.
[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, Openflow: Enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.
[8] M. M. Dessouky, W. Elkilani, and N. Alfishawy, A hardware approach for detecting the ARP attack, in Informatics and Systems (INFOS), 2010 The 7th International Conference on, Cairo, Egypt, 2010, pp. 1–8.
[9] F. Barbhuiya, S. Biswas, N. Hubballi, and S. Nandi, A host based des approach for detecting arp spoofing, in Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on, Paris, France, 2011, pp. 114–121.
[10] M. V. Tripunitara and P. Dutta, A middleware approach to asynchronous and backward compatible detection and prevention of arp cache poisoning, in Computer Security Applications Conference, 1999. (ACSAC’99) Proceedings, 15th Annual, Phoenix, UK, 1999, pp. 303–309.
[11] W. Lootah, W. Emck, and P. McDaniel, Tarp: Ticket-based address resolution protocol, Computer Networks, vol. 51, no. 15, pp. 4322–4337, 2007.
[12] D. Bruschi, A. Ornaghi, and E. Rosti, S-arp: A secure address resolution protocol, in Computer Security Applications Conference, 2003. Proceedings. 19th Annual, IEEE, 2003, pp. 66–74.

[13] S. Kumar and S. Tapaswi, A centralized detection and prevention technique against arp poisoning, in Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on, Kuala Lumpur, Malaysia, 2012, pp. 259–264.

[14] S. Y. Nam, D. Kim, and J. Kim, Enhanced arp: Preventing arp poisoning-based man-in-the-middle attacks, Communications Letters, IEEE, vol. 14, no. 2, pp. 187–189, 2010.

[15] S. Y. Nam, S. Djuraev, and M. Park, Collaborative approach to mitigating arp poisoning-based man-in-the-middle attacks, Computer Networks, vol. 57, no. 18, pp. 3866–3884, 2013.

[16] J. Gao and K. Xia, Arp spoofing detection algorithm using icmp protocol, in Computer Communication and Informatics (ICCCI), 2013 International Conference on, Coimbatore, India, 2013, pp. 1–6.

[17] H. Neminath, S. Biswas, S. Roopa, R. Ratti, S. Nandi, F. Barbhuiya, A. Sur, and V. Ramachandran, A des approach to intrusion detection system for arp spoofing attacks, in Control & Automation (MED), 2010 18th Mediterranean Conference on, Marrakech, Morocco, 2010, pp. 695–700.

[18] P. Pandey, Prevention of arp spoofing: A probe packet based technique, in Advance Computing Conference (IACC), 2013 IEEE 3rd International, Ghaziabad, India, 2013, pp. 147–153.

[19] N. Hubballi, S. Biswas, S. Roopa, R. Ratti, and S. Nandi, Lan attack detection using discrete event systems, IJA Transactions, vol. 50, no. 1, pp. 119–130, 2011.

[20] V. Ramachandran and S. Nandi, Detecting arp spoofing: An active technique, in Information Systems Security, S. Jajodia and C. Mazumdar, Eds. Springer, 2005, pp. 239–250.

Huan Ma received the BS degree in 2011 from University of Science and Technology Beijing, China. Now he is working towards the PhD degree in the School of Computer and Communication Engineering at University of Science and Technology Beijing, China. His main research interests include SDN and scheduling of network resources in cloud computing.

Zhenqiang Mi received his BS degree in automation and PhD degree in communication engineering, both from University of Science and Technology Beijing in 2006 and 2011, respectively. From 2015, he has been an associate professor with University of Science and Technology Beijing. His research interest includes service computing, multi-robot systems, and cloud computing in mobile environments.

James Yifei Yang received the BS degree in electronics and communication engineering from the University of Waterloo, in 2011, and the master’s degree in electronics and communication engineering from the University of Illinois at Urbana-Champaign, in 2013, where he is currently pursuing the PhD degree at the Department of Electrical and Computer Engineering. His research interests are in communication and networks, distributed algorithms, and recommendation systems.

Yang Yang received his PhD degree in information engineering from University of Science and Technology in Lillie, France, in 1988. He has been a professor of University of Science and Technology Beijing since 1988. His research interests include service science and cloud computing, image processing and pattern recognition, and grid technology.

Hao Ding received the BS degree from University of Science and Technology Beijing in 2010 from University of Science and Technology Beijing, China. Now he is working towards the PhD degree both in the School of Computer and Communication Engineering at University of Science and Technology Beijing, China. His main research interests include cloud computing and computer network.

Zenggang Xiong received his PhD degree in computer technology from University of Science and Technology Beijing, China, in 2009. He has been a professor of Hubei Engineering University. His research interest includes computer network, cloud computing, big data, and Internet of things.