PLOS ONE

RESEARCH ARTICLE

A decoy chain deployment method based on SDN and NFV against penetration attack

Qi Zhao[1], Chuanhao Zhang[2,3], Zheng Zhao[4]*

1 Computer Science and Technology College, Jilin University, Changchun, Jilin, China, **2** Department of Public Security Technology, Railway Police College, Zhengzhou, Henan, China, **3** National Digital Switching System Engineering & Technological R&D Center, Zhengzhou, Henan, China, **4** Department of network engineering, Zhengzhou Science and Technology Institute, Zhengzhou, Henan, China

* diyigemsn@hotmail.com

Check for updates

OPEN ACCESS

Citation: Zhao Q, Zhang C, Zhao Z (2017) A decoy chain deployment method based on SDN and NFV against penetration attack. PLoS ONE 12(12): e0189095. https://doi.org/10.1371/journal.pone.0189095

Editor: Hua Wang, Victoria University, AUSTRALIA

Received: August 9, 2017

Accepted: November 18, 2017

Published: December 7, 2017

Copyright: © 2017 Zhao et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: All relevant data are within the paper.

Funding: This study was funded by the Ministry of Public Security Technical Research Plan (2016JSYJB38) and the Scientific and Technological Research Program (172102210441) to Chuanhao Zhang. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Penetration attacks are one of the most serious network security threats. However, existing network defense technologies do not have the ability to entirely block the penetration behavior of intruders. Therefore, the network needs additional defenses. In this paper, a decoy chain deployment (DCD) method based on SDN+NFV is proposed to address this problem. This method considers about the security status of networks, and deploys decoy chains with the resource constraints. DCD changes the attack surface of the network and makes it difficult for intruders to discern the current state of the network. Simulation experiments and analyses show that DCD can effectively resist penetration attacks by increasing the time cost and complexity of a penetration attack.

Introduction

Internet has played an important role in various aspects of society, such as education [1], media [2], payment [3, 4], etc. However, the network security issue is becoming increasingly serious. In recent years, intrusion detection technology has made significant progress [5–7]. However, the current technology is still far from ideal in completely preventing intrusions. With the development of network attack technology and continual appearance of new attack methods, intruders are often able to circumvent security mechanisms and penetrate the network. Especially, the zero-day attacks cannot be defensed effectively.

Zero-day attack is a great challenge for defenders, in which attackers exploit unknown vulnerabilities of their target systems. Effective countermeasures, e.g. patch their systems or configure defense systems, cannot be launched since the defenders have no prior knowledge about the vulnerabilities in zero-day attacks. An example of zero-day attacks is Stuxnet worm [8] in 2010, which exploited four unknown vulnerabilities and compromised industrial control systems without being detected. Therefore, it is very necessary to provide network defenders an additional way to deal with such a risk and ensure the network security.

Penetration attack is a type of attack method that combines various network attack techniques and has explicit intentions, such as obtaining sensitive data, gaining administrative access to the network or destroying the network entirely. A penetration attack has the attribute

of gradualness, meaning that it often attacks network nodes one by one until it reaches sensitive targets. Penetration attacks are a kind of severe threat to the network security.

The honeypot [9] is the main defensive method against penetration attacks, which is an active defense technique aiming at cheating attackers. With a honeypot, attackers are lured to attack decoy nodes, such as decoy hosts or network services, and their attack behaviors can be caught [10]. Honeypot technologies include honeypot [11], honeynet [12] and honeytoken [13], in which fake data or forged applications are used for attracting attackers into traps so that attackers' behaviors can be analyzed and stopped efficiently. However, current honeypot technologies are based on an unrealistic assumption: that the attacks will be stopped as long as attackers are lured into a honeypot. Shakarian et al. [14] proposed a more realistic deception protection method based on moving target defense (MTD) [15]. Rather than stopping the attacks, this method delays the attack time and keeps the probability of a successful attack below a given threshold. However, this method assumes the attackers appear only at a fixed position in the network. In fact, the network can be attacked through multiple positions of the network. What's more, resource constraints are not considered in this method.

Taken together, the above methods are not ideal to defense penetration attacks for two reasons. Firstly, it is difficult for the traditional methods to deploy honeypots due to unavailability of global view of network. Thus, optimized strategies are not optimal. Secondly, traditional methods deploy strategies statically, which is a simplification of reality. In fact, the real attack defense situations are far more complex, and dynamic deployment of strategy can better protect the sensitive targets in networks. However, taking advantage of the SDN and NFV, the global network view can be accessed, thus dynamic service deployment can be achieved. In SDN+NVF architecture[16–18], decoy nodes can be deployed dynamically and efficiently to confuse penetration attackers and protect the network.

In this paper, a Decoy Chain Deployment (DCD) method based on "SDN+NFV" is proposed. DCD monitors the security state of the network globally based on the SDN controller and deploys decoy chains dynamically under certain resource constraints. DCD considers the fact that multiple attack sources and sensitive targets may exist in the network. Moreover, decoy chain strategies are devised based on a simulated annealing (SA) algorithm to maximize the benefit to the network defense.

Related work

The honeypot is a type of active defense technique with significant published researches studying it. The Argos honeypot [19] is built based on a virtual machine, which monitors the real guest OS and traces received network data using extended dynamic taint analysis. Thus, penetration attacks can be detected and attack features can be extracted automatically. Kuwatly et al. [20] proposed an adaptive honeypot system in a dynamic network environment where active detection and passive recognition tools are combined. Virtual honeypots are dynamically configured in this method. A highly interactive honeypot was proposed by Wagener et al [21]. This method learns attack behaviors and changes the configuration itself dynamically so that attackers are attracted into honeypots and their attack behaviors are revealed. An intelligent honeynet based on SDN, called HoneyMix, was proposed by Han et al [22] The programmability of SDN is utilized to conduct fine-grained control of the flows and the attacker is replied with the most desirable response. Unlike the methods mentioned above, DCD is designed to delay penetration attacks and reduces the probability of sensitive targets being compromised by intruders.

DCD shares the idea of MTD, which takes advantage of dynamically changing the attack surface of a system and repels the attack due to the difficulty of ascertaining the system's

current state. MTD aims to break down the assumption made by attackers of a static network and improves the security of the system by the variety [23–25]. Inspired by MTD, Jafarian et al. [26] proposed a host IP hopping method OF-RHM under SDN, which reduces the effectiveness of scanning attacks. RRM, a route hopping method, was proposed by Qi Duan et al. [27, 28] and can protect 90% of traffic flow from sniffing. Badishi proposed RPH [29], a random port hopping method, which can repel DDoS attacks by changing the communication port. Double hopping communication (DHC) was proposed in [30] and is able to defend against sniffer attacks by changing multiple network configurations dynamically. All of the above methods consider protecting the network before a certain attack is launched. However, DCD is designed to address situations when the network is suffering penetrations.

Based on MTD, Clark and et al. [31] proposed a defense strategy in which the IP addresses of decoy nodes can be hopped to prevent attackers from identifying decoy nodes. The IP addresses of both decoy nodes and real nodes are randomly renewed over time based on an optimum strategy determined by a formal analysis. The same team also modeled the interaction between the attacker and decoy nodes based on the game theory, and the optimized IP randomization strategy can be obtained by the equilibrium analyzing [32]. Both of the two methods above prevent attackers from detecting protected nodes by complicating the terminal nodes of the network with the addition of decoy nodes. However, in this paper, penetration attacks are prevented by increasing the complexity of the network topology. A similar decoy-based method was proposed in [14]. In this method, the graphical representation of network's logical layout is analyzed, and both the attack time cost and complexity of attacking have been increased by adding "distraction clusters" in the network. However, this method assumes that the intruder attacks a fixed target at only one position in the network without consideration of resource constraints. Nor does it consider the situation in which the intruders go back into the real network again after they fall into a distraction cluster. In this paper, multiple attack sources and sensitive targets are considered based on "SDN+NFV". We face a more realistic intruder model: intruders may go back into the network again to penetrate more nodes after they fall into a decoy chain.

Model building

DCD deploys decoy chains in the network based on the network paradigm of "SDN+NFV". In SDN, centralized control is adopted, where controller plays a core role. The controller can monitor the security status of the whole network and can find the possible attack sources using instruction detection technology [33, 34], attack trace [35–37] and forensic analysis [38, 39]. Taking advantage of SDN, DCD can deploy decoy chains with the knowledge of global network view and security status. NFV enables dynamic service deployment and rapid service delivery in the network. Combining with NFV, DCD can deploy decoy chains dynamically and efficiently, and deal with dynamic network security risks. DCD deploys decoy chains in the generic servers on the data plane to change the attack surface of the network based on the centralized control of the SDN network.

The intruders might appear on multiple network nodes, as there may be some less protected nodes or multiple potential intruders. And in the network, there are multiple sensitive targets where sensitive data is stored. If one of the sensitive targets is compromised by intruders, the defense of the network fails. Therefore, multiple attack sources and multiple sensitive targets should be considered in the penetration model. In addition, we only consider the penetration attack, where intruders attack networks through the nodes connecting directly.

The penetration topology model

In SDN, a network is defined as a node set $S = \{s_1, s_2, \cdots, s_n\}$, where s_i can be an access switch, a core switch or a middlebox, etc. The penetration attack can be represented as penetration topology model $Z = (S,R,\pi,f,O,T)$, where S is a network system, $R \in S \times S$ is a directed edge set representing the relations between nodes, π is defined as the compromise probability function $S \times S \rightarrow [0,1]$, or the probability of compromising node s' when the intruder has obtained the controlling authority of node S. π has two properties as follows.

$$\text{For} \forall (s, s') \notin R, \ \ \pi(s, s') = 0 \tag{1}$$

$$\text{For} \forall (s, s') \in R, \ \ \pi(s, s') > 0 \tag{2}$$

Formulas (1) and (2) show that if nodes s and s' are not connected directly, intruders cannot penetrate s' from s directly, otherwise they can penetrate node s' with a probability greater than zero. Based on existing security risk metric standards of network device (such as CVSS [40]) and attack graph-based probabilistic security metric [41], the probability of each node in the network being compromised can be evaluated.

The sweetness function f is defined as $S \times S \rightarrow \mathbb{R}^+$, which evaluates the attraction of the node to intruders. The value of $f(s, s')$ represents the willingness of penetrating node s' with the controlling authority of node s. The property of f is shown in formula (3). If s and s' are not directly connected, the sweetness of s' for s is zero.

$$\text{For} \forall (s, s') \notin R, \ \ f(s, s') = 0 \tag{3}$$

Let O and T denote the attack source set and sensitive target set, respectively. Intruders start the network attack by attacking nodes from O until an arbitrary sensitive target in T has been reached. In reality, the network is layered. Let $\mathcal{L}(s)$ denote the layer of node s. We only consider outside intruders that locates at the edge of network, while the sensitive targets are at the innermost layer of the network. Intruders have to penetrate the network layer by layer to get to a sensitive target. Therefore, the penetration topology model is layered as well. We show an example of penetration topology with 3 layers in Fig 1. The intruders (the solid squares) are present at the first layer of the penetration topology and sensitive targets (the hollow squares) are at the third layer.

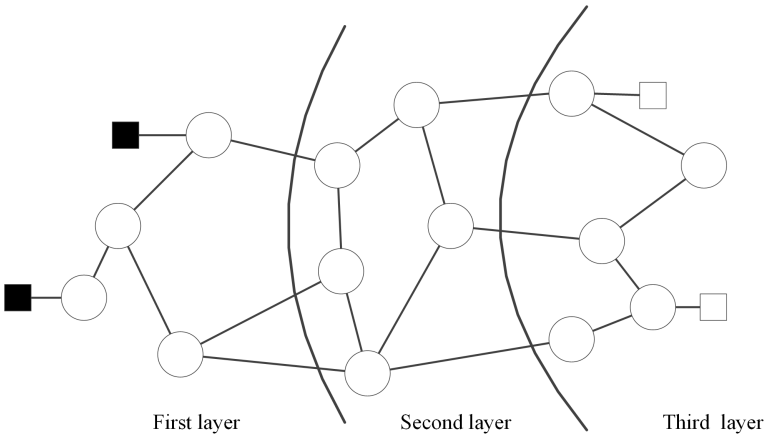

Fig 1. An example of layer structure of penetration topology.

https://doi.org/10.1371/journal.pone.0189095.g001

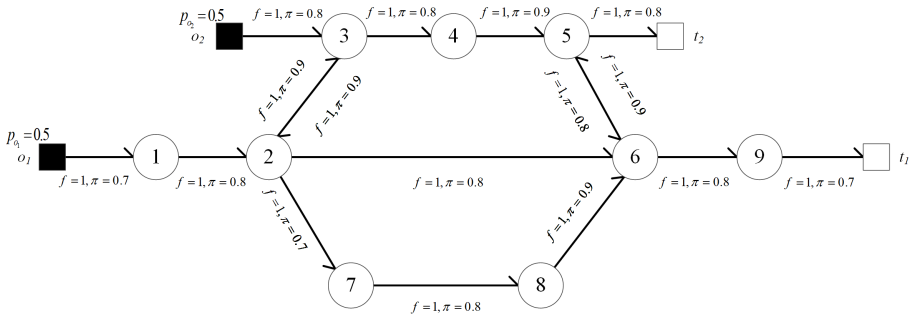

Fig 2. An example of penetration topology. (First layer: node 1, 2, 3; Second layer: node 4, 7, 8; Third layer: node 5, 6, 9).

https://doi.org/10.1371/journal.pone.0189095.g002

A penetration topology is represented as a directed graph and an example of penetration topology is shown in Fig 2. In this penetration topology, the attack source set and the sensitive target set are $O = \{o_1, o_2\}$ and $T = \{t_1, t_2\}$, respectively. Intruders will penetrate the network from the nodes in O with identical probability, i.e. $p_{o_1} = p_{o_2} = 0.5$. π and f are labeled on the directed edges in the penetration topology. The directed edges represent the penetration direction of an attacker.

The attack model

Devices, such as switches and middleboxes could be attacked, if attackers take advantage of their MAC addresses. Attackers could launch specific attacks taking advantage of the vulnerabilities of the devices by sending some carefully constructed packets to them. Vulnerabilities will be triggered, when these devices process the constructed packets. Thus, these devices will be penetrated. Due to the gradualness of penetration attacks, intruders the penetrate network node by node along the paths in the network's penetration topology. In this section, the definition of "penetration path" is described. Then, the formalization description of the penetration attack model is presented. Finally, the traceback of the penetration path is shown.

Penetration path. The node sequence through which the network is attacked by an intruder is called the penetration path, which is denoted as σ and satisfying in formula (4). $\omega(\sigma)$ represents a set of node pairs where two nodes are attacked successively in σ. The length of the penetration path is represented as $|\sigma|$, which is the number of nodes in σ.

$$(s, s') \in \omega(\sigma) \rightarrow (s, s') \in R \wedge \mathcal{L}(s) \leq \mathcal{L}(s') \tag{4}$$

For a penetration path σ, $next(s, \sigma) = \{s' | s \in \sigma \wedge s' \notin \sigma \wedge (s, s') \in R \wedge \mathcal{L}(s') \geq \mathcal{L}(s)\}$ is defined to represent the set of next nodes that can be potentially penetrated at node s on penetration path σ. In the penetration topology as shown in Fig 2, the penetration paths from O to T satisfying $|\sigma| \leq 5$ are $\sigma_1, \sigma_2, \cdots, \sigma_5$. The penetration paths are shown in Fig 3, where the number marked on each directed edge represents the node penetration probability (probability of one node being penetrated, detailed in Section 3.2.2). At the rightmost of corresponding paths, $penPathP(\sigma)$ is path penetration probability (probability of one path being penetrated, described in Section 3.2.2).

Penetration attack model. There are limits to an attacker's abilities due to limited cost he can afford. Moreover, network defense measures, such as intrusion detection, can detect intrusion behaviors. Therefore, the number of nodes that an intruder can penetrate continuously is limited. Let ξ denotes the maximum length of the penetration path, representing the ability of an intruder. Bigger ξ corresponds to greater ability of an intruder. Suppose that in a

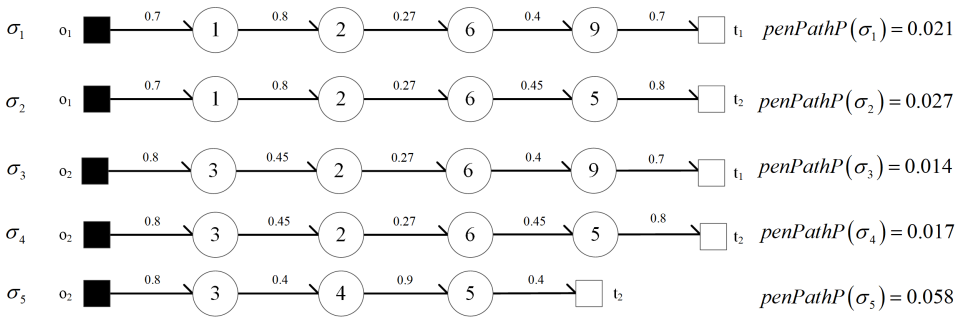

Fig 3. The penetration paths reaching sensitive targets with length less than 5. (The number marked on each directed edge represents the node penetration probability, which is detailed in Section 3.2.2. *penPathP* (σ) is path penetration probability, which is described in Section 3.2.2).

https://doi.org/10.1371/journal.pone.0189095.g003

penetration topology, s is the current node on a penetration path σ ($s \in \sigma$). The probability of an intruder selecting ($s' \in next(s, \sigma)$ as the next node is *selNodeP*, as shown in formula (5). This formula describes the fact that the intruders tend to select a more attractive node to penetrate.

$$selNodeP(s, s', \sigma) = \frac{f(s, s')}{\sum_{s'' \in next(s,\sigma)} f(s, s'')} \tag{5}$$

It can be assumed that for an intruder, selecting s' as the next node to penetrate is independent with compromising s' successfully. Hence, the probability of selecting s' and compromising s' for an intruder can be represented as *penNodeP*, which can be obtained by formula (6), called node penetration probability.

$$penNodeP(s, s', \sigma) = selNodeP(s, s', \sigma) \times \pi(s, s') = \frac{f(s, s') \times \pi(s, s')}{\sum_{s'' \in next(s,\sigma)} f(s, s'')} \tag{6}$$

Given the attack source s_0, an intruder can penetrate the network along penetration path $\sigma = \langle s_0, s_1, \cdots, s_n \rangle$, and reach s_n with probability $penPathP(\sigma)$ calculated from formula (7), called the path penetration probability. $p_{\sigma(0)}$ denotes the probability that the intruder appears at the first node of σ ($\sigma(0) \in O$).

$$penPathP(\sigma) = p_{\sigma(0)} \times \prod_{(s,s') \in \omega(\sigma)} penNodeP(s, s', \sigma) \tag{7}$$

As shown in Fig 3, the path penetration probabilities of $\sigma_1 \sim \sigma_5$ can be calculated based on formula (7). In the penetration topology as shown in Fig 2, the intruder successfully penetrates one node of T from O and satisfies $|\sigma| \leq 5$ with probability $\sum_{i \in [1,5]} penPathP(\sigma_i) = 0.137$.

Traceback of penetration path. An intruder launches a penetration attack from the first layer of the network, then he penetrates the network along the directly connected nodes. The layers of network are public information. The intruder knows the layer he is currently at and always selects a node to penetrate that is at the same or higher layer of the network. If all the nodes directly connected to the current penetrated node have already been compromised or the layers they belong to are lower than the current layer, the intruder will trace back along the penetration path and continue to penetrate the network through the first node with penetrable neighbor nodes.

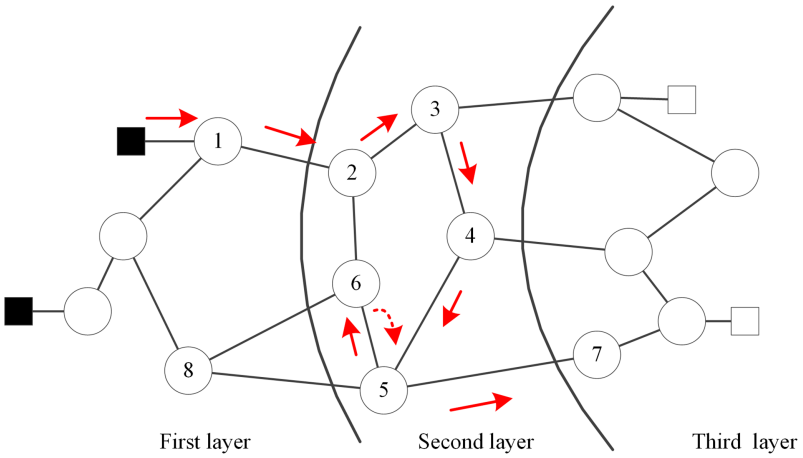

Fig 4. An example of traceback of the penetration path.

https://doi.org/10.1371/journal.pone.0189095.g004

Fig 4 shows an example of traceback of penetration path in a 3-layer network. The intruder starts from the first layer of the penetration topology and launches a penetration attack along the path $\sigma = \langle 1, 2, 3, 4, 5, 6 \rangle$. When the intruder continues to penetrate at node 6, he has to trace back to node 5 to find a new node to penetrate, since all the neighbor nodes except node 8, which is at lower layer, have been compromised. The uncompromised neighbors of node 5 include node 7 and 8. The intruder selects node 7 as the next target to penetrate because the layer of node 8 is lower than node 5. Then, the penetration path is updated with $\sigma' = \langle 1, 2, 3, 4, 5, 6, 7 \rangle$ and $(5,7) \in \omega(\sigma')$. The traceback of the penetration path described here is a realistic assumption of penetration attack.

Decoy chain model

A decoy chain is a one-way sequence of virtual machines that can function as decoy switches, middleboxes or terminal hosts. Once the intruder falls into a decoy chain, he will continue to penetrate the decoy nodes along the decoy chain. Decoy chains are different from traditional honeypots, as they do not stop intruders from attacking the network. Instead, intruders will be attracted into decoy chains so that their attacks are delayed, and the probability of sensitive targets being penetrated is decreased. The decoy chain is defined as $dc = (id, \Pi, F, l)$, where id is the unique identification of a decoy chain. Π is a set of compromised probabilities for the nodes in the decoy chain and F consists the sweetnesses for them. l is length of the decoy chain, or the number of virtual machines in the decoy chain.

There are 3 decoy chains in decoy chain set $DC = \{dc_1, dc_2, dc_3\}$, and their configurations are shown as follow.

$$dc_1 = \begin{cases} \forall \pi \in \Pi, \pi = 0.9 \\ \forall f \in F, f = 1 \\ l = 3 \end{cases} \quad dc_2 = \begin{cases} \forall \pi \in \Pi, \pi = 0.9 \\ \forall f \in F, f = 1 \\ l = 4 \end{cases} \quad dc_3 = \begin{cases} \forall \pi \in \Pi, \pi = 0.8 \\ \forall f \in F, f = 1 \\ l = 5 \end{cases}$$

The length of dc_1, dc_2 and dc_3 are 3, 4 and 5, respectively, and all their sweetnesses are 1. Multiple instances of each decoy chain can be deployed in the network. For the penetration topology in Fig 2, one decoy chain's deployment strategy is shown in Fig 5. In the penetration topology with this decoy chain deployment strategy, the intruder successfully penetrates one

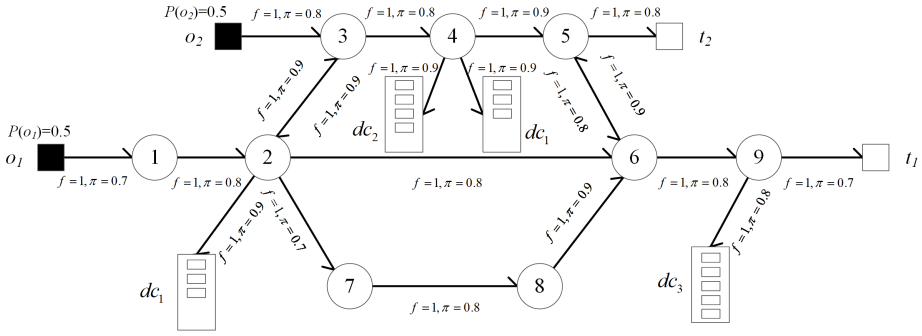

Fig 5. The penetration topology deployed with decoy chains.

https://doi.org/10.1371/journal.pone.0189095.g005

node of T from O and satisfies $|\sigma| \leq 5$ with probability $\sum_{i\in[1,5]} penPathP(\sigma_i) = 0.0781$. The probability is decreased by 43% comparing to the original network shown in Fig 2.

Decoy chain deployment

Within the framework of SDN+NFV, DCD utilizes the server resource to deploy decoy chains under centralized management of the controller. To avoid an intruder penetrating two identical decoy chain instances from one node, we assume that a decoy chain cannot be deployed in one server more than once. Given resource constraints, optimizing the deployment of decoy chains to maximize the benefit to the network defense is a DCD problem.

Decoy chain deployment model

Let Z denote the penetration topology of a SDN network S. Define $V = \{v_1, \cdots, v_n\}$ to denote the servers in the data plane, where one or more virtual machines can run. Assume every forwarding node in Z connects with a server, i.e., for $\forall s_i \in Z$, $\exists v_i \in V \wedge (v_i$ connects with $s_i)$. For potential attack sources and sensitive targets, the SDN controller generates the deployment strategies and deploys decoy chains in the data plane for minimizing the probability that sensitive targets are compromised. Let DC denote the set of decoy chains that are available to defender. Variable $x = \{x_i^j | i \in [1, |V|], j \in [1, |DC|]\}$ denotes one deployment strategy where $x_i^j \in \{0, 1\}$. $x_i^j = 1$ indicates that an instance of $dc_j \in DC$ is deployed in server v_i. Otherwise, no instance of dc_j is deployed in v_i. For Z, the DCD problem is to minimize formula (8), where Z^+ represents the penetration topology deployed with strategy x. Penetration probability $P_{Z^+}^{O,T}(x)$ is the probability that an intruder reaches T from O under the condition of Z^+.

$$\text{Min } DCD(x) = P_{Z^+}^{O,T}(x) \tag{8}$$

Under penetration topology Z^+, the probability of reaching a sensitive target satisfying $|\sigma| \leq \xi$ is shown in formula (9). $Path_{Z^+}^{O,T}(x, \xi)$ represents the set of penetration paths in Z^+ from O to T that have a length no bigger than ξ.

$$P_{Z^+}^{O,T}(x) = \sum_{\sigma \in Path_{Z^+}^{O,T}(x,\xi)} penPathP(\sigma) \tag{9}$$

Due to limited resources, such as number of CPU, memory size, HDD and bandwidth, the number of virtual machines running on a server are limited. In this paper, the resource constraints are simplified and the capacity of a server is used to represent all the resource provided

by a server. The capacity of a server is defined to be the maximum number of virtual machines that could run on it at the same time. For a server $v_i \in V$, its capacity is denoted as c_i; in other words, the number of virtual machines running on v_i is no more than c_i. The server capacity constraint in DCD is shown in formula (10), where $dc_j \in DC$ and l_j denotes the number of virtual machines in dc_j.

$$\text{For} \forall i \in [1, |V|], \sum_{j \in [1,|DC|]} l_j \times x_i^j \leq c_i \tag{10}$$

When multiple decoy chain instances are deployed in a single server, each decoy chain needs to occupy one port connecting to an SDN switch to simulate a real network branch, as shown in Fig 6. As the number of ports on servers and switches are limited, the number of decoy chains that a server can load is limited as well. For example, the load of decoy chains of the server in Fig 6 is less than 3. Given the maximum load h_i of server v_i, DCD satisfies the constraint as shown in formula (11).

$$\text{For} \forall i \in [1, |V|], \sum_{j \in [1,|DC|]} x_i^j \leq h_i \tag{11}$$

In the servers on data plan, other service functions also need to be deployed, such as firewalls and traffic monitors. Therefore, the total capacity that decoy chains can occupy is limited. Suppose that θ is the maximum service capacity utilization rate, then the capacity occupied by decoy chains satisfies the constraint as shown in formula (12).

$$\sum_{i \in [1,V]} \sum_{j \in [1,|DC|]} x_i^j \times l_j \leq \theta \sum_{k \in [1,|V|]} h_k \tag{12}$$

The solution algorithm of DCD problem

Decoy chain deployment is similar to the server chain deployment [42–44], but their targets are different. The target of DCD is to decrease the penetration probability from one attack

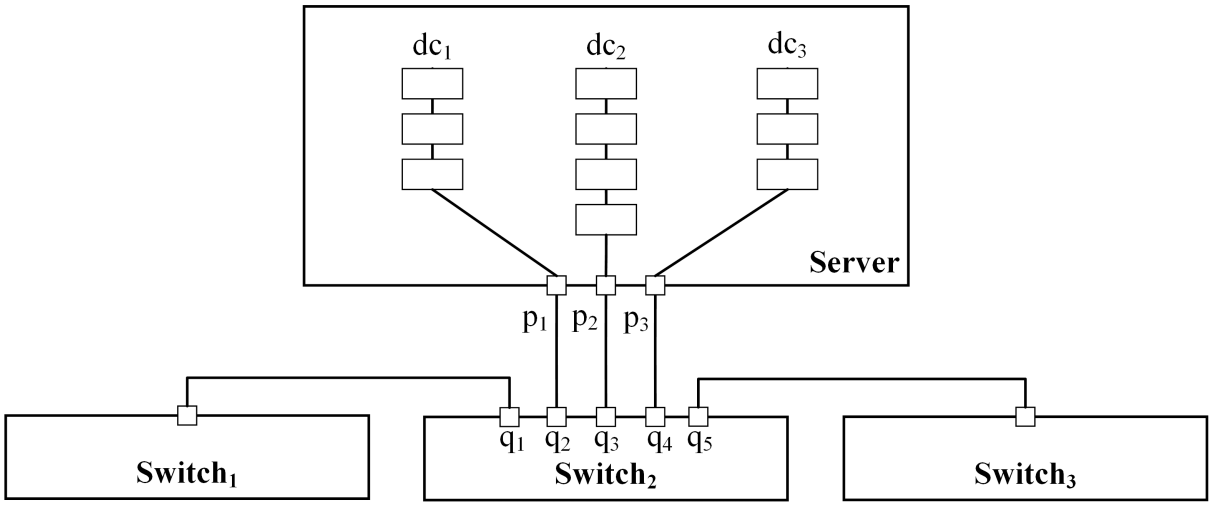

Fig 6. The connection between one server and one switch.

https://doi.org/10.1371/journal.pone.0189095.g006

source node to the sensitive target set. The solution to DCD problem is to find the most optimal position for decoy chains and to maximize the benefit to the network defense under certain resource constraints. An SA algorithm is utilized in this paper to solve the problem.

Penetration probability computation. For a decoy chain deployment strategy x, an intruder could select any attack source to launch a penetration attack. Algorithm *Penetrate-Probability* computes the probability that the intruder reaches a sensitive target in T from O (see Algorithm 1). In this algorithm, line (2) loops for each attack source $o_i \in O$ to get penetration probability from o_i to any sensitive target in T. Line (3) of the algorithm computes the probability that the intruder reaches a sensitive target in T from one attack source o_i. $p^0_{o_i}$ is the probability that the intruder is present at attack source o_i.

RecursionTraversi ng is a recursion algorithm that uses the depth-first search method to compute the penetration probability on each path from an attack source to a sensitive target set, as shown in Algorithm 2. *RecursionTraversing* starts the recursion from node s. If the current node is a sensitive target, it adds the penetration probability to variable *sum* and returns (lines (1)~(3)). Otherwise, it makes recursive call (line (4)~(17)). Firstly, a neighbor node of s in $next(s, \sigma)$ is selected (line (6)). The path is then updated, the penetration probability of the new path is computed and the recursive call is made (line (7)~(9)). If $next(s, \sigma) = \emptyset$, it traces back σ to find a node s' that satisfies $next(s', \sigma) \neq \emptyset$ (line (11)~(15)). The traceback will be stopped if a sensitive target is reached (line (16)~(17)).

Algorithm *RecursionTraversing* goes over all the paths through which an intruder may potentially attack the network. However, it is less applicable for large scale networks because the number of penetration paths increases exponentially with respect to the size of the network. To solve this problem, DCD uses a random sampling method to select penetration paths. As shown in Algorithm 3, *maxCnt* penetration paths are randomly selected in *RandomSampling*, and the penetration probability of these paths are computed. In general, the larger the size of network is, the bigger value of *maxCnt* should be selected, as there are more penetration paths in a larger network. *RandomSampling* is similar to the algorithm proposed in literature [14]. However, we consider the traceback possibility and the layers of the network, which is more realistic. In *RandomSampling*, a node s_i is selected as the penetrated node from set $next(s, \sigma)$ with probability $penNodeP(s,s_i,\sigma)$ for simulating the penetration attack (line (6)~(10)). If $next(s, \sigma) = \emptyset$, it traces back along σ until a penetrable neighbor node is selected (line (12)~(16)). If we use the random sampling method to evaluate penetrate probability from attack source set O to sensitive target set T, we need to replace the function *RecursionTraversing* in line (3) of Algorithm 1 with *RandomSampling*, described in Algorithm 3.

Simulated annealing algorithm. The DCD problem is a resource distribution problem which has $|DC|\times|V|$ decision variables and $2|DC|+1$ constraints. This is difficult to solve using 0–1 programming because of the large number of variables. As an optimization algorithm, SA algorithm [45] is ideal to solve this type of problem. SA is a heuristic method for solving global optimization problems with a large solution space. It can escape from local optima and

Algorithm 1. Penetrate probability from attack source set to sensitive target set.

Input: Z^+, O, T, ξ
Output: $P^{O,T}_{Z^+}$
PenetrateProbability(Z^+, O, T, ξ)
(01) $P = 0$
(02)**for** o_i **in** O
(03) $P \leftarrow P + p^0_{o_i} \times RecursionTraversing\,(Z^+, T, \xi, o_i, \emptyset, 1, 0)$
(04)**return** P

https://doi.org/10.1371/journal.pone.0189095.t001

Algorithm 2. Penetration probability from one attack source node to sensitive target set using recursion traversing.

Input: Z,T,ξ,s,σ,p,sum

Z: Penetration topology;

T: Sensitive target set;

ξ Maximum length of penetration path;

s: An attack source node;

σ: Current penetration path;

p: Probability of a penetration path;

sum: Probability of all the penetration paths from s to T;

Output: $P_Z^{s,T}$

$P_Z^{s,T}$ is the penetration probability from s to T in penetration topology Z within max length ξ.

RecursionTraversing (Z,T,ξ,s,σ,p,sum)

(01)**If** $s \in T$ //For s is a sensitive target

(02) $sum \leftarrow sum + p$

(03) **return** sum

(04)**else if** $|\sigma| < \xi$ //For s is not a sensitive target

(05) **if** $next(s, \sigma) \neq \emptyset$

(06) **for** $s_i \in next(s, \sigma)$

(07) $new\sigma \leftarrow \sigma \bigcup s_i$

(08) $newp \leftarrow p \times penNodeP(s,s_i,\sigma)$

(09) $sum \leftarrow RecursionTraversing(Z^+,T,\xi,s_i,new\sigma,newp,sum)$

(10) **else**

(11) Loop traceback σ to get a node s' **satisfy** $next(s', \sigma) \neq \emptyset$

(12) **for** $S_j \in next(s', \sigma)$

(13) $new\sigma \leftarrow \sigma \bigcup s_j$

(14) $newp \leftarrow p \times penNodeP(s',s_j,\sigma)$

(15) $sum \leftarrow RecursionTraversing\ (Z^+,T,\xi,s_j,new\sigma,newp,sum)$

(16) **if** any sensitive target in T is penetrated

(17) **return** sum

(18)**return** sum

https://doi.org/10.1371/journal.pone.0189095.t002

searches global optimal solution effectively. In addition, SA is also very simple and easy to implement. Therefore, SA is used to solve the DCD problem in this paper. The basic elements are defined as follows.

State expression: the binary-encoded vector is adopted as state expression. $x = \{x_i^j | i \in [1, |V|], j \in [1, |DC|]\}$ represents a state, where $x_i^j \in \{0, 1\}$.

Objective function: SA algorithm is used to solve non-constrained optimization problems. To solve DCD, which is a constrained optimization problem, a punishment function is designed to transfer DCD to an unconstrained optimization problem, as shown in formula (13), where 3 penalty terms are introduced.

$$\begin{aligned} \text{Min } E(x) = P_{Z^+}^{O,T}(x) + \left(\frac{\alpha}{T_k}\right)\sum_{i\in W}\left(\sum_{j\in[1,|DC|]} l_j \times x_i^j - c_i\right)^2 \\ + \left(\frac{\beta}{T_k}\right)\sum_{i\in W'}\left(\sum_{j\in[1,|DC|]} x_i^j - h_i\right)^2 + \varphi\left(\frac{\gamma}{T_k}\right)\left(\sum_{i\in[1,V]}\sum_{j\in[1,|DC|]} x_i^j \times l_j - \theta\sum_{k\in[1,|V|]} h_k\right)^2 \end{aligned} \tag{13}$$

α, β, γ are the penalty factors, and T_k is the temperature parameter. W is the set of servers

Algorithm 3. Penetration probability from one attack source node to sensitive target set using random sampling paths.

Input: Z,T,ξ,s,σ,p,sum

Output: $P_Z^{s,T}$

RandomSampling (Z,T,ξ,s,σ,p,sum)

(01) $cnt = 0$

(02) **while** $cnt < maxCnt$

(03) $\sigma \leftarrow s$

(04) $p \leftarrow p_0$

(05) **while** $|\sigma| < \xi$ **and** $s \notin T$

(06) **if** $next(s, \sigma) \neq \emptyset$

(07) Select s_i from $next(s, \sigma)$ with probability of $penNodeP(s,s_i,\sigma)$

(08) $\sigma \leftarrow \sigma \bigcup s_i$

(09) $p \leftarrow p \times penNodeP(s,s_i)$

(10) $s \leftarrow s_i$

(11) **else**

(12) traceback σ to get a node s' **satisfies** $next(s', \sigma) \neq \emptyset$

(13) Select s_j from $next(s', \sigma)$ with probability of $penNodeP(s',s_j,\sigma)$

(14) $path \leftarrow path \bigcup s_j$

(15) $p \leftarrow p \times penNodeP(s',s_j,\sigma)$

(16) $s \leftarrow s_j$

(17) $sum \leftarrow sum+p$

(18) cnt ++

(19) **return** sum

https://doi.org/10.1371/journal.pone.0189095.t003

with overloaded capacity when x is deployed. W' is the set of servers with overloaded decoy chains when x is deployed. φ is a Boolean variable, which equals 1 when the maximum capacity utilization rate exceeds θ. As the SA runs, T_k will decrease gradually. Accordingly, penalty terms will increase to ensure global searching at the beginning of SA and local searching at the end of SA. $E(x)$ is defined as the energy at state x.

Neighborhood: The neighborhood of x is defined as $N(x)$. Each element $x' \in N(x)$ is obtained by changing the value of an arbitrary $x_i^j \in x$ ($i \in [1,|V|], j \in [1, |DC|]$). At temperature T_k, the state transition probability from x to $x' \in N(x)$ is calculated with the Metropolis criterion [46], as shown in formula (14), where $\Delta E = E(x') - E(x)$. Metropolis criterion models the state transition of a thermodynamic system. In this transition, the energy content is being minimized. Metropolis criterion enables SA to jump out of local optimum and achieve the global optimum solution.

$$P_{T_k}(x,x') = \begin{cases} 1, & \text{if } E(x') \leq E(x) \\ \exp\left(-\dfrac{\Delta E}{T_k}\right), & \text{else} \end{cases} \tag{14}$$

Cooling rule: Cooling rule is used to simulate cooling process, which grantees the convergence of SA and further assures the global optimum solution of DCD problem. In this paper, the selected cooling rule is shown in formula (15), which was proposed in literature [46]. a is a constant slightly smaller than 1, which determines the speed of cooling. The larger the value is,

more slowly the temperature decreases. Empirically, $a = 0.95$. Thus, the temperature drops with a reasonable low speed and prevents SA from being trapped into a local optimum.

$$T(k+1) = a \cdot T(k), \quad k = 0, 1, 2, \cdots \tag{15}$$

The SA for DCD problem is designed as follows.

1. Initialize the parameters of SA. Choose an arbitrary initial solution x. Set annealing temperature as T_0, end temperature as T_f and iterator as k. $T_k = T_0$.
2. For the current solution x, generate a neighborhood solution $x' \in N(x)$. Calculate the increment of the energy value $\Delta E = E(x') - E(x)$.
3. If $\Delta E < 0$, set x = x' and go to step 4, otherwise generate $\mu = U(0,1)$. If $\exp\left(-\frac{\Delta E}{T_k}\right) < \mu$, set x = x'.
4. If thermal equilibrium is attained, go to step 5, otherwise go to step 2.
5. Decrease T_k, $k = k + 1$. If $T_k < T_f$, stop SA, otherwise go to step 2.

Instructions for implementation and simulation experiments

Instructions for implementation

DCD is designed based on the architecture of SDN and NFV and one possible way to implement it is shown in Fig 7. There are three planes in this design: the policy plane, control plane and data plane. The policy plane and the control plane can be built based on the SDN controller. On the control plan, a Topology Generator can be adopted to provide network topology information. The Security Status Monitor can be used to monitor the network security status

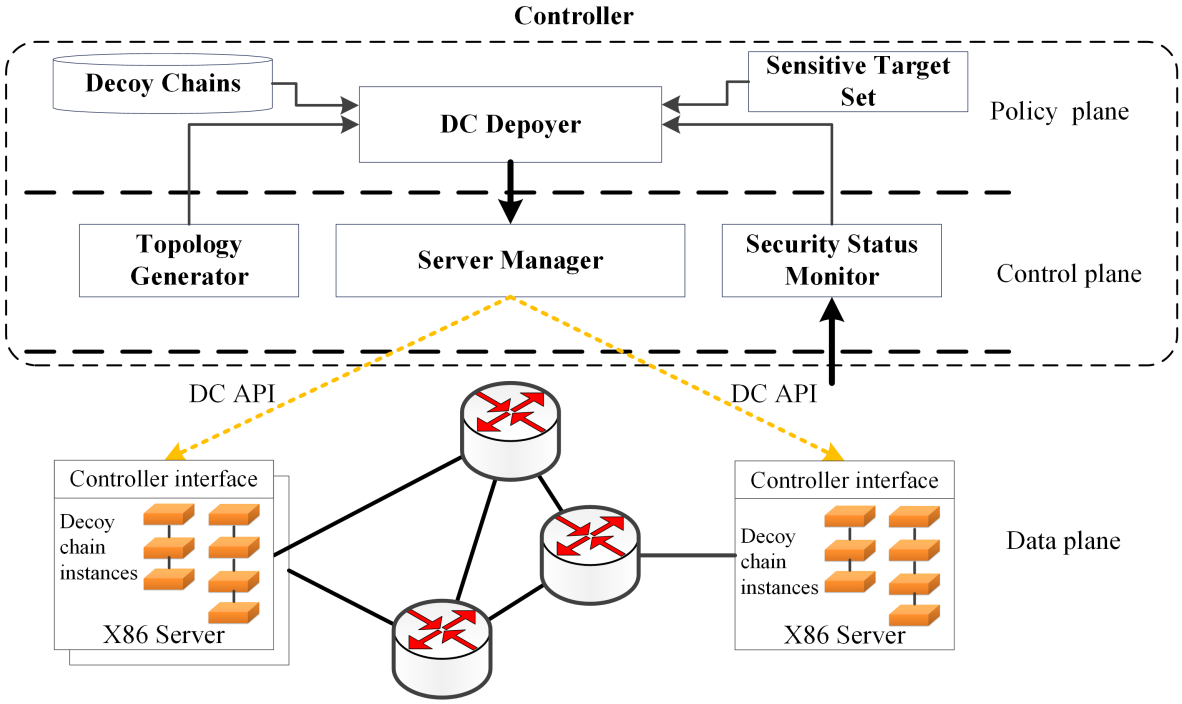

Fig 7. The schematic illustration of DCD system.

https://doi.org/10.1371/journal.pone.0189095.g007

as well as potential attack source information. On the policy plane, Sensitive Target Set can be set based on users' requires to provide sensitive targets information. With these essential information, a DC Deployer could be used to generate the decoy chain deployment strategies and deliver them to the Server Manager. At last, the decoy chain instances can be deployed in servers on the data plane through DC API, a southbound interface.

As the possible implementing scheme described above, the controller can control the data plane, collect security information from the network and quickly response to any network security threats. The controller is also able to directly control servers in the data plane and change the network attack surface by deploying the decoy chains in servers. When the network threat is removed, the controller will delete the decoy chain instances that have been deployed in the servers to decrease the server load and the resource consumption of the network. In order to reduce the latency of decoy chain deployment and economize bandwidth, the decoy nodes can be stored in the servers' disks. When a server receives commands from the controller through DC API, the decoy chains can be constructed quickly. The latency of deployment will be sufficiently shortened and no traffic will be caused by transporting of virtual machines.

Simulation experiments and evaluation

Penetration probability improvement (*PPI*) is introduced in this section to evaluate the effectiveness of DCD, as defined in formula (16). Z is the original penetration topology, while Z^+ is the penetration topology deployed with decoy chains.

$$PPI = \left(1 - \frac{P_{Z^+}^{O,T}}{P_Z^{O,T}}\right) \times 100\% \tag{16}$$

We implemented a DCD simulator with a C program with accordance to realistic networks, penetration attacks and decoy chain deployment. The DCD simulator complies with the basic nature and property of network, as well as realistic attack-defense interaction. In the experiments, BRITE [47], is used as the network topology generator. Four network random topologies are generated based on the Waxman model [47] with 16, 32, 48, 64 nodes (m = 2). All networks in our experiments have four layers, and each layer has the same number of nodes. Attack sources are in the outermost layer of the network, while sensitive targets are in the innermost layer. We assume that the π and f of each node in the network are 1. Every node in the network connects with a server. The capacity of these servers is denoted as c, which follows the distribution as shown in formula (17), where $F(x)$ represents the probability distribution function of normal distribution with $\mu = \sigma = 5$. The maximum load of decoy chains on server h follows the distribution as shown in formula (18), where $F'(x)$ represents normal distribution with $\mu = 3$, $\sigma = 1$.

$$P(c) = \frac{F(c+1) - F(c)}{F(16) - F(0)}, 0 \le c \le 15 \tag{17}$$

$$P'(h) = \frac{F'(h+1) - F'(h)}{F'(6) - F'(0)}, 0 \le h \le 5 \tag{18}$$

Penalty factors α, β, γ are all set to 1 in the experiments. Ten kinds of decoy chains with lengths ranging from 1~10 are used in the experiments. The maximal capacity usage rate θ is set to 0.5. The DCD simulator runs on a 64 bit computing platform with 2.53 GHz Intel Xeon CPU and 32G RAM.

Validation of the effectiveness of DCD. In this experiment, decoy chains are deployed using DCD in the four generated networks. We also generated four networks based on the BA

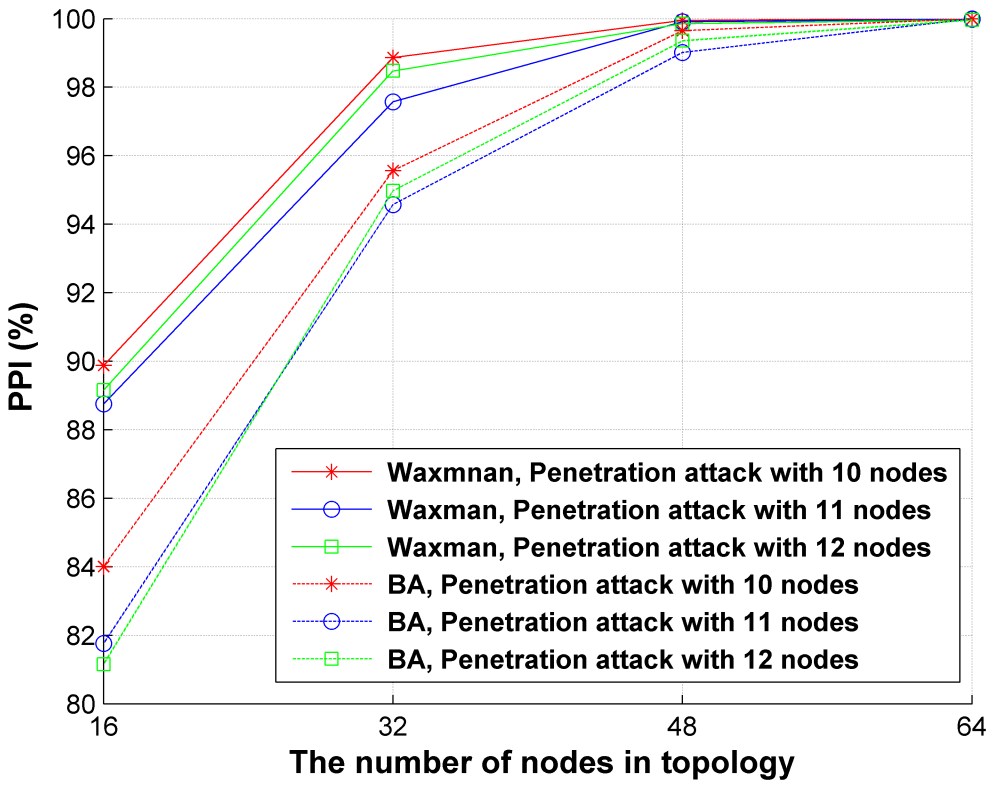

Fig 8. Comparison of *PPI* with different topologies and ξ.

https://doi.org/10.1371/journal.pone.0189095.g008

model [48] using BRITE for comparing the effects of topologies on our method. One attack source and one sensitive target are set in these networks. The maximum length of the penetration path ξ varies between 10~12. The penetration probability is computed using *RandomSampling* ($maxCnt = 10^4$). Multiple tests are conducted, and the average result is computed to eliminate the uncertainty brought by randomness. A series of *PPI* is obtained with different topologies and penetration path lengths, as shown in Fig 8.

As seen, *PPIs* are larger than 80% for all cases, which indicates that the probability of sensitive targets being penetrated by intruders decreased after deploying decoy chains. It also shows that the larger the network size is, the higher the *PPI* can be achieved, resulting in stronger resistance to penetration attacks. The reason is that more decoy chains can be deployed on larger networks, making them more complicated to be penetrated. It can be found that DCD could get similar results on the Waxman model network and BA model network. Therefore, DCD is effective on networks of different models.

Decoy chains are deployed to simulate parts of the network. It is difficult for intruders to distinguish real nodes from decoy nodes when they try to penetrate the network. Assume that an intruder penetrates the network using the random-walk method: he starts from an attack source and selects an adjacent node according to the probability calculated by formula (6). If no adjacent nodes can be selected, he will trace back and continue to select nodes to penetrate until a sensitive target is compromised. The average lengths of penetration paths that compromise sensitive targets in both Waxman networks and BA networks with and without DCD are compared, and results are shown in Fig 9. As we can see, the average length of the penetration path with DCD is 1.069 times greater than that without DCD. A greater length in the penetration path indicates increased attack time cost. The reason is that intruders easily falls into the

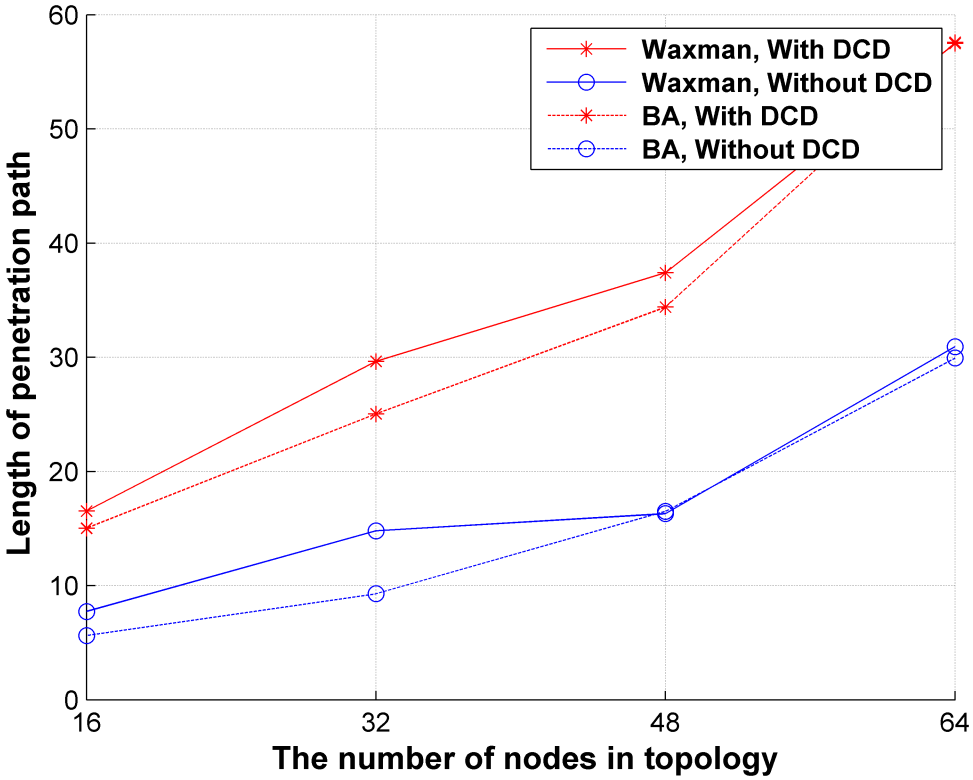

Fig 9. Comparison of the length of penetration paths with and without DCD.

https://doi.org/10.1371/journal.pone.0189095.g009

trap of the decoy chains when they penetrate the network with DCD. The results of DCD upon the two network models are very similar, which means our method is robust to different networks.

Comparison between recursive traversal and random sampling algorithm. A series of experiments are conducted using the generated network topologies to compare the influence on DCD when recursive traversal and random sampling are used. In the experiments, the two algorithms are used to calculate energy in SA, and they generate decoy chain deployment strategies $x^{recursion}$ and x^{sample}, respectively. In this experiment, 10^4 penetration attacks are launched using random-walk to the network with one of the two strategies. The comparison of *PPI* is shown in Fig 10. The horizontal axis represents ξ and the vertical axis represents *PPI*. These results show that the random sampling method can achieve a higher *PPI* in all topologies. This is because the deployment of the decoy chains using the recursive traversal method aims at all penetration paths. On the contrary, the random sampling method tends to deploy decoy chains for the paths that have higher penetration probabilities. Furthermore, these paths with higher penetration can be more likely to be penetrated using random-walking. Therefore, the SA using random sampling better suits the DCD problem and achieves higher *PPI* rate.

The time costs of SA using the recursive traversal algorithm and random sampling algorithm are compared under several generated network topologies. In the random sampling algorithm, we set *maxCnt* to 10^4 and the time costs are shown in Fig 11. It can be concluded that SA using recursive traversal algorithm is more efficient than using random sampling algorithm when the scale of the network is small. The reason is obvious: there are fewer penetration paths in a small-scale network, and those paths can be traversed with less time cost. As the

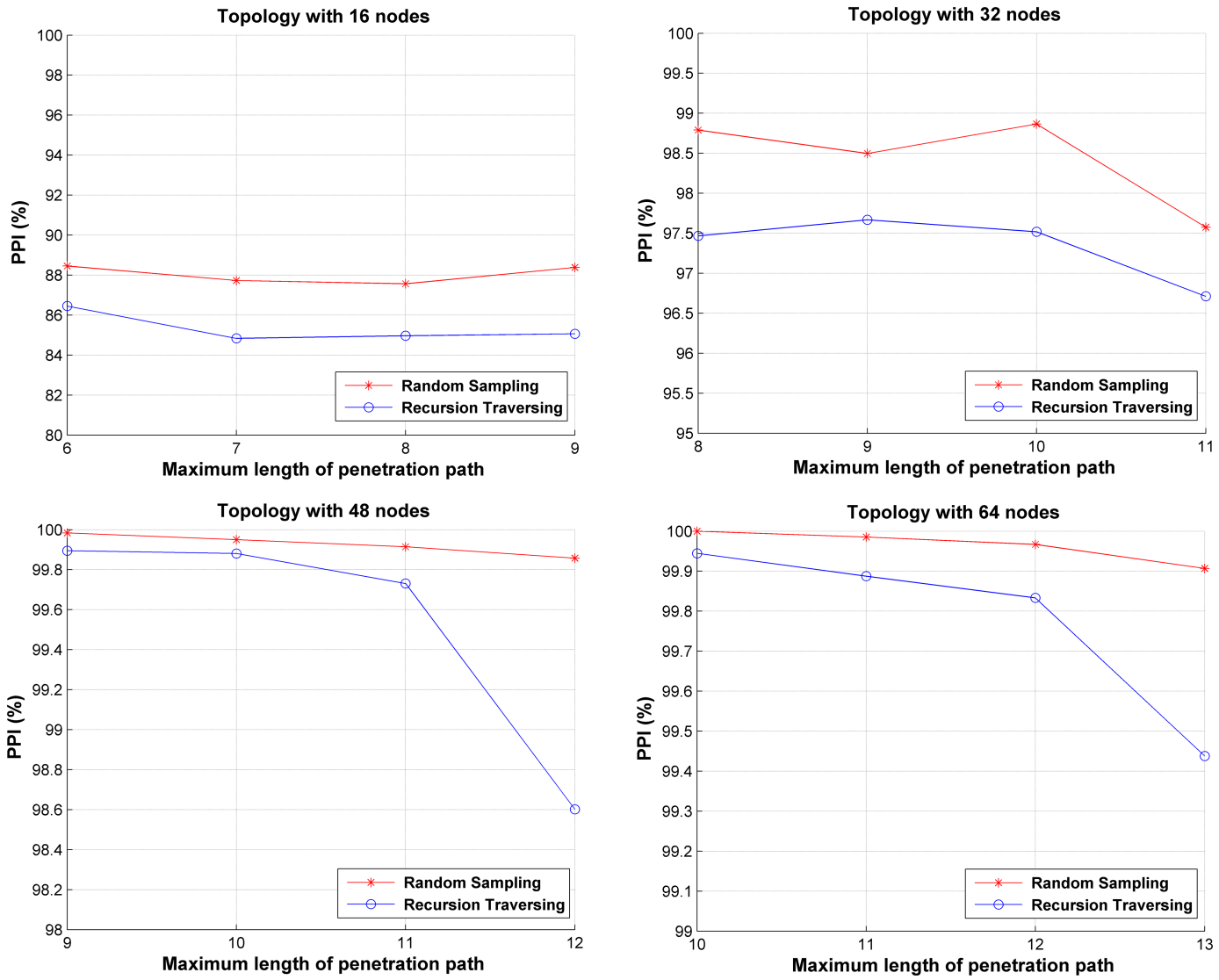

Fig 10. Comparison of *PPI* using recursive traversal and random sampling algorithm.

https://doi.org/10.1371/journal.pone.0189095.g010

network scale and the maximum length of the penetration path grow, the number of possible penetration paths increases exponentially. Therefore, the time cost of SA using the recursive traversal algorithm increases exponentially. However, the growth of the time cost is much less when the random sampling algorithm is used, as the number of penetration paths remain unchanged. Therefore, the random sampling algorithm can get a lower time cost in larger-scale networks.

Comparison of greedy algorithm and DCD. According to literature [14], it is helpful to reduce the penetration probability when the decoy chains are deployed on nodes with small degrees. Therefore, we compared the effectiveness of DCD and greedy algorithm. The greedy algorithm deploys decoy chains as many as possible to the nodes with the smallest degrees under the resource constraints. The comparison between the greedy algorithm and DCD is made in experiments with the generated network topologies. We set *maxCnt* to 10^4 in this

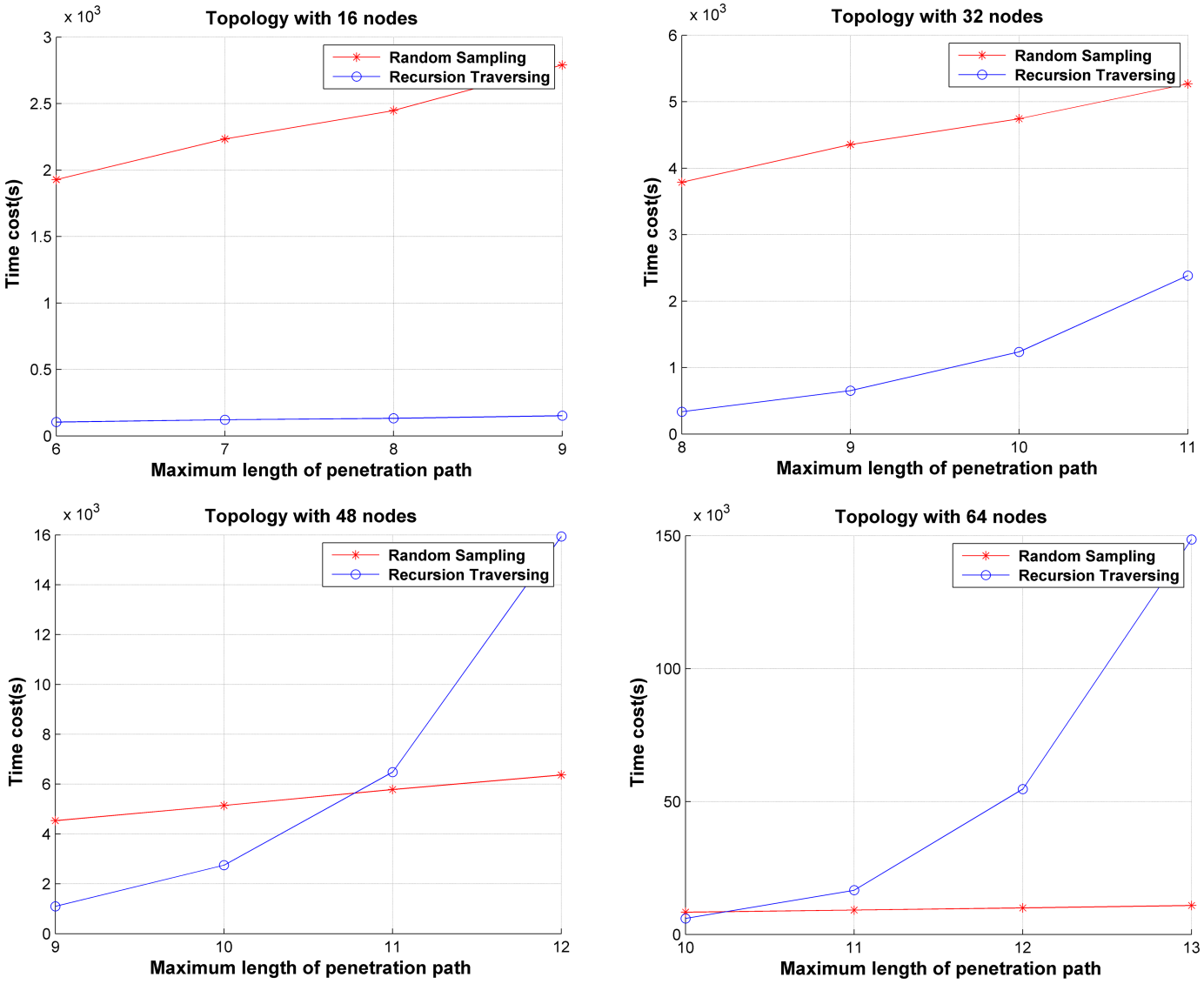

Fig 11. Comparison of time cost between SA with recursive traversal and random sampling algorithm.

https://doi.org/10.1371/journal.pone.0189095.g011

experiment and the *PPIs* are shown in Fig 12. As seen, the greedy algorithm can reduce the penetration probability ($PPI > 0$), which is consistent with literature [14]. However, it provides much lower security compared with DCD. DCD tries to find the global optimum solution, while the greedy algorithm can only give a local optimal solution. Therefore, DCD can provide better security by reducing probability of sensitive targets being compromised in a greater degree.

DCD with multiple attack sources and targets. DCD is able to deploy decoy chains for multiple attack sources and sensitive targets. The generated network with 64 nodes is used in this experiment. Two nodes are randomly chosen from the first layer of the network as the attack source set $O = \{o_1,o_2\}$ and two nodes from the fourth layer constitute the sensitive target set $T = \{t_1,t_2\}$. The intruder appears at o_1 and o_1 with equal probability and we set $\xi = 12$. Random sampling algorithm ($maxCnt = 10^4$) is used to calculate the penetration probability

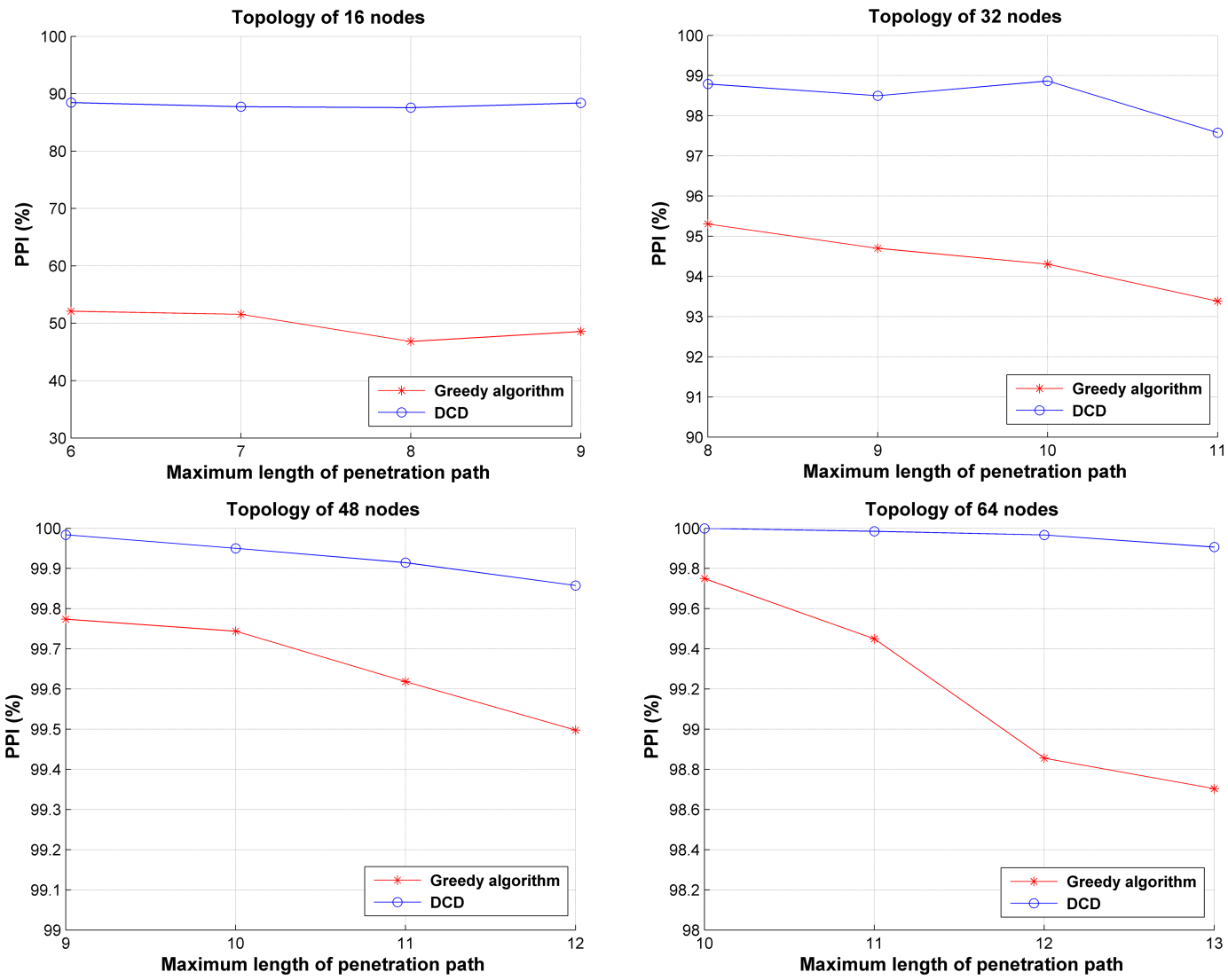

Fig 12. Comparison between greedy algorithm and DCD.

https://doi.org/10.1371/journal.pone.0189095.g012

$P_Z^{\{o_i\}\to\{t_j\}}$ $(i, j \in \{1, 2\})$. The relative probability RP_{ij} is defined as formula (19).

$$RP_{ij} = \frac{P_Z^{\{o_i\}\to\{t_j\}}}{\max\{P_Z^{\{o_h\}\to\{t_k\}} | o_h \in O, t_k \in T\}}, \quad i \in [0, |O|], j \in [0, |T|] \tag{19}$$

The RP_{ij} of the each OT pair (o_i, t_j) $(o_i \in O, t_j \in T)$, is obtained in the experiment, as shown by the solid line in Fig 13. Then, the decoy chains are deployed in the network using DCD. The *PPI* of each pair (o_i, t_j) is shown by the dashed line in Fig 13. The result shows that OT pairs with higher *RPs* achieve higher*PPIs*, meaning that a better security is achieved by DCD for an OT pair with higher *RP*. DCD considers the global optimation for deploying decoy chains. Thus, the OT pairs with the higher *RPs* are considered preferentially. Therefore, an OT pair with higher *RP* achieves a higher *PPI* after deploying decoy chains.

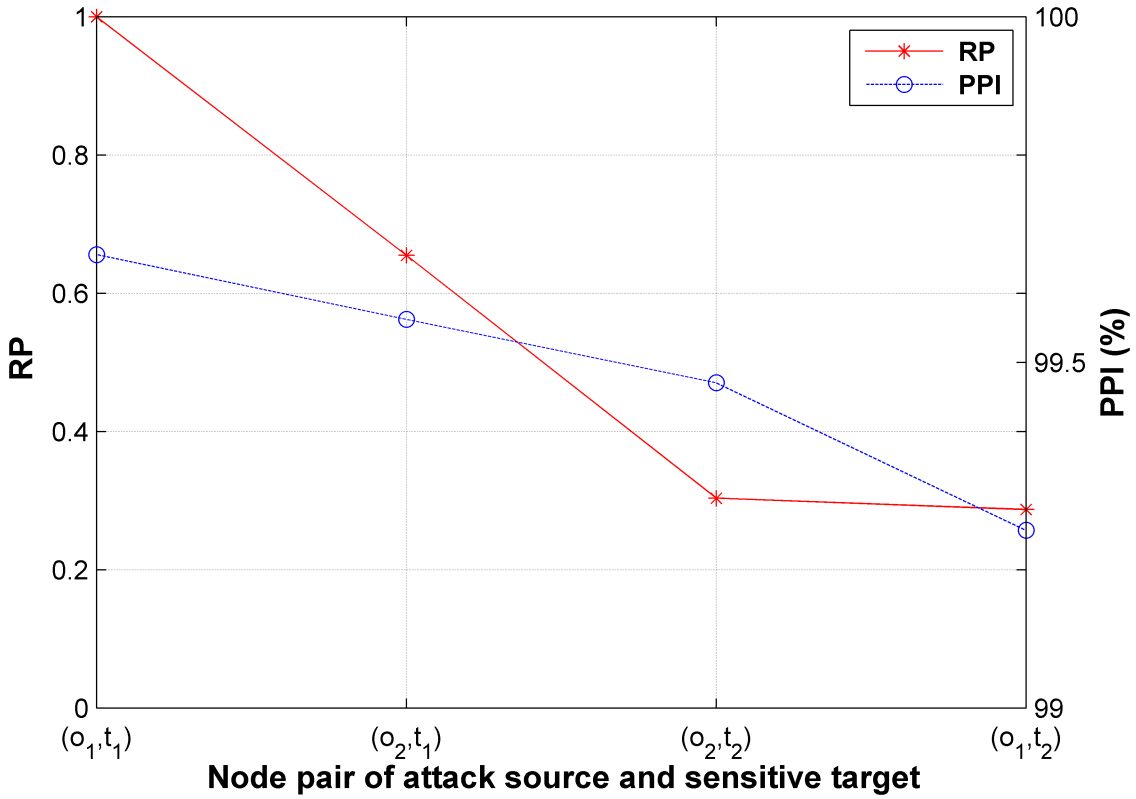

Fig 13. Deployment of the decoy chains with multiple attack sources and sensitive targets.

https://doi.org/10.1371/journal.pone.0189095.g013

Discussion

We view DCD as a promising method to defense penetration attacks. As the simulation experiments shown, DCD can effectively decrease the probability of sensitive targets being compromised. We believe that DCD has a great potential in improving the security of network, even though the performance is not ideal. Further optimization could greatly facilitate its application.

In this study, we concentrate on outside intruders that appear at the edge of network. Accordingly, the decoy chains are deployed for these outside intruders. However, the insider intruders who appear at inner layers of the network may bypass several layers of the network. Therefore, their penetration actions may not be effectively defensed by the current DCD. The defense of inside attack will be further studied in future work.

In the experiments, the time cost for generating decoy chain deployment strategy is not ideal. The reasons are: 1) In the experiments, we choose a big value for the number of random samples of paths ($maxCnt = 10^4$) to adequately test the security of DCD, which leads to large time cost of random sampling algorithm. 2) Both the design and implementation of DCD are preliminary and can be optimized using parallelism techniques. Moreover, the performance could be further improved by executing part of the operation offline, instead of online, even though DCD is designed for responding network security threats online. More specifically, decoy chain deployment strategies could be calculated in advance, since the attack sources (terminals with poor protection) and sensitive targets (terminals with sensitive data) do not change frequently. Furthermore, for deploying decoy chain, if DCD sends the whole decoys to the servers, both the time cost and bandwidth cost will be large. Considering the availability of

DCD, the decoy nodes can be stored in disks of the servers in advance. When a server receives commands from the controller, the decoy chains can be constructed at once so that the latency could be greatly shortened.

In this paper, the capacity of a server is defined to represent the maximum number of virtual machines running on the server. However, the capacity of a server is a coarse-grained resource constraint and a more fine-grained approach could be considered. One possible way would be calculating multiple resource constraints respectively, such as the number of CPU, memory size, hard disk driver, and bandwidth. However, the fine-grained resource constraints will make the decoy chain model much more complex.

Conclusion and future work

In this paper, MTD is introduced to deploy the decoy chains in the network for changing the attack surface. DCD, a decoy chain deployment method, is proposed based on SDN and NFV. DCD creates a second defense line for the network, which can delay the penetration attack and decrease the probability that sensitive targets are compromised. Centralized control of SDN is utilized to deploy decoy chains under resource constraints, and multiple attack sources and sensitive targets are considered. SA algorithm is used in DCD to achieve the maximum benefit for network defense. Experiments show that DCD can effectively defend against penetration attack. Future improvements could be achieved in three aspects as follows: 1) How to reduce the time cost of DCD; 2) How to deploy chains dynamically based on security monitoring to increase the efficiency of DCD; 3) In this paper, the decoy chain composing of decoy nodes has no branch. Therefore, it is easy to arouse intruders' suspicions. Instead of the decoy chain, decoy-net, a network consists of decoy nodes, will also be explored in future work.

Author Contributions

Conceptualization: Zheng Zhao.

Data curation: Qi Zhao, Chuanhao Zhang.

Formal analysis: Chuanhao Zhang, Zheng Zhao.

Investigation: Zheng Zhao.

Methodology: Chuanhao Zhang, Zheng Zhao.

Project administration: Chuanhao Zhang.

Resources: Zheng Zhao.

Software: Qi Zhao, Chuanhao Zhang.

Supervision: Zheng Zhao.

Validation: Zheng Zhao.

Visualization: Qi Zhao.

Writing – review & editing: Qi Zhao, Chuanhao Zhang, Zheng Zhao.

References

1. Linn MC, Davis EA, Bell P. Internet Environments for Science Education: Lawrence Erlbaum Associates; 2005.
2. Fang IE. Alphabet to Internet: media in our lives: Routledge; 2015.
3. Mcknight LW, Bailey JP. Internet Economics. Mit Press Books. 1997; 49(6):569–571.

4. Wang H, Cao J, Zhang Y. A flexible payment scheme and its role-based access control. IEEE Transactions on Knowledge & Data Engineering. 2005; 17(3):425–436.
5. Modi C, Patel D, Borisaniya B, Patel H, Patel A, Rajarajan M. A survey of intrusion detection techniques in cloud. Journal of Network and Computer Applications. 2013; 36(1):42–57.
6. Zuech R, Khoshgoftaar TM, Wald R. Intrusion detection and big heterogeneous data: A survey. Journal of Big Data. 2015; 2(1):1–41.
7. Butun I, Morgera SD, Sankar R. A survey of intrusion detection systems in wireless sensor networks. Communications Surveys & Tutorials, IEEE. 2014; 16(1):266–282.
8. Falliere N, Murchu LO, Chien E. W32.Stuxnet Dossier. White Paper, Symantec Corp, Security Response. 2011; 5:6.
9. Spitzner L. Honeypots: tracking hackers. 2003. https://pdfs.semanticscholar.org/669d/5738698653184b345b4e6fd2e13529aadba4.pdf.
10. Kumar A, Kumari S. A Survey on Honeypots Security. IITM Journal of Management and IT. 2015; 6 (1):44–50.
11. Tang Y, Lu X, Hu H, Zhu P. Honeypot technique and its applications: A survey. MINIMICRO SYSTEMS SHENYANG. 2007; 28(8):1345.
12. Spitzner L. The honeynet project: Trapping the hackers. IEEE Security & Privacy. 2003;(2):15–23.
13. Spitzner L. Honeytokens: The Other Honeypot. Boston: Addison-Wesley; 2002. http://www.securityfocus.com/infocus/1713.
14. Shakarian P, Kulkarni N, Albanese M, Jajodia S. Keeping Intruders at Bay: A Graph-theoretic Approach to Reducing the Probability of Successful Network Intrusions. E-Business and Telecommunications: Springer; 2014. p. 191–211.
15. Jajodia S, Ghosh AK, Subrahmanian V, Swarup V, Wang C, Wang XS. Moving Target Defense II. Application of game Theory and Adversarial Modeling Series: Advances in Information Security. 2013; 100:203.
16. Huang H, Li P, Guo S. Traffic scheduling for deep packet inspection in software-defined networks. Concurrency & Computation Practice & Experience. 2017; 29(16). https://doi.org/10.1002/cpe.3967
17. McKeown N. Software-defined networking. INFOCOM keynote talk. 2009; 17(2):30–32.
18. ISG N. Network Functions Virtualisation (NFV)-Network Operator Perspectives on Industry Progress. ETSI, Tech., 2013.
19. Portokalidis G, Slowinska A, Bos H, editors. Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. ACM SIGOPS Operating Systems Review; 2006; 40(4): 15–27.
20. Kuwatly I, Sraj M, Al Masri Z, Artail H, editors. A dynamic honeypot design for intrusion detection. Pervasive Services, 2004 ICPS 2004 IEEE/ACS International Conference on; 2004: 95–104.
21. Wagener G, State R, Engel T, Dulaunoy A, editors. Adaptive and self-configurable honeypots. Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on; 2011: 345–352.
22. Han W, Zhao Z, Doupé A, Ahn G-J, editors. HoneyMix: Toward SDN-based Intelligent Honeynet. Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization; 2016: 2016: 1–6.
23. Wang H, Jia Q, Fleck D, Powell W, Li F, Stavrou A. A moving target DDoS defense mechanism. Computer Communications. 2014; 46:10–21.
24. Carvalho M, Ford R. Moving-target defenses for computer networks. IEEE Security & Privacy. 2014; (2):73–76.
25. Antonatos S, Akritidis P, Markatos EP, Anagnostakis KG. Defending against hitlist worms using network address space randomization. Computer Networks. 2007; 51(12):3471–3490.
26. Jafarian JH, Al-Shaer E, Duan Q, editors. Openflow random host mutation: transparent moving target defense using software defined networking. Proceedings of the first workshop on Hot topics in software defined networks; 2012: 127–132.
27. Duan Q, Al-Shaer E, Jafarian H, editors. Efficient Random Route Mutation considering flow and network constraints. Communications and Network Security (CNS), 2013 IEEE Conference on; 2013: 260–268.
28. Jafarian J, Al-Shaer E, Duan Q. Formal Approach for Route Agility against Persistent Attackers. In: Crampton J, Jajodia S, Mayes K, editors. Computer Security—ESORICS 2013. Lecture Notes in Computer Science. 8134: Springer Berlin Heidelberg; 2013: 237–54.
29. Badishi G, Herzberg A, Keidar I. Keeping denial-of-service attackers in the dark. Dependable and Secure Computing, IEEE Transactions on. 2007; 4(3):191–204.

30. Zhao Z, Gong D, Lu B, Liu F, Zhang C. SDN-based Double Hopping Communication against sniffer attack. Mathematical Problems in Engineering. 2016; 2016(2): 1–13.
31. Clark A, Sun K, Poovendran R, editors. Effectiveness of IP address randomization in decoy-based moving target defense. Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on; 2013: 10–13.
32. Clark A, Sun K, Bushnell L, Poovendran R. A Game-Theoretic Approach to IP Address Randomization in Decoy-Based Cyber Defense. Decision and Game Theory for Security: Springer; 2015: 3–21.
33. Ujcich BE, Rausch MJ, Nahrstedt K, Sanders WH. IRMA via SDN: Intrusion Response and Monitoring Appliance via Software-Defined Networking. 2015. https://www.ideals.illinois.edu/bitstream/handle/2142/88342/irma_ideals.pdf?sequence=2.
34. Chen P-J, Chen Y-W, editors. Implementation of SDN based network intrusion detection and prevention system. Security Technology (ICCST), 2015 International Carnahan Conference on; 2015: 141–146.
35. Khan S, Gani A, Wahab AWA, Shiraz M, Ahmad I. Network forensics: Review, taxonomy, and open challenges. Journal of Network and Computer Applications. 2016; 66:214–235.
36. Xiang Y, Zhou W, Guo M. Flexible Deterministic Packet Marking: An IP Traceback System to Find the Real Source of Attacks. IEEE Transactions on Parallel & Distributed Systems. 2009; 20(4):567–80.
37. Xiang Y, Li K, Zhou W. Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics. IEEE Transactions on Information Forensics & Security. 2011; 6(2):426–37.
38. Wang W, Daniels TE. A graph based approach toward network forensics analysis. ACM Transactions on Information and System Security (TISSEC). 2008; 12(1):4.
39. Smith R, Japkowicz N, Dondo M, Mason P. Using unsupervised learning for network alert correlation. Advances in Artificial Intelligence: Springer; 2008. p. 308–319.
40. Mell P, Scarfone K, Romanosky S. Common Vulnerability Scoring System. Security & Privacy IEEE. 2006; 4(6): 85–89.
41. Wang L, Islam T, Long T, Singhal A, Jajodia S, editors. An Attack Graph-Based Probabilistic Security Metric. Data and Applications Security Xxii, Ifip Wg 113 Working Conference on Data and Applications Security, London, Uk, July 13–16, 2008, Proceedings; 2008, 5094: 283–296.
42. Huang H, Guo S, Wu J, Li J, editors. Joint middlebox selection and routing for software-defined networking. IEEE International Conference on Communications; 2016: 1–6.
43. Huang H, Guo S, Wu J, Li J. Service Chaining for Hybrid Network Function. IEEE Transactions on Cloud Computing. 2017. https://www.researchgate.net/publication/317932285_Service_Chaining_for_Hybrid_Network_Function.
44. Huang H, Li P, Guo S, Liang W, Wang K. Near-Optimal Deployment of Service Chains by Exploiting Correlations between Network Functions. IEEE Transactions on Cloud Computing. 2017, In Press.
45. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. science. 1983; 220 (4598):671–680. https://doi.org/10.1126/science.220.4598.671 PMID: 17813860
46. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. The journal of chemical physics. 1953; 21(6):1087–1092.
47. Medina A, Matta I, Byers J. BRITE: a flexible generator of Internet topologies. Boston University, 2000. https://dl.acm.org/citation.cfm?id=864341.
48. Barabási A-L, Albert R. Emergence of scaling in random networks. Science. 1999; 286:509–512. PMID: 10521342