Linkbait: Active Link Obfuscation to Thwart Link-flooding attacks

Qian Wang, *Member, IEEE,* Feng Xiao, Man Zhou, *Student Member, IEEE,* Zhibo Wang, *Member, IEEE,* Qi Li, *Member, IEEE,* Zhetao Li

Abstract—Link-flooding attack (LFA) has emerged as a serious threat to Internet which cuts off connections between legitimate hosts and targeted servers by flooding only a few links (e.g., target links). Several mechanisms have been proposed to mitigate LFA, however, they can only mitigate LFA after target links have been compromised by adversaries. Based on the fact that adversaries rely on network linkmap to discover weakness of the network, in this paper, we propose an active LFA mitigation mechanism, called Linkbait, that actively and preventively mitigates LFA by providing a fake linkmap to adversaries. Inspired by Moving Target Defense (MTD), we propose a link obfuscation algorithm in Linkbait that selectively reroutes probing flows to hide target links from adversaries and mislead them to consider some bait links as target links. By providing the faked linkmap to adversaries, Linkbait can actively mitigate LFA even without identifying bots and does not affect flows from legitimate hosts. In order to further reduce the junk traffic generated by adversaries from entering the network, we propose a bot detection algorithm in Linkbait that extracts unique traffic patterns from LFA and leverages Support Vector Machine to accurately distinguish bots from legitimate hosts. Finally, we evaluate the feasibility of implementing Linkbait in real Internet, and evaluate its performance by using both a real-world testbed and large-scale simulations. The analyses and experiments results demonstrate the effectiveness of Linkbait.

Index Terms—Link-flooding attack, Obfuscation, DDoS defense

1 Introduction

Botnet-driven distributed denial-of-service (DDoS) attack [1], [2], which consumes resources of targeted servers to cause a denial of service (DoS) attack for legitimate hosts, is one of the most serious threats to the Internet because it always leverages legitimate TCP-like flows to launch attacks. A recent example of DDoS attack is that the Mirai botnets brought down much of America's internet in 2016 and was likely the largest of its kind in history [3]. A new type of DDoS attack, called link-flooding attack (LFA), has been proposed recently [4], [5], which is even more difficult to be detected or mitigated. Unlike traditional DDoS attacks which mainly consume the resources of the end targets, LFA utilizes distributed botnets to deplete the bandwidth of certain network links (e.g., target links) to disconnect entire domains from the Internet. To be precise, it does not attack the end targets directly, but instead depletes the bandwidth of target links, which makes the detection and defense systems at the end servers irrelevant.

The bots manipulated by an adversary send TCP-like flows to the target area just as legitimate users do. Because of its indirectly attack strategy, the victims may not receive any malicious traffic even they are under attacks. Moreover,

- Qian Wang, Feng Xiao, Man Zhou and Zhibo Wang are with the State Key Lab of Software Engineering, School of Computer Science, Wuhan University, Wuhan 430072, China. E-mail: {qianwang, f3i, zhouman, zbwang}@whu.edu.cn.
- Qi Li is with the Graduate School at Shenzhen, Tsinghua University, Shenzhen, China. E-mail: liqi@csnet1.cs.tsinghua.edu.cn.
- Zhetao Li is with the the College of Information Engineering, Xiangtan University, Xiangtan, Hunan 411105, China. E-mail: liztchina@gmail.com.

it is challenging to distinguish between bots and legitimate hosts because the bots can also use valid IP addresses and the low-rate attack flows from each bot conform to the Internet protocol. With these unique characteristics, LFA has quickly emerged in real-world attacks [6], [7], and it is very difficult to be detected or mitigated by traditional defense mechanisms.

LFA has attracted the great attention of researchers and some works recently have been proposed to detect and mitigate it [8], [9], [10], [11], [12], [13], [14], [15], [16]. However, existing mechanisms mainly try to mitigate LFA by simply imposing policies on all traffic instead of differentiating flooding traffic from legitimate traffic with different rules. In addition, all these mechanisms take effects after networks have been congested, which cannot guarantee the availability of target links or thoroughly thwart adversaries' intention of introducing congestions to the network, even some of them are able to backtrack bots after LFA happens. Therefore, it is urgently needed to design a preventive mechanism that captures adversaries' intention early and treats their flows with special policies in advance, so that we can preventively mitigate LFA without target links being congested by adversaries.

The mainstream of LFA (e.g., the Cross Fire attack [5]) relies on the linkmap of the network to figure out target links and then launches LFA effectively. That is, the core is to build an accurate linkmap of the network. In order to obtain an accurate linkmap, an adversary will manipulate all his bots to send probing flows to the network to gather the link information towards as many servers in the target area as possible. However, a legitimate host usually would not gather link information like the adversary does. Thus, we argue that this process becomes an important pattern

that can be used to distinguish bots from legitimate hosts. Moreover, if we could mislead the adversary to build a wrong linkmap, LFA will inevitably miss fire.

In this paper, we propose an active link obfuscation mechanism, called Linkbait, to actively mitigate LFA by providing a fake linkmap to adversaries. Inspired by Moving Target Defense (MTD) [17] [18], we propose a link obfuscation algorithm in Linkbait by selectively rerouting probing flows to obfuscate links, so that target links are hidden from adversaries and some bait links are misjudged as target links by adversaries. Since adversaries rely on the linkmap to discover target links and launch attacks, a fake linkmap will force adversaries to miss fire inevitably. Linkbait uses several bait links to fake target links which could attract attentions from adversaries while hiding the true target links, so that LFA can be avoided in advance without target links being compromised by adversaries. In particular, our bait link construction strategy and selectively flow rerouting policy reduce the probability of bait links being congested by flooding attacks. Furthermore, in order to reduce the junk traffic generated by adversaries from entering target links, we propose a bot detection algorithm that extracts unique traffic patterns from LFA and leverages SVM to accurately distinguish bots from legitimate hosts. In this way, we propose a pre-congestion LFA mitigation and detection system, while the precursors focused on digging out adversaries after critical information are leaked or the network has been compromised.

In order to realize our mechanism, there are three major challenges that need to be addressed.

- Link obfuscation: Linkbait intends to hide target links and use bait inks to fake target links. However, numerous flows are randomly distributed in the network. How to distract the attention of the adversary from true target links to bait link is a challenging problem.
- LFA resistance for bait links: Linkbait uses dynamic rerouting to mislead adversaries to build a fake linkmap. The bait links will be exposed to and attacked by the adversary on behalf of target links, which may suffer from congestion and further affect the flows from legitimate hosts. Therefore, how to construct the bait links without suffering congestion in the network is challenging.
- Pre-congestion bot detection: Linkbait tries to detect
 bots before the links are compromised by the adversary, which is totally different from existing LFA
 mitigation works detecting bots after the congestion
 happens. Hence, using only the attack pattern is not
 sufficient to detect bots. Moreover, it is possible that
 some legitimate hosts are identified as bots. Therefore, how to achieve an accurate bot detection while
 minimizing the false positive rate is a challenging
 problem.

We propose Linkbait to solve these challenges, and the main contributions of this work are summarized as follows:

 We propose a novel LFA mitigation mechanism, called Linkbait, to preventively mitigate LFA before it starts to congest the network. To the best of our

- knowledge, we are the first to distinguish suspicious hosts before flooding happens by intentionally providing a fake linkmap. In addition, we treat suspicious hosts with different rules so as to avoid them further introducing congestions to our network, which is totally different from existing works.
- We propose two link algorithms for Linkbait. The first is link grouping algorithm which aims at finding suitable links to build bait links in the network, and the second is link obfuscation algorithm that selectively reroutes probing flows to hide target links from adversaries and mislead them to consider some bait links as target links. By providing the faked linkmap to adversaries, Linkbait can actively mitigate LFA without affecting legitimate flows from legitimate hosts.
- We propose a bot detection algorithm which extracts unique traffic features from both the linkmap construction phase and the flooding phase, and leverage SVM to accurately distinguish bots from legitimate hosts.
- We evaluate the performance of Linkbait with implementation in a software defined network (SDN) testbed and large-scale simulations. The experimental results show that Linkbait can effectively mitigate LFA with a very low network latency.

The remainder of this paper is organized as follows. We introduce the system model and the background of LFA in Section 3. We present the design of Linkbait in Section 4 and the security analysis in Section 5. We then evaluate the performance of Linkbait in Section 6. Finally, we conclude the paper in Section 7.

2 RELATED WORK

In this section, we briefly discuss the state-of-the-art of LFA detection and mitigation. After Kang et al. proposed the Cross Fire attack [5], many works on LFA detection and mitigation have been proposed and they mainly focus on two aspects of LFA: flooding traffic and probing information.

Mitigation on flooding traffic: Lee et al. proposed Codef [9], a collaborative defense mechanism between autonomous systems (ASes) to mitigate LFA. However, the coordination between ASes or ISPs is not readily available yet due to their competitive relationship and the latency in cooperation. Liaskos et al. proposed a novel framework which implements online traffic engineering (TE) and continuously re-routes traffic in a manner that makes persistent mitigation after LFA events happen [10]. Gkounis et al. also investigated the interplay of TE and LFA [11]. Woodpecker [12] proposed a centralized TE scheme based on the upgraded nodes. Aydeger et al. proposed a SDN based model leveraging TE dynamically to reroute traffic on the suspected target links as long as it is congested [14]. However, adapting frequently changing routing policy for all traffic on core network is not quite feasible, since it fails to consider the network topology condition (such as bandwidth) and its dependence on SDN makes it inapplicable in real-world network. Generally speaking, these special TE mitigations temporarily reroute legitimate traffic along with flooding one to other links, which do not eliminate flooding

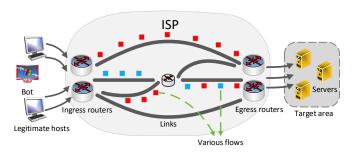


Fig. 1. System model.

traffic from the network, so they can only serve as a temporary solution towards LFA. Kang et al. designed a SDN based system, called SPIFFY [13], that leverages temporary bandwidth extension to identify flooding traffic during LFA happens. However, SPIFFY requires to extend bandwidth of the bottlenecked core link in a short time (TBE) which has imposed strong requirements for bandwidth and link infrastructure.

Information oriented mitigation: Xue et al. proposed LinkScope [8], a detecting system that employs both the end-to-end and the hop-by-hop network measurement techniques to capture abnormal path performance degradation for detecting LFA. However, his work focuses on link state monitoring which falls short in botnet tracing. Hence it has limited effect on eliminating flooding traffic and picking out the adversary. Hirayama et al. [16] regarded the traffic of traceroute as the sign of a up-coming link-flooding attack, so as to alarm the supervisor of ISP to deploy defence, but only employing the increase of traceroute as the sign of LFA probably leads to a high false-positive rate of alarming. MoveNet [15] employs virtual networks to offer constant, dynamic and threat-aware reallocation of critical network resources to deceive attackers knowledge about critical network resources, which provides an abstract methodology to counter DDOS. However, it is merely a general framework of DDOS mitigation, and corresponding mechanisms should be added if MoveNet want to detect and mitigate LFA.

All the existing works focus on a temporary solution that cannot thoroughly eliminate flooding traffic or prevent them from entering target links. In addition, these solutions only take effects after the network has been compromised, even though some of them can backtrack the bots after attacks happen. In contrast, we propose an preventively LFA mitigation in this paper that can distinguish suspicious traffic from legitimate traffic at an early stage and impose special policies on them to mitigate LFA before target links are compromised.

3 PRELIMINARIES

In this section, we first introduce the link-flooding attack (LFA), and then present the system model.

3.1 Link-flooding Attack (LFA)

LFA targets links in the core of the network and creates a large number of attack flows crossing the targeted links to flood and virtually disconnect them. There are mainly two kinds of LFA: The first is the Coremelt attack [4]. It

utilizes bots to send attack traffic to other bots. This attack leverage bot pairs, whose communication paths share the links in the Internet core, to congest the network. And the second is the Crossfire attack [5], which coordinates bots to send legitimate-looking low-rate traffic to the attackerchosen publicly accessible servers (e.g., HTTP servers) in a way that their routes cross the link targets in the core Internet. Compared to the Coremelt attack, the Crossfire attack exhibits a lower requirements for the location and distribution of bots and therefore adversaries can manipulate more bots to effectively attack the victim, so it is more flexible and threatening. In particular, adversaries tend to choose the latter when they target at large networks (e.g., ISP), in which they are difficult to find and collude enough bot pairs. Hence, we mainly focus on the Crossfire attack in this paper and we employ LFA to denote such kind of attacks. To compromise the victim, the adversary first discovers the target links of the network and then manipulates a large number of bots to isolate the targeted victims from the Internet by flooding flows to the target links. It can be described as the following two steps: link information gathering and flooding.

Link information gathering: To launch LFA, the adversary will use all his bots to query link information towards as many servers in target area as possible. Usually the adversary leverages network diagnostic tools (e.g., traceroute) to gather layer-3 router links. The link information gathered by the adversary is called *linkmap*. We call any host which performs such link information querying for legitimate or malicious purpose as a *link-prober*.

We argue that linkmap is different from network topology [19]. The linkmap consists of all the 2-dimension router information from source hosts to destination servers while the network topology focuses on the infrastructure of 3-dimension router physical connectively relationship. To be precise, the linkmap describes the routing policy towards the target area in the ISP. In order to figure out the best attack-cost strategy, the adversary tends to attack links which can be occupied by as many bots as possible because he can inject more junk traffic to these links. As a consequence, links which can be attacked by enough bots are chosen as *target links*.

Flooding: In this step, the adversary manipulates a large number of bots to persistently send TCP-like flows to congest the target links by consuming their bandwidth. Note that a rational adversary will cautiously manipulate his bots in a reasonable rate to avoid being detected by the rate-based detecting mechanisms. In addition, the adversary also constantly checks the target links to see whether they are successfully congested.

3.2 System model

In this paper, as shown in Figure 1, we focus on networks with two edges (ingress routers and egress routers). The networks can be ISPs where servers in the target area are linked with the egress routers to provide services to the public. Hosts outside the network (on the left side) can access the servers in the target area (on the right side) via the ingress routers. In particular, the legitimate hosts as well as the adversaries can visit the target area by using

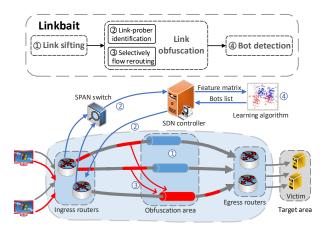


Fig. 2. System overview with Linkbait.

suitable protocols (e.g., hosts can establish communications with web servers using HTTP).

As we mentioned in Section 3.1, adversaries tend to choose the Crossfire attacks due to its flexibility. Hence, an adversary need to manipulate a large number of bots to obtain the linkmap of the network by sending probing flows to the network. He figures out the target links between the ingress/egress routers, and then launches LFA to the target links in the target area.

4 LINKBAIT DESIGN

In this paper, we propose a new mechanism, called Linkbait, to preventively mitigate LFA with active link obfuscation. The key idea of Linkbait is to provide an obfuscated linkmap to the adversary by imposing differential policies on probing flows, and mislead attacks from bots to the faked target links while hiding the true target links. Different from existing LFA mitigation mechanisms, Linkbait is an preventively attack mitigation mechanism that can early mitigate LFA before it congests the network.

In this section, we first give a high-level overview of Linkbait, and then elaborate on the design of Linkbait.

4.1 Linkbait Overview

The design principle of Linkbait components is to find susceptible target links and obfuscate traffic in them. To achieve that, Linkbait consists of three components, as shown in Figure 2: link sifting, link obfuscation and bot detection. In link sifting, we check the flow distribution of all links in the network to figure out the target links that may be flooded by bots and also select appropriate links as bait links to fake target links. Link obfuscation tries to provide a fake linkmap for the adversary by selectively rerouting of probing flows, so that the true target links are hidden while the bait links will be misjudged as target links and attacked by the adversary. It is worth noting that a bait link contains multiple links which can efficiently mitigate LFA to the bait link. Although LFA can be mitigated by link obfuscation, we further leverage a supervised learning algorithm to accurately distinguish bots from legitimate hosts, and certain measures can be taken to reduce the junk traffic going through the network.

4.2 Link Sifting

Link sifting¹ aims to figure out potential target links of the network and select appropriate links to fake target links. We call the faked target link as *bait links*. To realize this purpose, link sifting has two phases: link analysis and link grouping. The former tries to obtain the whole network information including all the links and their flow densities, and the latter figures out the target links and select appropriate links to form bait links.

4.2.1 Link Analysis

In this paper, we deem that a single link is a stable router sequence from an ingress router to an egress router, which serves as a communication channel between hosts and servers in the target area. Since links in the ISP are dynamically changing and the ISP might well only possess a coarse-grained linkmap, it is a difficult job, even for the ISP itself, to gather fine-grained information needed by Linkbait. Hence, we propose a method to obtain the whole network information.

The method is called *looking glass tracing* (LG tracing), which leverages existing network diagnostic tools (e.g., traceroute) to collect link information. In order to obtain the complete link information in the network, the ISP needs to hire a large number of hosts distributed in different locations to trace links, which is expensive and impossible for a single ISP due to its limited resources. Fortunately, there are many public available servers maintained by other ISPs providing traceroute services. We call these servers as LG servers and they can be remotely accessed for the purpose of querying routing information. These servers are distributed around the world, which are similar to the distribution of bots. Therefore, with the help of LG servers, LG tracing leverages existing network diagnostic tools to collect link information of the ISP [20]. As the adversary also uses existing network diagnostic tools to obtain target links, the links we obtain can cover the links obtained by the adversary as long as the number of LG servers manipulated by the ISP is large enough.

Flow density represents bandwidth utilization for links. According to the universal power-law property of flow density distribution [5], the more flows can be created through one link, the higher flow density it has. Thus, we estimate the flow density of a link by calculating the number of LG servers going through the link.

4.2.2 Link Grouping

After gathering the link information, we can compute the flow density of each link and then figure out which links are most likely to be flooded. In this paper, we use the algorithm in [5] to figure out the true target links.

In order to hide the true target links from the adversary, we select some links to fake target links and reroute the probing flows of bots to these faked links. We call these faked links as *bait links*. As a consequence, the adversary will obtain a fake linkmap and misjudge the bait links as

1. Link sifting should not be confused with linkmap construction of the adversary in [5], though they all gather links information towards a specific area. Linkmap construction mainly focuses on discovering the links which can be consumed by bots.

the target links. Note that a bait link in our mechanism is not a single link but is composed of several links. There are two reasons to construct a bait link in this way. First, in order to fake a target link, it should have large flow density after flow rerouting. The most convenient way to realize this purpose is to reroute the probing flows of multiple links to one converge link. Second, the bait links will suffer from flooding attacks from bots. The congestion can be reduced if the attacks are distributed into multiple links of each bait link. However, it is difficult to ensure that linkmap obfuscation on these links can affect enough flows in the network due to the limited link resources in some target networks. To solve this problem, we must design a link grouping algorithm that can cover as many as possible flows in the network when minimize the "cost" of adding links into bait links.

Problem Formulation: We formulate the link grouping problem as a weighted set cover problem. Let f_i denotes each single flow sending from individual hosts to the target network, and the total set of flows in the network is denoted by $F = \{f_1, f_2, ..., f_{\varphi}\}$. For each links, there are several flows going through it. Let $T(l_i) = \{f_{i_1}, f_{i_2}, \cdots\}$ be the set of flows going through the link l_i . Suppose there are n flows, so the flows going through all the links are denoted by $L = \{T(l_1), T(l_2), ..., T(l_N)\}$. It is obviously that

$$\cup_{i=1}^{N} T(l_i) = F$$

In the weighted set cover problem, defining a weight function is very important. This nonnegative weight function $w:L\to\mathbb{R}$ should reflect the cost of each link. A bait link with less cost is supposed to obfuscate more flows than others while their consumptions of link resources are the same. Let b_i denote a bait link and $T(b_i)$ is the set of flows going through b_i . Our objective is to find a set of bait links which can minimize the total cost while covering as many flows as possible. With this objective, the link grouping problem is formulated as follows.

$$\arg \min_{B} \sum_{b_{i} \in B} w(b_{i})$$
s.t.
$$\bigcup_{b_{i} \in B} T(b_{i}) = F$$
(1)

The weight of b_i , $w(b_i)$, can be characterized by two factors: the number of links in b_i , and the flow density of b_i . Note that the flow density of b_i , denoted by ρ_{b_i} , is the total number of flows in $T(b_i)$.

We first characterize the influence of ρ_{b_i} to $w(b_i)$. Suppose there are M hosts communicating with the target area. Let $F(h_i)$ denote the set of flows between servers in the target area and a host h_i outside the target area. Therefore, we have

$$\bigcup_{i=1}^{M} F(h_i) = F = \bigcup_{j=1}^{N} T(l_j)$$
 (2)

Generally speaking, linkmap obfuscation can be formulated as a *Bernoulli experiment*: for any hosts, p_a approaches the overall proportion of its flows obfuscated in the network, which can be illustrated by Equation 3. Hence, higher flow density of a bait link ρ_{b_i} corresponds to a higher p_a .

As a result, increasing flow density of bait links leads to a larger amount of bots obfuscated by Linkbait.

$$p_a \approx \sum_{b_i \in B} \rho_{b_i} / \|F\| \tag{3}$$

However, n_{b_i} should not be too large for the following reasons. Linkbait utilizes existing links in the target area to construct bait links, but such link reuse is resource-constrained. The smaller the target area is, the more difficult to find sufficient links to meet the requirement of ρ_{b_i} . Hence, n_{b_i} should be kept small to ensure that bait links can be flexibly implemented in networks of different size.

As a result, the weight $w(b_i)$ of a bait link b_i should consider both ρ_{b_i} and n_{b_i} , which can be described as follows.

$$w(b_i) \propto n_{b_i}/\rho_{b_i}$$
 (4)

Grouping algorithm: The weight set cover problem is a well-known NP-hard problem [21]. Thus, the formulated link grouping problem is a NP-hard problem. We propose a greedy link grouping algorithm to solve the link grouping problem. As we stated above, a bait link should increase its flow density ρ_{b_i} while maintain a small n_{b_i} . Hence, we search for links to construct bait links according to two principles. First, the links of a bait link can be chosen from several normal links which support a certain amount of traffic instead of links with very low flow density so that we can increase ρ_{b_i} . Second, to reduce n_{b_i} , partial flows to true target links should also be rerouted to bait links. Since true target links is of high ρ_{b_i} , redirecting their flows is an effective way to largely increase the ρ_{b_i} while keep n_{b_i} small at the same time. According to the principles of selecting links, we design the link grouping algorithm. Its formal description is stated as follows:

Algorithm 1 Greedy Link Grouping.

Input:

- 1: Total flows F;
- 2: Flows grouped by links L;
- 3: Bait link coverage threshold τ ;

Output:

- 4: Bait link set *B*;
- 5: % initiate all sets
- 6: B ← ∅
- 7: $F' \leftarrow F$
- 8: % add proper links into B
- 9: repeat
- 0: $l \leftarrow argmax_{X \in L} |X \cap F| / w(X)$
- 11: $B \leftarrow B \cup l, L \leftarrow L \setminus \{l\}, \text{ and } F' \leftarrow F' \setminus L$
- 12: **until** $||B||/||F|| > \tau$

In the greedy link grouping algorithm, Linkbait tries to find the links that best match the two principles(i.e., the least w) and combines them as bait links, until flows in bait links have a satisfying coverage to the total flows in the network.

4.3 Linkmap Obfuscation

The adversary manipulates a large number of bots to create probing flows to the network to obtain the linkmap.

Linkmap obfuscation is proposed to provide a fake linkmap to the adversary and use several bait links to fake target links. To realize this purpose, link obfuscation is proposed with two steps: link-prober identification and selectively flow rerouting. It is worth noting that a link-prober can be a bot or a legitimate host since a legitimate host may also create probing flows to the network just like a bot does. Therefore, the linkmap obfuscation should not affect the objective of legitimate hosts.

4.3.1 Link-prober Identification

This step aims to identify all the link-probers and label their probing flows for the rerouting purpose. As we mentioned earlier, a link-prober can use network diagnostic tools to create probing flows to query the IP information of every hop. In particular, we focus on identifying the traffic generated by traceroute since it is one of the most widely used network diagnostic tools used by both adversaries and the legitimate hosts (e.g., [16] also uses traffic of traceroute to detect links attacked by LFA). It is worth noting that the framework of link-prober also works for other network diagnostic tools due to their similar traffic patterns.

The probing flows enter the network via the ingress routers, so we can implement a real-time monitoring on the ingress traffic to identify the probing flows. However, frequent operations on the ingress routers will introduce heavy burden to them and increase network latency, so we separate the flows towards the target area from numerous traffic and mirror them into a SPAN switch. In this paper, we leverage the SDN controller to analyze the mirrored traffic instead of performing this work on the ingress routers. The controller will examine all the flows through the SPAN switch and identify the probing flows according to the unique features of traceroute.

The probing flows generated by traceroute have two unique features: repeated invalid destination port and different TTL from the same source, which can help us to distinguish the probing flows from other TCP-like flows. Traceroute leverages ICMP Time Exceeded message responded from routers to discover IP-level nodes along router paths. Hence, it continuously sends packets with different TTL values. In addition, traceroute must choose the destination UDP port number to be an unlikely value (e.g., >30,000), making it improbable that an application at the destination is using that port [22]. When traceroute receives a "port unreachable", it knows its task has finished.

Bots that send a sequence of packets containing different TTL and invalid dest ports, to collect link information toward the target area, differ a lot from those legitimate hosts. Since the links to transform traffic from specific areas are always stable (or several hops fluctuation), traffic from normal users (not link-probers) always arrive the ingress of ISP with a comparatively stable TTL as well as a valid dest port, while traffic from link-probers must carry different TTL as well as invalid dest ports. With these two unique features, we can distinguish the probing flows from other flows, and the hosts creating probing flows will be identified as link-probers. Even if the adversary perceives the existence of Linkbait and mutates his probe traffic (e.g., randomizing TTL or prolonging the interval between each packet) to

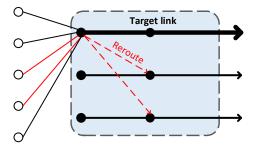


Fig. 3. Rerouting policy for probing flows to a target link. Black lines which are entering the target link are legitimate flows whereas the red one are labeled probing flows.

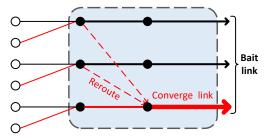


Fig. 4. Rerouting policy for probing flows to a bait link. Black lines entering the bait link are legitimate, and the red one are labeled probing flows

evade from identification, Linkbait can still pick out these probers. This situation will be discussed in Section 5.

After identifying the link-probers, flow tables are installed on the ingress routers to label probing flows of link-probers in a real-time manner. Note that only probing flows of link-probers are labeled while TCP-like flows of link-probers are not modified. These labeled probing flows will be recognized and rerouted in the next step.

4.3.2 Selectively Flow Rerouting

In order to provide a fake linkmap to the adversary, we propose a selectively flow rerouting policy to reroute probing flows accordingly. The basic idea is to reduce probing flows to the target links and increase probing flows to the bait links, so that the bait links will be misjudged as target links by the adversary. We emphasize that only probing flows will be rerouted while other TCP-like flows from the link-probers or other hosts will not be labeled or rerouted. In particular, we have different rerouting policies for probing flows to the target links and the bait links respectively.

Rerouting policy for probing flows to the target link: The true target links are obtained by Linkbait in link sifting. We would like to reroute the probing flows to the target links to other links randomly, so that the adversary cannot figure out the true target links. In order to hide the target links from the adversary, we associate a set of links to each target link, which are called *branch links*. The branch links can be any links in the network which should have small communication latency with the target link. For each probing flow to the target link, as shown in Figure 3, it will be randomly rerouted to one of the branch links of the target link. This random rerouting policy reduces the flow density to each target link, and makes the link information changes

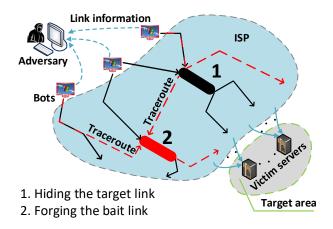


Fig. 5. Illustration of selectively rerouting of probing flows.

dynamically to link-probers so they cannot figure out the true target links.

Rerouting policy for probing flows to the bait link: We intend to use bait links to fake target links, so that the adversary will be misled. In our mechanism, a bait link is not a single link, but contains several links. In order to fake a target link, for the links in a bait link, the one with the largest bandwidth will be selected as the converge link. As shown in Figure 4, for any probing flow to the links of a bait link, it will be rerouted to the coverage link of the bait link. This will increase the flow density of the coverage link and mislead the judgement of the adversary.

Figure 5 illustrates how the probing flows from the bots of the adversary to the target area are selectively rerouted. According to the rerouting policy, only the probing flows (denoted by the red dashed) through the target link will be randomly distributed to its branch links, and the probing flows through the links of a bait link will be converged to its converge link, while the TCP-like flows (denoted by black lines) still go through their original links without any rerouting. Note that branch links of a target link can belong to a bait link. This selectively flow rerouting has several advantages. First, the true target links will be hidden and protected. Second, the bait links will be misjudged as target links by the adversary. Moreover, when the adversary launch attacks to the bait links with TCP-like flows, these flows will not be rerouted and still go through their original links. As a consequence, neither the true target links nor the bait links will be congested by the adversary, so that LFA becomes useless.

We emphasize that only probing flows will be rerouted while other flows from the link-probers will not be labeled or rerouted. If we rerouted all the flows, although the true target links is hidden from the adversary, the converge link of each bait link will be congested by LFA. It is also worth noting that legitimate hosts will not be influenced by selectively flow rerouting. The objective of legitimate hosts is to testify the router-level connectivity, which is different from the objectives of the adversary who wants to obtain the linkmap and the target links. Since selectively flow rerouting only modifies partial hops in a link, traceroute with legitimate purpose can still obtain a valid result.

The linkmap obfuscation component plays the role

of traffic monitor and flow injector in the network. In this paper, we leverage Differentiated Services Code Point (DSCP) [23], which can be agilely adapted into traditional network and generates little disturbances to legitimate users, to deploy linkmap obfuscation component (Please refer to Section 5.2 for the details). During the link-prober identification stage, DSCP is introduced to label the probing flows. The probing flows, marked as a special services in DSCP by ingress routers, will be transmitted by normal routers in ISP, while flows with the special label, will be discriminated by the router in bait links and target links. As a consequence, these links can redirect probing flows according to different policies. Hence, once probing flows enter bait links or target links, it will be redirected so as to obfuscate routes. Note that the DSCP method is only one of various solutions to deploy Linkbait, other suitable methods can be adopted if necessary.

4.4 Bot detection

Different from exiting LFA mitigation mechanisms, Linkbait can actively mitigate LFA by link obfuscation even without the effort of bot detection. However, the network still suffers from junk traffic generated by the bots. In this section, we further propose the bot detection component to accurately distinguish bots from legitimate hosts. It is worth noting that bot detection of Linkbait is different from that of existing schemes. Since we issue early warning about suspicious bots before links are congested, we can leverage routing policies (e.g., TE) to regulate their flows instead of simply preventing them from entering our network. In this way, these legitimate hosts who are identified as malicious bots by Linkbait will not be severely disturbed.

The biggest difference between bots and legitimate linkprobers is that the former tries to dig out as many links as possible while the latter only queries one or two links at one time. Generally speaking, an adversary spends a relatively long time to construct the linkmap since the probing flows were gradually generated by a large number of bots. Once the linkmap is obtained, the adversary will fully utilizes all his bots to flood the target links with TCP-like flows. With our link obfuscation, the adversary will converge his junk flows into bait links at the same time. In this paper, we propose to leverage the unique patterns during the linkmap construction phase and the flooding phase to accurately distinguish bots from legitimate hosts. In particular, we monitor the long-term traceroute traffic and also continuously monitor the short-term flooding traffic with a sliding window. By combining these features, we leverage a supervised learning algorithm to accurately distinguish bots from legitimate hosts.

4.4.1 Feature extraction

We extract two features corresponding to the unique traffic patterns of the bots during the flooding phase and the linkmap construction phase. The first feature is called flooding matrix, which represents the short-term flooding traffic patterns of a link-prober. The second feature is called traceroute matrix, which represents the long-term traffic patterns for linkmap construction of a link-prober. It is worth noting that the flooding matrix is the main feature

that we extract and the traceroute matrix, which aims at decreasing the false-positive rate, is an optional choice for Linkbait. Hence, the bot detection is still effective even though traceroute matrix is not perfectly gathered. This situation will be discussed in Section 5.4.

Flooding Matrix (FM): Once the linkmap is obtained, the adversary will fully utilize all its bots to flood the target links with TCP-like flows. FM represents the flooding behaviors of a host which accesses links during the flooding phase. However, it is unknown to us when the adversary will launch the flooding attacks. To solve this problem, we use a sliding window to continuously detect the flooding behaviors. Note that any host is suspected of being a bot even if it is not a link-prober, hence every host which accesses links should have FMs.

A sliding window consists of n intervals and the sliding windows moves with one interval each time. Let I_i denote the ith interval of a sliding window. We use fs_{ij} to represent the traffic, denoted by the number of bytes, going through $link_i$ during I_j . The FM for a host i during a sliding window can be represented as follows.

$$FM_{host_{i}} = \begin{cases} link_{1} & I_{2} & \cdots & I_{n} \\ link_{2} & fs_{11} & fs_{12} & \cdots & fs_{1n} \\ fs_{21} & fs_{22} & \cdots & fs_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ fs_{m1} & fs_{m2} & \cdots & fs_{mn} \end{cases}$$
(5)

As the sliding windows moves, we can obtain a lot of FMs for each host.

Traceroute Matrix (TM): TM represents the traceroute behaviors of a link-prober during a detection period DT whereas DT is divided into multiple subperiods. Let T_i denote the ith subperiod of DT. For each link-prober, we use ft_{ij} to represent the traceroute frequency of the link-prober towards $link_i$ during T_j .

Suppose there are m links and n subperiods. Thus, the TM for a host i is represented as follows.

$$TM_{host_{i}} = \begin{cases} I_{1} & T_{2} & \cdots & T_{n} \\ link_{1} & ft_{11} & ft_{12} & \cdots & ft_{1n} \\ ft_{21} & ft_{22} & \cdots & ft_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ ft_{m1} & ft_{m2} & \cdots & ft_{mn} \end{cases}$$
(6)

Therefore, each link-prober has a TM to represent the traffic pattern during the detection period of DT. The value of DT depends on how much time spent by the adversary to construct the linkmap, which varies from an adversary to another. Generally speaking, link distribution dynamically changes due to load-balancing, so the adversary will launch LFA immediately after linkmap construction. We set DT to be 5 days since adversaries usually take less than 5 days to obtain the linkmap. Note that DT cannot be too large or too small. If DT is too small, we cannot discover the traceroute behaviors of all bots manipulated by the adversary since it gradually uses bots to obtain the linkmap. While if DT is too large, too much efforts are wasted for monitoring the large traffic in the network.

4.4.2 Classification

With the extracted features, we then leverage a supervised classification algorithm to distinguish bots from legitimate hosts. Each link-prober has one TM and many FMs while other hosts only have FMs. In particular, we combine all FMs together to form the joint-FM for each host. In our experiments, we collect a ground truth dataset where each sample has a label to indicate the corresponding host is a bot or a legitimate user (e.g., 0 indicates a bot and 1 indicates a legitimate user). We divide the ground truth dataset into a training set and a testing set. The training set is used to train a supervised classifier which then predicts the label of each sample in the testing set. The classification accuracy can be calculated by comparing the predicted labels of samples in the testing set with their true labels. In particular, a linear multiclass Support Vector Machine (SVM) classifier implemented by libSVM3 [24] is employed for accurate classification.

5 Discussion

In this section, we first analyze why the adversary cannot congest the network under Linkbait, and then discuss the feasibility of identifying the link-probers under various circumstances. At last, we discuss the effectiveness when adversaries perceive Linkbait.

5.1 Influence Linkbait can exert on adversaries

The adversary aims at disjointing connections between the target area and Internet, so it tends to congest as many links as possible by using a limited number of bots. Suppose each bot has an upstream bandwidth U.

LFA without Linkbait: To saturate a target link with bandwidth B, the adversary utilizes $N_p = B/U$ bots whose flows can go through the target link. Using N_p bots guarantees a robust congestion even if there is no legitimate flows in the link. We consider this as an ideal condition because not every bot can be fully utilized in common situations. Some bots are not able to congest target links chosen by the adversary, so that they keep unused in the attack. Let us denote the amount of these unused bots as N_{un} ($N_{un} \ll N_p$). Thus, the number N_b of bots which the adversary need to finish the attack to a target area including n target links is calculated as

$$N_b = n \cdot N_p + N_{un} \tag{7}$$

LFA with Linkbait: Linkbait hides the true target links and misleads the attacks of the adversary to the bait links. Since the target links are hidden from the adversary, so they would not be attacked or congested. In our experiments, as shown in Table 1, we found that almost all the traffic to the target area go through the limited number of target links. In other words, once the target links are protected, the traffic to the target area would not be congested.

Let us then consider the attack to the bait links. Suppose a bait link consists of M normal links. When the adversary created TCP-like flows to congest the bait link, the flows will be distributed to M links for each bait link. Let $\alpha_i B$ denote the bandwidth of the i link of a bait link. In order to congest

the bait link, the adversary is supposed to use N_l bots which is calculated as follows.

$$N_l = \frac{\alpha_1 B + \alpha_2 B + \dots + \alpha_M B}{U} = \sum_{i=1}^M \alpha_i N_p$$
 (8)

That is, the number of bots required to flood a bait link is $\sum_{i=1}^{M} \alpha_i N_p$. We can observe that if $\alpha_i = 1$, the attack cost of the adversary has been forced to increase from N_p to MN_p . Moreover, the more number of links to form a bait link, the higher the attack cost is required for the adversary.

5.2 Linkbait is feasible in real-world networks

In this section, we discuss whether Linkbait is feasible in real-world applications. We mainly discuss it from two perspectives: the simplicity of implementing Linkbait and little disturbances on legitimate users.

In Linkbait, we choose DSCP, which is served as the identifier of flows, to build the link obfuscation component. Employing DSCP to build Linkbait has the following advantages. First, DSCP has been widely supported by most vendors (e.g., Cisco [25] and Huawei [26]), so Linkbait based on DSCP is compatible with current hardwares. Second, given that DSCP has been employed by ISP for many purposes, such as load-balancing, reconfiguring DSCP to adapt Linkbait requires no more effort than simply adding one rule even in large scale networks, which reduces the burden of deployment and maintenance. Third, the logical structure of linkmap obfuscation component, which consists of an ingress monitor and distributed obfuscation injectors, can be rendered by DSCP perfectly.

For the little disturbances on legitimate users, there are two situations to consider. When network failures happen outside the obfuscation areas, traceroute with legitimate purpose (e.g., network diagnosis) can still obtain a valid result and thus figure out the node of failures, because selectively flow rerouting only modifies partial hops which only distribute in obfuscation areas. Once nodes which cause the failure are from obfuscation areas, various recovering mechanisms are provided by DSCP can agilely handle the failures so as to fast recover its network usability.

5.3 Link-prober identification is versatile to various probers

In Linkbait, we identify a host as a link-prober if it repeatedly creates flows to reach every hop of a link. The identification is feasible because the fact that a link-prober can only fetch informations of one hop in a link every time he requests if he wants to obtain link information towards the target area.

As we have mentioned, the adversary uses network diagnostic tools to collect link information of the network. The most significant characteristic of these tools is that it probes only one hop every time. This is due to the intrinsical nature of Internet routing protocols that packets will be informed the next hop only after reaching a router. Since a link-prober has no idea where his packet will be directed, he must query hops in the link repeatedly using tools like traceroute. Hence, the probing flows reveal the same feature no matter which network diagnostic tools are used.

In addition to that, it is the complicated hop discovering pattern, which the adversary must obey, that reliefs the real-time reacting requirement for Linkbait. Since the adversary requires a comparatively long time to discover an entire link, Linkbait has enough time to perform traffic analysis and identify the probers before real links are disclosed.

5.4 Case study: What if adversaries perceive the existence of Linkbait

In most situations, adversaries perform link-probe stage in a comparatively constant period. However, once the adversary perceives the existence of Linkbait, he might launch link-probe in a very flexible way so as to evade from the sliding windows based detection. In the following, three different methods of link-probe are presented and we discuss the effectiveness of Linkbait against them.

Prolonging interval between each link-probe: An adversary might probe links in a very long time interval using a large number of bots. Once a bot can send intermittent probes in a long time period and escape from the detection window, it seems that Linkbait cannot detect these bots. However, this is not true for Linkbait.

Firstly, we combine two features to perform fine-grained bot detection. Even TM can be annulled by the adversaries' countermeasure, FM which reflects the short-time flooding pattern during the flooding stage of the adversary will still remain effective, since the adversary must launch attacks with all his bots at the same time in order to deplete the bandwidth. In the absence of TM generated by traceroute, the detection accuracy can still be satisfying, which possesses a bot detection rate higher than 86%. The probe feature which aims at improving detection accuracy is just an optional choice for Linkbait. Therefore, Linkbait's effectiveness will not be diminished without it. As we have mentioned in bot detection, Linkbait can actively mitigate LFA by link obfuscation even without the effort of bot detection. Even though bots might escape from detection windows, they still get deceived. The floods towards a bait will encounter no more than stable transmission. Secondly, the linkmap accuracy of long-term probing suffers from inevitably periodic link changes in ISP, which diminishes the adversaries' threats.

Randomizing packet headers for each packet: The attacker can also randomize header of packets to evade from link-prober identification. But it is in vain due to the fundamental differences between the link-probe of bots and normal access. By repeatedly sending the same TTL from one bot, the adversary seems to mask the fact that his bots carry changing TTL. However, it exacerbates the other feature, the invalid ports. Hence, Linkbait can still find out the bots.

One in all link-probe: It is common for a adversary to compromise a large network and manipulate all hosts in the network. Since these bots share the same or nearby IP segments, they also share similar links towards the target area. Hence it is feasible for the adversary to reduce the risk of being captured by reducing the number of bots performing link-probe. He can just employ partial bots in the network instead of all to perform link-probe and then launch the attack with all bots. In this way, the bots which



Fig. 6. Geographical distribution for LG servers of Telia (red pins) and Cogentco (blue pins).

do not perform link-probe do escape from the link-prober identification. However, once the bots which perform link-probe are identified, the whole IP segment can be marked as suspected. Linkbait can still identify the suspected bots which belong to these IP segment during the flooding. When these bots try to compromise the bait links, restricts can be imposed on them so as to remove these junk traffic out of the network.

6 Performance Evaluation

In this section, we evaluate the performance of Linkbait by using both real-world experiments and large-scale simulations. In particular, we first implement link sifting in real-world networks to show the feasibility of deploying Linkbait in a real ISP. We then implement a prototype on a real SDN testbed to see whether Linkbait can obfuscate the linkmap while maintaining a low network latency. Finally, we simulate large-scale LFA and evaluate the accuracy of bot detection with large-scale simulations.

6.1 Internet-scale Link Sifting

Bait links are the key to the success of Linkbait, which can fake target links and selectively reroute flows. Therefore, we implement link sifting in real-world Internet to see whether we can find enough bait links to attract the attention of the adversary.

We choose five places as the target areas in Internet. In particular, five ASes scattering in the United States are chosen in our experiments. We leverage 126 LG servers provided by Telia [27] and Cogentco [28] to launch LG tracing. Both of them provide web interfaces for users to run traceroute. As seen in Figure 6, these LG servers are globally distributed in 33 countries, and 68 of these servers are provided by Telia while 58 are provided by Cogentco. By using globally distributed LG servers we could find as many paths as possible whose destinations are the ASes we choose.

6.1.1 Basic link information

The basic link information of the five ASes are collected using LG tracing, and the detailed information for each AS is shown in Table 1. The *Avg Hop Num* represents the average number of hops in all paths towards each AS. The *Non-identical Path* represents the number of different links from all LG servers to each AS. The flow density of a link is

TABLE 1
Basic link information for 5 ASes.

Info Area	Avg Hop Num	Non-identical Paths	Coverage of Top-15 flow-density links
AS1	12	603	1.0
AS2	12	804	0.84
AS3	16	497	1.0
AS4	14	353	1.0
AS5	12	792	0.81

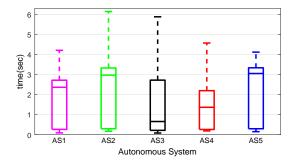


Fig. 7. Time for traceroute.

estimated by the times it appears in the results of traceroute of all the LG servers. The *Coverage of Top-15 flow-density links* is the fraction of the number of LG servers which can access the top 15 highest flow density links to the total number of LG servers. It can be used to estimate the flow distribution of the network. As seen in Table I, the coverage for AS1, AS3 and AS4 reach 100% because almost all paths from different LG servers to each AS share the same link at the end servers, making the coverage 100%. The network size can be seen from the non-identical paths. We observe that AS4 has the least number of paths whereas AS2 has the most, which indicates that AS3 is the smallest network whereas AS2 is the largest. In this way, we can guarantee that our algorithm works for various network topologies.

6.1.2 Traceroute time cost

To investigate the relationship between link latency and the scale of ASes, we show the time spent of every single hop for each AS during traceroute in Figure 7. We can see that the longest time spent for one hop is 6.14s due to network latency, whereas the lowest is 0.076s. The average time spent for discovering one node of all 5 ASes is 1.72s. Combining with Table 1, we observe that networks with more number of non-identical paths have a longer average time cost on every hop. This is because more number of non-identical paths leads to a more complex and larger network, which possesses a comparatively long geographical distance and therefore a large latency.

6.1.3 Link coverage

As we mentioned, link sifting tries to hide the target links and find as many bait links as possible to obfuscate target links with the least cost. *Link coverage* here is estimated by the fraction of the number of LG servers which can be obfuscate by bait links to the total number of LG servers. The higher the link coverage, the better obfuscation the network.

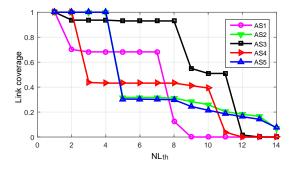


Fig. 8. Link coverage for five target areas.

In our experiments, we measure the link coverage for the 5 ASes with our grouping algorithm. Since the minimum number of links to form a bait link NL_{th} demonstrates resistibility of Linkbait against flood, we change NL_{th} while maintaining the least cost with our grouping algorithm. It is worth noting that we cannot obtain the actual bandwidth or the flow density of links, ρ_{b_i} , in ISP, so we use the fraction of the number of LG servers that can travel through the link to estimate. Figure 8 shows the link coverage against the variation of NL_{th} . We can see that the link coverage decreases as NL_{th} increases, which is because less links in the network are chosen to form bait links. However, almost for all ASes, the link coverage can reach 70% when $NL_{th} \leq 4$. The only exception is AS4 which mainly because its smallscale network has limited number of links for sifting. Based on these observations, we argue that link sifting can realize a satisfying linkmap obfuscation with an appropriate NL_{th} that makes Linkbait stable enough.

6.2 Evaluation using Real Testbed

We then implement Linkbait in a real testbed to evaluate its performance. In particular, we focus on the rerouting latency introduced by Linkbait since our mechanism should not affect legitimate hosts or be perceived by the adversary. Therefore, the lower latency introduced by Linkbait, the better link obfuscation to the adversaries. Thus, we build Linkbait on a real testbed to evaluate the rerouting latency.

6.2.1 Prototype Implementation

We employ software defined network (SDN) to implement the DSCP based network prototype over physical nodes and links provided by Cloudlab [29]. Cloudlab provides physical machines across three sites around the United States: Utah, Wisconsin, and South Carolina. It also provides 2×10 Gbps network interfaces to every node via SDN. It is worth noting that SDN is not the prerequisite of Linkbait implementation, since the DSCP method can be deployed in traditional networks as well. We implement Linkbait on the Floodlight [30] controller. We use OVS [31] to virtualize layer-2 switches which support OpenFlow 1.3 [32] to perform selectively rerouting. Note that links in our experiments consist of layer-3 routers whereas only rerouting sites are implemented as switches. In addition, we leverage iPerf [33] to emulate the legitimate TCP-like flows in the network.

We build a experimental network with two edges (ingress/egress routers) in the prototype. We build such a network because its structure is similar to a simplified ISP

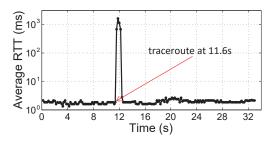


Fig. 9. Real-time RTT change for legitimate hosts. Traceroute is performed at 11.6s.

\$traceroute wikipedia.org tracerouse wikipedia.org (66.230.200.100), 64 hops max, 44 byte packets
1 124.ae0.xrl.3d12.xs4all.net (194.109.21.1) 0.305 ms 0.360 ms 0.405 ms
2 0.so-6-0-0.xrl.tc2.xs4all.net (194.109.21.1) 0.305 ms 0.736 ms 0.673 ms
3 ams-1x-000.wvfiber.net (195.69.145.58) 0.638 ms 0.601 ms 0.551 ms
4 lon-c00-pos-4-0.0048-ams-posl1-0.wvfiber.net (63.223.28.201) 7.512 ms 7.427 ms 7.494 ms
5 nyc60-pos-1-0.0048-lon-c00-pos-3-0.wvfiber.net (63.223.28.145) 84.108 ms 83.804 ms 83.995 ms
6 66.216.1181 (66.16.1181) 83.435 ms 83.278 ms 83.384 ms
7 ash-c01-tge-3-3.TG-nyc-c01-1-1.wvfiber.net (66.216.1.161) 89.563 ms 89.554 ms 89.551 ms
8 atl-c01-tge-3-3.TG-nyc-c01-1-1.wvfiber.net (66.216.1.161) 89.563 ms 89.554 ms 69.556 ms
9 cpp-hostway.wvfiber.net (63.223.8.26) 103.678 ms 103.609 ms 103.630 ms
10 e1-12.co2.as30217.net (64.156.55.105) 113.014 ms 113.004 ms
11 10ge5-1.csw5-pmtpa.wikimedia.org (84.40.25.102) 113.153 ms 113.251 ms 113.100 ms

Fig. 10. An example of traceroute on Wikipedia.org.

network. A bait link with three parallel links is deployed between the edges. Among the three link, there is a converge link L_c for link obfuscation, and there is another link L_o through which both link-probers and legitimate users communicate with the target area. In addition, there is a link L_l which only contains legitimates flows. It is worth noting that the effect of flow rerouting can be demonstrated even with only one bait link, since rerouting in a large system just uses a combination of several bait links.

6.2.2 Rerouting Latency

The Rerouting latency is an important metric for evaluating the performance of Linkbait. On one hand, the network jitter caused by rerouting should not disturb legitimate hosts. On the other hand, since latency may produce a deviation for the result of link-probers (traceroute), we should reduce this jitter in case the adversary perceives that the linkmap has been faked.

Rerouting Impact on legitimate hosts. To investigate the impact on legitimate hosts, we measure Round-Trip Time (RTT) during three different stages in Linkbait. Figure 9 illustrates the RTT change for legitimate hosts. The link L_o supports all link-probers' traffic before 11.6s. Our mechanism identifies and reacts immediately after a link-prober performs traceroute at 11.6s. The response time includes the time for link-prober identification and the time for pushing corresponding labeling flow table to Openflowenabled switches. As shown in Figure 9, legitimate hosts who communicate with the target area via L_o experience a temporary block. However, RTT quickly returns to a normal value when Linkbait handles traceroute of the link-prober.

Rerouting Impact on Link-probers. To investigate the impact on link-probers, we record the result of traceroute during rerouting policy takes effect. Figure 10 demonstrates an example of traceroute to wikipedia.org. When traceroute runs, traceroute outputs the list of traversed routers in a simple text format, together with the timing information. With the list of routers, a link-prober restores a link. In addition, the link-prober can observe whether every hop

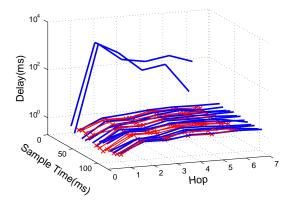


Fig. 11. The timing information of traceroute time cost in a probing flow rerouting period. Each line reflects traceroute latency fluctuation as time changes for every hop. Blue lines represent changes for Linkbait whereas red lines are the baselines without our defending policy.

works fine according to its traceroute delay. An abnormal delay may alert the adversary, which leads to a failed obfuscation.

Figure 11 shows the timing information for traceroute command with and without our rerouting policy. We create probing flows every 7 ms into the network. Each blue line shows the average router response time for each hop of a probing flow with the rerouting policy. Each red line shows the average router response time for each hop of a probing flow without rerouting policy, which is considered as a baseline.

From Figure 11, we observe that probing flows experience an inevitable temporary block around the second hop when Linkbait starts to reroute its flow to L_c . This is because the rerouting operation occurs at that hop. We also observe that the latency of the following hops drops quickly, which indicates that our rerouting policy only blocks several hops rather than the whole link. Hence, Linkbait achieves a robust flow rerouting from L_o to L_c . We also observe that time spent for querying every hop with the rerouting policy only slightly differs from that of baselines. The average time spent for every hop is 1.32ms whereas that of baselines is 1.24ms. As a consequence, Linkbait seamlessly obfuscates a linkmap before adversaries obtain the real one. In this experiment, we evaluate the effect of Linkbait on network delay over physical nodes. Our system generates a short delay to identify traceroute and deploy rerouting policy before network delay backs to normal.

6.3 Evaluation using Large-scale Simulation

In this section, we further evaluate the performance of bot detection of Linkbait by using large-scale simulations. We use the real link information collected from the real-world Internet in Section 6.1 in the experiments of simulations. We deploy 100 bots, 190 legitimate hosts and 20 servers in the target area. The legitimate hosts send packets to servers at different rates in order to get services from them. The bots send flooding flows to launch LFA to the servers.

In Linkbait, we extract the FM and TM features and use SVM to distinguish bots from legitimate hosts. The evaluation mainly focus on (1) bot detection rate and (2) false-positive rate. Let TP denote the number of correctly identified bots,

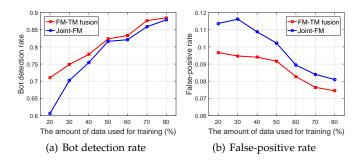


Fig. 12. The impact of the percentage of data used for training on the performance of bot detection when joint-FM features or fused FM-TM features are used.

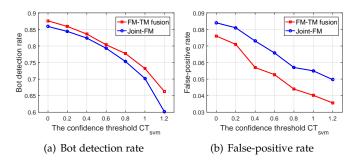


Fig. 13. The impact of confidence threshold on the performance of bot detection when joint-FM features or fused FM-TM features are used.

TN denote the number of correctly identified legitimate hosts, FP denote the number of legitimate hosts wrongly identified as bots, FN denote the number of bots wrongly identified as legitimate hosts. Then, the *bot detection rate* is defined as

bot detection rate =
$$\frac{TP}{TP + FN}$$
. (9)

The false-positive rate is defined as

$$false-positive \ rate = \frac{FP}{TN + FP}. \tag{10}$$

We expect that the bot detection rate should be as high as possible while the false-positive rate should be as low as possible.

Performance vs. Data used for training: We first evaluate the performance against the percentage of data used for training. Figure 12 shows the bot detection performance against the percentage of joint-FM features or fused FM-TM features used for training. The bot detection rate increases and the false-positive rate decreases as the percentage of data used for training increases. This is because the SVM classifier is more accurate with larger amount of training data. We can also observe that the bot detection rate using fused features is better than using only joint-FM features under the same parameters. The false-positive rate using fused features is lower than using only joint-FM features under the same parameters. Therefore, it is better to combine joint-FM and TM together to accurately distinguish bots from legitimate hosts. When 80% data are used for training, the bot detection rate can reach 88.5% while the false-positive rate drops under 7.5%.

TABLE 2 Mechanism comparison.

Feature Scheme	Technology	Traffic Affected	Time to take effect	Adversary-Detecting Method	Need Collaboration
Linkbait	Moving Target Defense	Partial traffic	Before congestion	Machine Learning	N
Spiffy	Traffic Engineering	All traffic	On congestion	Temporary Bandwidth Expansion	N
CoDef	Collaborative routing	All traffic	On congestion	Rerouting Compliance Test	Y
Liaskos	Traffic Engineering	All traffic	On congestion	Traffic Engineering Optimal Mapping	N

Performance vs. Confidence Threshold CT_{svm} : The label of a new sample predicted by SVM classifier is attached with a confidence score which indicates the confidence level of predicting the sample as that category. We only treat the predicted bots as real bots when the confidence score of predicted bots is larger than a Confidence Threshold CT_{svm} . Figure 13 shows the impact of confidence threshold on the performance of bot detection when 70% data are used for training. We can see that both the bot detection rate and the false-positive rate decrease as the confidence threshold increases. This is because more low-confidence bots judgements are refused when CT_{svm} is higher. As a result, less legitimate hosts are wrongly identified as bots and suspected bots are more likely to slip away. We can also observe that using TM and FM features together can achieves better performance than using joint-FM features only. When CT_{svm} is 0.6, the false-positive rate is near 5% while the bot detection rate is still above 80%. Note that the detection accuracy can still be satisfying even without TM features.

If bots are detected, ISP can employ various methods to eliminate their flooding traffic from the network in a low false-positive rate. It is worth nothing that LFA only occurs on bait links. As we illustrate in Section 5.1, bait links with Linkbait can resist congestion with its extended bandwidth. Hence, LFA can be early mitigated before it takes effects on true target links.

6.4 Comparison with Existing Schemes

In this section, we compare Linkbait with the state-of-the-art of LFA mitigating schemes (e.g., Spiffy [13], CoDef [9] and Liaskos [10]).

Table 2 shows their comparisons from five aspects. All the other three mechanisms are based on TE while Linkbait is inspired by MTD and uses link obfuscation to mitigate the effects of LFA. Linkbait only reroutes the probing flows, and the normal TCP flows will not be affected while the other three mechanisms affect all the traffic flows. Time to take effect tells us when the system starts to handle LFA. As a matter of fact, the earlier the system reacts, the little congestion the network will suffer. In particular, Linkbait uses link obfuscation to mitigate LFA before the attacks congest the links while the other mechanisms take effect when the attacks already happen. Also, Linkbait can be implemented by one single ISP without no collaboration with other ISPs, which reduces the cost and complexity. Therefore, we can conclude that Linkbait can actively mitigate LFA before the the network is congested without affecting legitimate flows, and has low implementation cost and high bot detection rate.

7 CONCLUSION

In this paper, we propose Linkbait to actively mitigate LFA by providing a fake linkmap to the adversary. To the best of our knowledge, we are the first to early mitigate LFA before congestion happens, which is total different from existing works that mitigate LFA after the links are comprised by adversaries. The core of Linkbait is link obfuscation that selectively reroutes probing flows to hide target links from adversaries and mislead them to consider bait links as target links. Furthermore, we extract unique traffic features from both the linkmap construction phase and the flooding phase, and leverage SVM to accurately distinguish bots from legitimate hosts. The experiments with real-world testbed and large-scale simulations demonstrate the feasibility and effectiveness of Linkbait. The experimental results show that Linkbait introduces a very small rerouting latency and achieves a high bot detection rate while maintaining a low false positive rate.

REFERENCES

- J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," in *Proc. of ACM SIGCOMM*, 2004, pp. 39–53.
- [2] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, 2013.
- [3] DDoS attack using mirai botnet https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.
- [4] A. Studer and A. Perrig, "The coremelt attack," in *Proc. of ES-ORICS*, 2009, pp. 37–52.
- [5] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *Proc. of IEEE S&P*, 2013, pp. 127–141.
- [6] How extorted e-mail provider got back online after crippling DDoS attack http://arstechnica.com/security/2015/11/ how-extorted-e-mail-provider-got-back-online-after-crippling-ddos-attack/.
- [7] Hearthstone and other Blizzard games suffer DDoS attacks http://www.pocketgamer.biz/news/64018/hearthstone-and-other-blizzard-games-suffer-ddos-attacks/.
- hearthstone-and-other-blizzard-games-suffer-ddos-attacks/.
 [8] L. Xue, X. Luo, E. W. Chan, and X. Zhan, "Towards detecting target link flooding attack," in *Proc. of LISA*, 2014, pp. 90–105.
- [9] S. B. Lee, M. S. Kang, and V. D. Gligor, "Codef: collaborative defense against large-scale link-flooding attacks," in *Proc. of ACM CoNEXT*, 2013, pp. 417–428.
- [10] C. Liaskos, V. Kotronis, and X. Dimitropoulos, "A novel framework for modeling and mitigating distributed link flooding attacks," in *Proc. of IEEE INFOCOM*, 2016.
- [11] D. Gkounis, V. Kotronis, C. Liaskos, and X. Dimitropoulos, "On the interplay of link-flooding attacks and traffic engineering," in *Proc. of ACM SIGCOMM*, 2016, pp. 5–11.
- Proc. of ACM SIGCOMM, 2016, pp. 5–11.
 [12] L. Wang, Q. Li, Y. Jiang, and J. Wu, "Towards mitigating link flooding attack via incremental sdn deployment," in Proc. of IEEE ISCC, 2016, pp. 397–402.
- [13] M. S. Kang, V. D. Gligor, and V. Sekar, "Spiffy: Inducing costdetectability tradeoffs for persistent link-flooding attacks," in *Proc.* of NDSS, 2016.
- [14] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using sdn-based moving target defense," in *Proc.* of IEEE Local Computer Networks (LCN), 2016, pp. 627–630.

[15] E. Al-Shaer and S. F. Gillani, "Agile virtual infrastructure for cyber deception against stealthy ddos attacks," in *Cyber Deception*. Springer, 2016, pp. 235–259.

[16] T. Hirayama, K. Toyoda, and I. Sasase, "Fast target link flooding attack detection scheme by analyzing traceroute packets flow," in

Proc. of IEEE WIFS, 2015, pp. 1–6.

[17] F. Gillani, E. Al-Shaer, S. Lo, Q. Duan, M. Ammar, and E. Zegura, "Agile virtualized infrastructure to proactively defend against cyber attacks," in *Proc. of IEEE INFOCOM*, 2015, pp. 729–737.

[18] D. Kewley, R. Fink, J. Lowry, and M. Dean, "Dynamic approaches to thwart adversary intelligence gathering," in *Proc. of IEEE DIS-*CEX, 2001, pp. 176–185.

[19] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *Proc. of ACM SIGCOMM*, 2002, pp. 133–145.

[20] Looking Glass server https://en.wikipedia.org/wiki/Looking_ Glass_server/.

[21] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[22] K. R. Fall and W. R. Stevens, TCP/IP illustrated, volume 1: The protocols. addison-Wesley, 2011.

[23] Y. Bernet, "The complementary roles of rsvp and differentiated services in the full-service qos network," *IEEE Communications Magazine*, vol. 38, no. 2, pp. 154–162, 2000.

[24] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, no. 3, p. 27, 2011.

[25] Cisco http://www.cisco.com.

[26] Huawei http://www.huawei.com/en/.

[27] Telia's LG http://lg.telia.net.

[28] Cogentco's LG http://www.cogentco.com/en/network/looking-glass.

[29] Cloudlab http://www.cloudlab.us/.

[30] Floodlight http://www.projectfloodlight.org/floodlight/.

[31] OpenVswitch http://openvswitch.org.

[32] OpenFlow Switch Specification v1.3 https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf.

[33] IPerf https://iperf.fr.

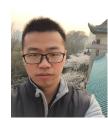


Qian Wang received the B.S. degree from Wuhan University, China, in 2003, the M.S. degree from Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, China, in 2006, and the Ph.D. degree from Illinois Institute of Technology, USA, in 2012, all in Electrical Engineering. He is currently a Professor with the School of Computer Science, Wuhan University. His research interests include wireless network security and privacy, cloud computing security, and applied

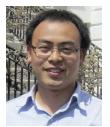
cryptography. Qian is an expert under "1000 Young Talents Program" of China. He is a co-recipient of the Best Paper Award from IEEE ICNP 2011. He is a Member of the IEEE and a Member of the ACM.



Feng Xiao is an undergraduate in the School of Computer, Wuhan University, China. He is currently working at the NIS&P Lab, where he works with immense zeal to seek and realize any fancy ideas about cyber security. He was the recipient of the first prize in the National Undergraduate Information Security Contest, China in 2016, and Second Prize and Most Potential Student Prize of 2015 National SDN innovative programming Contest. He won the National Scholarship in 2016, and Yuanyi scholarship in 2015.



Man Zhou is working toward the M.S. degree at the School of Computer, Wuhan University, China. His research interests include mobile system and security, and wireless network security. He published two papers on IEEE ICDCS'15 and ACM MobiCom'16. He was the recipient of the first prize in the "National Undergraduate Internet of Things Competition, China" in 2015, and the first prize in the "National Graduate Contest on Application, Design and Innovation of Mobile-Terminal, China" in 2016.



Zhibo Wang received the B.E. degree in Automation from Zhejiang University, China, in 2007, and his Ph.D degree in Electrical Engineering and Computer Science from University of Tennessee, Knoxville, in 2014. He is currently an Associate Professor with the School of Computer, Wuhan University, China. His currently research interests include crowdsourcing, mobile sensing and computing, recommender systems and privacy-preserving data mining. He is a member of IEEE and ACM.



Qi Li is an associate professor in the Graduate School at Shenzhen, Tsinghua University. His research interests include system and network security, particularly in Internet security, mobile security, and security of large-scale distributed systems. Li has a PhD in computer science from Tsinghua University. Contact him at qi.li@sz.tsinghua.edu.cn.



Zhetao Li is a professor in College of Information Engineering, Xiangtan University. He received the B. Eng. degree in Electrical Information Engineering from Xiangtan University in 2002, the M. Eng. degree in Pattern Recognition and Intelligent System from Beihang University in 2005, and the Ph.D. degree in Computer Application Technology from Hunan University in 2010. From Dec. 2013 to Dec. 2014, he was a post-doc in wireless network at Stony Brook University. From Dec. 2014 to Dec. 2015, he was

an invited professor at Ajou University. His research interests include wireless communication and multimedia signal processing, with about 30 papers published. He received the Second Prize of Fok Ying Tung Education Foundation Fourteenth Young Teachers Award in 2014.