LinkScope: Towards Detecting Target Link Flooding Attacks

Lei Xue, Xiaobo Ma, Xiapu Luo, Edmond W.W. Chan, Tony T.N. Miu, Guofei Gu

***Abstract*—A new class of target link flooding attacks (LFA) can cut off the Internet connections of a target area without being detected because they employ legitimate flows to congest selected links. Although new mechanisms for defending against LFA have been proposed, the deployment issues limit their usage since they require either additional modules to enhance routers or using the software-defined network (SDN) to replace the traditional routers. In this paper, we propose a novel framework that employs both the end-to-end and the hop-by-hop network measurement techniques to capture abnormal path performance degradation for detecting LFA and then locate the target links or areas whenever possible, and develop a prototype of the framework named *LinkScope*. Although using network measurement to capture network anomaly is not new, we tackle a number of challenging issues, such as conducting large-scale Internet path monitoring via non-cooperative measurement so that users do not need to install *LinkScope* on every host, profiling the performance of asymmetric Internet paths, and detecting LFA. The extensive evaluation in a testbed and the Internet shows that with limited bandwidth and computational overhead *LinkScope* can achieve timely detection and diagnosis of LFA with high detection rate and low false positive rate.**

I. Introduction

DDoS attacks remain one of the major threats to the Internet, and there is a significant increase in the number and the size of DDoS attacks recently [1], [2], not to mention the 300 Gbps direct flooding attacks on Spamhaus and the record-breaking 400 Gbps NTP reflection attack on CloudFlare. It is not difficult to detect such traditional bandwidth DDoS attacks, because the attack traffic usually reaches the victim and has an obvious difference from legitimate traffic [3].

Recent research has discovered a new class of target link flooding attacks (LFA) that can effectively cut off the Internet connections of a target area (or network) *without* being detected [4], [5]. For example, the LFA against NetEase in 2015 flooded the major links connected to NetEase's game services and made it unavailable for 9 hours [6]. Moreover, LFA has been used by attackers to flood selected links of four major Internet exchange points in Europe and Asia [7]. To launch LFA, the attackers first select persistent links that connect the target area to the Internet and have high flow density, and then instruct bots to generate legitimate traffic between bots and public servers to congest those links [5]. If the paths among bots cover the target area, the attackers can also send traffic among themselves to clog the network [4].

Lei Xue, Xiaobo Ma, and Xiapu Luo are with the Department of Computing at the Hong Kong Polytechnic University; Xiaobo Ma is also with the MOE KLINNS Lab at Xi'an Jiaotong University. Edmond W.W. Chan is with Akamai Inc.; Tony T.N. Miu is in Nexusguard Limited; Guofei Gu is with the Department of Computer Science and Engineering at Texas A&M University.

Detecting LFA is difficult because 1) the target links are selected by attackers. Since the target links may be located in an AS different from that containing the target area and the attack traffic will not reach the target area, the victim may not even know he/she is under attack; 2) each bot sends low-rate protocol-conforming traffic to public servers, rendering signature-based detection useless; 3) bots can change traffic patterns to evade abnormal traffic pattern-based detection.

Traditional anomaly detection [8] and traffic filtering systems (e.g., IDS, IPS) deployed at the network perimeter *cannot* handle LFA by design. The reason is that, to degrade the connectivity of the target network, LFA overwhelms the selected upstream links of the target network, which are beyond the monitoring scope of the traditional systems. Due to the direct control over the traffic crossing critical links, router-based approaches have been proposed to defend against LFA and other smart DoS attacks [7], [9]–[16]. Despite their effectiveness, their availability is limited in the foreseeable future as a result of the constraints by practical factors, such as deployment overhead [7], network externalities [17], and ISP commercial adoption incentives [18] (see Section VII for details). Similarly, although several promising SDN-based defense mechanisms [19]–[22] have been proposed, they may not be immediately deployed because they need to replace the traditional routers with SDN devices. Therefore, it is urgent to devise a framework for developing practical systems that have visibility to LFA and are immediately deployable without modifying/affecting existing Internet routing infrastructure. Such systems can help victims detect LFA and locate the links under attack whenever possible so that victims could ask help from the corresponding providers to mitigate such attacks.

We fill this gap by proposing a novel framework that employs end-to-end and hop-by-hop non-cooperative measurement techniques to achieve the goal mentioned above. Our framework exploits the nature of LFA [5], including 1) LFA causes severe congestion on persistent links because light congestion cannot disconnect the target area from the Internet; 2) to evade detection and avoid route changes [5], the period of congestion caused by LFA is much shorter than that caused by traditional bandwidth DDoS. Meanwhile, the congestion period caused by LFA is long enough to cause damages to victims; 3) to cut off the Internet connections of a target area, LFA keeps attacking important links. These observations motivate the design of four major modules in our framework, including (1) topology collection and analysis module, (2) non-cooperative path measurement module, (3) attack detection module, and (4) target link localization module. Their functionalities will be detailed in Section II. Basi-

cally, by actively collecting samples of network path performance (through non-cooperative path measurement module), our framework employs abnormal performance degradation to detect LFA (by attack detection module), and leverages the information collected through various measurement (through topology collection and analysis module and non-cooperative path measurement module) to locate the links under attack whenever possible (by target link localization module).

Our major contributions lie in proposing a novel framework for defending against LFA and developing a practical LFA detection system (dubbed *LinkScope*) based on this framework after tackling several challenging technical issues, including:

- Since the target links are selected by attackers, victims have to monitor as many paths as possible. However, the majority of existing network measurement systems have limited scalability because they require installing measurement tools on both ends of each path [23]–[25]. We solve this issue from two aspects. First, we design new non-cooperative measurement approaches that only need the installation on one end of a path, and the other end could be any TCP-based server. Therefore, it can cover many more paths than the existing systems. Second, we strategically select target links for probing [26].
- To handle the prevalence of asymmetric routes [27], our new non-cooperative measurement approaches can differentiate the performance metrics on the forward path and the reverse path, and thus allow users to infer which path(s) is under attack.
- By conducting hop-by-hop measurement, our framework attempts to locate the target link or the target area on the forward path whenever possible. Although our framework may not locate the target link on the reverse path in the absence of reverse traceroute data, we will explore possible solutions, such as reverse traceroute, looking glass [28]–[34], in future work. To our best knowledge, *LinkScope* is the first system that can conduct both end-to-end and hop-by-hop non-cooperative measurement to detect LFA.
- We develop *LinkScope*, a new and practical LFA detection system based on our framework, after addressing a number of challenging issues, and conduct extensive experiment to evaluate it. The results in a testbed and the Internet show that introducing limited bandwidth and computational overhead *LinkScope* can achieve timely detection and diagnosis of LFA with high detection rate and low false positive rate.

Roadmap. We give an overview of the new framework in Section II, and detail the design and the implementation of *LinkScope* in Section III and Section IV, respectively. Section V and Section VI present the evaluation results of *LinkScope* obtained in a testbed and the Internet, individually. We review the related work in Section VII and discuss the limitations in Section VIII. Finally, we conclude the work in Section IX.

II. Framework Overview

Our framework consist of four major modules. First, the topology collection and analysis module identifies potential target links, because LFA aims at important persistent links to cut off the Internet connections of the target area. Moreover, it will enumerate a set of end-to-end paths that cover these links, which will be used by the other modules for detecting LFA and localizing the links under attack. Depending on the criteria of important links, this module may list different links and paths. *LinkScope* adopts the topology analysis mechanism in [26], which is described in Section III-A.

Second, the non-cooperative path measurement module performs scheduled end-to-end and hop-by-hop measurements on the paths identified by the first module. We design new non-cooperative measurement methods, which are presented in Section III-B, because existing measurement approaches do not fulfill the requirements. More precisely, the majority of network performance measurement approaches (e.g., iperf [23]) require installing measurement tools on both ends of each path, thus limiting their scalability [25], [35], [36]. Although some tools (e.g., Ping, Pathneck [37]) support non-cooperative performance measurement by sending ICMP/UDP packets, their results may be biased because firewall/IDS/IPS is likely to drop such packets [38]. For example, when we run both *LinkScope* and Pathneck [37] to conduct hop-by-hop measurements from the campus network to randomly selected web servers, the load packets of Pathneck are dropped by the firewall whereas *LinkScope* can still obtain the right results because *LinkScope* conducts the measurements in an established TCP connection. We extend existing non-cooperative measurement patterns to create new measurement methods suitable for capturing the performance anomalies caused by LFA in practice, and combine them together for better performance. Moreover, we propose a new algorithm for scheduling the non-cooperative measurement. It is worth noting that existing scheduling algorithms are designed for cooperative measurement [39]–[42], and they cannot be directly used to schedule the non-cooperative measurement because the remote servers are not under our control. Our schedule algorithm aims to achieve two goals: each target link is probed "often enough" and the probing traffic is "low enough", as suggested in [39].

Third, the attack detection module first turns the raw measurement results into feature vectors, and then uses anomaly detection algorithms to capture the abnormal performance degradation caused by LFA (Section III-C). Since there are lots of anomaly detection methods, we compare six popular ones (i.e., Holter-Winter [43], Euclidean Distance [44], Mahalanobis Distance [45], Heterogeneous Distance [46], Polynomial Regression [47] and One-Class SVM [48]), and select Holter-winter for our detection algorithm because it has the best performance. It is worth noting that the detection algorithm could be replaced by others whenever necessary.

Fourth, the target link localization module conducts further active measurement to locate the links under attack (Section III-D). The majority of existing algorithms for locating link failures or infrastructure outages cannot infer the links under LFA attack because they rely on BGP update information [49] whereas LFA avoid triggering route changes.

Procedure Fig. 1 shows the major steps in the procedure of our framework. The first step uses the topology collection and analysis module to identify potential target links and a set of end-to-end paths covering those target links. Second, depending on the available resources, the non-cooperative path

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2815555, IEEE Transactions on Information Forensics and Security

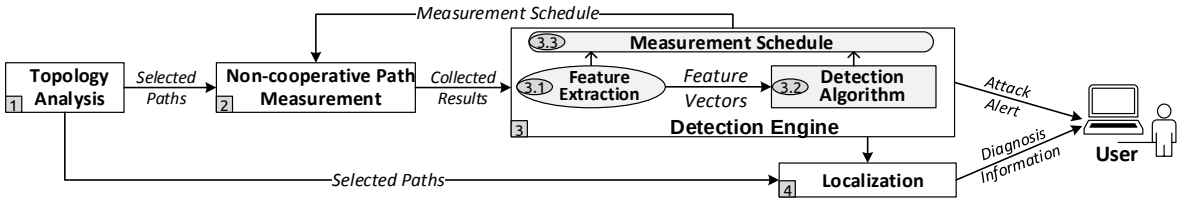
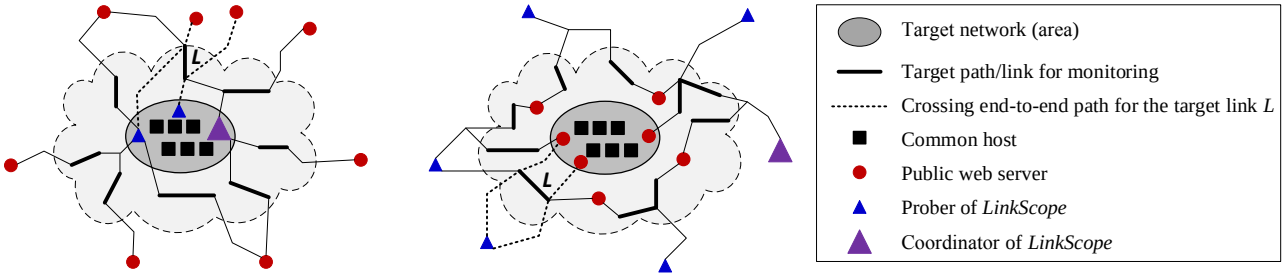


Fig. 1. The procedure of our framework.



(a) Self-initiated deployment (b) Cloud-based deployment

Fig. 2. The deployment strategies of our framework.

measurement module conduct measurements on the selected paths according to the schedule derived from the historic measurement results. In the third and the fourth steps, the attack detection module constructs feature vectors from the raw measurement results and leverages the anomaly detection algorithm to determine the existence of LFA. If an attack is detected, the target link localization module will be activated to infer the links under attack by further measuring a set of selected paths crossing the abnormal path and synthesizing the measurement results.

Deployment Our framework has two deployment strategies. Fig. 2(a) shows the first one, named self-initiated deployment, where the probers are deployed on hosts within the target network. By selecting web servers in different autonomous systems (AS), a prober can measure many diverse Internet paths for covering all possible target links. The second scenario, as illustrated in Fig. 2(b), is the cloud-based deployment, where probers are deployed on a group of hosts (e.g., virtual machines (VMs) in different data centers) outside the target network and measure the paths between themselves and hosts close to the target network.

In both strategies, the coordinator schedules the measurements on multiple paths and collects all the results, because the detection is based on the performance of all paths being measured. By running probers in diverse networks and/or selecting web servers in various ASes/areas, the paths being measured may cover all possible target links.

III. Design of *LinkScope*

To demonstrate the feasibility of the new framework, we design and develop *LinkScope*, a new and practical LFA detection system, and describe its major modules as follows.

A. Topology Collection and Analysis Module

LinkScope first selects paths to be monitored by following LFA's strategy, because lots of links can be flooded by LFA and the selected paths should cover the target links. In fact, LFA usually chooses the bottleneck links [5], [50] as the target links in order to maximize the damages to the target area [26].

We conduct network topology collection to get the routes of the target area, and then analyze the collected topology to identify the target links. More precisely, given a target network, we first analyze the network topology between it and its upstream ASes by performing paris-traceroute [51] from a group of hosts (e.g., VMs in clouds or looking glasses [52]) to web servers close to or within the target network, or using systems like Rocketfuel [53] and topological data from CAIDA [54]. From the topology, we can identify target links following LFA's strategy that selects persistent links with high link occurrence [26]. The link occurrence of a link is defined as the number of Internet paths crossing that link between hosts and public servers in the target network.

Given a set of potential target links denoted as $L = \{l_1, l_2, ..., l_M\}$, we select a set of paths for measurement, which is indicated by $P = \{p_1, p_2, ..., p_N\}$. Since there may be more than one path traversing certain target links, we define three rules to guide the path selection:

- For the ease of locating target links, paths that contain one target link will be selected.
- The number of paths sharing the same remote host should be minimized to avoid potential mutual interference [55]. It is desirable that each path has a different remote host.
- The number of paths initialized by one prober should be minimized to avoid self-induced congestion.

During LFA detection, we monitor the target links through running *LinkScope* on the sources of these paths to probe the destinations of these paths.

B. Non-cooperative Path Measurement Module

LinkScope detects LFA by conducting non-cooperation path measurement. Since LFA congests the target links, it leads to anomalies in path performance metrics, including:

- Packet loss rate, which increases when the link is clogged;
- Round-trip time (RTT), which increases because of the full queue in routers under attack;
- Jitter, which may have significant variations when bots send intermittent bursts of packets to congest the link [56], thus leading to variations in the queue length;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2815555, IEEE Transactions on Information Forensics and Security

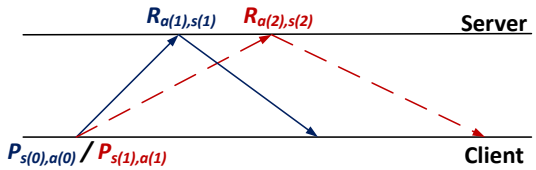


Fig. 3. Round trip probing (RTP) pattern.

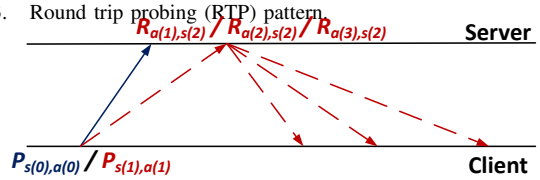


Fig. 4. Extended two way probing (eTWP) pattern with $w = 3$.

- Number of loss pairs [57], which may increase as a pair of probing packets may often see full queues due to LFA;
- Available bandwidth, which decreases because the target link is congested;
- Packet reordering, which may increase if the router under attack transmits packets through different routes;
- Connection failure rate, which increases if the target area has been isolated from the Internet due to severe LFA.

Besides measuring the above metrics, *LinkScope* should also support the following features:

- Conduct the measurements within an established TCP connection to avoid the biases or noises due to the network equipment that processes TCP/UDP packets in a different manner and/or discards all but TCP packets belonging to the established TCP connections.
- Perform both end-to-end and hop-by-hop measurements, for quickly detecting the anomalies caused by LFA and localizing the target links/areas, respectively.
- Measure one-way path metrics because of the prevalence of asymmetric routing.

To fulfill these requirements, *LinkScope* contains the following three probing patterns:

1) Round Trip Probing (RTP): We proposed the Round Trip Probing (RTP) pattern to measure RTT, one-way packet loss, and one-way packet reordering in [35]. As shown in Fig. 3, each RTP measurement involves sending two back-to-back probing packets (i.e., $P_{s(0),a(0)}$ and $P_{s(1),a(1)}$) with customized TCP sequence number (i.e., s(0) and s(1)) and acknowledgement number (i.e., a(0) and a(1)) to the remote host. The advertising window of each probing packet is set to 2 maximal segment size (MSS), allowing the server to return two response packets (i.e., $R_{a(1),s(1)}$ and $R_{a(2),s(2)}$) immediately upon receiving each probing packet. By analyzing the sequence numbers and the acknowledgement numbers in the response packets, we can decide whether there is packet loss/packet reordering occurred on the forward path or the reverse path. Servers that support TCP options like timestamp or SACK (Selective Acknowledgement) can ease the detection of packet loss on the forward path [35]. Moreover, RTT can be measured as the duration from sending $P_{s(0),a(0)}$ to receiving $R_{a(1),s(1)}$.

2) Extended Two Way Probing (eTWP): We proposed the original Two Way Probing (TWP) pattern for measuring one-way capacity in [58]. The extended Two Way Probing (eTWP) pattern has similar probing packets as that of TWP. The difference is that eTWP will induce more response packets from the remote host than TWP does. For example, the two response packets triggered by TWP can reflect four packet loss patterns, whereas, if eTWP triggers three response packets, eight packet loss patterns could be profiled. In addition, TWP just measures reverse capacity with the packet pair from server, and eTWP can measure the available bandwidth through triggering a packet train from the server. Hence, *LinkScope* can profile more loss patterns and characterize the changes of available bandwidth. As shown in Fig. 4, TWP (or eTWP) involves sending two back-to-back probing packets (i.e., $P_{s(0),a(0)}$ and $P_{s(1),a(1)}$). The first probing packet uses zero advertising window to prevent the server from sending back responses on the arrival of $P_{s(0),a(0)}$. In TWP, the advertising window in $P_{s(1),a(1)}$ is equal to 2×MSS so that it will trigger two packets from the server [58]. Since a packet train can characterize more loss patterns than a packet pair [59], we enlarge the advertising window in $P_{s(1),a(1)}$ from 2 to w ($w > 2$) in eTWP. Note that increasing w requires *LinkScope* to handle more patterns of response packets.

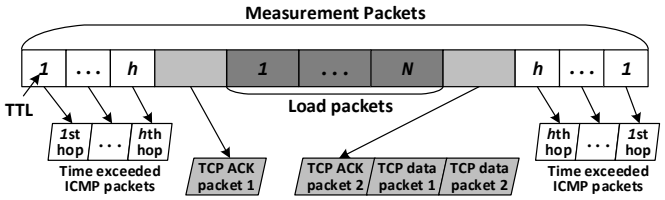


Fig. 5. Modified recursive packet train (RPT) pattern.

Since the server will dispatch w packets back-to-back if its congestion window allows, we calculate the time gap between the first and the w-th packet, denoted as G_r, and define θ_r to characterize the available bandwidth on the reverse path.

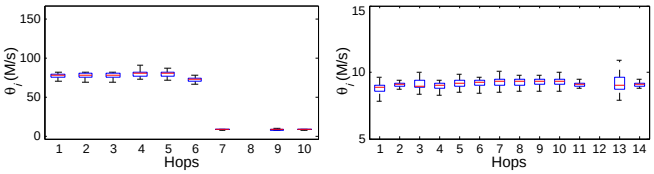
$$\theta_r = \frac{MSS \times (w-1)}{G_r} \tag{1}$$

Note that θ_r may not be equal to the real available bandwidth [60] but its reduction could indicate congestion [37].

3) Modified Recursive Packet Train (mRPT): Hu et al. proposed the original recursive packet train (RPT), which was employed in Pathneck for detecting the location of a network path's bottleneck [37]. The original RPT consists of a group of load packets, and a set of TTL-limited measurement packets and Pathneck uses UDP packets to construct RPT. We modify RPT to support end-to-end and hop-by-hop measurements in a TCP connection and remove redundant packets because network elements may handle UDP and TCP packets differently and the TCP packets that do not belong to a TCP connection may be dropped by firewall.

Fig. 5 illustrates the modified RPT, denoted as mRPT, where each rectangle is a probing packet, and each parallelogram indicates a response packet triggered by a probing packet. mRPT has h pairs of measurement packets, whose TTL values are equal to the number in those rectangles. Since a router will send back a time exceeded ICMP packet when a packet's TTL becomes zero, a pair of ICMP packets will be sent back after mRPT passes through a router. We use $G_{I(i)}$ to denote the time gap between the two ICMP packets from the i-th

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2815555, IEEE Transactions on Information Forensics and Security



(a) Path from Korea to Hong Kong (b) Path from Taiwan to Hong Kong

Fig. 6. θ_i measured on two paths to Hong Kong.

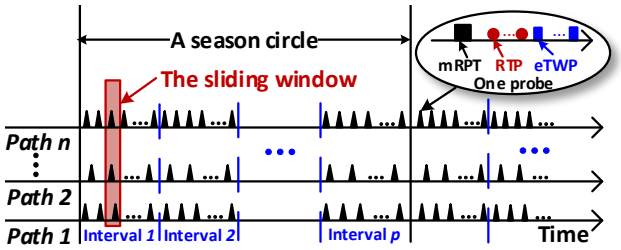


Fig. 7. The measurement schedule.

hop. *LinkScope* does not use a fixed number of measurement packets because we do not want them to reach the server and LFA usually targets on links outside the victim's network. Instead, *LinkScope* first determines h by doing a traceroute. If there are H hops form the probing client to the target server, we specify $h < H$. Thus, even the pair of measurement packets that have the maximum TTLs (i.e., h) will be dropped by the h-th hop before reaching the target server.

The load packets are customized TCP packets that belong to an established TCP connection and carry an invalid checksum value or a TCP sequence number so that they will be discarded by the server. Besides, we can let the load packets be dropped at specific hops by configuring the TTL values of the load packets. There are two special packets (i.e., R1 and R2, which are in the original RPT) between the load packets and the measurement packets. They have the same size as the load packets and work together to accomplish two tasks: (1) each packet triggers the server to send back a TCP ACK packet so that the prober can use the time gap between these two ACK packets, denoted as G_A, to estimate the interval between the head and tail load packets; (2) induce two TCP data packets from the server to start the measurement through RTP [35]. To achieve these goals, *LinkScope* prepares a long HTTP request whose length is equal to two load packets and puts half of it to R1 and the remaining part to R2. To force the server to immediately send back an ACK packet on the arrival of R1 and R2, we first send R2 and then R1, because a TCP server will send back an ACK packet immediately when it receives an out-of-order TCP segment or a segment that fills a gap in the sequence space [61].

To characterize the per-hop and the end-to-end available bandwidth, we define θ_i $(i = 1, \ldots, h)$ and θ_e as shown in Equation 2 and 3 respectively.

$$\theta_i = \frac{S_L \times (N+2) + S_M \times ((h-i) \times 2 + 1)}{G_{I(i)}} \tag{2}$$

$$\theta_e = \frac{S_L \times (N+1)}{G_A} \tag{3}$$

S_L and S_M are the sizes of load packets and measurement packets respectively, and N is the number of load packets.

Note that since the packet train cannot be controlled after each hop, θ_i (or θ_e) may not be a precise estimate of per-hop available bandwidth (or end-to-end available bandwidth), but their large decrement indicates serious congestion [37]. As LFA leads to severe congestion on target links, θ_i of the target link and θ_e of the path covering the target link will decrease.

Fig. 6 shows θ_i on two paths to a web server in our campus, whose last four hops are located in the campus network. Since the last but two hops do not send back ICMP packets, there is no θ_i on that hop. On the path from Korea to Hong Kong, θ_i drops from around 80Mbps to around 9Mbps on the 7th hop, because the bandwidth of each host in the campus network is limited to 10Mbps. On the path from Taiwan to Hong Kong, θ_i is always around 9Mbps. It may be due to the fact that the first hop's available bandwidth is around 9Mbps.

4) Measurement Schedule: Fig. 7 illustrates how *LinkScope* schedules the measurement. One probe consists of conducting mRPT measurement once and performing RTP and eTWP measurements m. In Fig. 7, we use the black triangle to represent one probe, and employ the black square, red cycle and blue rectangle (of the ellipse sub-figure) to denote a mRPT, RTP and eTWP measurement, respectively. The probing rate on the path i means the number of probes conducted per second on that path, which is represented by f_i.

Since the measurements introduce additional network traffic (evaluated in Section V-D), we propose an adaptive probing schedule to dynamically adjust the probing rate and the number of RTP/eTWP measurements in one probe according to the network performance variation for reducing the volume of measurement traffic. Let $f_{Max}/f_{Min}/f_{Cur}$ denote the maximum/minimum/current probing rate and $m_{Max}/m_{Min}/m_{Cur}$ represent the maximum/minimum/current number of RTP/eTWP measurements in one probe.

In the initial period, the probing is conducted with f_{Max} and m_{Max}. If the network performance measured for one day with f_{Cur} and m_{Cur} is stable, we adjust the probing schedule for the next day according to (4) and (5).

$$f_{Nxt} = \begin{cases} f_{Cur} \times \alpha & f_{Cur} \times \alpha > f_{Min} \\ f_{Cur} & f_{Cur} \times \alpha \leq f_{Min} \end{cases} \tag{4}$$

$$m_{Nxt} = \begin{cases} m_{Cur} \times \alpha & m_{Cur} \times \alpha > m_{Min} \\ m_{Cur} & m_{Cur} \times \alpha \leq m_{Min} \end{cases} \tag{5}$$

Here, α is less than 1 (0.75 by default).

If abnormal network performance is observed when $f_{Cur} < f_{Max}$ and $m_{Cur} < m_{Max}$ and the response packets can still be received, *LinkScope* restores to the initial configuration (i.e., $f_{Cur} = f_{Max}$ and $m_{Cur} = m_{Max}$) for measuring the target paths. If severe congestion happens on the paths and *LinkScope* cannot connect to the server, the RTP and eTWP measurement cannot get any results, because no response packets will be received. In this case, since the mRPT measurement can still receive the ICMP packets from the hops before the congested links, *LinkScope* will only conduct the mRPT measurement on the congested paths with f_{Max}.

C. Attack Detection Module

1) Feature Extraction: We define three metric vectors for storing the measurement results of three types of metrics, as listed in Table I, including the selected performance metrics

TABLE I
DETECTION METRICS AND THEIR DEFINITIONS.

Direction	Metric	Meaning of the metric
Forward path	θ_e	Characterizing available bandwidth through mRPT.
	L_{fr}	Packet loss rate from RTP.
	L_{ft}	Packet loss rate from eTWP.
	P_{fr}	Loss pair rate from RTP.
	P_{ft}	Loss pair rate from eTWP.
	R_{fr}	Packet reordering rate from RTP.
	R_{ft}	Packet reordering rate from eTWP.
Reverse path	θ_r	Characterizing available bandwidth through eTWP.
	L_{rr}	Packet loss rate from RTP.
	L_{rt}	Packet loss rate from eTWP.
	P_{rr}	Loss pair rate from RTP.
	P_{rt}	Loss pair rate from eTWP.
	R_{rr}	Packet reordering rate from RTP.
	R_{rt}	Packet reordering rate from eTWP.
Round-trip path	RTT	Round-trip time.
	J_b	Round-trip time variation (jitter).
	F_{rb}	Connection failure rate in RTP.
	F_{tb}	Connection failure rate in eTWP.

of the forward path, the reverse path and round-trip path, respectively. As a result, *LinkScope* can detect anomalies on different paths and locate the direction of the anomalies, thus easing the localization of the target link/area.

2) Detection Algorithm: Since the attack on a target link may result in the anomaly of multiple paths sharing the attacked link, *LinkScope* firstly identifies the performance anomalies on the individual path using the Holt-Winters forecasting algorithm [43], and then makes the detection decision by taking into account the anomalies of multiple paths.

Anomaly Detection on One Path. Since the network performance usually follows the diurnal pattern, *LinkScope* uses the Holt-Winters [43] forecasting algorithm to identify abnormal path performance by comparing the measurement results with the predicted values. This algorithm is practical for a reliable and robust forecast regarding trend and seasonal variations.

The detection algorithm has the following major steps.

Step 1: *Forecast*. The measurement results of each metric are organized as a time series, denoted as $\{X_1, X_2, \ldots, X_t\}$. We aim at predicting X_{t+k} at time t for k steps ahead, and denote the predicted value as $\hat{X}_t(k)$. Let α, β, γ indicate the smoothing parameters.

When a new observation X_t is available at time t, the Holt-winters algorithm updates local mean level L_t, local trend T_t and local seasonal index I_t as follows:

$$L_t = \alpha(X_t - I_{t-p}) + (1-\alpha)(L_{t-1} + T_{t-1}) \quad (6)$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \quad (7)$$

$$I_t = \gamma(X_t - L_t) + (1-\gamma)I_{t-p} \quad (8)$$

Then, the predicted value $\hat{X}_t(h)$ is:

$$\hat{X}_t(h) = L_t + hT_t + I_{t-p+1+(h-1) \bmod p} \quad (9)$$

for $h = 1, 2, \ldots, k$. The initialization values of level, trend and seasonal index are set to the average value in the first period (i.e., $L_0 = X_0$, $T_0 = X_1 - X_0$ and $I_0 = 0$). The smoothing parameters α, β, γ are estimated iteratively by minimizing the squared errors of expired estimations.

We define two periods for the detection, namely interval and seasonal circle, as shown in Fig. 7. Let p denotes the number of intervals within one seasonal circle. Since the measurement results usually have diurnal patterns (e.g., Fig. 19(c) and Fig. 20(c)), we set a seasonal circle as 1 day (i.e., 24 hours) and a time interval as 1 hour (i.e., $p = 24$) by default.

To identify performance anomalies, for each time interval in the seasonal cycle, we calculate the predicted confidence intervals as the deviation range of the predicted value, which are updated through exponential smoothing:

$$e_t = \gamma|X_t - \hat{X}_t| + (1-\gamma)e_{t-p}, \quad (10)$$

where e_t is the predicted deviation of time interval t, and the confidence interval of X_t is $(X_t - \delta_- * e_{t-p}, X_t + \delta_+ * e_{t-p})$. δ_- and δ_+ are the two scaling factors to control the width of confidence intervals.

Step 2: *Detection*. After each probing, $LinkScope$ compares the measured value of each metric with the corresponding forecasting confidence interval. The measured value is abnormal if it falls outside its corresponding confidence interval. Moreover, if more than μ metrics have abnormal values, the probing result is regarded as an anomaly. Eventually, if more than ν continuous probing results are abnormal, the path is detected as being attacked .

Attack Detected Based on Multiple Paths. Since LFA congests selected target links [26], end-to-end paths containing these links will have performance anomalies. *LinkScope* raises an alert if the number of abnormal paths containing the same target links at time t, which is denoted as $Pa(t)$, reaches the threshold ψ_a. Since the probing frequency may not be the same for all paths, we define a sliding window (as shown in Fig. 7) with the length $w = \max\{1/f_1, 1/f_2, ..., 1/f_n\}$, where n is the number of paths under monitoring, and then count the number of abnormal paths within each sliding window to detect LFA.

D. Target Link Localization Module

When LFA is detected on the forward path, *LinkScope* tries to locate the target link in two steps. First, we eliminate the links before the target link, and then we eliminate the links behind the target link. We use an example in Fig. 8 to illustrate the steps, where bots send traffic to the server to congest the link between H_i and H_{i+1}. First, based on the hop-by-hop measurement results from mRPT, *LinkScope* infers that the path from H_1 to H_{i-1} is not under attack. Second, according to the topology analysis, *LinkScope* will perform measurement on other paths that cover the hops after H_i, such as P_1 going through H_{i+1} and P_{k-1} covering H_{i+k}. If a new path (e.g., the one covering H_{i+j}) does not have poor performance like the original path, then the target link is in the area from H_i to H_{i+j-1}. The rationality behind this approach comes from the nature of LFA that congests a target link so that all paths crossing that link will suffer from similar performance degradation. In contrast, other paths will not have such patterns.

Since the paths identified in Section III-A may not cover all hops on the original path, we propose the following steps to look for new paths.

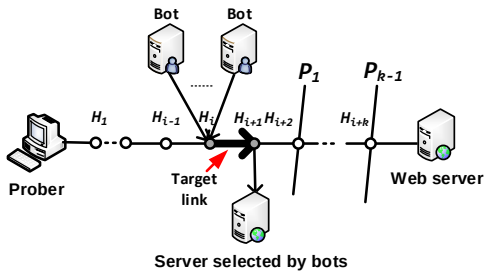


Fig. 8. Locating a target link.

1) For a hop, H_k, we utilize high-speed scanning tools such as Zmap [62] to look for web servers in the same subnet as H_k, which can be determined through systems like traceNET [63]. If a web server is found, *LinkScope* performs traceroute to this web server and checks whether the path to the server goes through H_k.
2) We look for web servers located in the same AS as H_k and then check whether the paths to those web servers go through H_k.
3) We look for web servers located in the buddy prefix [64] as H_k and then check whether the paths to those web servers go through H_k.
4) If no such path can be found, we check the next hop.

Note that the paths and routes selection is a continuous task instead of being taken only during the probing phase, and the system maintains a data set containing all information about the collected paths and routes of the target network. We acknowledge that this method may not be applied to reverse paths because it is difficult to get the traceroute on the reverse path (i.e., from the remote host to the prober). In future work, we will explore two possible approaches to tackle this issue. First, the victim may combine both self-initiated measurement and cloud-based measurement after anomalies are detected. Second, We will use reverse traceroute [65] or looking glass [66] to obtain the traceroute on the reverse path.

IV. Implementation of *LinkScope*

We next present the crucial technical issues in implementing *LinkScope* which is comprised by four modules and the architecture is shown in Fig. 9.

A. Control Module

The control module has two major components, namely, the topology analysis component for selecting paths to be monitored for a target region and the attack detection component for revealing the attacks. Note that only the coordinator has the control module.

The topology analysis component employs the strategy described in Section III-A to choose the paths to be measured, and the attack detection component decides whether an attack happens based on the algorithm introduced in Section III-C. Before probing, the control module first selects the paths to be monitored, then schedules the corresponding probers (including itself) to conduct the measurement with specified target servers on these paths. During probing, it collects the measurement results from all probed paths and decides whether attack alarm should be raised. Once an attack is detected, the control module tries to locate the links under attacks following the localization mechanism mentioned in Section III-D.

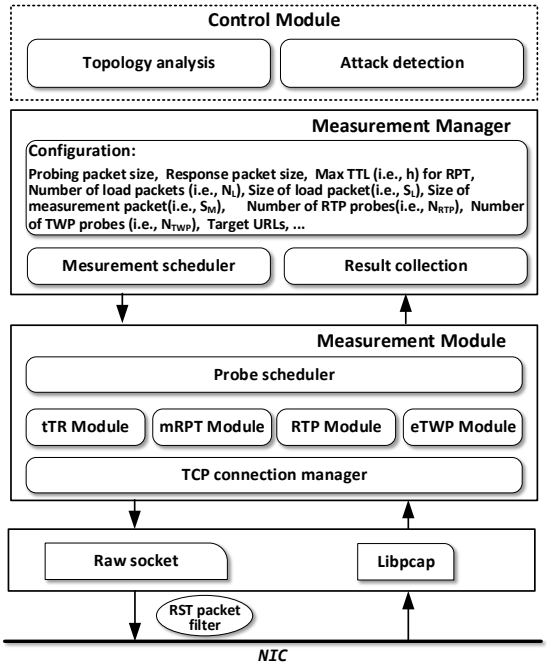


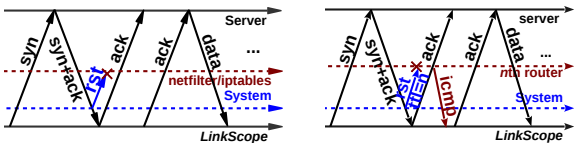
Fig. 9. The architecture of *LinkScope*.

B. Measurement Manager

The measurement manager is implemented to schedule the measurements including RTP, eTWP, and mRPT, and collect the measurement results. The designs of the measurements are not limited to specific application layer protocol. We use HTTP as the driving protocol because a tremendous number of web servers are publicly available for the measurement. In future work, we will explore other protocols.

We develop a tool named *WebChecker* to collect the information about paths and remote servers. It conducts traceroute to determine the number of hops between a prober and the server for setting h so that the measurement packets in mRPT can reach the network perimeter of the server. *WebChecker* also enumerates suitable web objects (i.e., size $\geq$ 10K bytes) in a web server to facilitate the measurement and outputs a set of URLs. It prefers static web objects (e.g., figure, pdf, etc.) starting from the front page of a website. Moreover, *WebChecker* checks whether the web server supports TCP options, including Timestamp, SACK, and HTTP header options such as Range [67]. These options could simplify the process of *LinkScope* and enhance its capability. For example, if the server supports MSS option, *LinkScope* can control the size of response packets. Supporting Timestamp and SACK can ease the detection of forward path packet loss [35].

The measurement scheduler manages a set of probing processes, each of which conducts the measurement on a path. To avoid self-induced congestion, the measurement scheduler will determine when the measurement for a certain path will be launched and how long a path will be measured. By default, the probing packet size, the response packet size, and the load packet size are set to 1500 bytes. For mRPT, the number of load packets is set 20 and the size of measurement packet is



(a) Use netfilter/iptables. (b) Modify TTL

Fig. 10. RST packet filter.

60 bytes. All these parameters are configurable. The collected measurement results will be sent to the attack detection component. Moreover, the measurement manager will dynamically adjust the probing schedule using the algorithm described in Section III-B4.

C. Measurement Module

In the measurement module, the probe scheduler manages the measurements on a path. After finishing one probe that consists of a mRPT measurement and m RTP and eTWP measurements (i.e., sending the probing packets and processing the response packets), the probe scheduler will deliver the parsed measurement results to the measurement manager and schedule a new probe.

The mRPT, RTP, and eTWP modules are in charge of preparing the probing packets and handling the response packets according to the corresponding patterns. Before conducting the measurement based on mRPT, *LinkScope* sets each measurement packet's IPID to its TTL. Since each pair of measurement packets will trigger two ICMP packets, *LinkScope* inspects the ICMP packet's payload, which contains the IP header and the first 8 bytes of the original packet's data, for matching it to the measurement packet.

It is worth noting that all the measurements in one probe are performed within *one* TCP connection. Such approach can mitigate the negative effect due to firewall and unstable routes, because stateful firewall will drop packets that do not belong to any established TCP connection and load balancer may employ the five-tuple of <src IP, src Port, dst IP, dst Port, Protocol> to select routes. The TCP connection manager will establish and maintain TCP connections. If the server supports TCP options like MSS, Timestamp, and SACK, the TCP connection manager will use MSS option to control the size of response packet (i.e., the server will use the minimal value between its MSS and the MSS announced by *LinkScope*). It will also put the SACK-permitted option and TCP timestamp option into the TCP SYN packet sent from *LinkScope* to the server.

Since *LinkScope* needs to control the establishment of TCP connections and customize probing packets (e.g., sequence number, acknowledgement number and advertising window), all packets are sent through the raw socket. Moreover, *LinkScope* uses the libpcap library [68] to capture all response packets and then parses them for computing performance metrics.

As the number of hops between a prober and the server may change over time, we develop a module named tTR to perform TCP traceroute regularly to monitor the routing changes for assuring that the measurement packets in mRPT can just reach the network perimeter of the server. Moreover, when the hops of a path change, tTR sends this event to the coordinator for re-scheduling the probing task if necessary.

D. RST Packet Filter

Since *LinkScope* constructs all packets by itself and sends them out through raw socket, OS does not know how to handle the response packets, thus it will send back an RST packet to the server to close the TCP connection. We propose two approaches to filter out RST packets generated by OS.

As shown in Fig. 10(a), if the system supports netfilter/iptables [69], we use it to drop all RST packets except those generated by *LinkScope*. We differentiate the RST packets from OS and that from *LinkScope* through the identification field of the IP header because *LinkScope* assigns a unique value to this field of its RST packets.

Since some hosts do not support netfilter/iptables, such as the Planetlab nodes [70], we propose a new method to tackle this issue as shown in Fig. 10(b). *LinkScope* first establishes a TCP connection with the remote server using stream socket (i.e., SOCK_STREAM), and then uses the function $setsockopt$ to set the TTL value in each packet generated by OS to a small value so that it will not reach the web server. Moreover, *LinkScope* utilizes the libpcap library to capture the TCP three-way handshaking packets generated by OS to record the initial sequence numbers selected by the local host and the web server along with other information related to the TCP connection, such as source port and TCP options. After that, *LinkScope* will create and send probing packets through raw socket with the help of such information.

V. Evaluation in the Testbed

We carry out extensive experiments in a controlled testbed to evaluate *LinkScope*'s functionality and overhead for answering the following questions.

- *RQ1*: Can *LinkScope* effectively measure the performance of network paths?
- *RQ2*: Can *LinkScope* accurately capture the changes in performance metrics due to various LFA attacks?
- *RQ3*: How much measurement traffic is introduced by *LinkScope*?
- *RQ4*: What is the additional overhead brought to the prober and server by *LinkScope*?

A. Testbed

Fig. 11 shows the topology of our testbed that connects to the Internet through the campus network. All hosts run Ubuntu 12.04. The attack traffic is generated by D-ITG [71] and transmitted from Host 2 and Host 6 to Host 1 and Host 5, respectively, to congest the router in red circle. The attacking packet sizes are uniformly distributed between 600 and 1400 bytes. Host 3 is a bridge for emulating packet loss and packet reordering, and Host 4 is an NAT-enable router providing port forwarding to connect the web server and the LAN to the Internet. In our experiment, LAN is the target network, and *LinkScope* is deployed on Prober1 and Prober2 following the cloud-based and the self-initiated deployment, respectively.

B. RQ1: Evaluating The Path Measurement Module

We first compare the measurement results of packet losses and packet reordering on both forward and reverse paths with

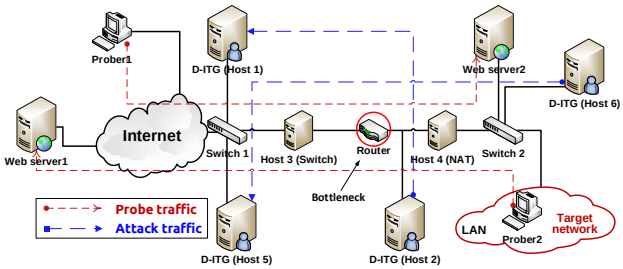
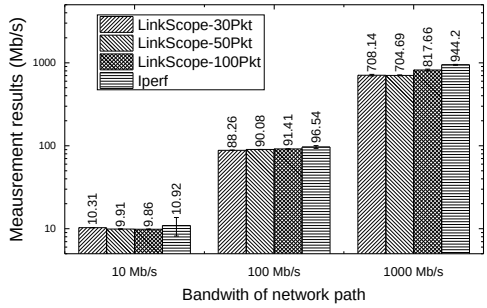
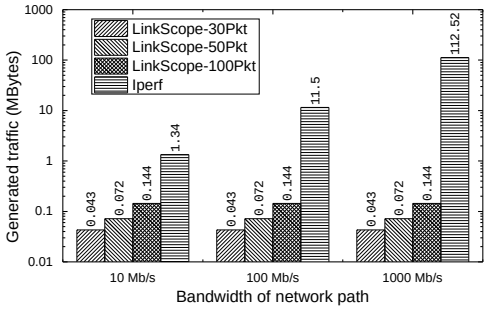


Fig. 11. The topology of the testbed.



(a) Measurement results



(b) Generated traffic

Fig. 12. Comparison of different measurement configuration.

the ground truth. More precisely, we capture and compare the packets at both probers and web server side to obtain the ground truth of packet loss and packet reordering. The results show that *LinkScope* can successfully recognize all packet losses and packet reordering cases.

Then, we compare the measured θ_e and θ_r with the results from iperf under three bandwidths (i.e., 10Mb/s, 100Mb/s, and 1000Mb/s), which we configured by the Router in Fig. 11. *LinkScope* uses 30, 50 and 100 payload packets of 1500 bytes, respectively, and iperf is scheduled to do measurement using TCP packets with interval of 1 second. We repeat each test 30 times. The measurement results with error bars are shown in Fig. 12(a), and the volume of measurement traffic is shown in Fig. 12(b). Fig. 12(a) illustrates that compared with the real available bandwidth, although *LinkScope* may underestimate it, its measurement result is close to that from iperf, and the result from *LinkScope* is more stable than that from iperf. It is worth noting that *LinkScope* uses much less measurement traffic than iperf does as shown in Fig. 12(b).

Summary: LinkScope can measure the performance of network paths accurately and effectively.

C. RQ2: Evaluating The Attack Detection Module

We first evaluate the ability of *LinkScope* on attack detection based on network performance changes. The experiment shows the measured performance metrics under different deployment strategies, and the second experiment evaluates whether *LinkScope* can identify different types of LFA attacks. Also, we evaluate the detection rate of *LinkScope*.

Attack Detection For the self-initiated deployment, as shown in Fig.13, before the volume of attack traffic reaches the capacity of the target link on around $t = 3000$s, the RTT is about 102ms and stable (Fig.13(c)) and θ_e decreases gradually (Fig.13(a)), whereas the RTT jitter increases obviously (Fig.13(c)). After the volume of attack traffic equals to and/or exceeds the capacity of the target link, congestion happens on the target link and the packets are queued so that the RTT values increase to around 120ms and the RTT jitter become more unstable as shown in Fig.13(c). Moreover, Fig.13(b) shows that severe packet losses are observed on the forward path. Consequently, since most packets in the mRPT packet train are lost, θ_e cannot be measured after $t = 3000$s as shown in Fig.13(a). Besides, from the experiment results, we can also find the measurement results are various with the changes of the attack traffic. For the cloud-based deployment, as shown in Fig.14, after the attack traffic is generated, θ_r becomes unstable (Fig.14(a)) and the RTT jitter increases obviously(Fig.14(c)). Similarly, after the volume of attack traffic equals to and/or exceeds the capacity of the target link, the RTT values increase to around 120ms and the RTT jitter become more unstable as shown in Fig.14(c). Moreover, Fig.14(b) shows severe packet losses on the reverse path.

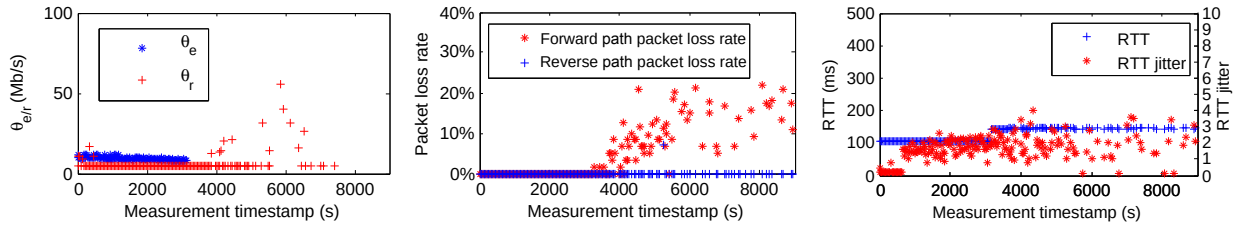
Attack Type Identification We emulate four types of LFA in the testbed and use the abnormal changes in θ_e to illustrate the effects due to different attacks. In this experiment, the target link's bandwidth is set to 100 Mb/s. Fig. 15(a) shows θ_e under pulsing LFA where the attacker transmits high-volume bursts of traffic to congest the bottleneck [56]. Without attack, θ_e is close to the available bandwidth. Under the attack, since the bottleneck is severely congested and all connections are broken, θ_e becomes zero.

Fig. 15(b) illustrates θ_e under LFA with two attack traffic rates: 80Mb/s and 40Mb/s. An attacker may change the attack traffic rate for evasion. We can see that when the attack rate decreases (or increases), θ_e increases (or decreases), meaning that *LinkScope* can capture the changes in available bandwidth. Fig. 15(c) represents θ_e under gradual LFA where the attack traffic rate increases from zero to a value equal to the capacity of the bottleneck. It emulates the attack scenario in the Internet where the traffic sent from different bots may not reach the bottleneck simultaneously, thus showing the gradual increase in the attack traffic rate. Although the TCP connection for measurement is broken when the attack traffic rate almost reached its maximal value, the decreasing trend of available bandwidth can be employed to raise an early alarm.

Since LFA will cause severe intermittent congestion on target links, we can use different patterns in performance metrics to distinguish it from other scenarios, such as long-term flooding and cable cut that will disable the Internet connection for a long period.

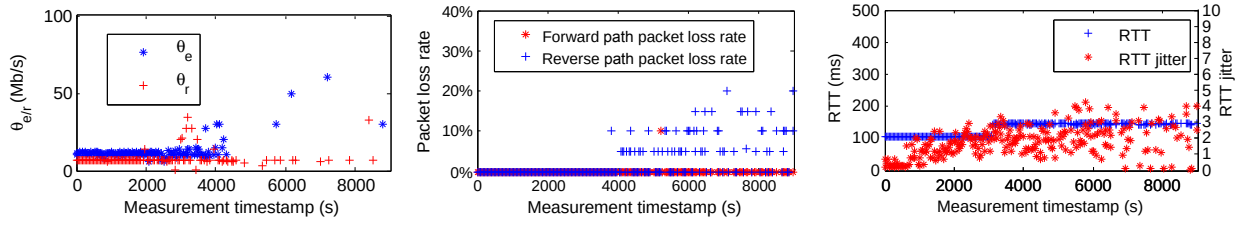
Detection Rate Evaluation To evaluate *LinkScope*'s detection rate, we follow the above setting to launch three types of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2815555, IEEE Transactions on Information Forensics and Security



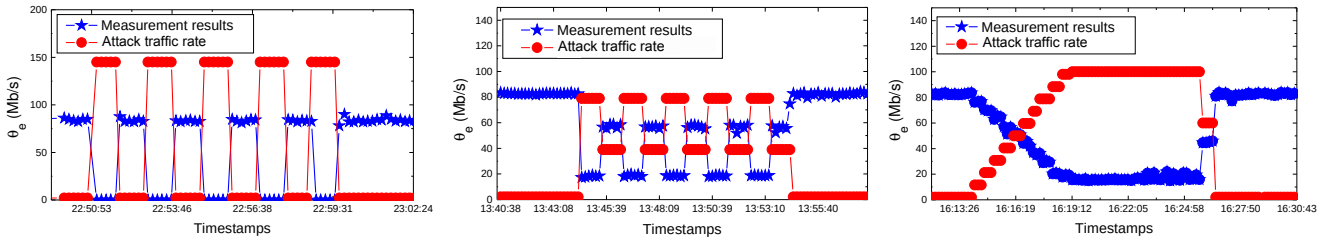
(a) $\theta_{e/r}$ (b) Packet loss. (c) RTT and RTT jitter.

Fig. 13. Metrics measured with self-initiated measurement (the emulated attack traffic increases from 0Mb/s at $t = 600$ and arrives at 10Mbs at $t = 3000$).



(a) $\theta_{e/r}$ (b) Packet loss. (c) RTT and RTT jitter.

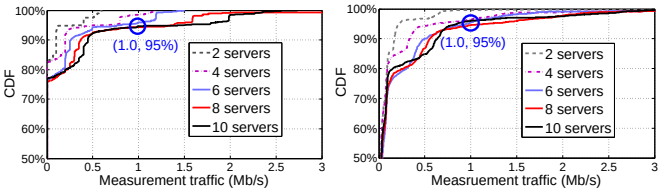
Fig. 14. Metrics measured with Cloud-based measurement (the emulated attack traffic increases from 0Mb/s at $t = 600$ and arrives at 10Mbs at $t = 3000$).



(a) Pulsing LFA. (b) Variational LFA. (c) Gradual LFA.

Fig. 15. Available bandwidth measured with different attacks with 100Mbit/s configuration.

attacks (i.e., Pulsing LFA, Variational LFA and Gradual LFA) on the bottleneck link because attacking a real Internet path would lead to ethical issues. In this experiment, to emulate the cloud-based measurement, we deploy *LinkScope* on Google VMs (i.e., prober1)and takes web server2 as the destination in Fig. 11. Besides, to emulate the self-initiated measurement, we run *LinkScope* in probe2 and specify web server2 as the destination. Note that both the link between prober1 and web server2 and the link between prober2 and web server 2 are probed through the Internet. In addition, to simulate the chaotic cross traffic in the testbed, we also generate TCP traffic between web server 2 and host5 with D-ITG. Precisely, to generate these traffic, we implement a python script that can generate five TCP flows at the same time. Each flow lasts for 3 seconds, and this script interactively generates flows with random mean packet size s between 500 bytes and 1500 bytes. In the same flow, the packet sizes follow normal distribution with mean size s and stand deviation 500 (i.e., "-n s 50"). Also, the flows have burst inter-departure time between packets, and the ON and OFF period durations are random variables, which we specify with parameters "-B N p 100 W 1000 100", namely, the former is a normal distribution with average p and stand deviation 100, whereas the latter is a Weibull distribution with shape 1000 and scale 1000. In this experiment, we also change the bandwidth of the bottleneck link to simulate the network with different bandwidth (i.e., 10Mb/s, 100Mb/s and 1000Mb/s), and p is 100, 1000 and 10000 for these bandwidths respectively. Then, we observe that the detection rates are always 100%. The false positive detection rate is evaluated on the Internet in Section VI-C.



(a) Tx traffic. (b) Rx traffic.

Fig. 16. The CDF of the volume of measurement traffic with different number of destinations.

Summary: *LinkScope* can accurately capture the performance anomalies due to various LFA attacks, and *LinkScope* can effectively and timely detect LFA attacks.

D. RQ3: Quantifying The Network Overhead

To quantify the network overhead introduced by *LinkScope*, we run multiple *LinkScope* on a prober for measuring the paths to D ($D = \{2, 4, 6, 8, 10\}$) destinations simultaneously. We obtain the Tx (Transmit) and Rx (Receive) bytes from */proc/net/dev* once per second for computing the volume of Tx/Rx traffic. Each experiment lasts for 2400 seconds. Fig. 16 depicts the cumulative distribution of the Tx/Rx measurement traffic. We can see that for 95% time both Tx and Rx measurement traffic are less than 1Mb/s even when 10 paths are probed at the same time.

Summary: *LinkScope* generates only a small amount of measurement traffic and provides configurable settings.

TABLE II
THE CPU UTILIZATIONS AND AVERAGE LOAD OF THE PROBER AND THE WEB SERVER DURING MEASUREMENT.

No. of Probing Processes	Probing rate (Hz)	Prober		Web sever	
		Load	CPU	Load	CPU
0	0	0.01	0.3%	0.00	0.5%
1	2	0.06	0.3%	0.00	0.5%
1	10	0.10	0.3%	0.01	0.6%
2	10	0.10	0.4%	0.01	0.6%
10	10	0.11	1.7%	0.02	0.7%
50	10	0.23	2.4%	0.08	0.8%
100	10	0.47	2.7%	0.09	0.8%

TABLE III
THE PERCENTAGE OF LOCALIZATION LINKS THAT CAN BE COVERED BY PROBING PATHS.

Target	1	2	3	4	All
Switzerland	92%	94%	89%	95%	97%
Hong Kong	89%	93%	86%	93%	91%
Japan	86%	93%	86%	92%	89%
Singapore	83%	92%	87%	92%	89%
Taiwan	84%	84%	86%	92%	86%

E. RQ4: Quantifying The System Overhead

To evaluate the system overhead introduced by *LinkScope*, we use htop [72] to measure the client's and web server's average load and CPU utilization when *LinkScope* runs with different configurations. The client is equipped with Intel 3.4 GHz i7-4770 CPU, 16G memory, and 1Gbps NIC, and the web server is equipped with Intel 2.83 GHz Core(TM)2 Quad CPU and runs Apache2. Table II lists the results for both the client and the server. The first line represents the load and CPU utilization without *LinkScope*, and we ensure that no other routine processes are executed on both machines during the measurement. We can see that even when there are 100 probing processes with 10Hz probing rates, the average loads and average CPU utilizations are still very low on both machines, especially for the web server.

Summary: *LinkScope* brings negligible overhead to both the probers and target web servers.

VI. EVALUATION ON THE INTERNET

To gain insight into the real-world performance, we also conduct extensive experiments to evaluate *LinkScope* on the Internet by answering the following questions.

RQ1: Can *LinkScope* effectively identify the target links and monitor them?

RQ2: Can *LinkScope* effectively measure the performance of network paths?

RQ3: Can *LinkScope* accurately detect LFA attacks?

A. RQ1: Network Topology Analysis

Target Link Identification In this experiment, we select the networks in Switzerland, Hong Kong, Japan, Singapore and Taiwan as the target networks for protection. We employ 6 Google VMs locating in different regions, 107 PlanetLab [70] nodes and 480 looking class routers, distributed in 267 cities of 33 countries, as probers, and randomly select around 1000 web servers in each target network as the remote hosts. Then we conduct traceroute on the probers to collect the routing information from the probers to the remote hosts. Eventually, we collect the routing information that includes 875861 paths and 56039 different links in total. Fig. 17 shows that the normalized link occurrences [26] with ranks in the routes to these five networks, and the results indicate that there are routing bottlenecks existing in all these five networks. As a result, we just need to probe the target links with high link occurrences to monitor the performances of the target networks. We also analyze the relationships between the probed links and the paths that pass these links, and the results are shown in Fig. 18. From the results, we can find the top 200 links carry around 80% paths and the top 400 links carry around 90% paths for all these five networks.

TABLE IV
FALSE POSITIVE RATES WITH DIFFERENT Pa FOR THE CLOUD-BASED DEPLOYMENT SCENARIO.

Destination Information	$Pa \geq 1$	$Pa \geq 2$	$Pa \geq 3$	$Pa \geq 4$
University site in US	0.34%	0.00%	0.00%	0.00%
University site 1 in Hong Kong	0.65%	0.00%	0.00%	0.00%
University site 2 in Hong Kong	4.43%	0.09%	0.00%	0.00%
Game site in Hong Kong	1.20%	0.03%	0.00%	0.00%
Government site in Hong Kong	3.84%	0.03%	0.00%	0.00%
Game site in Taiwan	0.71%	0.06%	0.03%	0.00%

Target Link Monitoring We evaluate whether *LinkScope* can monitor the target links effectively in this experiment. We choose the top 400 links as the target links when the links are ranked by normalized link-occurrence. Since *LinkScope* cannot run the looking glass servers, only the paths from Google VMs or planetlab nodes can be measured, namely we can only monitor the target links in the paths that can be measured. In this experiment, we find 93.6%, 95.4%, 94.8%, 91.6% and 94.6% of the target links can be monitored, when Switzerland, Hong Kong, Japan, Singapore and Taiwan are specified as the target network respectively.

Target Link Localization To evaluate the effectiveness of locating target link when LFA is launched, we select the top 30 links as the target links for each network. Since we locate the flooded link through measuring the links behind the target links, in this experiment, we respectively assume that the first link, first two links, first three links, first four links and all links behind the target link (i.e., localization links) are used to locate the target link as introduced in Section III-D. To locate the flooded target links, we need first find paths that traverse the localization links and do not traverse the target links. Also, since *LinkScope* cannot run on the looking glass servers, these paths should be from PlanetLab nodes or Google VMs to the web servers in the target network. Hence, in this evaluation, we choose the Google VMs and PlanetLab nodes as the clients, and the evaluation results are shown in Table III. We can find that, for each configuration, there are more than 80% localization links that can be probed. It means that, if we just use the randomly selected 1000 web servers in each target network as destinations, we can effectively choose more than 80% of the localization links to locate the target links when LFA is detected.

Summary: With network topology analysis, *LinkScope* can efficiently identify the target links for measurement, and the target link localization module can effectively locate the flooded links when attacks are detected.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2815555, IEEE Transactions on Information Forensics and Security

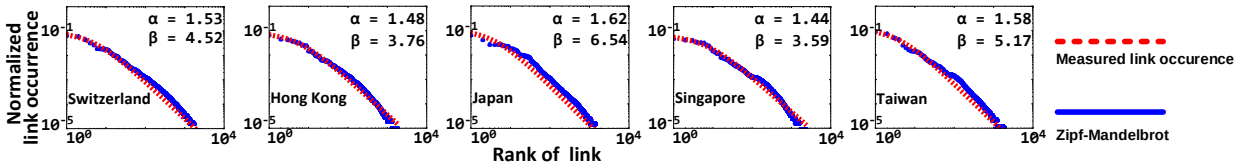


Fig. 17. Normalized link occurrence/rank in tracked routes to the target networks.

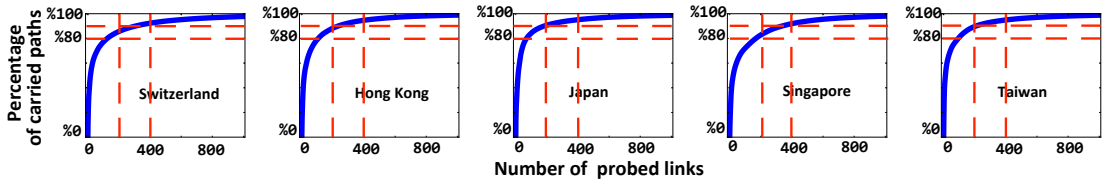
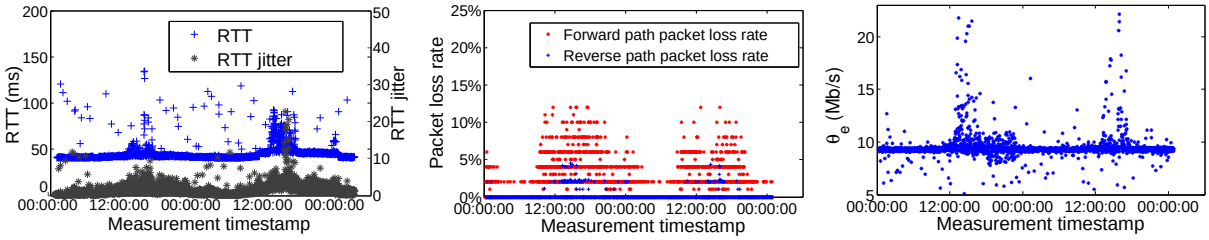
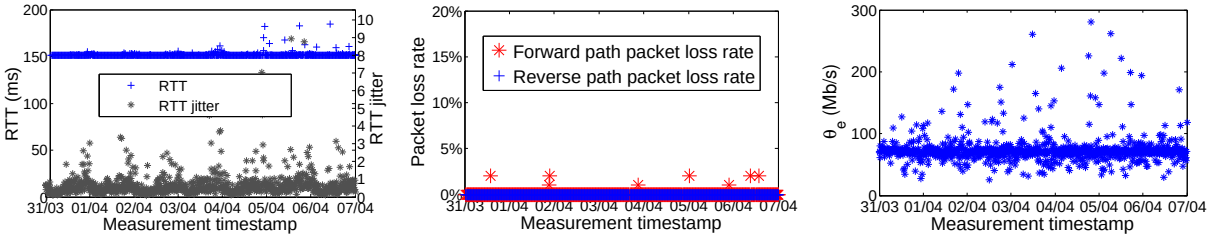


Fig. 18. The cumulative distribution function (CDF) between the percentage of the carried paths and the probed links.



(a) RTT and RTT jitter. (b) Packet loss rate. (c) θ_e.

Fig. 19. Performance metrics measured on the path from Amsterdam to Hong Kong for two days.



(a) RTT and RTT jitter. (b) Packet loss rate. (c) θ_e

Fig. 20. Performance metrics measured on the path from Santa Barbara to Taipei for seven days.

B. RQ2: Evaluating The Path Measurement Module

We run *LinkScope* on PlanetLab nodes in this experiment. Fig. 19 shows the performance metrics measured on the path from a prober in Amsterdam to a web server in Hong Kong for two days. It demonstrates the diurnal patterns in forward/reverse path packet loss, RTT, and jitter. The path performance is better and more stable in the period from 00:00 to 12:00 than that during the period from 12:00 to 24:00. The increased loss rate may affect the measurement of θ_e as some measurement results deviate during the period from 12:00 to 24:00 as shown in Fig. 19(c).

Fig. 20 demonstrates the performance metrics measured on the path from a prober in Santa Barbara (US) to a web server in Taipei for seven days. This path has stable performance. For example, RTT is around 150ms and the jitter is less than 10 as shown in Fig. 20(a). Moreover, the jitter is less during the period from 1:00 to 9:00 than the other time of each day. The loss rate is less than 2% and there is no packet reordering. The estimated θ_e is around 75Mbps as illustrated in Fig. 20(c). Since LFA will cause severe congestion during a short duration, it will result in obvious abrupt changes in the performance metrics and get caught by *LinkScope*.

Summary: *LinkScope* can measure the performance of real Internet paths effectively.

C. RQ3: Evaluating The Attack Detection Module

We first evaluate whether *LinkScope* can efficiently probe the web servers protected by the firewalls and IDSs. In this experiment, we run *LinkScope* to probe nine web sites protected by Radware (four sites), CloudFare (one site), DOSarrest (two sites) and Neustar (two sites), for one day, and we get stable and valid measurement results for all these 9 sites.

We evaluate *LinkScope*'s false positive rate using Internet measurement results on different paths, and then assess its detection rate. Since Google provides VMs in four different areas(i.e., US-east, US-central, Asia-east and Eur-west), we run *LinkScope* in Google VMs located in these four areas to probe six public web servers, which fall into three categories (university site, government site and game site) and are in three different areas (US, Taiwan and Hong Kong). Then, we collect the probing results of the 24 paths (4 Probers $\times$ 6 Servers) to evaluate *LinkScope*'s false positive rate. Here, we assume that there is no attack launched on these paths during the probing.

We schedule the four probers to perform measurement on each path once per 180s for 10 days. The measurement results of the first three days are used to forecast the confidence intervals and the results from 4th to 10th day are employed to compute the false positive rate. Table V lists the results with different μ and ν. Note that μ decides whether a probing result is abnormal according to the number of abnormal metrics,

TABLE V
FALSE POSITIVE RATE ON A SIGNAL PATH WITH DIFFERENT μ AND ν.

Destination Information	Client Location	$\mu=1$			$\mu=2$			$\mu=3$		
		$\nu=1$	$\nu=2$	$\nu=3$	$\nu=1$	$\nu=2$	$\nu=3$	$\nu=1$	$\nu=2$	$\nu=3$
University site in US	US-east	4.3%	2.1%	1.1%	0.7%	0.2%	0.1%	0.0%	0.0%	0.0%
	Asia-east	12.1%	5.8%	3.1%	0.4%	0.1%	0.0%	0.0%	0.0%	0.0%
	US-centeal	6.2%	2.4%	1.1%	0.2%	0.0%	0.0%	0.0%	0.0%	0.0%
	Eur-west	7.9%	2.9%	1.2%	0.4%	0.1%	0.0%	0.0%	0.0%	0.0%
University site 1 in Hong Kong	US-east	10.1%	6.0%	3.9%	1.5%	0.5%	0.2%	0.0%	0.0%	0.0%
	Asia-east	1.5%	0.2%	0.1%	0.4%	0.5%	0.0%	0.0%	0.0%	0.0%
	US-centeal	5.2%	1.9%	0.8%	0.3%	0.0%	0.0%	0.0%	0.0%	0.0%
	Eur-west	6.8%	2.9%	1.3%	0.2%	0.0%	0.0%	0.0%	0.0%	0.0%
University site 2 in Hong Kong	US-east	41.0%	31.1%	25.3%	4.9%	1.4%	0.4%	0.4%	0.0%	0.0%
	Asia-east	9.2%	3.1%	1.1%	2.8%	0.5%	0.1%	0.0%	0.0%	0.0%
	US-centeal	40.4%	27.1%	18.7%	7.5%	2.2%	0.6%	0.3%	0.1%	0.0%
	Eur-west	16.2%	8.4%	5.1%	1.7%	0.3%	0.0%	0.1%	0.0%	0.0%
Game site in Hong Kong	US-east	25.1%	24.5%	24.2%	2.2%	0.9%	0.4%	0.0%	0.0%	0.0%
	Asia-east	5.5%	5.3%	5.3%	1.1%	0.2%	0.0%	0.0%	0.0%	0.0%
	US-centeal	2.7%	2.0%	1.6%	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%
	Eur-west	15.7%	15.5%	15.4%	0.4%	0.3%	0.2%	0.0%	0.0%	0.0%
Government site in Hong Kong	US-east	15.8%	10.6%	7.7%	3.8%	1.7%	0.8%	0.1%	0.0%	0.0%
	Asia-east	18.2%	9.2%	5.3%	4.6%	1.2%	0.3%	0.4%	0.1%	0.0%
	US-centeal	18.0%	12.8%	9.8%	2.0%	1.0%	0.5%	0.0%	0.0%	0.0%
	Eur-west	6.8%	2.1%	0.6%	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%
Game site in Taiwan	US-east	22.2%	11.9%	7.1%	1.3%	0.4%	0.1%	0.1%	0.0%	0.0%
	Asia-east	13.8%	6.5%	3.4%	0.9%	0.2%	0.1%	0.0%	0.0%	0.0%
	US-centeal	16.4%	7.9%	3.7%	0.9%	0.1%	0.1%	0.1%	0.0%	0.0%
	Eur-west	11.9%	4.8%	2.2%	0.8%	0.2%	0.1%	0.0%	0.0%	0.0%

TABLE VI
FALSE POSITIVE RATES WITH DIFFERENT Pa FOR THE SELF-INITIATED DEPLOYMENT SCENARIO.

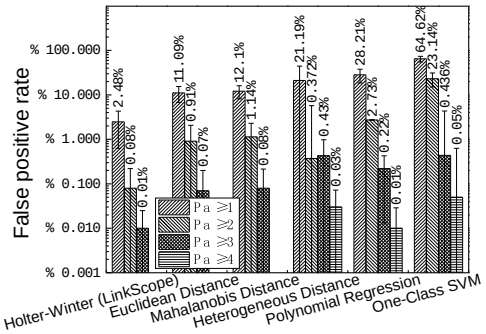
Client Location	$Pa \geq 1$	$Pa \geq 2$	$Pa \geq 3$	$Pa \geq 4$	$Pa \geq 5$	$Pa \geq 6$
VM in US-east	4.36%	0.29%	0.03%	0.00%	0.00%	0.00%
VM in Asia-east	2.09%	0.00%	0.00%	0.00%	0.00%	0.00%
VM in US-centeal	3.39%	0.03%	0.00%	0.00%	0.00%	0.00%
VM in Eur-east	0.08%	0.00%	0.00%	0.00%	0.00%	0.00%

and ν decides if an attack is launched based on the number of consecutive abnormal probing results. As a result, *LinkScope*'s false positive rate decreases with the increment of μ and ν. By default, $\mu = 2$ and $\nu = 2$. Table V shows that all false positive rates are lower than 2.5% with the default configuration.

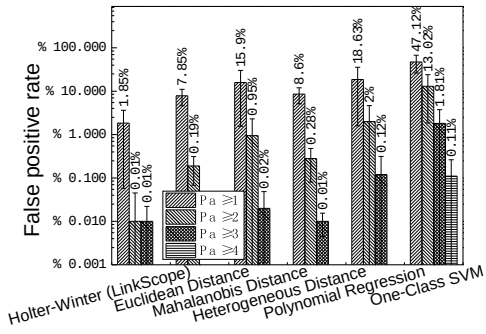
Since *LinkScope* detects LFA attacks based on the performances of multiple paths, we evaluate its false positive rates when more than one path share target links. We first evaluate the false positive rate for the cloud-based deployment scenario. In this case, there are 6 target areas, each of which is covered by four monitored paths from Google VMs. Table IV shows the false positive rates. When $Pa = 2$ (i.e., not less than 2 paths having abnormal performance at the same time), the false positive rates are less than 0.1% for all paths.

Then, we let the four networks containing Google VMs be the target areas, and evaluate the false positive rates for the self-initiated deployment scenario. In this case, each VM in the target area monitors 6 paths (i.e., from the VM to the 6 public web servers). Table VI shows the false positive rates. When Pa is 2, all false positive rates are less than 0.3%.

Comparison with Other Detection Algorithms In order to compare our detection approach with other popular detection systems and evaluate the effectiveness of the multiple-based detection module, we first compare the false positive detection rates of the detection modules implemented based on various detection algorithms. Since these detection systems are designed for different features and evaluated on various



(a) False positive rete of self-initiated deployment.



(b) False positive rete of cloud-based deployment.

Fig. 21. False positive detection rate.

data sets, we implement our detection module based on five algorithms (i.e., Euclidean Distance [44], [73], Mahalanobis Distance [45], [74], Heterogeneous Distance [46], [75], [76], Polynomial Regression [47], [77] and One-Class SVM [48], [78]) adopted by other detection systems for comparison, and the comparison results are shown in Fig. 21. From the results, we can the false positive detection rates of the detection module based on other algorithms are all higher than the false positive detection rate of *LinkScope*'s detection module. Besides, we can also find the multiple-path based detection

module can effectively reduce the false positive detection rate with both self-initiated (i.e., Fig. 21(a)) and cloud-based (i.e., Fig. 21(b)) deployments, For example, when attacks are detected based on the measurement results that there are more than three paths with abnormal performance (i.e., $Pa \geq 3$), the mean false positive detection rates are less than 2% for all detection algorithms.

Summary: *LinkScope* can effectively detect LFA, and the multiple-path based detection approach obviously reduces the false positive detection rate.

VII. Related work

LFA detection is essentially an anomaly detection problem [79], whose objective is to find abnormal performance degradation resulting from attacks [80]–[83] against critical links.

Traditional anomaly detection systems deployed at the network perimeter [8], although important security strategies, *cannot* handle LFA by design. The reason is that, to degrade the connectivity of the target network, LFA selects to overwhelm the upstream links of the target network, which are *invisible* to the systems deployed downstream. Additionally, an attacker can instruct bots to generate traffic flows, destined for networks near the target network, to converge at the target link and then congest it. In this way, these flows may never reach the traditional systems. Even if some flows reach the systems, the attacker could instruct bots to generate traffic flows indistinguishable from legitimate ones [5].

To cope with LFA, moving the countermeasures upstream then naturally arises. Specifically, due to the direct control over the traffic crossing critical links, router-based approaches have been proposed to defend against LFA and other smart DoS attacks [7], [9]–[16]. Despite their experimental effectiveness, their availability is limited in the foreseeable future for three reasons. First, additional deployment overhead is needed to activate router-based approaches. This is because router-based approaches require control over EGP (External Gateway Protocol, e.g., BGP)/IGP (Internal Gateway Protocol, e.g., OSPF) routers in autonomous systems (ASs), leading to the deployment of control servers in a number of ASs such as in [7], or extensive customized configuration and adaptation efforts for each router such as in [11] (e.g., configuring parameters and thresholds) and [16] (e.g., modifying the router to deceive the traceroute). Second, even if such overhead is ignorable, some router-level operations, although designed for security purposes, may incur unexpected risk (e.g., route flapping) due to network externalities [17]. Therefore, one needs to evaluate the effects of router configuration changes before deploying them on a live network. Current router-based approaches, however, have not considered this important issue. Third, router-based approaches rely on the collaboration between ASs to defend against LFA, whereas those (attack source) ASs where the attack traffic originates lack incentives to reroute the attack traffic to bypass the victim link (note that, in BGP, it is the source AS that decides the AS path towards a destination). This is because the collateral damage to each (attack source) AS posed by the attack traffic is negligible, while rerouting the attack traffic may break the commercial agreement between ASs and induce economic cost [18]. Although several promising SDN-based defense mechanisms [19]–[22] have been proposed, they may not be immediately deployed because they need to replace the traditional routers with SDN devices.

Therefore, a new solution aware of LFA and meanwhile immediately deployable without modifying/affecting existing Internet routing infrastructure is desirable. *LinkScope* is such a solution, constituting a practically feasible line of defense against LFA on today's Internet. It has two major features ensuring the effectiveness. First, instead of passively inspecting traffic for discovering anomalies, it conducts *active* measurement to cover as many paths as possible. The measurement is *non-cooperative* because the measurement agent is installed at one end without the cooperation of the other end, maximizing the network paths covered by each measurement agent and hence rendering our system scalable and flexibly deployable. Second, it provides a *novel framework* that captures the negative effect of LFA on a series of comprehensive performance metrics. Their comprehensiveness and joint effect facilitate accurate awareness of network path performance. Moreover, combining multiple network path anomaly to detect LFA empowers *LinkScope* to have rare false alarms.

Several widely deployed platforms on the Internet are also devoted to active measurement, offering a paradigm to deploy *LinkScope*. RIPE Atlas [84] measures Internet connectivity and reachability (e.g., whether a system is accessible from a different location and the corresponding RTT) using Ping, Traceroute, etc. Its primary feature lies in a network of probes with global coverage. Route Views [85] is a tool for obtaining real-time global backbone routing information based on data from BGP routers. Looking Glass [86] could achieve the same purpose, but provide only a constrained view of the routing system because of its dependence on probing servers that are unlikely to be deployed unlimitedly. Although these platforms are not dedicated to LFA detection, in the extreme case wherein LFA disconnects the target network, they can detect it due to the unreachability. However, the attacker, for the long-term payoff, designs LFA to be relatively stealthy, i.e., degrading (but never disconnecting) the link performance at utmost while avoiding triggering routing changes [5].

Our work differs from existing network tomography techniques, which cannot be applied to locate the target link because of impractical assumptions (e.g., multicast [87], source routing [88]). Binary tomography may be used for identifying faulty network links [89], but it just provides coarse information [90] and is not suitable for locating the link targeted by LFA, because it adopts assumptions for network fault (e.g., only one highly congested link in one path [91], faulty links nearest to the source [92]). LFA can easily invalidate them. Moreover, the probers in network tomography create a fully meshed measurement network [93], [94] whereas in our scenarios there is only one or a few probers that may not communicate with each other.

Although *LinkScope* is a detection system that *cannot* defend against LFA directly, it can help quickly identify the specific link under attack. Then, the identified victim link can be the input of traffic engineering methods such as proposed in [95] and [96] to mitigate the effect of LFA, or even pinpoint

the bots behind the attack by performing in-depth analysis of the traffic traversing the victim link [50].

VIII. Discussion

Although we cannot launch LFA against real links in the Internet due to ethical issues, our experiments on the testbed show that *LinkScope* can detect LFA and locate the target links.

Since *LinkScope* leverages non-cooperative measurement techniques, users only need to deploy it on the client side. This type of measurement techniques, if inappropriately used, may induce additional measurement traffic noise to the server side (e.g., open web servers on the Internet). However, due to the adjustable measurement rate, *LinkScope* can keep the measurement rate to a minimum to avoid affecting the normal operation of the servers. Moreover, it only introduces low overhead to remote servers as shown in Section V-E, under the premise of effective attack detection.

While different measurement approaches could be employed in our framework, users may need to adjust the detection methods, because not all measurement approaches support all metrics listed in Table I. For example, many approaches only support one or a few metrics (e.g., *POINTER* for packet reordering [97], *Sting* for packet loss [98], etc.). If several metrics are needed, users have to run several tools, thus wasting the bandwidth. Moreover, it would be good for users to first examine the metrics measured by a selected measurement approach and learn its pros and cons before adoption. For instance, compared with the Round Trip Probing (RTP) pattern, httping [99] is more susceptible to the load of remote server as shown in [35]. We also found that the RTT measurement of some HTTP-based tools largely deviates from that suggested in RFC2681 [100]. As another example, *pingb* cannot accurately measure asymmetric connections because it cannot distinguish between the time dispersion on the forward path and that on the reverse path [101]. We showed that the Two Way Probing (TWP) pattern can address this issue [58].

The attacker may evade our detection by frequently attacking the links with lower link occurrences. However, such evasion policy can further degrade the attack effect on the target area. To defend against such degraded attack, we can add more resources and monitor more paths.

IX. Conclusion

In this paper, we propose a novel framework and develop a new prototype of the framework, *LinkScope*, to detect a new class of target link-flooding attacks (LFA) and attempt to locate the target link or area whenever possible. By exploiting the nature of LFA that causes severe temporal congestion on links important to the target area, *LinkScope* employs both the end-to-end and hop-by-hop non-cooperative measurement techniques to capture the abnormal performance degradation due to LFA. Moreover, it correlates the measurement data and the traceroute data to infer the target links. After addressing a number of challenging issues, we have developed a prototype system based on the detection approach and conducted extensive evaluations in a testbed and the Internet. The results show that *LinkScope* can quickly detect LFA with high accuracy and low false positive rate. In future work, we will conduct large-scale and continuous measurements to evaluate *LinkScope* and investigate the optimal deployment of *LinkScope*.

References

[1] Incapsula, "2013-2014 DDoS threat landscape report," 2014.
[2] ARBOR, "DDoS and security reports," http://www.arbornetworks.com/asert/, 2014.
[3] M. Geva, A. Herzberg, and Y. Gev, "Bandwidth distributed denial of service: Attacks and defenses." *IEEE Security & Privacy*, vol. 12, 2014.
[4] A. Studer and A. Perrig, "The coremelt attack," in *Proc. ESORICS*, 2009.
[5] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *Proc. IEEE SP*, 2013.
[6] http://www.freebuf.com/articles/network/67107.html.
[7] S. Lee, M. Kang, and V. Gligor, "Codef: collaborative defense against large-scale link-flooding attacks," in *Proc. ACM CoNEXT*, 2013.
[8] M. Bhuyan, D. Bhattacharyya, and J. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, 2013.
[9] S. Lee and V. Gligor, "Floc: Dependable link access for legitimate traffic in flooding attacks," in *Proc. IEEE ICDCS*, 2010.
[10] A. Athreya, X. Wang, Y. Kim, Y. Tian, and P. Tague, "Resistance is not futile: Detecting ddos attacks without packet inspection," in *Proc. WISA*, 2013.
[11] A. Shevtekar and N. Ansari, "A router-based technique to mitigate reduction of quality (roq) attacks," *Computer Networks*, vol. 52, 2008.
[12] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin, "Flow level detection and filtering of low-rate DDoS," *Computer Networks*, vol. 56, 2012.
[13] C. Chang, S. Lee, B. Lin, and J. Wang, "The taming of the shrew: mitigating low-rate tcp-targeted attack," *IEEE Trans. On Network Service Management*, Mar. 2010.
[14] X. Luo and R. Chang, "Optimizing the pulsing denial-of-service attacks," in *Proc. IEEE DSN*, 2005.
[15] T. Hirayama, K. Toyoda, and I. Sasase, "Fast target link flooding attack detection scheme by analyzing traceroute packets flow," in *Proc. IEEE WIFS*, 2015.
[16] Q. Wang, F. Xiao, M. Zhou, Z. Wang, and H. Ding, "Mitigating link-flooding attacks with active link obfuscation," *Arxiv*, 2017.
[17] N. Feamster, J. Winick, and J. Rexford, "A model of bgp routing for network engineering," in *Proc. ACM SIGMETRICS*, 2004.
[18] A. Lodhi, A. Dhamdhere, and C. Dovrolis, "Open peering by internet transit providers: Peer preference or peer pressure?" in *Proc. IEEE INFOCOM*, 2014.
[19] P. Xiao, Z. Li, H. Qi, W. Qu, and H. Yu, "An efficient ddos detection with bloom filter in sdn," in *Proc. Trustcom/BigDataSE/ISPA*, 2016.
[20] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using sdn-based moving target defense," in *Proc. IEEE LCN*, 2016.
[21] L. Wang, Q. Li, Y. Jiang, and J. Wu, "Towards mitigating link flooding attack via incremental sdn deployment," in *Proc. ISCC*, 2016.
[22] M. S. Kang, V. D. Gligor, and V. Sekar, "Defending against evolving ddos attacks: A case study using link flooding incidents," in *Cambridge International Workshop on Security Protocols*, 2016.
[23] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The tcp/udp bandwidth measurement tool," *http://dast.nlanr.net/Projects*.
[24] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput," *ACM CCR*, vol. 32, no. 4, 2002.
[25] M. Crovella and B. Krishnamurthy, *Internet Measurement: Infrastructure, Traffic and Applications*. Wiley, 2006.
[26] M. S. Kang and V. D. Gligor, "Routing bottlenecks in the internet – causes, exploits, and countermeasures," in *Proc. ACM CCS*, 2014.
[27] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker, "On routing asymmetry in the internet," in *Proc. IEEE GLOBECOM*, 2005.
[28] http://www.cogentco.com/lookingglass.php, 2017.
[29] http://lg.eurorings.net/index.cgi, 2017.
[30] http://lg.level3.net/traceroute/lg_tr_output.php, 2017.
[31] http://lg.riss.ro/cgi-bin/lg.cgi, 2017.
[32] http://stats.uninett.no/cgi-bin/lg.cgi, 2017.
[33] http://lg.telia.net/, 2017.
[34] https://ssp.pme.gin.ntt.net/lg/lg.cgi, 2017.
[35] X. Luo, E. Chan, and R. Chang, "Design and implementation of TCP data probes for reliable and metric-rich network path monitoring," in *Proc. USENIX ATC*, 2009.
[36] X. Luo, L. Xue, C. Shi, Y. Shao, C. Qian, and E. Chan, "On measuring one-way path metrics from a web server," in *Proc. IEEE ICNP*, 2014.
[37] N. Hu, L. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks: Algorithms, measurements, and implications," in *Proc. ACM SIGCOMM*, 2004.

[38] S. J. Templeton and K. E. Levitt, "Detecting spoofed packets," in *Proc. DISCEX III*, 2003.

[39] P. Barford, N. Duffield, A. Ron, and J. Sommers, "Network performance anomaly detection and localization," in *Proc. IEEE INFOCOM*, 2009.

[40] A. D. Jaggard, S. Kopparty, V. Ramachandran, and R. N. Wright, "The design space of probing algorithms for network-performance measurement," in *Proc. ACM SIGMETRICS*, 2013.

[41] S. Gangam and S. Fahmy, "Mitigating interference in a network measurement service," in *Proc. IEEE iWQoS*, 2011.

[42] M. Fraiwan and G. Manimaran, "Scheduling algorithms for conducting conflict-free measurements in overlay networks," *Computer Networks*, vol. 52, no. 15, 2008.

[43] C. Chatfield, "The holt-winters forecasting procedure," *Journal of the Royal Statistical Society. Series C*, vol. 27, no. 3, 1978.

[44] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *ACM SIGCOMM CCR*, 2005.

[45] K. Rieck and P. Laskov, "Language models for detection of unknown attacks in network traffic," *Journal in Computer Virology*, 2007.

[46] S. Teng, H. Du, W. Zhang, X. Fu, and X. Li, "A cooperative network intrusion detection based on heterogeneous distance function clustering," in *Proc. IEEE CSCWD*, 2010.

[47] K. Futamura, A. Karasaridis, E. Noel, P. Reeser, A. Sridharan, C. Johnson, and P. Velardo, "vdns closed-loop control: A framework for an elastic control plane service," in *Proc. IEEE NFV-SDN*, 2015.

[48] E. G. da Silva, A. S. da Silva, J. A. Wickboldt, P. Smith, L. Z. Granville, and A. Schaeffer-Filho, "A one-class nids for sdn-based scada systems," in *Proc. IEEE COMPSAC*, 2016.

[49] V. Giotsas, C. Dietzel, G. Smaragdakis, A. Feldmann, A. Berger, and E. Aben, "Detecting peering infrastructure outages in the wild," in *Proc. ACM SIGCOMM*, 2017.

[50] M. S. Kang, V. D. Gligor, and V. Sekar, "Spiffy: Inducing cost-detectability tradeoffs for persistent link-flooding attacks," in *Proc. NDSS*, 2016.

[51] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with Paris traceroute," in *Proc. ACM IMC*, 2006.

[52] A. Khan, T. Kwon, H. Kim, and Y. Choi, "AS-level topology collection through looking glass servers," in *Proc. ACM IMC*, 2013.

[53] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proc. ACM SIGCOMM*, 2002.

[54] CAIDA, "Overview of datasets, monitors, and reports," https://www.caida.org/data/overview/.

[55] D. Croce, E. Leonardi, and M. Mellia, "Large-scale available bandwidth measurements: Interference in current techniques," *IEEE Transactions on Network and Service Management*, vol. 8, no. 4, 2011.

[56] X. Luo and R. Chang, "On a new class of pulsing Denial-of-Service attacks and the defense," in *Proc. NDSS*, 2005.

[57] E. Chan, X. Luo, W. Li, W. Fok, and R. K. Chang, "Measurement of loss pairs in network paths," in *Proc. ACM IMC*, 2010.

[58] E. Chan, A. Chen, X. Luo, R. Mok, W. Li, and R. Chang, "TRIO: Measuring asymmetric capacity with three minimum round-trip times," in *Proc. ACM CoNEXT*, 2011.

[59] R. Koodli and R. Ravikanth, "One-way loss pattern sample metrics," RFC 3357, Aug. 2002.

[60] J. Sommers, P. Barford, and W. Willinger, "Laboratory-based calibration of available bandwidth estimation tools," *Microprocess. Microsyst.*, vol. 31, no. 4, 2007.

[61] M. Allman, V. Paxson, and E. Blanton, "Rfc5681: Tcp congestion control," 2009.

[62] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *Proc. USENIX SEC*, 2013.

[63] M. Tozal and K. Sarac, "Tracenet: An internet topology data collector," in *Proc. ACM IMC*, 2010.

[64] J. Li, T. Ehrenkranz, and P. Elliott, "Buddyguard: A buddy system for fast and reliable detection of ip prefix anomalies," in *Proc. ICNP*, 2012.

[65] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. Wesep, A. Krishnamurthy, and T. Anderson, "Reverse traceroute," in *Proc. USENIX NSDI*, 2010.

[66] A. Khan, T. Kwon, H. Kim, and Y. Choi, "As-level topology collection through looking glass servers," in *Proc. ACM IMC*, 2013.

[67] Y. Lin, R. Hwang, and F. Baker, *Computer Networks: An Open Source Approach*. McGraw-Hill, 2011.

[68] "Libpcap: Packet capture library," http://www.tcpdump.org.

[69] Netfilter, http://www.netfilter.org.

[70] "Planetlab," https://www.planet-lab.org/, 2017.

[71] A. Dainotti, A. Botta, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, 2012.

[72] htop, http://hisham.hm/htop/.

[73] W. Bongiovanni, A. Guelfi, E. Pontes, A. Silva, F. Zhou, and S. Kofuji, "Viterbi algorithm for detecting ddos attacks," in *Proc. IEEE LCN*, 2015.

[74] D. Bayarjargal and G. Cho, "Detecting an anomalous traffic attack area based on entropy distribution and mahalanobis distance," *International Journal of Security and Its Applications*, 2014.

[75] M. Dharmadhikari and V. Kolhe, "Anomaly extraction using association rule with the heterogeneous detectors," in *Proc. IEEE ICICES*, 2014.

[76] I. Jingle and E. Rajsingh, "Colshield: an effective and collaborative protection shield for the detection and prevention of collaborative flooding of ddos attacks in wireless mesh networks," *Human-centric Computing and Information Sciences*, 2014.

[77] B. Gupta, P. Agrawal, A. Mishra, and M. Pattanshetti, "On estimating strength of a ddos attack using polynomial regression model," in *Proc. ICACC*, 2011.

[78] L. Teng, S. Teng, F. Tang, H. Zhu, W. Zhang, D. Liu, and L. Liang, "A collaborative and adaptive intrusion detection based on svms and decision trees," in *Proc. IEEE ICDMW*, 2014.

[79] M. Thottan and C. Ji, "Anomaly detection in ip networks," *IEEE Trans. on Signal Processing*, vol. 51, no. 8, 2003.

[80] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the dos and ddos problems," *ACM Computing Surveys (CSUR)*, vol. 39, no. 1, 2007.

[81] G. Loukas and G. Öke, "Protection against denial of service attacks: a survey," *The Computer Journal*, vol. 53, no. 7, 2010.

[82] S. 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.

[83] M. Bhuyan, H. Kashyap, D. Bhattacharyya, and J. Kalita, "Detecting distributed denial of service attacks: Methods, tools and future directions," *The Computer Journal*, Mar. 2013.

[84] "Ripe atlas," https://atlas.ripe.net/, 2017.

[85] "Route views," http://www.routeviews.org/, 2017.

[86] "Traceroute," http://www.traceroute.org/, 2017.

[87] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: recent developments," *Statistical Science*, vol. 19, 2004.

[88] L. Ma, T. He, K. Leung, A. Swami, and D. Towsley, "Identifiability of link metrics based on end-to-end path measurements," in *Proc. ACM IMC*, 2013.

[89] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "NetDiagnoser: troubleshooting network unreachabilities using end-to-end probes and routing data," in *Proc. ACM CoNEXT*, 2007.

[90] S. Zarifzadeh, M. Gowdagere, and C. Dovrolis, "Range tomography: combining the practicality of boolean tomography with the resolution of analog tomography," in *Proc. ACM IMC*, 2012.

[91] H. Nguyen and P. Thiran, "The boolean solution to the congested ip link location problem:theory and practice," in *Proc. IEEE INFOCOM*, 2007.

[92] N. Duffield, P. Avenue, and F. Park, "Network tomography of binary network performance characteristics," *IEEE Trans. Information Theory*, vol. 52, no. 12, 2006.

[93] Q. Zheng and G. Cao, "Minimizing probing cost and achieving identifiability in probe based network link monitoring," *IEEE Trans. Computers*, vol. 62, no. 3, 2013.

[94] Y. Zhao, Y. Chen, and D. Bindel, "Towards unbiased end-to-end network diagnosis," *IEEE/ACM Trans. on Networking*, vol. 17, no. 6, 2009.

[95] C. Liaskos, V. Kotronis, and X. Dimitropoulos, "A novel framework for modeling and mitigating distributed link flooding attacks," in *Proc. IEEE INFOCOM*, 2016.

[96] D. Gkounis, V. Kotronis, C. Liaskos, and X. Dimitropoulos, "On the interplay of link-flooding attacks and traffic engineering," *SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 2, 2016.

[97] X. Luo and R. Chang, "Novel approaches to end-to-end packet reordering measurement," in *Proc. ACM IMC*, 2005.

[98] S. Savage, "Sting: A TCP-based network measurement tool," in *Proc. USENIX USITS*, 1999.

[99] "httping," https://www.vanheusden.com/httping, 2017.

[100] L. Xue, X. Ma, X. Luo, L. Yu, S. Wang, and T. Chen, "Is what you measure what you expect? factors affecting smartphone-based mobile network measurement," in *Proc. IEEE INFOCOM*, 2017.

[101] "pingb: Bandwidth measuring ping," http://www.florian.ca/pingb, 2017.