On Modeling Link Flooding Attacks and Defenses

XIN WANG\textsuperscript{1,2}, XIAOBO MA\textsuperscript{1,2}, JIAHAO PENG\textsuperscript{1,2}, JIANFENG LI\textsuperscript{3}, LEI XUE\textsuperscript{3}, WENJUN HU\textsuperscript{4}, AND LI FENG\textsuperscript{5}

\textsuperscript{1}MOE Key Lab for Intelligent Networks and Network Security, Xi’an Jiaotong University, Xi’an, China
\textsuperscript{2}Faculty of Electronic and Information Engineering, Xi’an Jiaotong University, Xi’an, China
\textsuperscript{3}Department of Computing, Hong Kong Polytechnic University, Hong Kong, China
\textsuperscript{4}Palo Alto Networks, Santa Clara, CA, USA
\textsuperscript{5}Dependable and Secure Computing (CDSC) of Wuhan Digital Engineering Institute (WDEI), Wuhan, China.

Corresponding author: Xiaobo Ma (e-mail: xma.cs@xjtu.edu.cn).

This work was supported in part by National Natural Science Foundation (61972313), Postdoctoral Science Foundation (2019M663725, 2021T140543), the Fundamental Research Funds for the Central Universities, and CCF-NSFOCUS KunPeng Research Fund, of China. Xiaobo Ma is also an XJTU Tang Scholar supported by Cyrus Tang Foundation.

\textbf{ABSTRACT} The increasing popularity of Internet of Things (IoTs) is making people universally connected and thus bringing the ease of life. Because of their sheer volume, weak security, and continual operation, IoT devices, along with many computer servers, are widely compromised to launch powerful distributed denial-of-service (DDoS) attacks. The emerging link flooding attacks (LFAs) are one type of such attacks that attract significant attention in both academia and industry against the routing infrastructure. The attack traffic flows originating from bots (e.g., compromised IoT devices) are deliberately aggregated at upstream critical links and grow intensified, gradually making a network connected to the critical links disconnected. Although LFAs are far more sophisticated than traditional DDoS attacks, whether such sophistication comes without a downside has never been investigated. In this paper, by modeling link flooding attacks and defenses, we tackle a series of questions concerning the practical issues of LFAs. Specifically, from the perspective of attacks, we advance a novel notion of \textit{strike precision}, and reveal that LFAs may exhibit \textit{attack interference} (i.e., unexpectedly interfere the connectivity of innocent networks) which might undermine the stealthiness and persistence of LFAs. From the perspective of defenses, we make the first step to study \textit{attack intention}, i.e., inversely inferring the target network to disconnect based on the identified links under attack. Furthermore, we consider a strong defender who employs traffic engineering to mitigate LFAs, and formulate the game-theoretic interactions between attackers and defenders. Our formulation demonstrates that LFAs can be effectively mitigated based on traffic engineering from a game-theoretic perspective. We also study practical issues of non-cooperative defenses (e.g., light-weight probe deployment, multi-protocol-based measurement).

\textbf{INDEX TERMS} DDoS Attack, Link Flooding Attack, Routing Security

\section{I. INTRODUCTION}

The increasing popularity of Internet of Things (IoTs) is making people universally connected and thus bringing the ease of life. Because of their sheer volume, weak security, and continual operation, IoT devices, along with many computer servers, are widely compromised to build botnets and launch powerful distributed denial-of-service (DDoS) attacks. The emerging link flooding attacks (LFAs) are one type of such attacks that attract significant attention in both academia and industry against the routing infrastructure. In contrast to traditional (DDoS attacks that target (end) computer servers \cite{1,2,3}, LFAs target (intermediate) critical links that constitute the Internet backbone \cite{5,6,7}. LFAs have recently come into practice after attracting the academia for years, posing severe threats to large-scale regional networks. For example, a few links of major Internet exchange points in Europe and Asia were flooded to disconnect the anti-spam service Spamhaus, with up to 300Gb/s traffic of LFAs \cite{8}.

LFAs are powerful, formidable and hard to defend against for three major reasons. First, stemming from coordinated flows between bots and public servers (e.g., web servers) or among bots, the attack traffic flows are aggregated at a critical link with a huge amount. As such, a network connected to the critical link will be gradually disconnected, as...
the aggregated attack traffic flows grow intensified. Second, the disconnected network, though having traditional DDoS countermeasures deployed at the network perimeter, might be blind to the attack, since the critical link is not within its administrative domain. Last, even for those ISPs who have the privilege to access the traffic flows crossing the critical link, it would be challenging for them to identify the attack traffic flows that are indistinguishable from legitimate ones.

Since the pioneering works of Studer and Perrig [9] and Kang et al. [10], considerable progress has been made to thwart LFAs, with a focus on detecting and mitigating the attack [5], [9]–[16]. In all state-of-the-art studies, it is commonly believed that LFAs are far more sophisticated than traditional server-targeted DDoS attacks by design. Nevertheless, whether such sophistication comes without a downside has never been investigated.

In this paper, by modeling link flooding attacks and defenses, we tackle a series of questions concerning the practical issues of LFAs. Specifically, we advance the notion of strike precision of LFAs, and reveal that LFAs may exhibit attack interference which might restrict their applicability from the adversary’s standpoint. More precisely, when the adversary aims at disconnecting a specific network (e.g., ) via attacking a critical link , the connectivity of the networks surrounding (e.g., ) might be interfered (i.e., attack interference), in whole or in part, depending on the connectivity importance of for . Due to attack interference, strike precision of LFAs would be lowered in the sense that LFAs, while disconnecting a network, may unexpectedly interfere the connectivity of some innocent networks nearby.

At first glance, the adversary lacks interest in attack interference. For a terrorist adversary interested in mass destruction, it is true. However, for a rational adversary in consideration of attack stealthiness and persistence, it is not true. The reason is that attack interference tends to incentivize different victim networks to collaborate in defending against LFAs. Such collaboration is an effective countermeasure against LFAs, since it increases individual networks’ visibility of coordinated attack traffic flows. The severer attack interference is, the stronger the incentive is. Therefore, for a rational adversary, attack interference and strike precision would be big concerns, especially in the long run.

From the perspective of attacks, we attempt to answer the following research questions:

- **RQ1**: To what extent does attack interference exist, and how do they affect strike precision, in traditional LFAs that only seeks strike efficiency, i.e., interrupting more routes (destined to ) by flooding fewer selected links?
- **RQ2**: What are the main factors affecting attack interference and strike precision? Can adversaries reduce attack interference and increase strike precision?

From the perspective of defenses, we are interested in the following research questions:

- **RQ3**: Given the identified target links that the adversary attacks, how to infer the target network that the adversary would like to disconnect (i.e., attack intention)?
- **RQ4**: Consider a defender who employs traffic engineering to mitigate LFAs, how to formulate the interactions between attackers and defenders from a game-theoretic perspective?
- **RQ5**: When traffic engineering is used in defending against LFAs, how would the attack effect and strike precision would be? Can we effectively mitigate LFAs?

To answer RQ1, we first quantify strike precision, as well as strike efficiency. Then, we propose the Strike-Efficiency-Oriented (i.e., SEO) flooding strategy for traditional LFAs. The strategy interrupts more routes by flooding fewer selected links. Following the SEO flooding strategy, we perform traditional LFAs to gain insight into their strike precision. To tackle RQ2, taking into account the attack interference induced by attacking each link, we propose the Strike-Precision-Aware Hybrid (i.e., SPAH) flooding strategy, for impeding the attack toward links with more severe attack interference while achieving larger strike efficiency. To address RQ3, we construct a mapping from the target networks to the sets of target links to effectively infer the target networks from the set of target links under attack. Finally, we demystify RQ4 and RQ5 from a game-theoretic perspective. To answer these questions, all experiments are conducted using real-world traceroute data.

We have substantially extended our previous conference paper [17], which only focuses on RQ1 and RQ2, to include three more research questions, i.e., RQ3, RQ4, and RQ5. Additionally, we study practical issues of non-cooperative defenses (e.g., lightweight probe deployment, multi-protocol-based measurement) and perform more experiments for RQ1 and RQ2. To our best knowledge, we are the first to advance the notion of strike precision and attack intention, and study attack intention for LFAs. Moreover, we design a series of models to systematically study the questions under investigation. Our contributions are summarized below:

- We quantify strike precision, as well as strike efficiency. Our quantification sheds light on a new angle of understanding LFAs, potentially fostering more research concerning the practical aspects of LFAs.
- Combining the greedy algorithm and the genetic algorithm, we propose the Strike-Efficiency-Oriented (SEO) flooding strategy for LFAs, for interrupting more routes by flooding fewer selected links. Following the SEO flooding strategy, we perform LFAs to gain insight into their strike precision. The experiments suggest that attack interference is pervasive and significantly lowers strike precision of LFAs.
- Taking into account attack interference induced by attacking each link, we propose the SPAH flooding strategy, for impeding the attack toward links with more attack interference while achieving larger strike efficiency. The experiments show that our strategy can substantially increase strike precision.
- From the perspective of defenses, we make the first step to study attack intention, i.e., inversely inferring the
attack intention (i.e., the target network to disconnect) based on the identified attacked links. Furthermore, we consider a strong defender who employs traffic engineering to mitigate LFAs, and formulate the game-theoretic interactions between attackers and defenders. Our formulation demonstrates that LFAs can be effectively mitigated based on traffic engineering from a game-theoretic perspective.

Roadmap. In Sec. II we present the background. Then, we describe the problem in Sec. III model flooding attacks in Sec. IV and defenses in Sec. V and conduct experiments in Sec. VII. Finally, we discuss open questions in Sec. VIII survey the literature in Sec. IX and conclude in Sec. X.

II. BACKGROUND

A. LINK FLOODING ATTACK

As illustrated in FIGURE 1, the goal of LFAs is to congest the target link so to disconnect the intended victim network from the rest of the network, wherein the intended victim network could be a university, a city or even a country. To accomplish such a goal, the adversary analyzes the routing topology and finds a target link (or a set of target links) connecting the intended victim network and the rest of the network. By clogging the target link, the adversary can disconnect the intended victim network, rendering legitimate users unable to communicate with the intended victim network. A typical way that the adversary clogs the target link is to coordinate a number of traffic flows originating from bots and destined to decoy servers (or bots) inside or near the intended victim network. Decoy servers refer to public servers with open services such as HTTP, FTP. The coordinated flows are required to be aggregated at the target link enormously. They are often indistinguishable from legitimate flows for evading detection.

In recent years, with the weak security configuration of IoT devices, IoT devices have been easily infected by hackers to launch massive LFAs. Mirai botnet, for example, takes advantage of the vulnerability and pervasiveness of IoT devices to overwhelm targets with massive DDoS attacks and potential LFAs. By scanning public IP addresses, using a brute-force way to find the default credentials of IoT devices and controlling the IoT devices to execute LFA malwares, Mirai can launch massive LFAs to degrade the Internet connection of the targets. Home routers, network-enabled cameras, and digital video recorders are the most vulnerable devices to

![FIGURE 1: The basic idea of LFAs.](image)

Mirai since these devices have less protection. Traditional LFAs, initiated by PCs and servers, are limited by the up-to-date patching, power-off time. In contrast, IoT devices are always online, lacking update-to-date patching. Therefore, IoT devices are rendering LFAs more powerful.

B. LINK FLOODING ATTACK DEFENSE

Defenders, on the other hand, tackle LFAs to keep the network working with little or no Internet connectivity degradation. Traffic Engineering (TE) is a powerful tool for defenders to react to LFAs [18]. The goal of TE is to find appropriate paths for minimizing the transmission delay and maximizing the available bandwidth [19] between sources and destinations.

Since LFAs’ traffic flows are indistinguishable from legitimate ones, distributing and re-routing traffic flows (including legitimate ones and malicious ones) over available paths would be an effective method to tackle LFAs. As illustrated in FIGURE 2 when the target link is flooded, defenders can deploy parallel links to re-route traffic flows [20]. In FIGURE 3 finding new routes consisting of existing parallel links is also practical. TE methods such as load balancing, re-routing and finding new routes can be implemented in the IP networks based on OSPF-routing [21]. OSPF is a commonly used intra-domain link state routing protocol for IP networks within a single autonomous system (AS). In OSPF, the weight of links can be changed by the network operator to determine the path route adopted by different packets [21]. By adjusting the weights of the flooded links, defenders can mitigate LFAs congestion.

III. PROBLEM DESCRIPTION

By clogging the target link, the adversary can disconnect the intended victim network, while it is challenging for the intended victim network to defend against LFAs using countermeasures deployed at its network perimeter. However, it can be argued that, though more sophisticated, powerful and hard to defend against, LFAs were born to have limitations in comparison to traditional DDoS attacks. As depicted in
IV. MODELING LINK FLOODING ATTACKS

In this section, from the perspective of attacks, we first systematically model attack interference. Then, we propose different link flooding strategies in consideration of their attack interference.

A. ATTACK INTERFERENCE

Consider an adversary who performs LFAs. Let $\mathcal{N}$ denote a network that the adversary would like to disconnect from the rest of the network. Essentially disconnecting $\mathcal{N}$ is to select a set of target (intermediate) links along the routes between hosts within $\mathcal{N}$ and hosts outside of it, and then clog them so that the routes can be interrupted. Note that a route comprises a sequence of links.

We denote the set of routes between hosts within $\mathcal{N}$ and hosts outside of it by $\mathcal{R}_N$, the set of links (that constitute $\mathcal{R}_N$) by $\mathcal{L}_N$, and the set of selected target links by $\mathcal{L}'_N \subseteq \mathcal{L}_N$. Normally, it is hard, if not impossible, to interrupt all routes in $\mathcal{R}_N$ by clogging the links in $\mathcal{L}'_N$. Suppose, by clogging the links in $\mathcal{L}'_N$, the set of routes that the adversary interrupts is $\mathcal{R}'_N \subseteq \mathcal{R}_N$. Let $\lvert \cdot \rvert$ be the operator calculating the cardinality of a set. On a limited budget of $|\mathcal{L}'_N|$, the adversary can strategically derive $\mathcal{L}'_N$ for a larger value of $|\mathcal{R}'_N|/|\mathcal{R}_N|$, i.e., interrupting the routes in $\mathcal{R}_N$ as many as possible. We also refer $|\mathcal{R}'_N|/|\mathcal{R}_N|$ as Attack Effect in the following sections.

To quantify the adversary’s return on investment (ROI), we define strike efficiency, denoted by $SE$, to represent the number routes that the adversary can interrupt by clogging one link on average.

**Definition 3.1 Strike Efficiency.**

$$SE = \frac{|\mathcal{R}'_N|}{|\mathcal{L}'_N|}.$$  \hspace{1cm} (1)

When the adversary performs LFAs to disconnect $\mathcal{N}$, the connectivity of networks surrounding $\mathcal{N}$ may be interfered. Let $\mathcal{N}'$ denote the network(s) whose connectivity is interfered. $\mathcal{R}_{\mathcal{N}'}$ denote the set of routes between hosts within $\mathcal{N}'$ and hosts outside of it, and $\mathcal{R}'_{\mathcal{N}'} \subseteq \mathcal{R}_{\mathcal{N}'}$ denote the set of interrupted routes due to the adversary clogging the links in $\mathcal{L}'_{\mathcal{N}'}$. We define attack interference, denoted by $AI$, posed to $\mathcal{N}'$ due to disconnecting $\mathcal{N}$, as $|\mathcal{R}'_{\mathcal{N}'}|/|\mathcal{R}_{\mathcal{N}'}|$. To minimize $AI$, $|\mathcal{R}'_{\mathcal{N}'}|$ should be minimized, since $|\mathcal{R}_N|$ is constant in a specific problem.

On a limited budget of $|\mathcal{L}'_{\mathcal{N}'}|$ and a certain value of $SE$, $|\mathcal{R}'_{\mathcal{N}'}|$ is determined. Therefore, minimizing $AI$ is equivalent to maximizing $|\mathcal{R}'_{\mathcal{N}'}|/(|\mathcal{R}_N| + |\mathcal{R}'_{\mathcal{N}'}|)$, which we define as strike precision and denote by $SP$.

**Definition 3.2 Strike Precision.**

$$SP = \frac{|\mathcal{R}'_{\mathcal{N}'}|}{|\mathcal{R}_N| + |\mathcal{R}'_{\mathcal{N}'}|}.$$  \hspace{1cm} (2)

It refers to the ratio of the number of interrupted routes associated with $\mathcal{N}'$ to the total number of interrupted routes by clogging the links in $\mathcal{L}'_{\mathcal{N}'}$. 

FIGURE 4: The basic idea of LFAs.
We have three remarks on strike efficiency and strike precision. First, to evaluate LFAs comprehensively, both strike efficiency and strike precision should be measured, whereas existing studies neglect the latter. Second, for attack stealthiness and persistence, a rational adversary tends to seek larger strike efficiency while ensuring higher strike precision. Third, strategically deriving $L'_N$ is the core to achieve larger strike efficiency and higher strike precision.

Next, we propose different link flooding strategies in consideration of strike efficiency and strike precision. Determining link flooding strategies are essentially to strategically derive $L'_N$ by setting different priorities for strike efficiency and strike precision.

**B. LINK FLOODING ATTACK STRATEGIES**

When strategically deriving $L'_N$, traditional adversaries in LFAs seek high strike efficiency. We, accordingly, present two strike-efficiency-oriented (SEO) strategies, namely, the Naive Strike-Efficiency-Oriented (N-SEO) strategy that we recalculate $L'_N$ by setting different priorities for strike efficiency and strike precision.

1) Naive Strike-Efficiency-Oriented (N-SEO) Strategy

Traditional adversaries heuristically derive $L'_N$ (i.e., the set of selected target links to clog) greedily [10]. They calculate $|\mathcal{R}'_N|/|\mathcal{R}_N|$ to quantify the degradation severity of the attack against $\mathcal{N}$. The N-SEO strategy is detailed in Algorithm 1.

**Algorithm 1:** Naive strike-efficiency-oriented strategy.

```
Input: $\mathcal{R}_N, L'_N, \nabla, \Phi$
Output: $L'_N$
$L'_N = \emptyset$, $R'_N = \emptyset$
for $l \in L'_N$ do
    calculate $R'_N(l)$;
end for
while $|\mathcal{R}'_N|/|\mathcal{R}_N| < \nabla$ do
    $l = \arg\max_{l \in L'_N \setminus L'_N} \delta_l(L'_N)$;
    if $L'_N(l) = \Phi$ then
        break;
    end if
    $L'_N = L'_N \cup \{l\}$;
    $R'_N = R'_N \cup R'_N(l)$;
end for
```

In the N-SEO strategy, adversaries iteratively select a new link $l \in L'_N \setminus L'_N$ maximizing the reward gain, $\delta_l(L'_N) = |\mathcal{R}'_N(L'_N \cup \{l\})|/|\mathcal{R}_N| - |\mathcal{R}'_N(L'_N)|/|\mathcal{R}_N|$, and insert $l$ into $L'_N$. Here, $R'_N(\cdot)$ is a function calculating the set of interrupted routes to $\mathcal{N}$ due to clogging a set of links. Note that we recalculate $R'_N(l)$ for each $l$ after removing the link with maximum reward gain at each iteration. This process repeats until $|\mathcal{R}'_N|/|\mathcal{R}_N|$ approaches a pre-specified goal $\nabla$, or $|L'_N|$ reaches the budget of the number of selected links $\Phi$.

2) Advanced Strike-Efficiency-Oriented (A-SEO) Strategy

To further improve strike efficiency achievable by the N-SEO strategy, we devise the A-SEO strategy in Algorithm 2 to drive $L'_N$ using the genetic algorithm. Specifically, the A-SEO strategy initializes $n$ different genes of the first generation. A gene is a binary vector indicating and corresponding to a set of selected target links. We denote the function that transforms the former to the latter by $gene2set()$, while the latter to the former by $set2gene()$. All genes are constructed based on $L'_N$, differing from each other in the subset of selected target links (i.e., $L'_N$).

The evolution consists of selection, crossover, and mutation. Selection is for retaining better genes in each generation via roulette-wheel selection [22]. That is, probabilistically selecting a gene, say $g$, among $n$ different genes in each generation, and a gene with a larger value of strike efficiency, say $SE(g)$, should be assigned a larger selection probability. In the early generations, $SE(g)$ varies to a wide extent. Hence, assigning $g$ with a larger value of $SE(g)$ a larger selection probability is easy. However, as generations evolve, the value of $SE(g)$ tends to become closer to each other, resulting in similar selection probabilities between genes.

To signify the discrepancy of selection probabilities between genes, we assign $g$ in $G_j$ a selection probability of Stoffa-$SE_j(g)$ ($SE_j(g)$) using the Stoffa method [23]. $SE_j(g)$ is calculated as follows:

$$SE_j(g) = \frac{e^{SE(g)/T_j}}{\sum_{i=1}^{n} e^{SE(g)/T_j}},$$

$$T_j = 0.99^{-1} T_1,$$

where $j = 1, 2, 3, \ldots$ represents the sequence number of generations, and $T_1$ is the temperature of the 1st generation.

Crossover and mutation are leveraged to achieve cross-generation gene improvement. The self-adaptive genetic (SAG) algorithm is employed to adaptively adjust $P_c(g_k, g_q)$ (i.e., the probability of crossover between $g_k$ and $g_q$ acquired via roulette-wheel selection), $P_m(g)$ (i.e., the probability of mutation of $g$). Specifically, when the value of $SE_j(g)$ becomes homogeneous, $P_c(g_k, g_q)$ and $P_m(g)$ defined below should be increased and otherwise decreased [24, 25].

$$P_c(g_k, g_q) = \begin{cases} P_{c1} \cdot \frac{(SSE - SSE')}{SSE - SSE'}, & SSE' < SSE \text{ avg}, \\ P_{c1}, & \text{otherwise} \end{cases}$$

1 Unless otherwise stated, we remove the subscript $j$ representing the number of generations for the simplicity of representation.
where \( P_{cl}, P_{c2}, P_{m1}, P_{m2} \) are empirically specified constants, and the remaining notations are given in (7). Intuitively, the above equations render \( g \) with a value of \( \text{SSE}(g) \) smaller (resp. larger) than the average have larger (resp. smaller) \( P_{c}, (g_{k}, g_{t}) \), \( P_{m}(g) \). In this way, low-quality genes are likely to be improved.

\[
\begin{align*}
\text{SSE}' &= \max \{ \text{SSE}(g_{k}), \text{SSE}(g_{t}) \}, \\
\text{SSE}^\text{avg} &= \frac{1}{j} \sum_{g \in G} \text{SSE}(g), \\
\text{SSE}^\text{max} &= \max \{ \text{SSE}(g) | g \in G \}. 
\end{align*}
\tag{7}
\]

To prevent a generation \( G_{j} \) from degrading to genes of lower quality, we introduce the evolution-with-restart mechanism. Specifically, when \( \text{SSE}^\text{max}_{j} = \max \{ \text{SSE}(g) | g \in G_{j} \} \) is smaller than 0.99\( \text{SSE}^\text{max} \), we assign all the elements in \( G_{1} \) to \( G_{j} \). The evolution runs until we reach the \( j \)'th generation satisfying \( \text{SSE}^\text{max}_{j} \) exceeds \( \text{SSE}^\text{max} \). We derive \( L'_{N} \) from \( \text{arg max}_{g} \{ \text{SSE}(g) = \text{SSE}^\text{max}_{j} | g \in G_{j} \} \).

We summarize the A-SEO strategy in Algorithm 2.

**Algorithm 2:** Advanced strike-efficiency-oriented strategy.

**Input:** \( R_{N}, L_{N}', n, L'_{N} \) from the N-SEO strategy

**Output:** \( L'_{N} \)

\( G_{1} = \bigcup_{t=1,2,...,n-1} \{ \text{set2gene}(X_{t}) \} \}

random target links | \{ \text{set2gene}(L_{N}') \} |

\( \text{SSE}_{\text{avg}}^\text{max} = \max \{ \text{SSE}(g) | g \in G_{1} \} \}

**while** \( \text{SSE}^\text{max}_{j} \leq \text{SSE}^\text{max} \)

**do**

\( j++ \)

\( G_{j} = \emptyset \)

**while** \( |G| < n \)

**do**

select \( g_{k} \) and \( g_{t} \) over probability distribution \( \{ \text{SSE}_{\text{avg}} \} | g \in G_{j-1} \}

\( g_{j} = G_{j} \cup \{g_{k}, g_{t}\} \)

**generate a random float** \( r, 0 < r < 1 \)

if \( r < P_{m}(g_{k}, g_{t}) \) then

\( g_{k}', g_{t}' = \text{crossover}(g_{k}, g_{t}) \)

\( G_{j} = G_{j} \cup \{g_{k}', g_{t}'\} \)

**end if**

**end while**

**for** \( g \in G_{j} \)

**do**

if \( r < P_{m}(g) \) then

\( g_{m} = \text{mutation}(g) \)

\( G_{j} = G_{j} \cup \{g_{m}\} \)

**end if**

**end for**

\( \text{SSE}^\text{max} = \max \{ \text{SSE}(g) | g \in G_{j} \} \)

if \( \text{SSE}^\text{max} < 0.99 \text{SSE}^\text{max} \)

\( G_{j} = G_{j} \)

**end if**

**end while**

\( L'_{N} = \text{gene2set}(\text{arg max}_{g} \{ \text{SSE}(g) = \text{SSE}^\text{max} | g \in G_{j} \}) \)

\[ 3) \text{Strike-Precision-Aware Hybrid (SPAH) Strategy} \]

To seek strike precision along with strike efficiency, we propose the SPAH strategy in Algorithm 3. Specifically, we maximize \( |R'_{N}| / |R_{N}| \) and meanwhile minimize \( |R'_{N}| / |R_{N}| \) (i.e., \( AI \)). Adversaries iteratively select a link \( l \in L_{N} \setminus L_{N} \) maximizing the reward gain, \( \theta(l) = (|R'_{N}((L_{N} \cup \{l\}) - |R'_{N}((L_{N} \cup \{l\}) - |\{m(\{l\})\}) - |\{m(\{l\})\}) \) \). \( R'_{N}() \) calculates the number of interrupted routes to \( N \) due to clogging a set of links. This process repeats until \( |R'_{N}| / |R_{N}| \) approaches a pre-specified goal \( V \), or \( |L_{N}| \) reaches the budget of the number of selected links \( \Phi \).

**Algorithm 3:** Strike-precision-aware hybrid strategy.

**Input:** \( R_{N}, L_{N}', V, \Phi \)

**Output:** \( L'_{N} \)

\( L'_{N} = \emptyset, R_{N} = \emptyset ; \)

**for** \( l \in L_{N} \)

**do**

\( \text{calculate } R'_{N}(l) \)

**end for**

while \( |R'_{N}| / |R_{N}| < V \)

**do**

\( l = \text{arg max}_{l \in L_{N} \setminus L_{N}} \theta(l) \); \( \Phi(l) \)

if \( L'_{N} = \emptyset \) then

\( \text{break} \);

**end if**

**end while**

\( L'_{N} = L'_{N} \cup \{l\} \); \( \Phi = R'_{N} \cup R'_{N}(l) \);

**for** \( l \in L_{N} \)

**do**

\( \text{calculate } R'_{N}(l) \)

**end for**

\[ V. \text{MODELING LINK FLOODING DEFENSES} \]

In this section, from the perspective of defenses, we first study attack intention, i.e., inversely inferring the target network that the adversary would like to disconnect by LFA. Then, we formulate the game-theoretic interactions between attackers and defenders so to effectively mitigate LFA.

\[ A. \text{ATTACK INTENTION} \]

Since the pioneering work of Xue [6], substantial progress has been made to detect LFA and infer the target links under attack. Normally, sophisticated adversaries choose the target links that are over three hops away from the target area [10]. In this case, even though the target links under attack are successfully detected, the target network area that the adversary would like to disconnect, i.e., attack intention, remains unclear.

To effectively infer the target networks from the set of target links under attack, we construct a map from the target networks to the sets of target links. We take the set of target links when adversaries are trying to interrupt 50% of routes as the anchor target links. Let \( N_{1} \) and \( N_{2} \) denote the intended and innocent network respectively. We define the percentage of overlapped anchor links, denoted by \( O(N_{1}, N_{2}) \), below:

\[ O(N_{1}, N_{2}) = \frac{|L'_{N_{1}} \cap L'_{N_{2}}|}{|L'_{N_{2}}|} \]

\[ (8) \]
TABLE 1: The values of $O(N_1, N_2)$ for each combination of $N_1$ and $N_2$.

| $N_2$ | CH | SG | TW | JP | HK |
|-------|----|----|----|----|----|
| CH    | 1  | 0  | 0  | 42.86% | 0 |
| SG    | 0  | 1  | 4.76% | 14.29% | 14.29% |
| TW    | 0  | 7.69% | 1  | 7.69% | 7.69% |
| JP    | 15.00% | 15.00% | 5.00% | 1  | 10.00% |
| HK    | 0  | 16.00% | 4.00% | 8.00% | 1 |

In TABLE 1, we measure $O(N_1, N_2)$ for each combination of target networks when attackers use the N-SEO strategy. The overlap for any combination of different target networks is small. Only a small number of anchor target links for the innocent networks will be flooded when adversaries are attacking the intended network. Therefore, we can easily infer the target network accordingly by monitoring the anchor target links. According to the mapping, the set of target links are unique for a specific target network. Assume the attacker only attacks a target network at a time. If most of the links in a set of the target links are flooded, we can figure out the target network. We also measure $O(N_1, N_2)$ for the SPAH strategy, and find that for each combination of $N_1$ and $N_2$, $O(N_1, N_2)$ equals 0. Therefore, there is no overlap of the target links for the SPAH strategy. As a consequence, one can easily figure out the target network for the SPAH strategy.

B. A GAME-THEORETIC FORMULATION OF LINK FLOODING ATTACKS AND DEFENSES

We formally design the link flooding attack and defense game. As for attackers, they will select a set of target links $L_{X}^{c}$ to congest. By congesting these links, attackers will obtain a reward $r(L_{X}^{c})$ at the cost of $c(L_{X}^{c})$ due to renting bots. Attackers will also endure a loss $i(L_{X}^{c})$ brought by attack interference. Attackers will attack a set of links $L_{X}^{c}$ besides the intended ones, which brings their corresponding $r(L_{X}^{c}), c(L_{X}^{c}), i(L_{X}^{c})$. The overall gain of the attacker will be formulated as

$$G_1 = r(L_{X}^{c}) + c(L_{X}^{c}) + r(L_{X}^{c}) + r(L_{X}^{c}) + c(L_{X}^{c}) + i(L_{X}^{c}).$$

Defenders, on the other hand, will use a cost $c_d(L_{X}^{c}, L_{X}^{c})$ to detect the flooded links. Successfully detecting the flooded links will assign the defender a reward $r_d(L_{X}^{c}, L_{X}^{c})$. To tackle the LFA's, the defender will perform traffic engineering to reroute attack traffic to a set of links $L_{p}^{c}$. The defender can obtain a reward $r_d(L_{p}^{c})$ and a cost $c_d(L_{p}^{c})$. Traffic engineering may also introduce attack interference $i_d(L_{p}^{c})$ to the innocent areas. The overall gain of the defender equals

$$G_2 = r_d(L_{X}^{c}, L_{X}^{c}) + c_d(L_{X}^{c}, L_{X}^{c}) + r_d(L_{p}^{c}) + c_d(L_{p}^{c}) + i_d(L_{p}^{c}).$$

The attacker needs to pay a cost $c_r$ on probing the network to figure out the traffic engineering method that the defender deployed. The probing results will provide the attack with a reward $r_p$. Then, the attacker will flood a subset of parallel links $L_{p}^{c} \subseteq L_{p}^{c}$ with a reward $r(L_{p}^{c})$ and a cost $c(L_{p}^{c})$. Therefore, the overall gain of the attacker becomes

$$G_1 = r(L_{X}^{c}) + c(L_{X}^{c}) + r(L_{X}^{c}) + r(L_{X}^{c}) + c(L_{X}^{c}) + i(L_{X}^{c}) + c_p + r(L_{p}^{c}) + c(L_{p}^{c}).$$

(11)

Apparently, attempting to flood all the parallel links is the optimal strategy for the adversary. If the adversary tries to flood a new set of links, defenders will be able to detect the adversary’s attack intention and use traffic engineering to protect the new set of flooded links. In such a case, the adversary will never degrade the Internet connection.

Given $L_{p}^{c}$, the adversary chooses $L_{p}^{c} \subseteq L_{p}^{c}$ that maximize Equation (11). Formally, $L_{p}^{c}$ will be

$$L_{p}^{c} = \arg\max_{L_{p}^{c}} \{r(L_{X}^{c}) + c(L_{X}^{c}) + r(L_{X}^{c}) + r(L_{X}^{c}) + c(L_{X}^{c}) + i(L_{X}^{c}) + c_p + r(L_{p}^{c}) + c(L_{p}^{c})\}.$$  

(12)

The defender should choose a set of links $L_{p}^{c} \supseteq L_{p}^{c}$ to deploy traffic engineering. Therefore, $L_{p}^{c}$ should be

$$L_{p}^{c} = \arg\max_{L_{p}^{c}} \{r_d(L_{X}^{c} \cup L_{X}^{c}) + c_d(L_{X}^{c} \cup L_{X}^{c}) + r_d(L_{p}^{c}) + c_d(L_{p}^{c}) + i_d(L_{p}^{c})\}.$$  

(13)

Initially, the adversary should choose a set of target links $L_{X}^{c}$ and a set of links beyond the intended ones $L_{X}^{c}$ to congest. Considering the traffic engineering that a defender will use, the adversary would choose links with fewer parallel links to attack. To seek such links along with strike efficiency, we propose the LPSE strategy in Algorithm 4. Specifically, we maximize $SE$ and minimize $L_{p}$. Adversaries iteratively select a link $l \in L_{X} \setminus L_{X}$ maximizing the reward gain, $\theta_l(L_{X}) = (\|R_{X}^{l}(L_{X}) \setminus \{l\})| - \|L_{X} \setminus \{l\})| - (\|R_{X}^{l}(L_{X}) \setminus \{l\})| - \|L_{X} \setminus \{l\})|$, where $L_{X}$ calculates the number of parallel links and $\varepsilon$ denotes a parameter for defendees choose to maximize his gain. This process repeats until $|R_{X}^{l}(L_{X}) \setminus \{l\})| \varepsilon \|L_{X} \setminus \{l\})|$ reaches the budget of the number of selected links $\Phi$.

For attackers who are concerned about both attack interference and traffic engineering deployed by defenders. They will design an attack strategy to maximize $SE$, and minimize $L_{p}$ and $AI$. To seek such goals, we propose the LP-SPA strategy in Algorithm 5. Adversaries iteratively select a link $l \in L_{X} \setminus L_{X}$ maximizing the reward gain, $\theta_l(L_{X}) = (\|R_{X}^{l}(L_{X}) \setminus \{l\})| - \|L_{X} \setminus \{l\})| - (\|R_{X}^{l}(L_{X}) \setminus \{l\})| - \|L_{X} \setminus \{l\})|$, replacing $\theta_l(L_{X})$ with $\theta_l(L_{X})$ will generate the LP-SPA strategy.

Note that the defender will use traffic engineering to mitigate the congestion. For the defender, the best response is to choose a set of links $L_{p}^{c}$ that satisfy Equation (13) to perform traffic engineering. For the adversary, the best action is to
maximize his reward considering the traffic engineering that will be used on the links. That is,

\[ (\mathcal{L}'_N, \mathcal{L}'_X) = \arg \max \{ r(\mathcal{L}'_N) + c(\mathcal{L}'_N) + r(\mathcal{L}'_X) + c(\mathcal{L}'_X) + i(\mathcal{L}'_X) \}. \] (14)

VI. PRACTICAL ISSUES OF NON-COOPERATIVE DEFENSES

The attack target of LFAs is the key links around the target network area. Attackers control a large number of zombies to access public servers around the target network area, congest the critical links along the access paths, and ultimately cut off the connection between legitimate users and the target network area. In order to achieve such a goal, attackers first need to get the topology around the target network and figure out the critical links.

A feasible defense strategy against LFAs is to carry out long-term monitoring of the status of target links. However, it is difficult to observe the status of a target link separately. As a consequence, we need to indirectly infer whether a link is under attack by measure the network path consisting of that link. Since a path normally contains many routers that cannot be directly controlled, we monitor a path through non-cooperative measurement methods. Moreover, because there are many paths around the target network and the links of different paths are redundant, the efficiency of monitoring all paths is extremely low and will cause a large measurement overhead. Therefore, we design a lightweight and effective scheme that enables defenders to quickly select the most valuable paths for monitoring.

A. LIGHTWEIGHT PROBE DEPLOYMENT

A conventional approach for performing non-cooperative network measurement is to deploy many distributed probes that may be highly dispersed in various locations. These distributed probes have IP addresses of different networks, thereby offering plenty of measurement vantage points. However, deploying distributed probes is rather cost-intensive due to the high cost of renting and maintaining servers. To address such a limitation, we propose a lightweight probe deployment approach based on the dynamic virtual private server (VPS). The dynamic VPS can continuously switch IP addresses in different regions through keeping dialing-up repeatedly using just a single physical machine flexibly and randomly.

Our experimental results show that a dynamic VPS in China can randomly switch IP addresses in 26 provinces, including 181 cities. We deploy shell scripts on the dynamic VPS to periodically obtain new IP addresses and save the results to the database. The experiment lasted 335 days and obtained a total of 715,612 unique IP addresses. Based on the above finding, we propose a method of topology discovery based on dynamic VPS. Using dynamic VPS, we can obtain a large number of highly dispersed IP addresses easily and quickly. These IP addresses will be used as the probe host for topology discovery, and their distribution will affect the coverage and redundancy of network topology discovery performance.

In practice, since the IP address pool of a dynamic VPS is generally composed of IP addresses in different cities, it is necessary to sample the approximate distribution of IP addresses for a period of time in advance. Then, we need to assign corresponding measurement targets to the dynamic VPS based on the distribution of the IP addresses. When performing topology discovery using the dynamic VPS, we adaptively adjust the corresponding target IP address (where the measurement packets are destined) according to the IP address of the dynamic VPS (where the measurement packets originate) and the historical measurement results. In this way, the efficiency of topology discovery could be improved and link redundancy could be reduced.

To be more specific, every time when the IP address of the

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
dynamic VPS is switched randomly via automatic dial-up, we query the location of the current IP address, and compare it with historical IP address records. If the current IP address is close to historical IP address records, the redundancy of topology discovery result is expected to be high. In such a case, we dial up again to switch to a new IP address. Otherwise, we will assign the current IP address as the next scheduled measurement target. The above process repeats until we collect sufficient topology information of the target network using IP addresses of the dynamic VPS. Next, we integrate the information collected by different IP addresses to comprehensively analyze the topology around the target network.

**B. EFFECTIVE TOPOLOGY PROBING**

After completing topology collection of the target network, we analyze the data and find out paths suitable for long-term monitoring. To ensure the comprehensiveness of the monitoring, one needs to monitor all links around the target network to cover the set of target links that an attacker may attack. However, there are a large number of redundant links between different paths, and we need to select paths that can cover as many links as possible to reduce the number of paths to monitor.

Therefore, how to select an optimal set of paths from the collected topology data is an important issue. In addition, when measuring the optimal set of paths, the following requirements should also be satisfied to achieve good performance: (1) the paths should cover at least one target link; (2) the measurement traffic towards an individual target link is minimized (since measurement brings additional traffic load); (3) the tasks assigned to probe should not be paralyzed so as to avoid mutual-interfere between measurement packets; (4) high-frequency measurement on the same public server should be prohibited so as not to be filtered by the firewall.

In order to find the optimal set of paths, we propose two path selection algorithms in consideration of the above requirements. We next detail the two path selection algorithms.

1) Path selection based on greedy algorithm

To find the smallest set of measurement paths to cover all links, an intuitive method is to choose a path that maximizes the intersection with the links of the remaining paths each time. We propose a fast path selection method based on the greedy algorithm. The greedy algorithm continuously expands the selection set. Therefore, it is necessary to choose the path that brings the highest profit under the current situation. In our experiment, how much profit a path can bring depends on its intersection with the remaining paths. The profit of a path, say \( P_a \), is calculated as follows.

\[
P_a = \left| \bigcup_{i=1}^{\#link(a)} \text{path}(\text{link}_i) \right|
\]

(15)

where \( \#\text{link}(a) \) represents the number of links of \( a \), \( \text{link}_i \) denotes the \( i \)th link of \( a \), and \( \text{path}(\text{link}_i) \) is the set of paths consisting of \( \text{link}_i \).

At the beginning of the algorithm, the selected path set is empty. To calculate the profit of each path, all links along the path need to be included. We then compare different paths. In each round of the greedy algorithm, the path with the largest profit will be selected and added to the selected path set. Accordingly, the links of the selected path are added into the selected link set.

When calculating the profit of each path, only the links that do not appear in the selected link set are considered. Similarly, when calculating the total number of paths including a link, only the paths that have not been selected are considered, and paths in the selected path set would not be considered. This means that the profit of each path is not only determined by links that it contains, but also related to the previously selected path set. The path selection based on the greedy algorithm is shown in Algorithm 6.

**Algorithm 6: Greedy Path Selection Algorithm.**

Input: Dict
Output: Path num
initialize linkDictNum = ∅, linkSet = ∅
function GetMinPathNum(Dict)
for srcDst in Dict do
  for link in Dict[srcDst] do
    if link not in linkDictNum then
      linkDictNum[link] = 0
    end if
    linkDictNum[link] += 1
  end for
end for
linkSet = set(linkDictNum.keys())
result = greedySelect(linkSet, linkDictNum)
return len(result)
function GreedySelect(linkSet, linkDictNum)
while len(linkSet) > 0 do
  sendDictScore = dict()
  for srcDst in Dict do
    score = 0
    for link in Dict[srcDst] do
      if link in linkSet then
        score += linkDictNum[link]
      end if
    end for
    endDictScore[srcDst] = score
  end for
  sortedScore = sorted(endDictScore.iteritems(), key = lambda x: x[1], reverse = True)
  end while
  rSrcDst = sortedScore[0][0]
  pathSet.add(rSrcDst)
  for link in Dict[rSrcDst] do
    linkSet.remove(link)
  end for
  linkDictNum.pop(rSrcDst)
return pathSet

2) Path selection based on improved genetic algorithm

The greedy algorithm chooses the path with the largest profit in each round. Although computationally effective,
Steps are presented below.

The path selection problem could be simulated as the process of biological evolution. The next generation of individuals is generated through selection, crossover, mutation. The elimination of individuals with low fitness means that individuals with high fitness survive. Therefore, after the N-generation evolution of individuals with low fitness means that individuals with high fitness survive. Therefore, after the N-generation, the population tends to be consistent or local optimal, the crossover mutation probability is increased; Otherwise, the crossover mutation probability is reduced [25]. The crossover probability and mutation probability of the genetic algorithm are highly adaptable individuals.

The detailed algorithm is shown in FIGURE 5. The major steps are presented below.

- Chromosome Coding
  Each path is represented as a chromosome, and the genes on the chromosome represent the links that constitute the path. Let $P = [p_1, p_2, \ldots, p_n]$ represent a path, and $p_i (i = 1, 2, \ldots, n)$ is an individual link.

- Population Initialization
  To meet the requirements of the diversity and randomness of the initial population, we randomly pick up candidate paths. At the same time, there are certain restrictions when using the genetic algorithm for path selection. Specifically, when initializing the population, it is not enough to randomly select links to form a chromosome. Since the final requirement of covering all links with the least number of paths is required when constructing chromosomes, it must be satisfied that their descendants as many links as possible.

- Genetic Operator
  Genetic operations mainly use a series of genetic operators to perform operations. Traditional genetic algorithms include selection, crossover and mutation operations. It is worth noting that the crossover operator we use adopts a single-point crossover method. At the same time, due to the characteristics of network path, we choose to cross at the common link of two paths to avoid path discontinuity after crossover. If there are multiple common links, we randomly select one link for crossover; if there is no common link, no crossover operation will be performed. In addition, since the mutation operation will select a path to replace the original path, it may cause breakpoints in the path. Therefore, after the mutation operation, it is necessary to determine whether the newly generated individual meets the actual requirements. If the requirements are not met, the mutation operation repeats again.

  - Fitness function
    The fitness function is a performance index to judge the pros and cons of individuals. It is crucial for the genetic algorithm to find the optimal solution. In our context, the optimal solution is to cover as many links as possible with a limited number of paths. Therefore, from the perspective of reducing link redundancy, we design the fitness function as follows:

    $$ f = \frac{1}{\sum_{i=1}^{n} LR_i}, $$

    where $n$ represents the number of links contained in the current path, and $LR_i$ represents the redundancy of the $i$th link in the current population.

    The above algorithm uses the roulette-based random sampling method to select genes. The sampling probability of each gene equals the proportion of its fitness value to the total fitness of all genes, thereby genes with larger fitness values having higher probabilities of being selected.

    The crossover probability and mutation probability of genes are two key parameters affecting the convergence of the genetic algorithm. Increasing the crossover probability can speed up the generation of new genes at the risk of destroying high fitness genes. As the mutation probability grows large, the genetic algorithm will evolve into a random search algorithm. We employ adaptive cross mutation probability [25]. When the fitness of individuals in the gene population tends to be consistent or local optimal, the cross mutation probability is increased; Otherwise, the cross mutation probability is reduced [25]. The crossover probability $P_c$ and the mutation probability $P_m$ are calculated in equations (17) and (18), respectively.

    $$ P_c = \begin{cases} \frac{k_1(f_{max} - f)}{f_{max} - f_{avg}}, & f \geq f_{avg} \\ \frac{k_2}{k_2}, & \text{else}, \end{cases} $$

    $$ P_m = \begin{cases} \frac{k_3(f_{max} - f)}{f_{max} - f_{avg}}, & f_i \geq f_{avg}, \\ \frac{k_4}{k_4}, & \text{else}, \end{cases} $$

    where $f_{max}$ denotes the maximum fitness value of the gene in the population, $f_{avg}$ represents the average fitness value, $f$
represents the larger fitness value of the two genes to cross, and \( f' \) represents the fitness value of the gene to mutate. \( k_1, k_2, k_3, \) and \( k_4 \) are constants satisfying \( k_1 < k_2, k_3 < k_4 \). The adaptive cross-mutation probability setting allows genes with low fitness to have a higher probability of evolution. The path selection based on the improved genetic algorithm is detailed in Algorithm 7.

Algorithm 7: The Improved Genetic Algorithm for Path Selection

| Input: Dict |
| Output: Path num |
| function GetMinPathNum(Dict) |
| for src in Dict |
| for link in Dict[src] do |
| if link not in linkDictNum then |
| linkDictNum[link] = 0 |
| end if |
| end for |
| end for |
| linkSet = set(linkDictNum.keys()) |
| initPopulation = calFixedInitPopulation(cnt = 40) |
| curPopulation = list(initPopulation) |
| while gIndex <= 100 do |
| selectPopulation, persistList = genSelect(curPopulation, allNum, persistNum, gIndex) |
| cMaxFitness = 0 |
| cSumFitness = 0 |
| for i in xrange(len(selectPopulation)) do |
| cl, cp, cProfit = calProfit(selectPopulation[i]) |
| if cProfit > cMaxFitness then |
| cMaxFitness = cProfit |
| end if |
| cSumFitness += cProfit |
| end for |
| cAvgFitness = cSumFitness/len(selectPopulation) |
| crossover(selectPopulation, cMaxFitness, cAvgFitness) |
| mutation(selectPopulation, cMaxFitness, cAvgFitness) |
| for top in persistList do |
| selectPopulation.append(top) |
| end for |
| curPopulation = copy.deepcopy(selectPopulation) |
| gIndex += 1 |
| end while |
| for gene in curPopulation do |
| finalProfitList.append(calProfit(gene)) |
| end for |
| finalProfitList.sort(key = lambda x: x[1], reverse = True) |
| return len(finalProfitList) |

C. MULTI-PROTOCOL-BASED LINK MEASUREMENT

Through path selection, we obtain a set of paths to monitor. Then, the real-time status of each path and link could be observed via non-cooperative measurement. Non-cooperative measurement means that the deployment of measurement tools are required to be only single-ended, hence drastically reducing the deployment overhead. However, when performing non-cooperative measurement, protocols like ICMP may be dropped by some intermediate routers or end servers. Therefore, we propose a multi-protocol-based link measurement method based on Linkscope \(^5\) so to enhance the measurement robustness.

The measurement method we use is based on the packet train, i.e., a sequence of back-to-back packets. By sending a packet train, we can estimate the status of the links along a path based on the characteristics of response times. In addition, since the size of the packet train in our experiment is less than 50Kb, the amount of the measurement traffic is small. As a consequence, the measurement introduces limited overhead, hence suitable for long-term deployment.

As shown in FIGURE \(6\), our packet train is composed of measurement packets and load packets, which are represented by colored and shaded boxes, respectively. The measurement packet supports multiple protocol types, including TCP, UDP, and ICMP. The Time-to-Live (TTL) of the measurement packet increases symmetrically from both sides to the center of the train. Every time the packet train passes through a router, a pair of response packets will be triggered. Using these response packets, we can estimate the available bandwidth of each link along the path. In addition, because different networks may differ in packet filtering rules, packet train designed based on multiple protocols can adapt to sophisticated network environments. By default, we use the SYN connection request packet as TCP-based measurement packet, DNS request packet as UDP-based measurement packet, and ICMP echo request packet as ICMP-based measurement packet.

Using the multi-protocol-based link measurement method, we measure the selected path so to be aware of critical links of target network. Once an attacker launches an attack against a critical link, the defender could perceive the location of attacked link and in turn take defensive strategies based on the defense model in Sec. \(\S\).

VII. EXPERIMENTS

A. DATASET PREPARATION

We perform traceroute from source (Planetlab) servers distributed all over the world to target (public) servers in five regional networks. FIGURE \(7\) shows the locations of these servers. There are totally 81 source servers (40 in US, 7 in China, 5 in Canada, 4 in Australia, 4 in Brazil, 3 in Germany, and 18 in others), and 4,843 target servers (1,048 in E, 962 in C, 957 in B, 942 in A, 934 in D). The entire dataset consists of more than \(1.2 \times 10^6\) routes and \(1.2 \times 10^6\) links.

We consider part of the networks in five different regions as intended victim networks, namely, SubA in A, SubB in B, SubC in C, SubD in D, subE in E, and the remaining part as innocent victim networks (e.g., the network in A except SubA). In each attack, the adversary aims to disconnect one intended victim network by selecting target links to clog through the N-SEO strategy, the A-SEO strategy and the SPAH strategy. In the N-SEO strategy and the SPAH strategy, we assume \(\Phi = |L_N|\). In the A-SEO strategy, we set \(P_{c1} = 0.9, P_{c2} = 0.6, P_{m1} = 0.1, P_{m2} = 0.001\).
B. MEASURING STRIKE PRECISION UNDER DIFFERENT STRATEGIES

FIGURE 8 shows attack interference (AI) when \(|R'_N|/|R_N|\) (i.e., the percentage of interrupted routes) varies from 50% to 100%, with a step of 10%. Each line corresponds to a specific intended victim network.

We observe that, among the three strategies, the N-SEO strategy introduce the most attack interference, the SPAH strategy the least, while the A-SEO in the moderate, for a specific intended victim network. Apparently, the SPAH strategy substantially lowers attack interference by considering attack interference to innocent victim networks. In addition, for both the N-SEO strategy and the A-SEO strategy, AI grows roughly linearly as the percentage of interrupted routes increases. However, such a tendency is not significant for the SPAH strategy, indicating that no obvious attack interference would be introduced as more routes are interrupted.

Under a specific strategy, attack interference to networks such as SubD in D is relatively lower, whereas that to networks like SubA in A is relatively higher. The reason is that SubD is a small area for D, and SubA accounts for a large area for A. Therefore, the larger the ratio of the intended victim network size in the entire regional network is, the more interference to the entire network would be posed.

FIGURE 9 reports strike precision (SP) over \(|R'_N|/|R_N|\) under different strategies, as further cross-validate the results about attack interference. Specifically, the SPAH strategy achieves the largest \(SP\), the N-SEO strategy the least, while the A-SEO strategy in the moderate, for a specific intended victim network. Under the N-SEO strategy and the A-SEO strategy, the value of \(SP\) when SubD is disconnected is larger than those when the remaining intended victim networks are disconnected. This observation coincides with the lower attack interference when SubD is disconnected.

To exemplify the selected links intuitively, we plot the geographical locations of selected target links via the N-SEO strategy and the SPAH strategy, when \(N = \text{SubA}''\) (i.e., SubA in A) and \(|R'_N|/|R_N| = 0.8\) in FIGURE 10. We label the links commonly selected by two strategies (20 links in red), and the links exclusively selected by each strategy (22 links by N-SEO, and 39 links by SPAH, both in black). This example demonstrates that the set of selected links by the SPAH strategy are different from and larger than those by the N-SEO strategy, suggesting that, when interrupting the same number of routes, the SPAH strategy comes at the cost of more selected links to clog.

For terrorist adversaries who are interested in mass destruction, they may disconnect the Internet connection of more than one target regions. We consider the network of A and the network of one of the other four regions as intended victim networks, namely, A & B, A & C, A & D, A & E. The network of the left three regions are the innocent victim networks (e.g., the networks of C, D and E when attacking A and B simultaneously). We also measure the percentage of interrupt routes for the intended victim networks to show the attack impact for an individual network (e.g., the percentage of interrupted routes for A and the percentage of interrupted routes for B when attacking A and B simultaneously). In each attack, the adversary selects target links to clog through the N-SEO strategy. In both strategies, we assume \(\Phi = |L'_N|\).

We measure the number of target links when attacking the target regions simultaneously and separately. As shown in FIGURE 11 \(|R'_N|/|R_N|\) (i.e., the percentage of interrupted routes) varies from 50% to 100%, with a step of 10%.

As for attackers, simultaneously attacking multiple regions is more threatening since larger areas of networks are disconnected. Also, the defense can be harder to deploy because it is difficult for two separate regions to collaborate. We simulate LFAs when attackers want to interrupt the Internet connection of multiple regions. FIGURE 12 shows attack interference (AI) attacking two regions simultaneously. To make a comparison, we also simulate LFAs when attackers interrupt the Internet connection of one region in FIGURE 13. In this case, the other 4 regions are innocent networks.
(a) N-SEO strategy
(b) A-SEO strategy
(c) SPAH strategy

FIGURE 8: Attack interference ($AF$) as the percentage of interrupted routes varies under different strategies.

(a) N-SEO strategy
(b) A-SEO strategy
(c) SPAH strategy

FIGURE 9: Strike precision ($SP$) as the percentage of interrupted routes varies under different strategies.

(a) N-SEO strategy (42 links)
(b) SPAH strategy (59 links)

FIGURE 10: The locations of selected target links via different strategies, when $\mathcal{N}$ = “KLN” (i.e., SubA in A) and $|\mathcal{R}_\mathcal{N}'|/|\mathcal{R}_\mathcal{N}| = 0.8$.

(a) attack A&C
(b) attack A&B
(c) attack A&E
(d) attack A&D

FIGURE 12: Attack interference for all the other areas when attacking two regions simultaneously as the percentage of interrupted routes varies under the N-SEO strategy.

We observe that, $AF$ is relatively higher for all the four attacks than the $AI$ in FIGURE 13. This is because attacking multiple targets simultaneously requires more target links than attacking only one target. The extra target links result in more severe attack interference for the innocent networks. When adversaries launch LFAs to multiple targets, the attack also leads to more severe attack interference for the innocent networks. More severe attack interference tends to incentivize the victim networks to collaborate with the innocent networks in defending against LFAs. As shown in FIGURE 14, the SPAH strategy can effectively reduce the attack interference if attackers target one area.

We also measure the attack interference when attacking...
FIGURE 13: Attack interference (attacking one region) as the percentage of interrupted routes varies under the N-SEO strategy.

FIGURE 14: Attack interference for other (innocent) regions when attacking one region under the SPAH strategy.

FIGURE 15: Attack interference for surrounding regions when attacking two regions simultaneously as the percentage of interrupted routes varies under the N-SEO strategy.

FIGURE 16: Attack interference for surrounding regions when attacking two regions simultaneously as the percentage of interrupted routes varies under the SPAH strategy.

two regions simultaneously under the SPAH strategy. In FIGURE 16, the innocent networks are the networks surrounding the target network. For example, if attackers intend to attack SubA and SubC, the complement part in A and C are the victim networks. According to our results in FIGURE 15, the SPAH strategy has less attack interference than the N-SEO strategy. In FIGURE 17, the innocent networks are all the available networks except the target networks. For example, if attackers intend to attack A and C, the networks in SubE, B and D are the victim networks.

C. MEASURING STRIKE EFFICIENCY UNDER DIFFERENT STRATEGIES

We measure strike efficiency ($SE$) over $|\mathcal{R}_N'|/|\mathcal{R}_N|$ under different strategies. The result is shown in Figure 18.

For all strategies, as the percentage of interrupted routes increases, $SE$ drastically decreases. Compared with the N-SEO strategy, the A-SEO strategy has a higher value of $SE$. However, the strike efficiency of the SPAH strategy is just slightly lower than that of the SEO strategy. Since the SPAH strategy takes both strike precision and strike efficiency into account, it achieves high strike precision without significantly lowering strike efficiency.

We also measure strike efficiency ($SE$) over $|\mathcal{R}_N'|/|\mathcal{R}_N|$ under different strategies for flooding multiple targets. As shown in FIGURE 19, the N-SEO strategy performs better than the SPAH strategies when attacking multiple targets. However, when the attacker intends to interrupt over 80% of routes to the target areas. The SPAH strategy performs as well as the N-SEO strategy.

Combining the results above, we have the following conclusions. First, according to FIGURE 8 and FIGURE 9 on
average, when adversaries interrupt 80% of routes to the intended victim network by strategies that only consider strike efficiency, attack interference is severe and strike precision is low. More precisely, by the N-SEO strategy, we have $AI = 46.82\%$, $SP = 69.16\%$, and $AI = 40.36\%$, $SP = 72.18\%$ for the A-SEO strategy.

Second, the main factors affecting attack interference and strike precision include the percentage of routes adversaries would like to interrupt (i.e., a larger percentage results in more attack interference and lower strike precision), the ratio of the size of the intended victim network to that of the innocent victim network (i.e., a larger ratio generally leads to less attack interference and higher strike precision). To lower attack interference and improve strike precision, the SPAH strategy can be used. On average, when adversaries interrupt 80% of routes to the intended victim network by the SPAH strategy, $AI = 13.34\%$, $SP = 91.79\%$, at the cost of slightly more selected links to clog but without significantly decreasing strike efficiency.

**D. MEASURING ATTACK EFFECT & STRIKE PRECISION USING TRAFFIC ENGINEERING**

We evaluate the effects of traffic engineering against LFAs. Defenders, as illustrated in Sec. II-B, can deploy parallel links to defend against LFAs. Therefore, by distributing the flows to parallel links, the congestion of the target link can be relieved. However, as illustrated in FIGURE 20, it is possible to flood the succeeding link (i.e., the next link) of the target link. To measure the effect of LFAs when flooding the succeeding links of the target links, we measure the percent of interrupted routes when flooding the succeeding links and the corresponding attack interference in FIGURE 21. According to the results, it is effective to use parallel links to distribute routes because the percentage of interrupted routes and attack interference decreases significantly.

For the target link that has more than one succeeding link, LFAs may also congest the succeeding links of the target link. To figure out the scale of the congested links in the next hop and their corresponding attack effect, we calculate the percentage of interrupted routes if the succeeding links are congested. In FIGURE 22, we assume if 60%, 70%, 80%, 90% of routes of the target link traverses the succeeding link, the succeeding link is congested. In such a scenario, the corresponding attack effect is shown in FIGURE 22.

We measure the percentage of interrupted routes when flooding parallel links, and the corresponding attack interference is shown in FIGURE 23. We find that re-routing flows to parallel links can effectively relieve the congestion and lead to no significant attack interference to the target networks. For terrorist adversaries, they may be interested in flooding all parallel links. However, as the attack effect, attack interference and strike efficiency illustrated in FIGURE 24 attackers cannot interrupt more routes but have to endure more severe attack interference and less strike efficiency compared with the results in Sec. VII-B and Sec. VII-C. Therefore, re-routing flows to parallel links can effectively relieve the congestion.

Considering the traffic engineering that defenders use, adversaries will choose target links following the LPSE strategy. In FIGURE 25, we measure the attack effect, attack interference and strike efficiency of the LPSE strategy when $\varepsilon$ equals 5. Using the LPSE strategy, adversaries can select the links that have fewer parallel links. Therefore, there are fewer parallel links for the defender to choose so to deploy traffic engineering. Meanwhile, adversaries can effectively improve their strike efficiency using the LPSE strategy.

For adversaries who are concerned about both attack interference, the traffic engineering defense and strike efficiency, they may use the LP-SPAH strategy to launch LFAs. In FIGURE 26, we measure the attack effect, attack interference and strike efficiency of the LP-SPAH strategy. Adversaries can lower their attack interference while keeping good attack effect and high strike efficiency. We therefore conclude that the LP-SPAH strategy can effectively help the attacker to launch LFAs in a sophisticated way.

**E. EXPERIMENTS ON PRACTICAL ISSUES OF NON-COOPERATIVE DEFENSES**

1) Path Selection Algorithm Evaluation

In order to evaluate the performance of the two path selection algorithms in Sec. VI-B, we rented several dynamic VPS servers and used them to collect real network topology data. To ensure the validity and diversity of the network topology data, we used 588 geographically dispersed IP addresses across China to detect 100 public servers in Xi’an, China. All these IP addresses originate from the dynamic VPS servers. Finally, we collected a dataset containing 58,800 paths and 27,279 unique links. We used the two algorithms to calculate the number of paths required to cover all links. The minimum path required when using the greedy algorithm is 13366, and the number is 12796 when using genetic algorithm. To compare the effects of the two algorithms, we randomly
selected 1,847 paths, including 1,200 unique links. Then, we compare the number of paths required to cover different numbers of links when the two algorithms are employed or no algorithms are used (i.e., unprocessed). The specific results are shown in FIGURE 27. As the figure shows, as the link coverage increases, the number of paths required to measure also increases. It can be found that both the greedy-based and improved genetic-based path selection algorithms can cover the specified links with fewer paths, and the improved genetic algorithm slightly outperforms the greedy algorithm.

2) Multi-protocol Measurement Evaluation

Linkscope is capable of performing packet pair-based non-cooperative measurement to perceive abnormal link performance changes \(^5\), and the multi-protocol measurement method proposed in this paper mainly focuses on further enhancing its robustness. In our experiment, we use one host to probe 300 DNS servers, and compare measurement success rates of different protocols by changing the protocol type used by the packet train. TABLE 2 shows the results.
The success-num in the table refers to the number of IPs that can successfully receive the packet train and return response packets. When using Multi-protocol, as long as any one of the three protocols can get response packets, it will be considered as a successful measurement. We see that the success rate of multiple-protocol measurement is significantly higher than that of single-protocol measurement.

**VIII. DISCUSSION**

Adversaries can effectively increase the precision when performing LFAs via the SPAH strategy. However, such precision comes at the cost of lowered strike efficiency. That is, compared with the SEO strategy, the SPAH strategy has to clog more target links in order to interrupt the same number of routes. Fortunately, the number of target links
is not increased substantially, whereas the precision is improved significantly. Furthermore, according to our data, the number of average (traceroute) hops from the target links selected by the SPAH strategy to the intended victim network equals 6.76, indicating that the attacked links are beyond the administrative domain of the intended victim network and thus defending these links against the attack remains difficult. We, therefore, envision that there is a tendency that adversaries take into account strike precision when performing LFAs for attack stealthiness and persistence. Terrorist attackers may deploy LFAs by Mirai or other IoT botnets to launch LFAs in a more powerful and sly way. By combining the pervasiveness of IoT devices and the slyness of LFAs concerning attack interference, defenders can design LFAs that is more challenging to defend against. By analyzing the behaviors of the TCP traffic, the attackers may use to launch congestion, we model the attack stages of the attacker and give suggestions for the defenders.

Besides the cost of lowered strike efficiency, seeking strike precision may lead to the increase of the ratio of the routes crossing the set of selected target links and destined to the intended victim network to all the routes crossing the set of selected target links. Consequently, when coordinating attack traffic flows, adversaries have more choices to send attack traffic flows to decoy servers inside the intended victim network. In this case, for attack stealthiness and persistence, rational adversaries would construct attack traffic flows meeting two requirements. First, each attack traffic flow, when being inspected separately, is indistinguishable from legitimate flows. Second, these attack flows should exhibit diversity without similar behavioral patterns (e.g., visit different websites at different times) to evade correlation-based detection.

To defend against rational adversaries in consideration of strike precision, one solution is to monitor the performance (e.g., available bandwidth) of the set of links selected by the SPAH strategy in real time and trigger warnings upon anomaly occurs [6]. Defenders can also employ traffic engineering methods to defend against LFAs. According to our results, traffic engineering can effectively help to relieve the congestion of target links and help to figure out the intended attacking target network by only observing the congestion of target links. Due to the strong power of LFAs, we argue that it is still challenging to tackle LFAs using traffic engineering. Therefore, we model and formulate the game-theoretic interactions between the attacker and defenders.

The experimental results demonstrate traffic engineering can effectively relieve the congestion brought by LFAs and help to figure out the intended attacking targets by observing the target links. According to our results, defenders should follow the strategies we propose to deploy their defense. In addition, from the perspective of routing topology design, one can also increase the overlapping of the (intermediate) links of different networks, as well as the entropy of the distribution of link importance to different networks. Consider the extreme case where all links are equally important to different networks in terms of connectivity. It would be challenging for rational adversaries in consideration of strike precision to determine which network to disconnect, since clogging one link may interfere many other networks.

IX. RELATED WORK

The layered architecture of the Internet enables various types of connectivity structures [27]. CAIDA's Archipelago project [28] and DIMES [29] measured the router-level Internet topology via traceroute. Kang et al. analyzed the pervasiveness of routing bottlenecks in 15 countries and 15 cities around the world. Albert et al. [30] showed that if an adversary disables 4% of the highly connected routers, the entire Internet will be broken up into small isolated pieces. However, later work by Magoni [31] and Wang et al. [32] concluded that breaking the entire Internet may not be infeasible because of the vast number of routers or links that need to be disconnected. The IoT-originated DDoS attacks have attracted much academic attention. The large volume, pervasiveness, and high vulnerability of IoT devices make IoT devices great threats to Internet security [33], [34].

Recently, LFAs have gained attention in the literature. Kang et al. proposed LFAs that can effectively cut off the Internet connections of a target area without being detected [10], [13]. The Coremelt attack could be considered a special case of LFAs [9]. Since LFAs result in abnormal link performance, traditional active link (and path) measurement techniques, such as packet pair and packet train, could naturally facilitate the detection of LFAs. To apply these techniques in detecting LFAs, LinkScope employs both end-to-end and hop-by-hop measurement to detect the links under such attacks [5], [6].

To defend against LFAs, Gkounis et al. showed that both existing and novel traffic engineering modules can efficiently expose the attack. They implemented a defense prototype using simulation mechanisms and evaluated it extensively on multiple real topologies [16]. Attacks by cost-sensitive attackers try to fully utilize the bots’ upstream bandwidth. Kang et al. tackled this root cause that it is sufficient to perform a rate change test, where they temporarily increased the effective bandwidth of the bottleneck core link and observed the response. Attackers will be detected since they are unable to demonstrably increase throughput after bandwidth expansion. Kang et al. designed a software-defined network (SDN) based system called SPIFFY that addresses key practical challenges in turning this high-level idea into a concrete defense mechanism, and provided a practical solution to force a trade-off between cost and detectability for LFAs [14]. Traffic Engineering methods have also been proposed to tackle LFAs. Gkounis et al. propose that Traffic Engineering can efficiently expose the attack [16]. Liaskos et al. proposed that by using Traffic Engineering, attackers have to adopt a suspicious behavior to keep their attack, which makes themselves reveal their presence [20].

Game theory has also provided a lot of inspiration for defending against LFAs. Ma et al. proposed a randomized approach based on Stackelberg security game to optimize
X. ConcluSion

Despite making considerable research progresses regarding LFAs, existing studies do not consider attack interference and strike precision which might restrict the applicability of LFAs from the adversary’s standpoint. We take the first step to take into account these issues, and model LFAs from both the attack and defense perspectives. For rational adversaries concerning attack stealthiness and persistence, attack interference and strike precision would be big concerns, especially in the long run. Using real-world traceroute data, we demonstrate that current link flooding strategies that only seek strike efficiency (i.e., interrupting more routes by flooding fewer links) may result in poor strike precision, severely interfering the connectivity of the networks surrounding the intended victim network. The proposed hybrid strategy improves strike precision significantly at the cost of limited degradation of strike efficiency. From the perspective of defenses, we study attack intention, i.e., inversely inferring the attack intention (i.e., the target network to disconnect) based on the identified attacked links. Furthermore, we consider a strong defender who employs traffic engineering to mitigate LFAs, and formulate the game-theoretic interactions between attackers and defenders. Our formulation demonstrates that LFAs can be effectively mitigated based on traffic engineering from a game-theoretic perspective. Our study also reveals that practical issues such as light-weight probe deployment and multi-protocol-based measurement are important in non-cooperative defenses.

REFERENCES

[1] X. Luo and R. K. Chang, “Optimizing the pulsing denial-of-service attacks,” in Proc. IEEE DSN, 2005, pp. 582–591.

[2] ———, “On a new class of pulsing denial-of-service attacks and the defense,” in Proc. NDSS, 2005.

[3] Y. Tang, X. Luo, Q. Hua, and R. K. Chang, “Modeling the vulnerability of feedback-control based internet services to low-rate dos attacks,” IEEE Transactions on Information Forensics and Security, vol. 9, no. 3, pp. 339–353, 2014.

[4] C. Wang, T. T. Miu, X. Luo, and J. Wang, “Skyshield: A sketch-based defense system against application layer ddos attacks,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 3, pp. 559–573, 2018.

[5] L. Xue, X. Ma, X. Luo, E. W. Chan, T. T. Miu, and G. Gu, “Linkscope: Towards detecting target link flooding attacks,” IEEE Transactions on Information Forensics and Security, pp. 2423–2438, 2018.

[6] L. Xue, X. Luo, E. W. Chan, and X. Zhan, “Towards detecting target link flooding attack,” in Proc. USENIX LISA, 2014.

[7] X. Ma, J. Li, Y. Tang, B. An, and X. Guan, “Protecting internet infrastructure against link flooding attacks: A techno-economic perspective,” Information Sciences, 2018.

[8] P. Bright, “Can a ddos break the internet?” http://goo.gl/m6XJx, 2013.

[9] A. Studer and A. Perrig, “The coremelt attack,” in Proc. Springer ESORICS, 2009, pp. 37–52.

[10] M. S. Kang, S. B. Lee, and V. D. Gligor, “The crossfire attack,” in Proc. IEEE S&P, 2013.

[11] T. Hraymenta, K. Toyoda, and I. Sasase, “Fast target link flooding attack detection scheme by analyzing traceroute packets flow,” in Proc. IEEE WIFS, 2015.

[12] L. Wang, Q. Li, Y. Jiang, and J. Wu, “Towards mitigating link flooding attack via incremental SDN deployment,” in Proc. IEEE ISCC, 2016.

[13] M. S. Kang and V. D. Gligor, “Routing bottlenecks in the internet: Causes, exploits, and countermeasures,” in Proc. ACM SIGSAC, 2014.

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

[15] S. B. Lee, M. S. Kang, and V. D. Gligor, “Codef: collaborative defense against large-scale link-flooding attacks,” in Proc. ACM CoNEXT, 2013.

[16] D. Gkouis, V. Kotronis, C. Liaskos, and V. Dimitropoulos, “On the interplay of link-flooding attacks and traffic engineering,” ACM SIGCOMM Computer Communication Review, vol. 46, no. 2, pp. 5–11, 2016.

[17] J. Peng, X. Ma, J. Li, L. Xue, and W. Hua, “Shoot at a pigeon and kill a crow: On strike precision of link flooding attacks,” in Proc. NSS, 2018.

[18] M. Pióro and D. Medhi, Routing, flow, and capacity design in communication and computer networks. Elsevier, 2004.

[19] S. Balon, F. Skivée, and G. Leduc, “How well do traffic engineering objective functions meet te requirements?” in International Conference on Research in Networking, 2006.

[20] C. Liaskos, V. Kotronis, and V. Dimitropoulos, “A novel framework for modeling and mitigating distributed link flooding attacks,” in Proc. IEEE INFOCOM, 2016.

[21] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing ospf weights,” in Proc. IEEE INFOCOM, 2000.

[22] A. Lipowski and D. Lipowska, “Roulette-wheel selection via stochastic acceptance,” CoRR, abs/1109.3627, 2011.

[23] P. L. Stoffa and M. K. Sen, “Nonlinear multiparameter optimization using genetic algorithms: Inversion of plane-wave seismograms,” Geophysics, vol. 56, no. 11, pp. 1794–1810, 1991.

[24] J. Jing, M. Li-dong, L. Shu-ling, and J. Lin, “Simulation research based on a self-adaptive genetic algorithm,” in Proc. ICEIS, 2010.

[25] A. K. Qin and P. N. Suganthan, “Self-adaptive differential evolution algorithm for numerical optimization,” in Proc. 2005 IEEE congress on evolutionary computation, vol. 2. IEEE, 2005, pp. 1785–1791.

[26] K. Deb and H.-G. Beyer, “Self-adaptive genetic algorithms with simulated binary crossover,” Evolutionary computation, vol. 9, no. 2, pp. 197–221, 2001.

[27] W. Willinger and M. Roughan, “Internet topology research redux,” in Proc. ACM SIGCOMM Computer Communication Review, vol. 35, no. 5, pp. 71–74, 2005.

[28] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis et al., “Understanding the mirai botnet,” in Proc. USENIX Security, 2017.

[29] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, “Ddos in the ion: Mirai and other botnets,” Computer, vol. 50, no. 7, pp. 80–84, 2017.

[30] X. Ma, B. An, M. Zhao, X. Luo, L. Xue, Z. Li, T. Miu, and X. Guan, “Randomized security patrolling for link flooding attack detection,” IEEE Transactions on Dependable and Secure Computing, 2019.

[31] A. Aydeger, M. H. Manshaei, M. A. Rahman, and K. Akkaya, “Strategic defense against stealthy link flooding attacks: A signaling game approach,” arXiv preprint arXiv:1912.10073, 2019.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

XIN WANG is a master student with MOE Key Lab for Intelligent Networks and Network Security, Faculty of Electronic and Information Engineering, Xi’an Jiaotong University, Xi’an, China. He received his bachelor’s degree in Computer Science and Technology at China Agricultural University in 2020. His current research focuses on Internet routing measurement and analysis.
XIAOBO MA is an associate professor with MOE Key Lab for Intelligent Networks and Network Security, Faculty of Electronic and Information Engineering, Xi’an Jiaotong University, where he received a Ph.D. in Control Science and Engineering in 2014. He was a Post-Doctoral Research Fellow with The Hong Kong Polytechnic University. He is an XJTU Tang Scholar and a member of IEEE. His research interests include Internet measurement and cyber security.

JIAHAO PENG was a master student (when finishing this work) with MOE Key Lab for Intelligent Networks and Network Security, Faculty of Electronic and Information Engineering, Xi’an Jiaotong University, Xi’an, China. He received his master’s degree in Cyber Science and Engineering in 2021 at Xi’an Jiaotong University. His research focuses on Internet measurement and security.

JIANFENG LI received a Ph.D. in Control Science and Engineering from Xi’an Jiaotong University in 2018, and he was with MOE Key Lab for Intelligent Networks and Network Security, Faculty of Electronic and Information Engineering, Xi’an Jiaotong University, Xi’an, China. He is currently a Post-Doctoral Research Fellow with The Hong Kong Polytechnic University. His research interests include network security and privacy.

LEI XUE is a research assistant professor in Department of Computing at The Hong Kong Polytechnic University, where he also received the Ph.D. degree in Computer Science. He also interned in Qihoo360 during his Ph.D. study. Currently, his research topics mainly focus on mobile and IoT system security, program analysis, and automotive security.

WENJUN HU is a research manager with Palo Alto Networks where he leads a data driven security team. He received his master’s and bachelor’s degree in Electrical and Electronics Engineering from Xi’an Jiaotong University. His research interests include system security, data security and machine learning.