You Can Obfuscate, but You Cannot Hide: CrossPoint Attacks against Network Topology Obfuscation

Xuanbo Huang[†], Kaiping Xue[†] [*], Lutong Chen[†], Mingrui Ai[†],
Huancheng Zhou[‡], Bo Luo[§], Guofei Gu[‡], Qibin Sun[†]
[†] *University of Science and Technology of China,*
{hxb777,lutong98,amr2016}@mail.ustc.edu.cn; {kpxue,qibinsun}@ustc.edu.cn
[‡] *Texas A&M University, hczhou@tamu.edu; guofei@cse.tamu.edu*
[§] *The University of Kansas, bluo@ku.edu*

Abstract

Link-flooding attacks (LFAs) may disrupt Internet connections in targeted areas by flooding specific links. One effective mitigation strategy against these attacks is network topology obfuscation (NTO), which aims to obscure the network map and conceal critical links, preventing attackers from identifying bottleneck links.

However, we argue that the attackers can still discover critical links in the presence of NTO defenses. In this paper, we introduce the CrossPoint attacks to escape the security protections of state-of-the-art NTO defenses by exploiting two network traffic features: *correlated congestion* and *statistical disparities*. Although NTO defenses create a complex and seemingly robust virtual topology, distinct information is still discoverable due to conflicting design objectives and inherent features of the Internet, resulting in novel side channels. Through comprehensive experiments, including a measurement study on the Internet, we demonstrate CrossPoint attacks' high success rate (80%-95%), minor overhead (10%-20%), as well as attack stealthiness and feasibility.

1 Introduction

Large-scale botnet-driven distributed denial of service (DDoS) attacks continue to pose a significant threat to the Internet [2,14,15,31,42]. Despite the prevalence of end-host DDoS defenses, link-flooding attacks (LFAs) remain a substantial threat to the robustness of the Internet [19, 35]. LFAs aim to overwhelm network infrastructure, such as in-line routers and servers, rendering end-host defenses irrelevant. In recent years, LFAs have caused severe damage. For instance, a large-scale LFA in 2013 attacked 4 Internet exchange points between Europe and Asia, leading to numerous sites being offline for several hours [3, 25]. In 2022, Ithaca College's network infrastructure fell victim to an LFA [7]. CloudFlare's statistics from the same year indicated a 109% year-over-year increase in DDoS attacks targeting network infrastructure [2], underscoring the growing concern over such attacks.

To efficiently launch an LFA, the adversary needs to know about the topology of the target network. Without this knowledge, an attacker can only "guess" which flows share a common link, considerably reducing the attack efficiency. Nethide [29] demonstrates that such "guess" attacks require 5 times more flows, rendering them less practical. However, once attackers acquire knowledge of the network topology, they are equipped to disrupt a considerable volume of legitimate communication. For instance, the Crossfire attack [19] demonstrates the potential of topology-probing LFAs against critical infrastructure across US regions, potentially disabling up to 53% of Internet connections in some states. Therefore, LFAs consist of a probing stage followed by an attack stage [19,35]. In the probing stage, attackers use reconnaissance tools to gather knowledge of the network topology and identify bottleneck links that are potential targets. During the attack stage, the attacker coordinates attack flows to flood critical links.

In response, researchers have developed *proactive defenses* against attackers during the probing stage [13,21,22,29], as well as *reactive defenses* to mitigate LFAs during the attacking stage [9,18,25,26,32,34,36,38,39,41,44].

In recent years, *proactive defenses*, also known as automatic moving target defense (AMTD), have gained increasing popularity and attention in academia and industry [28]. According to a Gartner report [28], AMTD is recognized for its potential to significantly bolster security through a "defense in depth" strategy. In the industry, several vendors (*e.g.*, R6security, Dispel) offer AMTD for dynamic reconfiguration of network infrastructure, topology, and configuration to enhance the resilience of networks against various cyber threats [6]. AMTD is also deployed in critical systems [27]. In academia, researchers have proposed dynamic network topology obfuscation (NTO) defenses against LFAs [13,21,22,29]. NTO is an example of AMTD, which manipulates virtual topology to obscure the true network topology and conceal bottleneck links. State-of-the-art (SOTA) NTO techniques, such as NetHide [29] and EqualNet [21], provide strong secu-

[*]Corresponding author: Kaiping Xue, kpxue@ustc.edu.cn

rity guarantees by reducing the attacker's success rate to less than 1%, even when attackers have substantial resources (2-4 times that of a normal LFA).

Although SOTA NTO defenses [13, 21, 22, 29] can create complex and robust virtual topologies to confuse attackers, it is still possible to break the defenses. One intuitive approach is randomly selecting attack flows, such as blind or bottleneck random attacks [29]. However, these attacks are known to require substantial budgets against recent NTO defenses and have low success rates [29]. Another potential attack is to exploit hardware fingerprints to identify target links [21, 29]. Nevertheless, SOTA defenses are known to defeat similar attacks efficiently [21]. To our best knowledge, there does not exist an attack in the literature that achieves a high success rate with a low cost.

In this paper, we propose CrossPoint attacks that exploit novel side channels to escape SOTA NTO defenses effectively [13, 21, 22, 29]. We argue that some inherent lower-layer features of Internet traffic, such as propagation delay (a static feature) and congestion (a dynamic feature), cannot be completely concealed by higher-layer obfuscation mechanisms. Moreover, the *usability* objective [21, 29] of NTO schemes that require link failures to be correctly located may be employed to compromise link identity. A well-crafted attack may exploit such inherent and robust network features and the *usability* functions to compromise the security promises of NTO defenses and identify critical attack targets.

However, developing such a well-crafted attack is nontrivial. First, since attackers do not have any prior knowledge of the network, they cannot utilize static features, such as propagation delay, to distinguish virtual or physical links. Second, since the congestion probing tools are end-to-end, attackers cannot pinpoint the exact location of congestion and obtain knowledge of physical links by only measuring congestion. In this paper, we demonstrate that attackers can tackle these challenges by introducing statistical and correlation analysis.

First, we present an approach to identifying NTO-crafted virtual links by exploiting lower-layer static network features through *statistical disparities*. Although lower-layer characteristics, such as propagation delay, are difficult to exploit without prior knowledge, we demonstrate that, due to the usability properties of NTO schemes, attackers with sufficient bots can identify features of NTO-crafted links that are inconsistent with their inherent lower-layer characteristics. These inconsistencies allow attackers to pinpoint virtual links that appear more suspicious than others.

Second, we present an approach to discover physical links by analyzing *correlated congestion*. Since NTO defenses do not change underlying routing paths [13, 21, 22, 29], attackers can observe and measure Internet congestion to identify artificially concealed physical links. Figure 1 shows an example of a six-node topology, where link l_{BC} is the attacker's target. We demonstrate Nethide's virtual topology [29] with blue dashed lines, which protects the bottleneck link l_{BC} with two independent routes (A-C-D and E-B-F). When congestion occurs on l_{BC}, highly correlated RTT increases and packet loss indicates that the two parallel virtual paths utilize the same physical link. Nonetheless, the challenge arises because congestion on other links may interfere with the correlation analysis. To tackle this issue, we introduce a *control group* to filter out non-targeted congestion, allowing us to isolate and identify the specific congestion associated with the target link. Subsequently, attackers can calculate the Pearson correlation coefficient (PCC) matrix [10] to measure the similarities among congestion samples and identify the physical target links shared by each unknown path.

Essentially, the CrossPoint attacks exploit two side channels (inherent network features and congestion) to challenge NTO defenses under a black-box network, where the attacker only uses `traceroute` and `ping` as reconnaissance tools. With extensive experiments, we evaluate the performance of the CrossPoint attacks and show that they achieve high success rates (80% - 95%) in identifying bottleneck links at a small extra cost (10%-20%), even though the network is protected by SOTA NTO defenses [21, 29]. Moreover, even though we cannot test this attack on the Internet, we have designed a measurement study to demonstrate the feasibility and stealthiness of the CrossPoint attacks on the Internet. To pave the way for improving NTO defenses, we also discuss countermeasures and future NTO directions.

Contributions To summarize, this paper makes the following contributions:

1. We identify two fundamental weaknesses of the current NTO design: first, the routing compatibility leaks information regarding physical paths, and second, the usability features discover information about virtual links.
2. We present the CrossPoint attacks against NTO defenses. The attacks exploit *usability* with *statistical disparities* to infer virtual links and discover hidden links with *correlated congestion* of physical paths.
3. We demonstrate the effectiveness of CrossPoint attacks against state-of-the-art NTO defenses through comprehensive experiments, which show a high success rate of 80-95% with a low overhead of 10-20%.
4. Our large-scale measurement study demonstrates the feasibility and stealthiness of the CrossPoint attacks.

Ethical Considerations. This research aims to evaluate the effectiveness of SOTA NTO defenses in practice. We never attacked any real-world network infrastructures outside of the lab. Our experiments did not introduce any pressure on any ISP links. We use cloud servers in experiments (Section 6.2) to introduce real Internet noise instead of sending attack flows. Our Internet measurement study (Section 6.5) investigates the feasibility of the CrossPoint attacks with low-rate, benign probes of 0.36 KB/s ICMP flows.

The rest of this paper is structured as follows: Section 2 outlines background and related works. Section 3 contains the

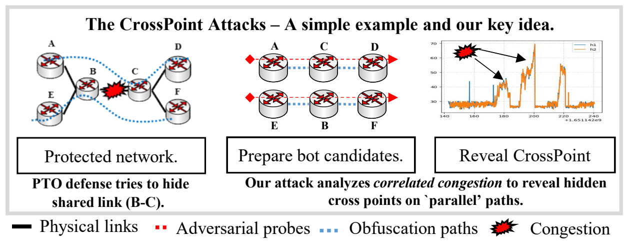
The CrossPoint Attacks – A simple example and our key idea.
A
B
C
D
E
F
Protected network.
PTO defense tries to hide shared link (B-C).
Prepare bot candidates.
Reveal CrossPoint
Our attack analyzes *correlated congestion* to reveal hidden cross points on `parallel' paths.
Physical links
Adversarial probes
Obfuscation paths
Congestion

Figure 1: A CrossPoint attacker discovers the hidden shared link by analyzing correlated congestion. We use the same instance with Nethide [29].

threat model and glossary. Section 4 analyzes the weakness of SOTA NTO defenses. Section 5 introduces CrossPoint attacks. Section 6 displays the evaluation results, and Section 7 discusses countermeasures. Section 8 gives discussions. Finally, we conclude in Section 9.

2 Background and Related Works

Link-flooding attacks [19,35] aim to target a limited number of critical links in a network, thereby inflicting substantial damage within a specific network area. For instance, Kang et al. [19] showcased how Crossfire attacks [19] could significantly disrupt over 53% of communication in some states by selecting approximately 20 critical links as targets. Consequently, the selection of critical targets within large-scale networks assumes paramount importance. To this end, Crossfire attacks employ probing tools like `traceroute` to estimate the importance of network links. Attackers calculate the *flow density* of each link by aggregating data from large-scale botnet probes utilizing `traceroute`. The concept of *flow density* is defined as the quantification of bot traffic transmitted through a particular link, which can be used to estimate the link's routing popularity within the network. Notably, links characterized by high flow density are considered prime targets for LFAs [19]. However, Crossfire attacks face limitations when confronted with SOTA NTO defenses.

Network topology obfuscation [13,21,22,29] manipulates `traceroute` responses to obfuscate the link's flow density to prevent attackers from finding profitable targets. This is achieved by strategically manipulating probing responses that present a virtual and resilient topology to potential attackers. In the virtual topology, attackers are left without a clear set of critical links to exploit. In essence, NTO schemes compel LFAs to transition from targeted critical-link attacks to random-target attacks. However, random-target attacks require a substantial budget and offer no guarantees on the effectiveness [29]. Another potential solution countering NTO schemes is exploiting the hardware fingerprints to discern virtual responses from physical ones. However, EqualNet [21] is engineered to produce virtual responses that are indistinguishable from their physical counterparts.

Nevertheless, this paper finds **two fundamental weaknesses** that can be exploited to bypass the obfuscation of SOTA NTO schemes. First, despite a robust virtual map being present, the lower-layer network traffic patterns can not be effectively hidden, allowing attackers to exploit such patterns to distinguish virtual responses. Second, NTO schemes have to provide the *usability* property for benign users, which degrades security when attackers launch statistic analysis. We hope these findings can help enhance the security of NTO schemes in practice.

3 Threat Model and Glossary

Threat model. We use the same threat model as previous attacks and NTO defenses [19,21,29,35]. In detail, the attacker has a limited number of bots located outside the target network to locate and congest at least one bottleneck link within the network. However, the attacker lacks prior knowledge about the network and can only employ limited reconnaissance tools, specifically `traceroute`, `ping`. It is assumed that the network does not prohibit normal ICMP requests and responses that are defined in RFC777 [30].

Most crucially, we assume that the network is protected by SOTA NTO defenses, such as Nethide [29] or EqualNet [21].

Additionally, we assume that at least one congestion event occurs on the target link. However, this assumption is optional and loose. If this condition cannot be satisfied, CrossPoint attacks can still work using only the statistical disparities technique without congestion information. Additionally, our measurement study indicates that most Internet paths experience a large number of congestion events, making it more likely that the bottleneck link will be affected.

Glossary. We give the glossary of definitions in this paper:

- **Paths and links.** A path is a flow route that may consist of multiple links.
- **Bot-path.** A bot can create different flows by altering destinations, and each flow corresponds to a dedicated routing path. In the following, we depict different bot-paths with $(b,p)_A, (b,p)_B, \cdots$. Under a NTO defense, the attacker needs to determine whether a bot-path conceals a profitable link.
- **Link capacity.** We assume that each bot-path sends the same volume of traffic during the attack stage. The link capacity can be represented by the number of available bot-paths passing through the link.
- **Flow density.** Flow density is a link attribute representing the number of bot-paths that pass through it. A link is considered profitable for the attacker if its flow density exceeds its link capacity.
- **Budget.** Budget is the maximum number of bot-paths that the attacker could employ in the attack stage.

The definitions of *link capacity*, *flow density*, and *budget* are the same as Nethide [29]. And the *bot-path* is considered as a *flow* in Nethide [29].

4 Security Analysis of NTO Defense

In this Section, we present the fundamental weaknesses of SOTA NTO defenses. Initially, we discuss how the *usability* property leads to a decline in *security*, ultimately resulting in *statistical disparities* in inherent and robust network features that help identify virtual links. Secondly, with unchanged routing paths, the observation of *correlated congestion* can assist in discovering physical links. These two weaknesses enable CrossPoint attackers to expose bottleneck links for flooding. In the following subsections, we delve into the respective root causes of these weaknesses.

4.1 The Statistical Disparities

The *Statistical disparities* refer to the static differential features between virtual and physical networks that higher-layer obfuscations cannot change. Although static features cannot be used to identify virtual or physical links directly, we have discovered a vulnerability stemming from the *usability* property of NTO schemes when attackers have sufficient bots to perform statistical analysis.

Security analysis on Nethide [29]. We briefly summarize the Nethide design detail for the next step of the analysis. Nethide formalizes *usability* as the optimization objective and formalizes *security* as hard constraints, represented by the optimization problem:

$$\begin{aligned} Max. \sum_{f \in F} (w_1 Acc(f) + w_2 Uti(f)) \\ s.t. \quad Security\ constraints, \end{aligned}$$

where f is a bot-path (a *flow* in Nethide). $Acc(f)$ calculates the similarity between virtual and physical paths, representing the accuracy of using virtual links to make routing decisions, while $Uti(f)$ calculates the probability of correctly locating link failures with virtual paths. The security constraints are solid requirements that some links must be concealed.

In SOTA NTO defenses [21,29], the *security* property takes precedence over *usability*. And the *usability* is achieved as a best-effort function. However, achieving the optimal *usability* objective can lead to undesired outcomes: the *security* is compromised. To comprehend this, it is essential to describe the approach for attaining the highest *usability* objective while ensuring *security*. For example, consider a scenario with three links, A, B, and C, where link B is a bottleneck and must be hidden. Without the *usability* objective, defense schemes are free to conceal any links in A, B, and C, or all of them, to provide good obfuscations. However, when under the objective to maximize the *usability* functions, *i.e.,* $Acc(f)$ and $Uti(f)$, hiding any links leads to these two objective decreases monotonically [29]. That is because removing any physical links in the network leads to inaccuracies in making routing decisions and locating link failures. Therefore, the best choice to maximize the *usability* is to hide link B without A and C, since it maintains the highest accuracy in terms of $Acc(f)$ and $Uti(f)$ for the other two links. To summarize, concealing non-bottleneck links like A and C does not contribute to meeting security constraints but decreases the *usability* objective. Therefore, the solver of the designed model tends to conceal the least number of links to satisfy the *security* constraints. Thus, our first observation is that the *usability* objective results in virtual links having a high probability of concealing bottleneck links, making identifying virtual links crucial in discovering critical targets.

Furthermore, since this design tends to conceal the least number of bottleneck links, an attacker can identify inherent features that show different statistical results from others. In practice, Nethide omits important nodes to offer near-optimal solutions, which means virtual paths appear shorter than physical paths in `traceroute` responses. However, the unchanged propagation delay may discover the actual length of the path. For example, empirical evidence shows that a long propagation delay indicates a long Internet path [5], but the probing results show the long path contains only a few nodes. Consequently, the attacker can guess that the path has been obfuscated. Nonetheless, since the network is a black box, it is challenging to establish the relationship between path length and propagation delay and determine whether the path is virtual or physical.

However, when attackers has sufficient bots to launch statistical analysis, they can identify relatively suspicious virtual links. In the case of path lengths and propagation delay, the attacker calculates *one-hop-delay* defined as $RTT/Hops$, indicating the relationship between propagation delay and the number of nodes along the paths. Even though the attacker does not know the actual relationship between path length and propagation delay in the target network, they can extract paths that are more suspicious than others (long propagation delay but few hops). Figure 2 depicts the statistical results of the $RTT/Hops$ attribute in 47 different topologies with realistic RTT settings (refer to Section 6.1 for details). The left Y-axis denotes the CCDF of $RTT/Hops$ between virtual and physical paths, while the right Y-axis shows the attacker's advantage defined as a Bayes probability $P(link == virtual|X \geq k)$, which increases as $RTT/Hops$ increases. Therefore, it is possible to extract some of the virtual links against Nethide.

Security analysis on EqualNet [21]. Similar to Nethide, EqualNet provides best-effort *usability* to enable benign users to debug link failures within a virtual topology effectively. However, the difference is that EqualNet guarantees accurate link failure within a subnet. To achieve this, EqualNet randomly assigns virtual nodes IP addresses within the same subnet as their physical nodes [21]. Nevertheless, similar to how propagation delay leaks the real path length, subnet-level IP addresses may also unintentionally expose information

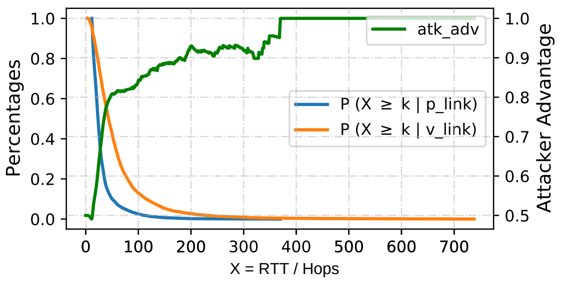

Figure 2: Statistical disparities of the physical propagation delay / virtual path length against Nethide in 47 different topologies. The attacker's advantage is a Bayes probability defined as $P(link == virtual|X \geq k)$.

about virtual links as a static feature. In this case, CrossPoint attackers treat all virtual nodes that have the same subnet prefix as one physical node. While this approach aggregates two physical nodes that share the same subnet prefix erroneously, it is enough to give a coarse-grained identification of virtual nodes and links, which can be used in the next step with *correlated congestion* to discover physical links accurately.

4.2 The Correlated Congestion

As the underlying routing paths remain unchanged under NTO defenses, it is possible to use information from lower-layer routing to identify links [13, 21, 22, 29]. In this study, we leverage correlated congestion to extract physical link information under NTO defenses.

Network congestion occurs when data transmission demands surpass the available bandwidth of a device, leading to significant delays and the possibility of packet loss [8]. It is rare for unrelated congestion events from different network locations to demonstrate similar congestion characteristics, due to the multifaceted nature of typical congestion determinants, such as unpredictable onset times, varying capacities of output queues, the arbitrary magnitude of burst traffic, and the diversity in router buffer sizes. Thus, congestion events with highly similar features observed at two points may indicate a common source, such as the same network device. This phenomenon, known as correlated congestion, serves as a valuable tool in pinpointing bottleneck links within a network. Many congestion control efforts already utilize correlated congestion to enhance performance, such as applications of multipath-TCP [23, 33, 40]. Therefore, it is possible to reveal physical links through correlated congestion events.

5 CrossPoint Attacks

5.1 Attack Overview

We assume an attacker has n bot-paths (defined in the glossary), and the target link's capacity is c, where $c < n$. The classic Crossfire attacks [19] can select sufficient bot-paths passing through the target with `traceroute` reconnaissance in the absence of protections. However, NTO defenses [21,29] obfuscate the `traceroute` responses to prevent Crossfire attacks. As a result, a Crossfire attacker will find there are only n' ($n' << c$) bot-paths passing through each link, making it impossible to flood any links with sufficient bot-paths. Therefore, the goal of our attack is to identify the remaining $c-n'$ bot-paths passing through the target.

The straightforward method is to randomly select the remaining bot-paths as *blind* or *bottleneck random* attacks [21, 29]. However, they are ineffective due to the protections of NTO defenses [29], achieving less than 1% success rate with 400% of the budget. Another potential method is to utilize hardware fingerprints to identify virtual and physical links. However, state-of-the-art NTO defense [21] can completely defend against such attacks.

CrossPoint attacks identify the remaining $c-n'$ bot-paths with *statistical disparities* and *correlated congestion*. Figure 3 shows an overview of our attacks. First, the attacker collects basic routes and delay information on the virtual map. Second, the attacker makes a coarse-grained classification of bot-paths based on *statistical disparities* to infer virtual bot-paths (*i.e.,* it hides something, potentially the target). Third, the classified virtual bot-paths are used to observe *correlated congestion*. In this step, we introduce *control group* to judge whether congestion happens on the target. If targeted congestion is found, the RTT samples are sent to other bot-paths, and correlations are calculated. A high correlation from a virtual bot-path means that it has a high probability of hiding the target link [23, 33, 40], making it suitable for flooding the target link.

To summarize, CrossPoint attacks have the following steps:

1. **Attack preparation.** The attacker has a bot-path set $S = \{(b,p)_A, (b,p)_B, \cdots\}$. To begin the attack, each bot collects basic information such as propagation delay and routing paths, to construct the (virtual) link map.
2. **Detecting virtual links with statistical disparities.** The attacker utilizes static features and calculates statistical results, then the attacker outputs the virtual bot-path set S_{sort}, where relatively more suspicious bot-paths are prioritized over others.
3. **Identifying physical links with correlated congestion.** The attacker coordinates selective bot-paths in S_{sort} to observe congestion and calculate correlations, clustering the bot-paths that share the same physical links and output the bot-path set S_{attack} for each potential target.

The CrossPoint attacks have two variants: the dedicated use of *statistical disparities* or the *correlated congestion*. The first variant, named the CrossPoint-SD attack, involves steps (1) and (2). This attack utilizes random suspicious bots to launch attacks. On the other hand, the CrossPoint-CC attack involves steps (1) and (3). This attack employs random bots to observe congestion. To achieve optimal performance, the CrossPoint-SD/CC attack combines both the SD and CC side

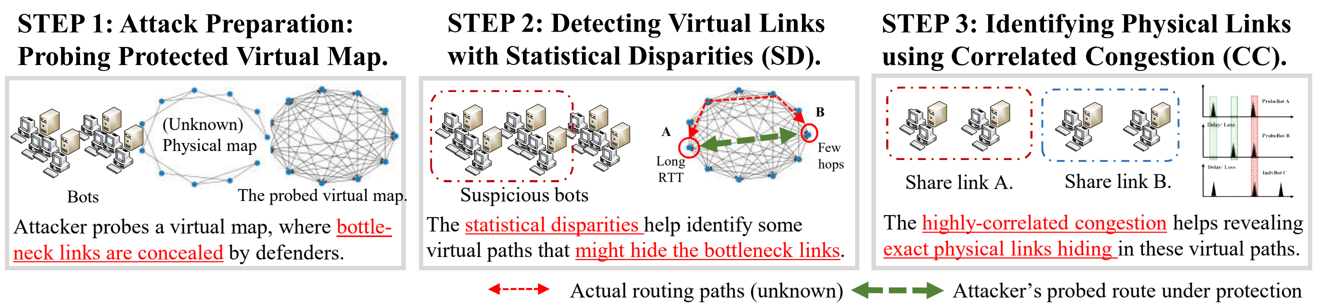
STEP 1: Attack Preparation: Probing Protected Virtual Map.
(Unknown) Physical map
Bots
The probed virtual map.
Attacker probes a virtual map, where bottle-neck links are concealed by defenders.
STEP 2: Detecting Virtual Links with Statistical Disparities (SD).
Suspicious bots
A
B
Long RTT
Few hops
The statistical disparities help identify some virtual paths that might hide the bottleneck links.
STEP 3: Identifying Physical Links using Correlated Congestion (CC).
Share link A.
Share link B.
The highly-correlated congestion helps revealing exact physical links hiding in these virtual paths.
Actual routing paths (unknown)
Attacker's probed route under protection

Figure 3: CrossPoint attacks consist of three steps. Firstly, each bot-path collects virtual routing paths and the propagation delay. Then, the attacker performs coarse-grained classification on virtual paths using statistical disparities. Finally, the attacker determines the attack bot-paths using correlated congestion.

channels with steps (1), (2), and (3).

5.2 Step 1: Attack Preparation

This step is the same as the Crossfire attack [19]. During this phase, each bot gathers essential information, including routing data, propagation delays, and available bandwidths, to prepare for attacks. To start this process, each bot sends `traceroute` probes to various potential destinations (other bots or public servers). The `traceroute` results are a series of IP addresses assigned to routers' interfaces, establishing a (virtual) map of network links. For each destination, bots can gather their available bandwidth through tools like `Pathneck` [17]. Next, each bot sends a series of `ping` messages (typically 10-20) to measure the propagation delay of each path. Then, the attacker computes the flow density for each network link and organizes them in descending order of flow density magnitude. Without NTO defenses, attackers can select the top-k critical links with the highest flow densities for their attacks. However, when NTO schemes are in place, the collected link map is virtual, and the top-k links may not be 'critical'. Consequently, CrossPoint attacks proceed to uncover natural critical links through subsequent stages.

5.3 Step 2: Detecting Virtual Links

CrossPoint attacks utilize the inherent features of the Internet (*e.g.* propagation delay, subnet addresses) to identify some virtual links crafted by NTO defenses. Since attackers face an unknown network without any prior knowledge, they cannot determine the value of these features that indicate whether a link is virtual or physical. As discussed in Section 4, CrossPoint attacks extract relatively more suspicious bot-paths with statistical analysis to identify virtual links.

Identify virtual bot-paths against Nethide [29]. For each bot-path denoted as $(b,p)_i$, we have the propagation delay represented as t_i, and the length of the `traceroute` path represented as h_i. As elaborated in Section 4, we employ the concept of the one-hop delay to estimate the relative likelihood of a path being virtual. To compute the one-hop delay for each bot-path, we use the formula t_i/h_i (propagation delay/number of hops). Subsequently, the attacker arranges all bot-paths in a descending order based on the magnitude of their respective one-hop delays. This results in the formation of a sorted set denoted as S_{sort}, where bot-paths with higher one-hop delays are consequently considered more suspicious.

Identify virtual bot-paths against EqualNet [21]. For each bot-path denoted as $(b,p)_i$, we employ the concept of *bot-path IP similarity* as a metric to measure the likelihood of concealing bottlenecks. First, we define *address similarity* as the count of IP addresses with the same subnet. Subsequently, the *bot-path IP similarity* is determined as the maximum *address similarity* observed within the routing paths. Following the computation of IP similarity for all bot-paths, we subsequently arrange the botnet set, denoted as S_{sort}, in a descending order based on their IP similarity values, which furnishes a descending sequence that reflects the likelihood of hiding bottlenecks in the virtual paths.

Finally, the attacker outputs S_{sort}, where the suspicious bot-paths are prioritized over others. Attackers can use these bot-paths in the next step to examine the critical links.

5.4 Step 3: Identifying Physical Links

Once the suspicious bot-paths containing virtual links have been identified, the next step is to identify the hidden physical links. As discussed in Section 4, NTO defenses only manipulate `traceroute` responses without altering the underlying routing paths. Therefore, our attacks utilize congestion in physical routing paths to expose hidden links. Specifically, *correlated congestion* on different paths can indicate that they share a common bottleneck [23,33,40]. As Figure 3 shows, attackers can cluster the unknown bot-paths with the correlation analysis of observed congestion samples.

However, the congestion location is a challenge. First, the `ping` probes are end-to-end, making it difficult to pinpoint the link where congestion occurs. Second, simultaneous congestion could happen on different links, which may pollute the correlation results. Therefore, CrossPoint attacks introduce the concept of *control groups* to help locate congestion.

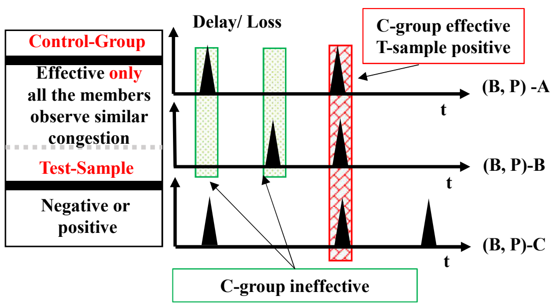

Figure 4: Example of using C-groups to distinguish congestion from a certain link.

Locate the congestion. To address this congestion location problem, we introduce a concept named *control group* comprised of bot-paths that share the minimal number of links necessary to monitor congestion on specific links. Figure 4 illustrates our idea, in which we assume that the routing paths of $(b,p)_A$ and $(b,p)_B$ intersect at a specific link. The objective for attackers is to determine whether the $(b,p)_C$ observed congestion also occurs on this particular link. To achieve this, we leverage $(b,p)_A$ and $(b,p)_B$ to establish a control group, essentially serving as a baseline reference. If congestion occurs on the target link, both $(b,p)_A$ and $(b,p)_B$ should experience the same *correlated congestion*. Consequently, if $(b,p)_C$ likewise observes this correlated congestion, it indicates that $(b,p)_C$ shares the same link with both $(b,p)_A$ and $(b,p)_B$.

In practice, two members in the control group may not be enough. Therefore, the attacker needs to establish a control group with the following rules:

1. Establish a basic control group with two known bot-paths that pass through the target link.
2. Augment the control group by introducing bot-paths that specifically incorporate the target link, to minimize the count of shared links.

Due to the *usability* design, SOTA defenses allow a certain number of bot-paths to retain their physical path, which aids the attacker in constructing control groups using the above-mentioned rules. For instance, Nethide [29] permits $n' < c$ bot-paths to pass through each link, where c is the link capacity, and n' is set to a sufficient value in practice to enhance *usability*. Similarly, EqualNet [21] utilizes virtual IP addresses belonging to the same subnet as their physical nodes, which can also be used to establish C-groups.

Observe the congestion and noise reduction. Since the Internet is noisy, we adopt a noise reduction technique that mitigates the impact of RTT jitters and Internet noise. Specifically, each bot sends `ping` traces at a certain speed (discuss in Section 6.5) and gains RTT traces. Then, we transform the RTT sequences into square waves while preserving critical features. We accomplish this by dividing constant RTT values into discrete buckets, with the lowest value representing propagation delay and the highest indicating packet loss. We define the congestion happening if an RTT sample increases more than 30% of its propagation delay.

Timestamp synchronization. Correctly analyzing congestion correlation necessitates time synchronization of each bot-path. However, the timestamp of the `ping` message is recorded when bot-paths receive ICMP responses, not precisely when congestion occurs. Therefore, we adjust the timestamp using the following correction:

$$t_{correct} = t_{timestamp} - RTT_{whole} + RTT_{target}/2,$$

where RTT_{target} is the delay between the bot and the target. This correction provides a more precise observation of targeted congestion events.

Quantify the correlation. We define *congestion correlation distance* to quantify the correlation between the congestion samples of $(b,p)_A$ and $(b,p)_B$. Specifically, we calculate the Pearson correlation coefficient (PCC) [10] between the two congestion samples, which measures how well the two samples are correlated. If the correlation coefficient is high, then the two samples are more likely to be derived from the same bottleneck.

When attackers observe congestion, they calculate PCC in the following steps. For a C-group containing n bot-paths, we denote the k-th bot-path's RTT sequence of congestion as $\mathbf{bp}_k(t), t \in [t_0, t_m]$. In detail, t_0 is the first RTT of a congestion sample, and t_m is the last. Therefore, all the bot-paths' RTT sequences are:

$$\mathbf{X}(t) = [\mathbf{bp}_1(t), \mathbf{bp}_2(t) \cdots \mathbf{bp}_n(t)]^T.$$

Next, we calculate the self covariance matrix as

$$\mathbf{C_x} = E\{[\mathbf{X}(t) - \mu_x][\mathbf{X}(t) - \mu_x]^T\} = \begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{pmatrix},$$

where c_{ij} represents the covariance of $\mathbf{bp}_i$ and $\mathbf{bp}_j$, while μ_x is the mean vector of $\mathbf{X}(t)$. Next, we calculate the PCC matrix as:

$$\mathbf{R}_x = \begin{pmatrix} \frac{c_{11}}{\sigma_1^2} & \cdots & \frac{c_{1n}}{\sigma_1\sigma_n} \\ \vdots & \ddots & \vdots \\ \frac{c_{n1}}{\sigma_n\sigma_1} & \cdots & \frac{c_{nn}}{\sigma_n\sigma_n} \end{pmatrix} = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{n1} & \cdots & r_{nn} \end{pmatrix},$$

where σ_i is the standard deviation of $\mathbf{bp_i}$. Each element in $\mathbf{R}_x$ represents two bot-paths' RTT similarity during time t. To find effective congestion, we calculate a distance between the observed congestion and an ideal congestion sample as

$$D = \underset{i,j \le n}{Max} \ (\Delta_x - \mathbf{R}_x)_{ij},$$

where Δ_x is an all-ones matrix that represents all the PCC of each bot-path are 100% similar to each other. We choose

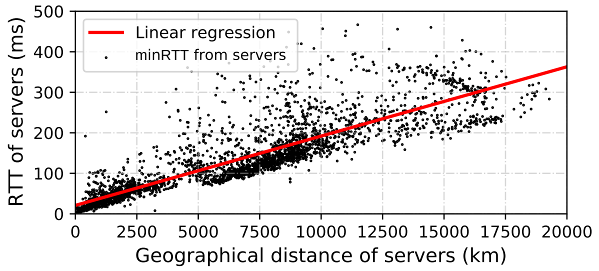

Figure 5: Linear regression of the real-world delay.

the maximum value in $(\Delta_x - \mathbf{R}_x)_{ij}$ as the distance because it describes how dissimilar it is from the ideal one.

Cluster the bot-paths. In determining whether a test sample conceals the target link, the attacker incorporates the test sample into the C-group and recalculates the distance, denoted as D, within the range of $[0,1]$. Subsequently, the attacker employs a threshold to assess the correlation between two instances of congestion. Drawing from the PCC definition [10], where PCC values within $[0,1]$ nearing 1 indicate a strong correlation, while values nearing 0 denote a lack of correlation, we establish a threshold at the midpoint, *i.e.,* 0.5, to determine whether two samples exhibit correlation.

Given that the distance D is inversely proportional to the PCC ($D \approx 1 - \text{PCC}$), if $D < 0.5$, it implies that the test sample traverses the target link. Conversely, if $D > 0.5$, the test sample does not traverse the target link. Given the C-group congestion traces, each bot-path can calculate the distance itself to know whether it passes through the target link. Then, each bot reports its available paths against a certain target to the attacker's command and control (C&C) server.

5.5 Implementation Details

This Section provides the implementation details of the CrossPoint bots and controllers.

Bot behavior. First, each bot sends a series of 10-20 `ping` and `traceroute` messages to determine the propagation delay and the obfuscated `traceroute` route. It then forwards the resulting `<src, dst, delay, route>` structure to the C&C server and waits for instruction. Upon receiving a command to observe congestion, the bot rapidly sends `ping` messages at a fixed rate of 10 packets per second with the shortest package length (36 bytes [30]). For each `ICMP_response` message, the bot extracts `<timestamp, icmp_seq, round_trip_time>` information and records them in a `csv` format file. The difference in `icmp_seq` values allows packet loss detection, while the round-trip-time facilitates congestion measurement. Then, if the bot belongs to a C-group, it transmits fractional `ping` traces containing a few seconds of correlated congestion to the C&C server. If the bot has suspicious paths, it receives fractional traces from C&C server and calculates the correlation. Finally, each bot reports available paths against the target to the C&C server.

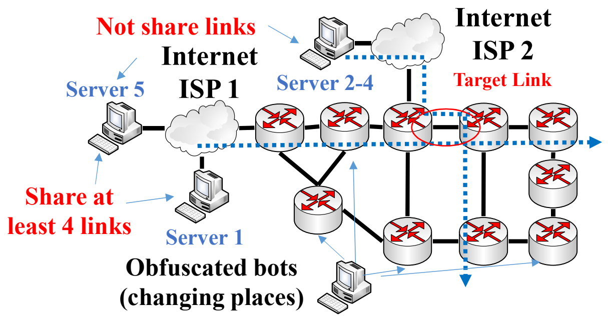

Figure 6: Testbed topology. Five servers from different cities are used to involve noise. Ten bots observe congestion.

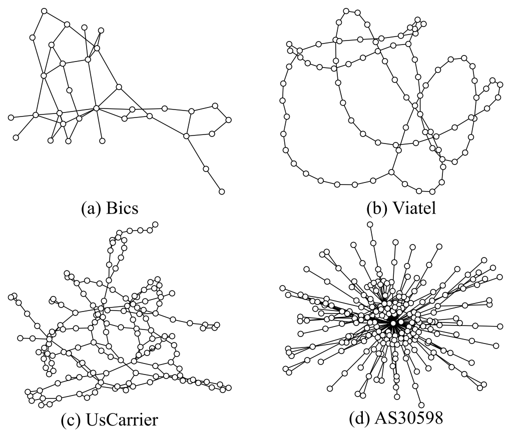

Figure 7: Simulation topologies: (a) Bics comprises 33 nodes and 48 links; (b) Viatel consists of 88 nodes and 92 links; (c) UsCarrier has 158 nodes and 189 links; (d) AS30598 contains 226 nodes and 314 links.

Controller behavior. Like conventional DDoS attacks, CrossPoint attacks are executed via botnets, necessitating C&C servers to coordinate the attack. The C&C server's interaction with the botnet encompasses several critical steps: it first collects probing data from the bots; then, using this data, it identifies the C-group (Section 5.4). Subsequently, the server dispatches the congestion traces from the C-group to each bot and awaits their confirmation on whether the targeted link is accessible. Once confirmation is received, the C&C server gives the final order for the bots to launch the attack.

6 Evaluation

Our evaluations consist of four parts. We assess the dedicated and combined performance of two attack variants and investigate the feasibility of CrossPoint attacks:

1. In Section 6.2, we construct a testbed to evaluate the performance of the *correlated congestion* attack (CrossPoint-CC). This evaluation involves real Internet noise and congestion from 5 servers on 2 continents.
2. In Section 6.3, we assess the single and overall performance of *statistical disparities*, *i.e.,* CrossPoint-SD attack) and CrossPoint-SD/CC attack, with the implemen-

tation of SOTA defenses [21, 29]. This evaluation considers a variety of network topologies, attack budgets, network robustness, and defense parameters.

3. In Section 6.4, we assess the scalability of CrossPoint attacks in different scaled network topologies. This evaluation contains 7 different topologies
4. In Section 6.5, we conduct a large-scale measurement study to investigate whether CrossPoint attacks can remain stealthy and be employed on the Internet.

We compare our attack with Coremelt [35], Crossfire [19], and Crossfire-Random attacks [29], against SOTA defenses including Nethide [29] and EqualNet [21].

6.1 Implementation

Topology. We select representative backbone networks from the Internet Topology Zoo [24] and CAIDA dataset [1], which are widely used in NTO evaluations [29]. The topologies and the corresponding attributes are shown in Figure 7.

Link delay and IP addresses. To emulate realistic link delays, we use the dataset from the Global-Ping-Statistics project [5], containing `ping` data from 247 different servers globally. We employ linear regression to fit the relationship between the geographical distance and the delays from the `ping` dataset, as illustrated in Figure 5. For each link in the testbed and simulations, we extract the longitude and latitude of the nodes, calculate the link delay based on the regression results, and then assign these delays using Linux traffic control in both the testbed and simulations. We assign IP addresses to each node using the CAIDA dataset's ITDK project [1].

Congestion. Congestion occurs when burst traffic exceeds the capacity of network devices [8, 40]. In our experiments, we replicate this process to ensure a realistic reproduction of congestion. To this end, we randomly launch each network node to send burst traffic. At certain times, this burst traffic competes for the bandwidth of the network devices, thereby reproducing short, unpredictable congestion events that mirror real-world congestion processes. Additionally, we also involve real Internet congestion as noise on our testbed by using cloud servers in Section 6.2.

Defenders. We implement Nethide [29] and EqualNet [21] using the OpenVswitch and the Ryu SDN controller to forward and respond to the `traceroute` messages. We use the prevalent shortest-path-first routing policy, in line with Nethide [29] and EqualNet [21].

Nethide [29] employs the optimal solution offered by the ILP solver [4], which promises a gap of less than 2%. We vary Nethide's *flow density reduction factor* parameter, representing different protection levels. EqualNet [21] uses the strictest *obfuscation threshold* ($\tau = 80\%$).

Attacker. In our experiments, the attacker has n^2 candidate attack bot-paths. The bots are uniformly distributed, in line with the Nethide configurations [29], and the defenders fully obfuscate all the bots and their paths. However, in each attack, the attacker can only use a limited number of bots due to budget constraints, which necessitates revealing sufficient bot-path information to counter NTO defenses.

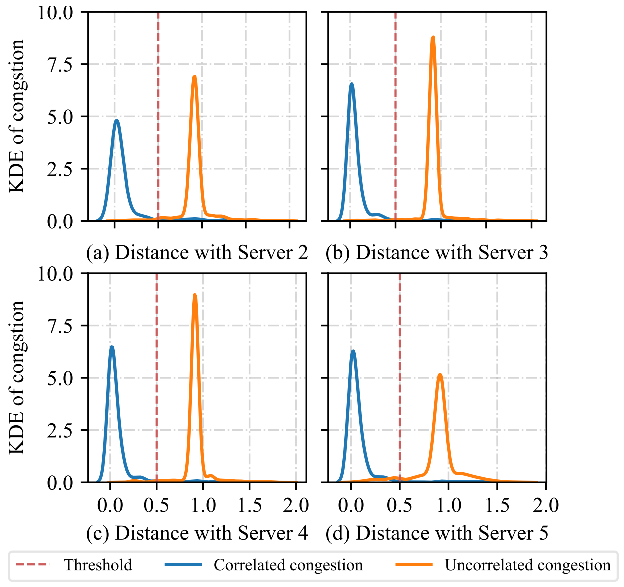

Figure 8: Kernel density estimation of two kinds of congestion on identifying the target link congestion. Figures (a), (b), (c), and (d) show the results of servers 2, 3, 4, and 5.

Table 1: Testbed experiment parameters.

SID*	Date (2022)	Duration	Noise Congestion
1	05-18 — 05-24	120.2 h	10K+
2	05-20 — 05-21	29.15 h	163
3	05-22 — 05-24	29.15 h	955
4	05-22 — 05-23	29.12 h	1384
5	05-18 — 05-19	29.12 h	3053

* SID = Server ID, Server 1 and 5 shares several links.

6.2 CrossPoint-CC Evaluations

Setup. This Section presents the evaluation of the effectiveness of the CrossPoint-CC attack. As depicted in Figure 6, we set up a testbed based on the Abilene topology, which includes several cloud servers and 10 bots for observing congestion. Notably, the cloud servers do not act as bots or generate additional traffic that could affect the ISP links. Table 1 summarizes the key details of our experiments. Specifically, a C-group consists of two servers, a fixed server 1 and a varying server 2-4, located in different regions to involve Internet noise. We also consider a scenario with a non-perfect control group establishment with servers 1 and 5 where overlapped ISP links more than the target are involved.

Performance of identifying shared links. To assess the attackers' effectiveness in identifying shared bottleneck links

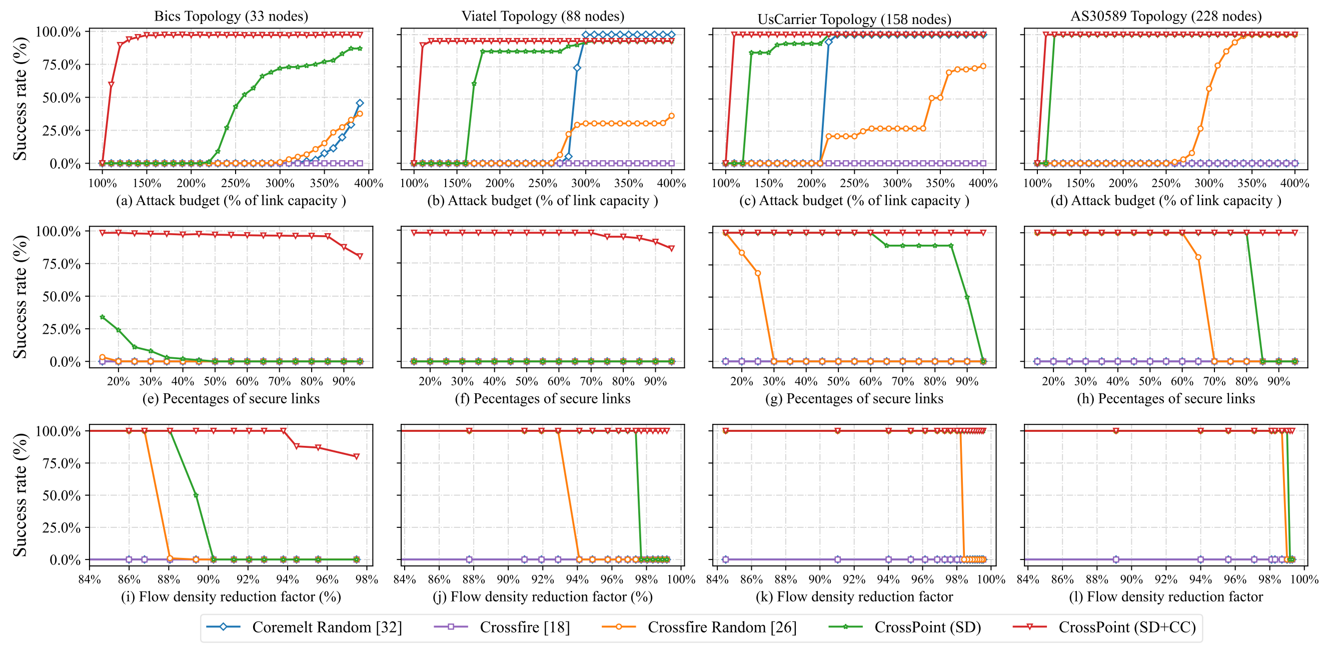

Figure 9: The CrossPoint attacks' effectiveness against Nethide [29] defense mechanism across four network topologies. Subfigures (a)-(d) illustrate the relationship between the attack success rate and the attack budget; subfigures (e)-(h) demonstrate how the success rate of CrossPoint attacks varies with the robustness of the network topology; subfigures (i)-(l) depict the success rate of CrossPoint attacks against varying levels of protection implemented by the Nethide defense.

Table 2: Congestion classification results

Metrics	Accuracy	Precision	Recall	F1 score
Fig.7(a)	96.2%	98.3%	94.4%	96.3%
Fig.7(b)	97.5%	98.4%	96.8%	97.6%
Fig.7(c)	97.7%	98.4%	97.3%	97.8%
Fig.7(d)	95.4%	94.2%	97.3%	95.7%

through correlated congestion, we analyze about 3000 independent data traces collected from a week-long observation of congestion events. Within each data trace, the challenge for attackers is to discern whether the bot-paths have shared bottlenecks amidst the background of Internet noise. Since the problem of whether bottlenecks are shared can be a binary classification problem, we employ Kernel Density Estimation (KDE) to visualize the classification outcomes.

As depicted in Figure 8, the x-axis represents the values of $D = Max_{i,j}\ (\Delta_x - \mathbf{R}x)ij$, which differentiate between two distinct congestion scenarios, while the y-axis indicates their kernel density. In this representation, blue signifies true positives (instances where congestion samples genuinely share bottleneck links), and yellow denotes false positives (instances without shared bottlenecks). The differentiation between the two congestion types is clear, suggesting that a straightforward threshold of 0.5 can effectively classify the samples.

Additionally, we evaluate the classification performance using four key metrics: accuracy, precision, recall, and the F1-score, summarized in Table 2. The results demonstrate that the congestion identification approach presented in this study achieves a minimum performance of 95% across these metrics, validating the practicality of using correlated congestion for bottleneck link identification within the Internet. Notably, the identification method outperforms some recent congestion control studies [40] due to the incorporation of time synchronization and enhanced noise mitigation techniques in our congestion correlation analysis algorithm.

6.3 CrossPoint-SD/CC Evaluations

Setup. In this Section, we focus on the effectiveness of the CrossPoint-SD attack and the CrossPoint-SD/CC attack with comprehensive experiments considering topology, budget, network robustness, and protection parameters.

Metric. To assess the effectiveness of the attack, we employ the attack's *success rate* as our evaluation metric. An attack is considered a *success* if the attackers manage to identify at least c bot-paths traversing a link, where c denotes the link capacity. To ensure reliability, we replicate each experimental condition 1,000 times, contributing to the data presented in Figure 9 and Figure 10.

Experiment 1: attack budget *v.s.* success rate. For LFA adversaries, understanding the correlation between attack budget and success rate is important. To this end, we design an experiment to evaluate the success rates of CrossPoint attacks alongside various other LFAs against NTO mechanisms. In this experimental setup, attackers are constrained to choose a limited number of bot-paths (defined as their budget) from

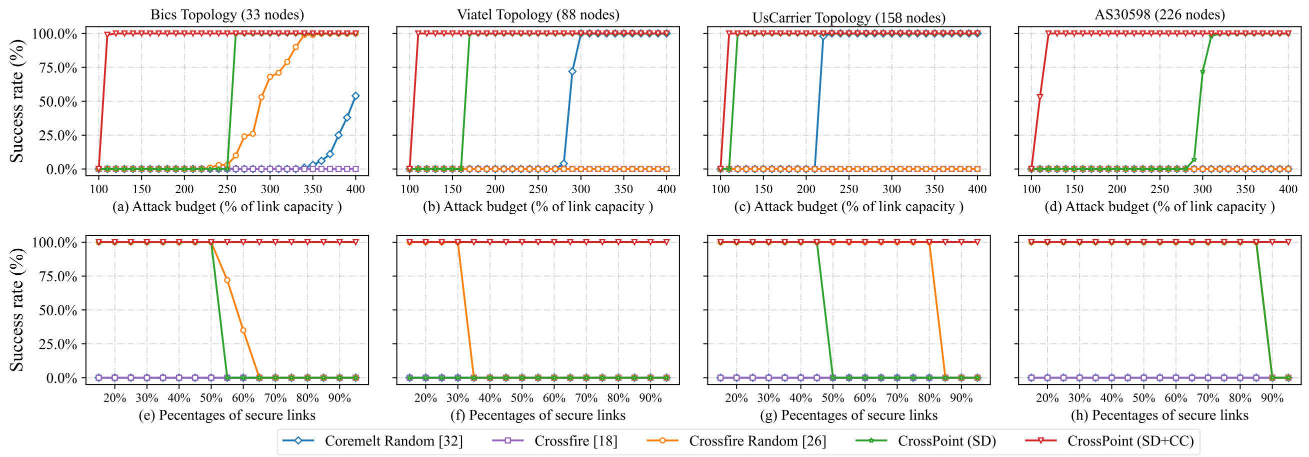

Figure 10: The CrossPoint attacks' effectiveness against EqualNet [21] defense mechanism across four network topologies. Subfigures (a)-(d) illustrate the relationship between the attack success rate and the attack budget; subfigures (e)-(h) demonstrate how the success rate of CrossPoint attacks varies with the robustness of the network topology.

their botnets to initiate attacks, aiming to flood at least one link, which necessitates a critical set of attack flows traversing the targeted link. However, NTO defenses may obscure the physical link information for each bot-path, complicating the attackers' efforts. Under these conditions, the Crossfire attack is unable to pinpoint the target link and its associated attack flows. Conversely, Crossfire+Random and Coremelt+Random strategies employ random bot-paths to mitigate some shortfall, while CrossPoint-SD and CrossPoint-SD/CC attacks leverage statistical disparities and a combination of these with correlated congestion to identify the requisite attack flows.

Figures 9(a)-(d) illustrate the success rates of various LFAs against the Nethide defense and their relationship to budget. Similarly, Figures 9(a)-(d) depict these success rates against the EqualNet defense. In both experiments, the x-axis represents the budget, scaling from 100% to 400% of link capacity, while the y-axis denotes the attack success rate, ranging from 0 to 1. According to the results, the success rates for most attack methodologies (except Crossfire) tend to escalate with an increase in the attacker's budget. Notably, the CrossPoint-SD/CC attack shows a high success rate with a minimal additional budget, significantly surpassing other LFAs, and the CrossPoint-SD method outperforms other random selection-based strategies. Both CrossPoint attacks show good performance because they exploit statistical disparities and correlated congestion, which are challenging for NTO defenses to hide effectively.

Moreover, the success rate of CrossPoint-SD/CC attacks shows consistency across different network topologies, achieving high success rates with an extra budget of 10% to 20%, demonstrating the general applicability of statistical disparities and correlated congestion in CrossPoint strategies. Nevertheless, the CrossPoint approach is not without its limitations. It invariably entails additional budget expenses in the range of 10%-20%. As inferred from the analysis in Section 6.2, this can be attributed to the attacker's approximately 95% accuracy in analyzing correlated congestion, leading to potential minor inaccuracies in flow judgment based on correlated congestion that necessitate a compensatory budget increase. Compared with the higher cost implications of random attack strategies, an extra expense of 10%-20% remains an affordable proposition for attackers.

Experiment 2: Topology robustness *v.s.* success rate. In addition to the budget, the success rate of LFAs is also influenced by network robustness. To understand the relationship between various robustness parameters and LFA performance, we design experiments that vary the link capacity parameters to change network robustness. Given that the flow density on each link is intrinsic to the topology and not readily alterable, adjusting link capacities allows us to control the number of bottleneck links, thereby presenting networks of differing robustness. For instance, a link inherently routing only 10 flows (due to the topology's design) becomes secure to flooding if its capacity is set to 11. We can control the proportion of links susceptible to LFAs by varying the link capacities across different topologies. For example, setting a robustness parameter x = 95% implies that only 5% of the network's links are vulnerable to LFAs, posing a significant challenge for attackers to accurately target one of these susceptible links within the virtual topology.

Figures 9(e)-(h) and 10(e)-(h) depict the relationship between the success rates of LFAs against the Nethide [29] and EqualNet [21] defenses across a spectrum of topological robustness settings. In these trials, we maintained a constant budget parameter at 120%. The x-axis in these figures denotes the topological robustness parameter, which varies from 20% to 95% secure links. As the proportion of secure links grows, the difficulty of achieving a successful LFA correspondingly increases. According to the results, in most scenarios, only the CrossPoint-SD/CC attack consistently maintains a high

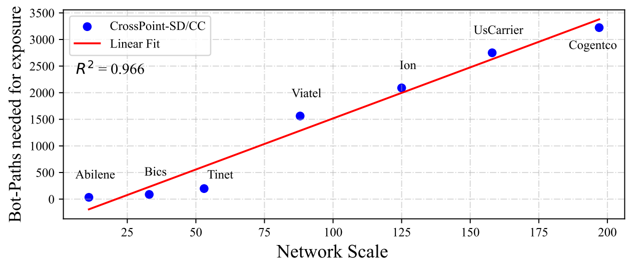

Figure 11: CrossPoint attacks' scalability across different sizes of topologies.

success rate in topologies with high robustness parameters, whereas the efficacy of other attacks reduces as robustness increases. This is because the reduction in vulnerable targets further enhances the defensive advantage, making it difficult to discover the true bottleneck link among the created virtual links. At the same time, the effectiveness of random attack strategies declines due to the increase in the number of invalid flows. Nevertheless, the CrossPoint-SD/CC attack remains largely unaffected, as the congestion information it leverages is independent of the topological robustness.

Experiment 3: Protection levels *v.s.* success rate. The protection levels of defense mechanisms may also influence the effectiveness of LFAs. To explore the impact of different protection level parameters on LFA performance, we conduct experiments that adjust the protection level settings of defensive mechanisms. Specifically, Nethide [29] employs a flow density reduction factor (FR) to vary protection levels across the network topology, depicted as

$$FR = 1 - \frac{Avg.FD_{virtual}}{Avg.FD_{physical}}.$$

The FR value indicates the topology's security level; a higher FR suggests enhanced security, making it increasingly challenging for LFA attackers to accurately discern the intended attack target and select the corresponding attack flows. On the other hand, EqualNet [21] utilizes an obfuscation parameter τ to denote different levels of protection. According to EqualNet, the obfuscation parameter τ can be adjusted up to 80%; hence for EqualNet, our evaluation focuses on the boundary of LFA performance under the strictest condition.

6.4 Scalability

To evaluate the scalability of CrossPoint attacks, we conduct experiments with seven real network topologies (Table 3) from the Internet topology zoo [24].

We explore several evaluation schemes to assess the scalability of CrossPoint attacks. The most intuitive approach is to study the success rate of attacks under different network scales. However, the success rate may not be the most suitable metric for understanding the scalability of attacks. This is because the success rate is closely linked to the attack budget, and maintaining a constant budget for scalability evaluations across varying topological scales makes nonsense. For instance, deploying 1000 bots in a small 10-node network might invariably lead to successful attacks, rendering such experiments on smaller topologies somewhat trivial. Conversely, a limited number of bots might not provide meaningful insights into the effectiveness of attacks on larger networks.

Table 3: Topology used in scalability experiments

Topology	Nodes	Edges	Topology	Nodes	Edges
Abilene	11	14	Ion	125	146
Bics	33	48	UsCarrier	158	189
Tinet	53	89	Cogentco	197	243
Viatel	88	92			

Therefore, our experiment focus shifts towards the required budget to achieve a specific success rate as a measure of scalability. Specifically, we study the number of bot-paths that attackers must reveal against defenders to attain a 90% success rate within different topologies. This approach allows us to better understand the resource demands attackers need to make as network sizes and complexities increase, offering a more practical understanding of attack scalability.

Figure 11 illustrates the number of bot-paths that need to be disclosed by CrossPoint attacks to achieve a 90% success rate across networks of varying scales. With the increase in network scale, there is a corresponding rise in the requisite number of bot-paths for CrossPoint attackers. In particular, the required bot number for CrossPoint-SD/CC attacks appears to grow linearly with the size of the target network. To validate this, we fit the required bot number and the network size into a linear model, as shown in Figure 11, and evaluate the goodness-of-fit with the R^2 coefficient [16]. The R^2 value of 0.966 confirms the linear relationship between the attack resources and the network size.

In practice, the required attack resource also remains affordable to the attackers. For example, attacking a network with about 200 nodes necessitates the disclosure of about 3000 bot-paths. Given the distributed nature of DDoS attacks, 3000 bot-paths is a manageable figure for attackers, especially considering that each bot may handle 100+ paths and report the results to the attackers.

In summary, the proposed CrossPoint attacks are highly scalable in that the required attack resource (bot-paths) increases linearly with the size of the target network. The CrossPoint attacks could pose practical threats to large-scale networks, as long as the attacker has a sufficient budget.

6.5 Measurement study

Setup. We conduct a measurement study on the Internet to understand the feasibility of our attacks. Our study took place from May to November 2022 and utilized 6 senders and 20 public receivers. We observed at least 120 paths and sent

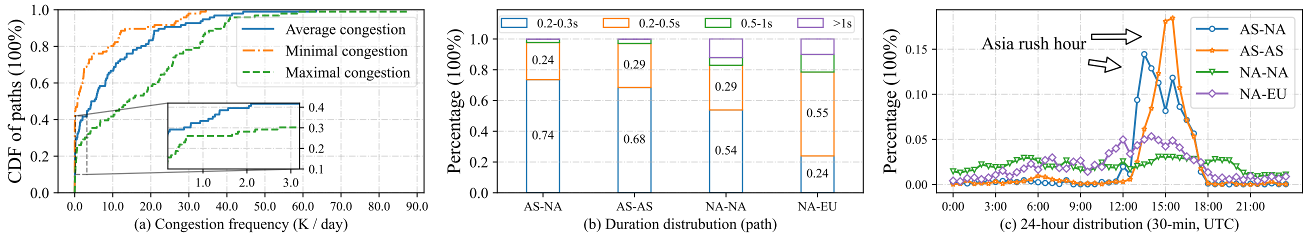

Figure 12: Measurement study of congestion. (a) CDF of the congestion frequency; (b) congestion duration distribution on four path groups; (c) a 24-hour congestion distribution on four path groups. AS = Asia, NA = North America, and EU = Europe.

almost 290 million ping packets, obtaining approximately 100.8 GB of aggregated data. Our senders were located in 6 cities globally, while the receivers consisted of industry and academia, including public DNS servers, universities, and worldwide organizations, on different continents. Each sender sends `ping` probes at a 10 packet-per-second speed, approximately 2.88 Kbps, which will not add pressure to any Internet links and servers.

Separate host-level congestion. Since our study relies on end-to-end delay probing, it's possible that the observed increase in delay might not solely indicate network link congestion but could also include host-level congestion, such as a busy destination server. Therefore, to refine our analysis, we design a method to exclude this host-level congestion. If a correlated delay increase is detected on all sources targeting the same server, this pattern might indicate a server-related slowdown. We examine cases where all sources to a particular destination report correlated congestion and classify these congestion events as host-level issues rather than Internet congestion.

Feasibility of CrossPoint attacks. Figure 12(a) illustrates the aggregated results of the congestion frequency's cumulative density function (CDF) on all paths. Notably, on average, 68% of the paths experience more than 1K congestion events per day. Even on the best days, half of the paths still incur at least 700 congestion events, which indicates an average interval of two minutes between congestion events. Furthermore, the results indicate that about 40% of Internet paths encounter 10K congestion events per day on average. These findings suggest that the Internet provides ample congestion resources that attackers can exploit to conduct CrossPoint attacks. It may take only a few hours to prepare for an attack on most links, even when the network performs optimally. Therefore, in practice, a CrossPoint attacker can launch a 24-hour observation on the Internet to collect sufficient congestion samples of a hidden link in most network paths. It is important to note that this time can be further shortened, as demonstrated in Figure 12(c), where some Internet paths exhibit a specific 'rush hour' around a certain time, during which the attacker can gain more useful congestion data.

Stealthiness of CrossPoint attacks. Given that congestion can occur suddenly, it may be necessary to employ a very high sampling rate to fully capture instances of congestion in real-world scenarios. However, in this study, we demonstrate that even if an attacker uses a lower congestion detection sampling rate to maintain stealth, such as 10 PPS ($\approx$ 2.88 Kbps), they can still observe most network congestion events.

As depicted in Figure 12(b), we present four examples of intercontinental link congestion, revealing that most network congestion durations last for more than 0.2 seconds. Consequently, at a sampling rate of 10 PPS, the attacker can differentiate between network noise generated by a single packet and network congestion involving at least two packets.

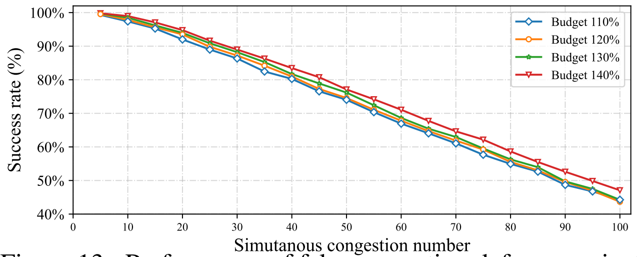

Figure 13: Performance of fake congestion defense against CrossPoint attacks.

7 Countermeasure

Since the CrossPoint attacks exploit congestion events to target their attack flows on specific physical links, a conceivable defense strategy is to deflect these attack flows by intentionally creating fake congestion events across various network links simultaneously. For instance, the defenders could craft correlated fake congestion on two distinct network links; botpaths observing these correlated congestion events might mistakenly believe that they pass through the same physical link and send attack flows to them. However, this misidentified botpath cannot contribute to attacks against the target, reducing the effectiveness of the attack.

To implement the *fake congestion defense*, the defenders follow three steps: (1) To generate congestion in a specific device, defenders can either delay incoming packets temporarily or inject burst traffic into buffer queues. The latter, injecting burst traffic, is more practical as it reproduces the natural congestion process without necessitating modifications to network devices. (2) To ensure that the artificial congestion events appear correlated, each network device must have the same bandwidth and receive the same volume of burst traffic simultaneously. Finally, (3) to deceive the attackers, the defenders must strategize on the timing and locations for generating fake congestion. Given the unpredictability of the attacker's target and timing, a practical approach for defenders is to create correlated congestion across random network devices periodically.

In this paper, we evaluate the effectiveness of fake congestion defense against CrossPoint attacks within the AS30598 topology, which comprises 226 nodes and 314 links. The defense strategy involves the use of Nethide [29] to protect the network while simultaneously generating various amounts of fake congestion through burst traffic injection. Figure 13 illustrates the correlation between the attacker's success rate and the number of simultaneous fake congestion events. Our experiment results indicate that as the number of simultaneous fake congestion events increases, the attacker's success rate decreases for all budget levels. This trend suggests that amplifying the number of links involved in fake congestion creation generally boosts defense effectiveness.

While this defense appears to be reasonably effective, however, it comes at a relatively high cost. Implementing this defense requires generating a significant amount of simultaneous fake congestion, which results in substantial burst traffic injection and may impact legitimate users. Firstly, creating short-term congestion on a device requires a considerable volume of burst traffic; for instance, injecting 20 Gb of data into a 40 Gbps queue could only congest the device for about 0.5 seconds. Moreover, achieving simultaneous fake congestion demands the involvement of a large number of network devices. For example, the results suggest that engaging around 90 devices in simultaneous fake congestion generation is necessary to decrease the success rate of CrossPoint attacks by 50%. Secondly, existing congestion control mechanisms are sensitive to congestion [12, 20, 37]; the fake congestion could activate TCP flows' congestion avoidance mechanisms, thereby reducing the throughput for legitimate users. As a result, bridging the gap between the concept of fake congestion defense and its practical implementation remains a challenge.

8 Discussion

Enhancing CrossPoint attacks. In scenarios where congestion events are infrequent or absent, attackers can either exploit existing *statistical disparities* patterns or proactively craft congestion. For instance, they might adopt approaches akin to SDN CrossPath attacks [11], where selected bots are orchestrated to dispatch short-term burst flows to random places while remaining stealthy [43].

Future design of NTO schemes. The CrossPoint attack highlights a fundamental issue in current NTO designs: while NTO strategies can conceal critical information at the upper layer, lower-level network traffic characteristics are inherently difficult to obfuscate, challenging the creation of defenses that align with upper-layer strategies. Nonetheless, introducing lower-level obfuscation mechanisms is possible. For example, fake congestion defense. However, a major challenge associated with fake congestion lies in the trade-off between maintaining benign user throughput and achieving defensive efficacy; a large amount of fake congestion can decrease user throughput, whereas a limited number of fake congestion might fall short of providing robust defense. To tackle this challenge, one potential NTO strategy could integrate upper-layer obfuscation algorithms, network topology, traffic patterns, and other attributes to strategically create fake congestion at key points to minimize the impact on legitimate users. Moreover, other network traffic characteristics that influence security should also be considered. For instance, propagation delay can leak information about path length. Therefore, future NTO schemes should avoid using short virtual paths to substitute long physical paths in detailed obfuscation algorithms. To sum up, future NTO designs should carefully balance the integration of various obfuscation techniques and considerations of network traffic characteristics to enhance security without impacting user experience.

9 Conclusion

Crosspoint attacks exploit inherent Internet features to escape SOTA NTO defenses with a high success rate and a low extra cost. Besides, our measurement study substantiates the practical feasibility and stealthy nature of these attacks. Consequently, although NTO defenses may succeed in obfuscating links, they fall short in concealing the entire network map due to incomplete obfuscation measures on inherent features. To enhance the future security of NTO schemes, we also discuss potential countermeasures. We hope this work could stimulate increased focus on the development of more robust NTO schemes to counteract LFAs effectively.

10 Acknowledgements

We thank all anonymous reviewers and our shepherd for their valuable comments and suggestions. This work of Xuanbo Huang, Kaiping Xue, Lutong Chen, Mingrui Ai, Qibin Sun was supported in part by the National Natural Science Foundation of China under Grant No. 62372425 and No. U19B2023, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

References

[1] The CAIDA UCSD IP Prefix-to-AS mapping in 2008, 2022. accessed: June, 2024.

[2] DDoS attack trends for 2022 q2, 2022. accessed: June, 2024.

[3] The DDoS that knocked spamhaus offline, 2022. accessed: June, 2024.

[4] Gurobi mathematical programming solver, 2022. accessed: June, 2024.

[5] Wondernetworks: A day in the life of the Internet, 2022. accessed: June, 2024.

[6] Ciso panel: The future of cyber is automated moving target defense., 2023. accessed: June, 2024.

[7] Cyberattack causes ic Internet connection to cut out., 2024. accessed: June, 2024.

[8] Venkat Arun, Mina Tahmasbi Arashloo, Ahmed Saeed, Mohammad Alizadeh, and Hari Balakrishnan. Toward formally verifying congestion control behavior. In *proceedings of the 35th ACM SIGCOMM Conference*, pages 1–16, 2021.

[9] Cristina Basescu, Raphael M. Reischuk, Pawel Szalachowski, Adrian Perrig, Yao Zhang, Hsu-Chun Hsiao, Ayumu Kubota, and Jumpei Urakawa. SIBRA: Scalable internet bandwidth reservation architecture. In *proceedings of the 23rd Network and Distributed System Security Symposium (NDSS)*, pages 1–16, 2016.

[10] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise Reduction in Speech Processing*, pages 1–4. Springer, 2009.

[11] Jiahao Cao, Qi Li, Renjie Xie, Kun Sun, Guofei Gu, Mingwei Xu, and Yuan Yang. The CrossPath attack: Disrupting the SDN control channel via shared links. In *proceedings of the 28th USENIX Security Symposium*, pages 19–36. USENIX, 2019.

[12] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue*, 14(5):20–53, 2016.

[13] Xuyang Ding, Feng Xiao, Man Zhou, and Zhibo Wang. Active link obfuscation to thwart link-flooding attacks for internet of things. In *proceedings of the 19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 217–224. IEEE, 2020.

[14] Seyed K. Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. Bohatei: Flexible and elastic DDoS defense. In *proceedings of the 24th USENIX Security Symposium*, pages 817–832. USENIX Association, 2015.

[15] Chuanpu Fu, Qi Li, Meng Shen, and Ke Xu. Realtime robust malicious traffic detection via frequency domain analysis. In *proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 3431–3446. ACM, 2021.

[16] Stanton A Glantz, Bryan K Slinker, and Torsten B Neilands. *Primer of applied regression & analysis of variance, ed*, volume 654. McGraw-Hill, 2001.

[17] Ningning Hu, Li Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. Locating internet bottlenecks: Algorithms, measurements, and implications. *ACM SIGCOMM Computer Communication Review*, 34(4):41–54, 2004.

[18] Min Suk Kang, Virgil D Gligor, Vyas Sekar, et al. SPIFFY: Inducing cost-detectability tradeoffs for persistent link-flooding attacks. In *proceedings of the 23rd Network and Distributed System Security Symposium (NDSS)*, pages 53–55. ISOC, 2016.

[19] Min Suk Kang, Soo Bum Lee, and Virgil D Gligor. The crossfire attack. In *proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)*, pages 127–141. IEEE, 2013.

[20] Changhoon Kim, Anirudh Sivaraman, Naga Katta, Antonin Bas, Advait Dixit, and Lawrence J Wobker. Inband network telemetry via programmable dataplanes. In *proceedings of the 30th ACM SIGCOMM*, volume 15, pages 1–2. ACM, 2015.

[21] Jinwoo Kim, Eduard Marin, Mauro Conti, and Seungwon Shin. Equalnet: A secure and practical defense for long-term network topology obfuscation. In *proceedings of the 2022 Network and Distributed System Security Symposium (NDSS)*. ISOC, 2022.

[22] Jinwoo Kim, Jaehyun Nam, Suyeol Lee, Vinod Yegneswaran, Phillip Porras, and Seungwon Shin. BottleNet: Hiding network bottlenecks using sdn-based topology deception. *IEEE Transactions on Information Forensics and Security (TIFS)*, 16:3138–3153, 2021.

[23] Min Sik Kim, Taekhyun Kim, Yong-June Shin, Simon S Lam, and Edward J Powers. A wavelet-based approach to detect shared congestion. *IEEE/ACM Transactions on Networking (TON)*, 16(4):763–776, 2008.

[24] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications (JSAC)*, 29(9):1765–1775, 2011.

[25] Soo Bum Lee, Min Suk Kang, and Virgil D Gligor. Codef: Collaborative defense against large-scale link-flooding attacks. In *proceedings of the 9th ACM Conference on Emerging Networking Experiments and Technologies (CoNext)*, pages 417–428, 2013.

[26] Christos Liaskos, Vasileios Kotronis, and Xenofontas Dimitropoulos. A novel framework for modeling and mitigating distributed link flooding attacks. In *proceedings of the 35th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–9. IEEE, 2016.

[27] Carl C. Manion. The emerging moving target defense market - what you need to know, 2023. accessed: June, 2024.

[28] Carl Manion Mark Pohto. Gartner emerging tech: Security — tech innovators in automated moving target defense, 2023. accessed: June, 2024.

[29] Roland Meier, Petar Tsankov, Vincent Lenders, Laurent Vanbever, and Martin Vechev. NetHide: Secure and practical network topology obfuscation. In *proceedings of the 27th USENIX Security Symposium (USENIX Security)*, pages 693–709. USENIX, 2018.

[30] J. Postel. Internet Control Message Protocol, 1981. RFC 777, Accessed: June, 2024.

[31] Ryan Rasti, Mukul Murthy, Nicholas Weaver, and Vern Paxson. Temporal lensing and its application in pulsing denial-of-service attacks. In *proceedings of the 36th IEEE Symposium on Security and Privacy (S&P)*, pages 187–198, 2015.

[32] Nagarathna Ravi, S. Mercy Shalinie, and D. Danyson Jose Theres. BALANCE: Link flooding attack detection and mitigation via hybrid-sdn. *IEEE Transactions on Network and Service Management (TNSM)*, 17(3):1715–1729, 2020.

[33] Dan Rubenstein, Jim Kurose, and Don Towsley. Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM Transactions On Networking (TON)*, 10(3):381–395, 2002.

[34] Jared M Smith and Max Schuchard. Routing around congestion: Defeating ddos attacks and adverse network conditions via reactive bgp routing. In *proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*, pages 599–617, 2018.

[35] Ahren Studer and Adrian Perrig. The coremelt attack. In *proceedings of the 14th European Symposium on Research in Computer Security (ESORICS)*, pages 37–52. Springer, 2009.

[36] Muoi Tran, Min Suk Kang, Hsu-Chun Hsiao, Wei-Hsuan Chiang, Shu-Po Tung, and Yu-Su Wang. On the feasibility of rerouting-based ddos defenses. In *proceedings of the 40th IEEE Symposium on Security and Privacy (S&P)*, pages 1169–1184. IEEE, 2019.

[37] Jingyuan Wang, Jiangtao Wen, Chao Li, Zhang Xiong, and Yuxing Han. Dc-vegas: A delay-based tcp congestion control algorithm for datacenter applications. *Journal of Network and Computer Applications*, 53:103–114, 2015.

[38] Juan Wang, Ru Wen, Jiangqi Li, Fei Yan, Bo Zhao, and Fajiang Yu. Detecting and mitigating target link-flooding attacks using sdn. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 16(6):944–956, 2019.

[39] Lei Wang, Qing Li, Yong Jiang, Xuya Jia, and Jianping Wu. Woodpecker: Detecting and mitigating link-flooding attacks via sdn. *Computer Networks*, 147:1–13, 2018.

[40] Wenjia Wei, Kaiping Xue, Jiangping Han, David SL Wei, and Peilin Hong. Shared bottleneck-based congestion control and packet scheduling for multipath tcp. *IEEE/ACM Transactions on Networking (TON)*, 28(2):653–666, 2020.

[41] Jiarong Xing, Wenqing Wu, and Ang Chen. Ripple: A programmable, decentralized Link-Flooding defense against adaptive adversaries. In *proceedings of the 30th USENIX Security Symposium*, pages 3865–3881. USENIX, 2021.

[42] Menghao Zhang, Guanyu Li, Shicheng Wang, Chang Liu, Ang Chen, Hongxin Hu, Guofei Gu, Qianqian Li, Mingwei Xu, and Jianping Wu. Poseidon: Mitigating volumetric DDoS attacks with programmable switches. In *proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*, pages 1–18. ISOC, 2020.

[43] Jing Zheng, Qi Li, Guofei Gu, Jiahao Cao, David K. Y. Yau, and Jianping Wu. Realtime ddos defense using cots sdn switches via adaptive correlation analysis. *IEEE Transactions on Information Forensics and Security (TIFS)*, 13(7):1838–1853, 2018.

[44] Huancheng Zhou, Sungmin Hong, Yangyang Liu, Xiapu Luo, Weichao Li, and Guofei Gu. Mew: Enabling large-scale and dynamic link-flooding defenses on programmable switches. In *proceedings of 43rd IEEE Symposium on Security and Privacy (S&P)*, pages 1625–1639. IEEE Computer Society, 2022.