2018 IEEE Symposium on Security and Privacy

Routing Around Congestion
Defeating DDoS Attacks and Adverse Network Conditions via Reactive BGP Routing

Jared M. Smith
VolSec: UT Computer Security Lab
University of Tennessee, Knoxville
Email: jms@vols.utk.edu

Max Schuchard
VolSec: UT Computer Security Lab
University of Tennessee, Knoxville
Email: mschucha@utk.edu

***Abstract*—In this paper, we present Nyx, the first system to both effectively mitigate modern Distributed Denial of Service (DDoS) attacks regardless of the amount of traffic under adversarial control and function without outside cooperation or an Internet redesign. Nyx approaches the problem of DDoS mitigation as a *routing problem* rather than a *filtering problem*. This conceptual shift allows Nyx to avoid many of the common shortcomings of existing academic and commercial DDoS mitigation systems. By leveraging how Autonomous Systems (ASes) handle route advertisement in the existing Border Gateway Protocol (BGP), Nyx allows the deploying AS to achieve isolation of traffic from a critical upstream AS off of attacked links and onto alternative, uncongested, paths. This isolation removes the need for filtering or de-prioritizing attack traffic. Nyx controls outbound paths through normal BGP path selection, while return paths from critical ASes are controlled through the use of specific techniques we developed using existing traffic engineering principles and require no outside coordination. Using our own realistic Internet-scale simulator, we find that in more than 98% of cases our system can successfully route critical traffic around network segments under transit-link DDoS attacks; a new form of DDoS attack where the attack traffic never reaches the victim AS, thus invaliding defensive filtering, throttling, or prioritization strategies. More significantly, in over 95% of those cases, the alternate path provides complete congestion relief from transit-link DDoS. Nyx additionally provides complete congestion relief in over 75% of cases when the deployer is being directly attacked.**

I. Introduction

Due to their high level of impact, combined with low degree of technical complexity, Distributed Denial of Service (DDoS) attacks continue to represent one of the largest unsolved and persistent threats on the Internet. Recent successful DDoS attacks by the Mirai botnet against root DNS providers [1] and core transit links [2] highlight both the lack of an effective deployed solution to DDoS attacks and the impact such attacks have on critical network infrastructure. By leveraging increased bandwidth, botnets have allowed strategic adversaries to launch devastating attacks with traffic flows too massive to be filtered by downstream victims. Furthermore, adversaries have begun targeting shared transit links located *outside* of the intended victim, rather than directly against the victim's end hosts, an attack methodology proposed in literature by Kang [3] and Studer [4], which we call *transit-link DDoS*. Even worse, transit-link attacks are occurring far more often in practice, with over 99% of observed attacks in quarter three of 2017 targeted at the core Internet infrastructure according to Akamai [5], with less than 1% targeted at the application layer.

However, research has yet to propose an effective and deployable DDoS mitigation strategy that addresses gaps in both literature and industry capabilities. These gaps arise from three core challenges. First, filtering and prioritization techniques [6], [7], [8], [9], [10], [11] claim to alleviate congestion by distinguishing between benign and malicious traffic, and then filtering out the malicious traffic. Despite advances in this area, filtering and related methods such as traffic prioritization require costly per-stream calculations, presenting scalability concerns with modern DDoS attacks. Additionally, recent work illustrates that state-of-the-art approaches to classifying malicious traffic can be defeated via strategic attackers using adversarial machine learning [12], [13]. Finally, filtering and prioritization strategies *cannot* be leveraged when attack traffic is directed at upstream transit-core links, or transit-link DDoS, and then dispersed to wanted locations on the Internet. By dispersing the traffic, the attacker ensures the malicious packets entirely avoid appearance at the victim Autonomous System (AS) while still forcing severe packet loss at the edge of the victim AS. The second core challenge originates in the industry's current approach to combating DDoS. Solutions proposed by CDN-backed solutions [14] become a test of who possesses more bandwidth, the defender or the adversary, a tenuous proposition in an era of multi-Tbps attack flows, an attack flow size the Mirai botnet and its variants have repeatedly achieved. Attacks such as Kang's Crossfire are also outside of the threat model considered by bandwidth-oriented DDoS defenses, which focus on protecting the last-mile links and provide no protection for transit links. Third and finally, theoretical systems such as SCION and SIBRA [15], [16], which integrate DDoS defense into the transit fabric of the Internet via bandwidth contracts between ASes, present exceptional promise but require a complete redesign of the Internet, raising doubt about realizing their benefits in the foreseeable future on a wide-scale.

Our system, called **Nyx**, aims to resolve the problem of modern DDoS and adverse network conditions by allowing

IEEE S&P '18, 21-23 May 2018, San Francisco, CA, USA

DOI 10.1109/SP.2018.00032

IEEE computer society

the defending or deploying network, specifically a multi-homed Autonomous System (AS), to isolate critical traffic from attack traffic at a path level, preventing the critical traffic from competing against malicious traffic for limited resources. An AS deploying our system, which we term the *Deployer AS*, when negatively impacted by a DDoS attack, will adjust the routes of outgoing *and* incoming traffic from a single remote *Critical AS*, known a priori, around links degraded by congestion. Essentially, Nyx allows the deployer to **route inbound critical traffic around congestion**. By considering DDoS mitigation as a *routing* problem rather than a *filtering* or *prioritization* problem, the deployer's capacity to successfully mitigate a DDoS attack is *not dependent* on the volume of malicious traffic. Our motivation for this shift in approach is clear, modern DDoS attacks often reach sustained traffic levels of 1 Tbps or more, CDNs and filtering mechanisms are incapable of providing an effective defense, but Nyx avoids these costly per-stream filtering decisions and bandwidth wars, providing a more scalable defense. In addition to avoiding the necessity of conducting costly processing of large attack flows, the deployer utilizing Nyx does not need to classify traffic as benign or malicious, since our system focuses solely on managing benign traffic from known critical networks. Example deployments of Nyx include protecting traffic to and from a remote compute resource such as a super computer at a national lab, or traffic from a piece of critical cyber-physical infrastructure, for example a smart grid. To accomplish our goals, Nyx achieves path isolation for inbound critical traffic by utilizing the existing functionality of the Border Gateway Protocol (BGP) at any single deployer AS *without outside cooperation*, thus allowing deployment onto the existing Internet routing infrastructure, while providing nearly the same benefits of theoretical and currently not well-tested systems such as Scion and SIBRA. In order to realize Nyx, we address three key challenges.

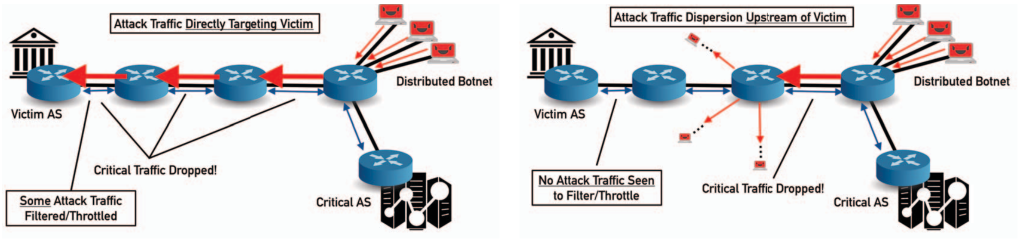
First, we address how the deployer AS can successfully maneuver both outgoing and *incoming* traffic off of attacked links. While altering outgoing paths is trivial, BGP provides the destination network no direct way to control incoming paths. We overcome this limitation by propagating alternative routes to the deployer's network and controlling which ASes propagate those routes via strategically triggered BGP loop detection. We achieve loop detection in routers beyond the deployer's control by selectively lying about the networks appearing on the advertised path. Our traffic engineering technique, which we term *Fraudulent Route Reverse Poisoning* or FRRP, works even if routers deploy origination integrity mechanisms such as RPKI. Nyx use existing traffic engineering techniques to cause the alternative paths to be more preferable with respect to packet forwarding, and utilizes FRRP to ensure that alternative paths propagate around, but never actually reach, the links under a DDoS attack.

Second, our deployer must limit the number of non-critical networks which also change their best path as a result of adjusting paths used by critical networks; a property we term *disturbance*. Disturbance can result in two undesired scenarios. Primarily, disturbance can result in malicious traffic flowing along the alternate path, resulting in the alternate path itself suffering a DDoS attack. Furthermore, even if the disturbed networks are not sources of attack traffic, irrelevant traffic from ASes other than the chosen critical AS might congest the alternative path, as it is likely not provisioned to handle a large amount of traffic beyond normal loads. In order to mitigate disturbance, we expand our path propagation control techniques, *preventing* the propagation of the alternate path to all networks outside of the critical network and the networks appearing along the alternative path.

Lastly, our system needs to ensure that the resulting alternative path has sufficient spare capacity to handle traffic from the critical network, along with traffic from any disturbed networks. If our system detects that the path is struggling to handle the added load, detectable by sampling packet loss on TCP flows to and from the critical network, it will attempt to search for a different alternative path. Nyx utilizes an evolutionary algorithm, where our fitness function attempts to minimize packet loss in TCP sessions with the critical network. Nyx samples new potential alternative routes by withdrawing the alternative path and attempting to re-propagate it, avoiding propagating the route to both links under DDoS attack and the congested links in the previous alternative path. Furthermore, Nyx *does not require* knowledge of either the malicious traffic sources (the ASes containing malicious bots) or the actual capacity of upstream links to search for alternate paths not under attack. Tables I and II described in Section III summarize all the information Nyx does and does not need in order to function.

We demonstrate the ability of Nyx to solve all three of these challenges using realistic Internet-scale simulations in which our system attempts to mitigate a variety of DDoS attack scenarios. We find that it is possible to route critical traffic around attacked links and onto any alternative path in 98% of cases where the primary link connecting the deploying AS to the Internet is attacked, which we call *traditional DDoS*, and all other cases where the attacked link is upstream of the deployer, or *transit-link DDoS* as illustrated by Kang and Studher with Crossfire and Coremelt [3], [4]. We see that implementing techniques to limit changes in the best path to the deployer of non-critical networks reduces unintended path changes to 10 networks on average, as opposed to between 1000 and 5000 networks prior to employing reduction strategies. In addition, we find that our system results in little to no added costs with respect to path length, and does not result in best paths switching to less economically advantageous routes. Lastly, we demonstrate that we can adjust incoming critical traffic onto alternative paths **with significantly less congestion** in 98% of cases regardless of the DDoS attack scenario, and we find that we can move critical traffic impacted by DDoS attacks onto **completely uncongested paths** in over 98% of cases for transit-link DDoS and on average 70% of the time for traditional DDoS.

The rest of this paper is laid out as follows. Section II provides the relevant background on DDoS and BGP. Sec-


Attack Traffic Directly Targeting Victim
Distributed Botnet
Victim AS
Critical Traffic Dropped!
Some Attack Traffic Filtered/Throttled
Critical AS
Attack Traffic Dispersion Upstream of Victim
Distributed Botnet
Victim AS
No Attack Traffic Seen to Filter/Throttle
Critical Traffic Dropped!
Critical AS

(a) **Traditional DDoS**: The victim AS is directly targeted

(b) **Transit-link DDoS**: Transit ASes upstream of the victim is targeted without sending traffic to the victim

Fig. 1: Traditional and transit-link DDoS. In the case of Transit-Link DDoS, which was demonstrated in the Coremelt [4] attack in 2009 and the CrossFire [3], filtering and throttling cannot be done since attack traffic is never seen by the victim.

tion III presents the details of our system design, including the adversarial model, design constraints, our approach to DDoS mitigation, research challenges addressed, and the mechanisms by which we realize our mitigation strategy. Section IV covers details of our simulation methodology and most importantly the evaluation results supporting our claims. Lastly, in Section V we compare our system to other DDoS mitigation systems and conclude with ongoing future work in Section VI.

II. Background

A. DDoS and Botnets

Volumetric Distributed Denial of Service attacks: DDoS attacks provide a high level of impact, combined with a low degree of technical complexity, which has resulted in an increased number of occurrences in recent years. Typically, DDoS attacks have originated from infected hosts on the Internet, as is the case with the Conficker botnet [17]; however, new botnets are often originating primarily in IoT-based devices, such as Mirai [1]. Along with changing sources, monitoring organizations have reported an increase in overall DDoS incidents of 83% from 2015 to 2016 [18]. More troubling, the bandwidth adversaries can harness to conduct DDoS attacks has been steadily increasing annually. Researchers have observed a more than 140% increase in attacks of greater than 100Gbps [18] from 2015 to 2016, with Mirai generating over *1 Tbps* of malicious traffic on multiple occasions. Historically, traditional DDoS attacks originating primarily in hosts see adversaries sending bot traffic directly at the victim network, forcing traffic at the edge of the victim's network to be dropped, thus significantly degrading quality of service. Throughout this paper, we will discuss how Nyx defends against traditional DDoS, which is illustrated in Figure 1a.

Transit-link DDoS: Recently, a new DDoS attack strategy has emerged, targeting core transit links that serve the victim host's *entire network*, which we call Transit-Link DDoS and is shown in detail in Figure 1b. In practice, transit-link DDoS has been seen in recent attacks on the major DNS provider Dyn [19], the prominent security journalist Bryan Krebs with KrebsOnSecurity [20], and the country of Liberia [2]. With transit-link DDoS, the adversary directs bot traffic upstream of the network that is the actual victim, which causes traffic directed to the target to be dropped far ahead of reaching it's final destination. In this case, the bots address their traffic to networks other than the victim, which ensures that the victim *cannot filter the traffic or blackhole it in any way*. Examples of these attacks in literature include the Coremelt attack [4] and the Crossfire attack [3]. The Coremelt attack is a transit-link DDoS attack that takes any number of N bots participating in the attack and sets up N^2 connections between them, inflicting significant damage to the transit core of the Internet. At the time of Coremelt's introduction, no other transit-link DDoS attacks existed, but since then, others have emerged, such as the Crossfire attack. Crossfire, in a method similar to Coremelt, directs traffic to "wanted" locations expecting the attack traffic, such that attack traffic can never be dropped or filtered by targeted ASes along the chosen attack paths. By doing so, Crossfire can bring down connections to selected critical servers in the transit-core simply by congesting their available capacity. In addition to routing around congestion caused by traditional DDoS, Nyx addresses transit-link DDoS as well, alleviating congestion from attacks for a single critical AS in nearly all cases, which we will discuss further in the next section, Section III.

B. Border Gateway Protocol and Inter-AS Routing

Before we discuss how Nyx operates, we review the Internet routing infrastructure, which despite performing beyond the highest expectations for several decades, is revealing flaws not seen or mitigated when first designed. Today, the modern Internet is composed of many autonomous systems, or ASes, which are sets of routers and IP addresses under singular administrative control [21]. Between ASes on the Internet, the Border Gateway Protocol [22] (BGP) is the de facto routing protocol. BGP allows the exchange of information between ASes about routes to blocks of IP addresses, allowing each AS to have knowledge of how to forward packets toward their destinations. BGP is a path-vector routing protocol with policies. This means that routes contain the path they traverse along with other qualities, and individual routers can define their own policies for which routes are considered best and then use the preferred routes to forward packets. BGP policies frequently extend beyond simply choosing the fastest or shortest routes: policies allow complex and flexible decisions based on the relationships between ASes. The decision process tells the router where to send traffic on a per-AS basis, and at each successive hop, the BGP routers along the way pick up the traffic and forward it to the destination through other ASes chosen based on their own policies.

A BGP traffic engineering technique that will be highly relevant to this work is *hole-punching* [22], [23]. In hole punching, a router advertises both a block of IP addresses and a de-aggregation of that block, each with different path properties. Since these IP blocks are technically different, BGP will treat them as routes to different destinations, allowing for more specific policies for certain blocks of IP addresses. These more specific routes will automatically be used, as routers always forward on the most specific matching IP block. We will discuss hole-punching further in Section III-B.

III. System Design

A. Routing Around Congestion

To combat the unmitigated threat posed by transit-link DDoS as well as mitigating traditional DDoS, we have designed **Nyx**. Nyx mitigates DDoS attacks by routing traffic between a Nyx deployer and a chosen critical AS, known ahead of time, around links degraded by a DDoS attack or other adverse network conditions. At a high level, Nyx makes attack traffic from botnets *irrelevant* with regard to network performance to and from the critical network, removing the need for filtering. By operating on a per-route basis, Nyx avoids having to make costly and difficult per-stream decisions. Nyx directly interacts only with the deployer's BGP routers and receives **no outside cooperation, including from the critical AS**, when routing around DDoS. Nyx only makes routing advertisements for IP addresses owned by the deployer, and therefore does not adversely affect the routing information of other ASes. Nyx's ability to route around congestion and make attack flows irrelevant is illustrated in Figure 2 for traditional DDoS and Figure 3 for transit-link DDoS, though we will describe how Nyx works in detail in the rest of this section.

Before we discuss how Nyx is deployed let us consider why prior solutions are insufficient. Recall that *traffic filtering and prioritization are ineffective* against modern DDoS with multi-Tbs traffic flows. Furthermore, the transit-link DDoS attacks proposed in literature, for example Crossfire and Coremelt [3], [4], as well as real-world attacks seen against Liberia [2] and others [5], do not send their attack traffic directly at the targeted AS. This property of transit-link DDoS eliminates even the possibility for the victim to apply any filtering to incoming traffic since critical traffic is dropped upstream and typically outside the control of the victim AS. Nyx approaches the problem differently by exerting control over strictly the benign traffic, without relying on filtering or prioritization techniques. Since our system focuses on the problem of *route selection*, Nyx can utilize normal BGP and traffic engineering techniques to **route around congestion** and DDoS attacks, enabling the deployer to communicate with any upstream AS without loss of quality of service even when targeted by a multi-Tbps attack.

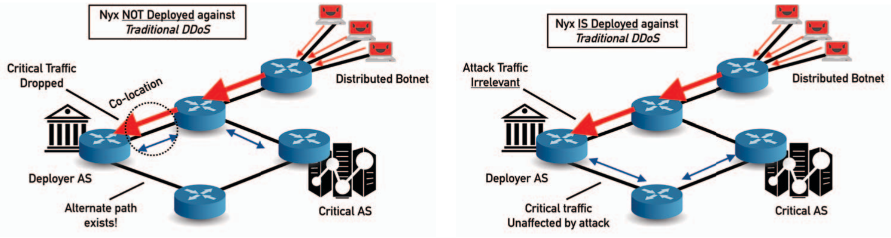
1) Realistic Deployment: Unlike prior systems which mitigate transit-link DDoS via bandwidth contracts [16], Nyx requires no outside cooperation from other ASes, including the critical AS. Furthermore, Nyx does not have any knowledge of where attackers originate. Nyx only assumes it knows the AS relationships via open-source data from CAIDA [24]. In Tables I and II in the Appendix, we summarize the information required and not required by Nyx, including the easily retrievable public sources of the information.

Our system also does not have knowledge about the location of bots in the topology or what DDoS attack flows look like. Instead, Nyx continuously uses packet flow performance as an indicator that the current path between the critical AS and the deployer AS is congested. Additionally, Nyx does *not* need information about the bandwidth or capacity of links on the Internet. The simulator which this work uses to validate Nyx utilizes a bandwidth model for the capacity of links in the topology, but this information is not known to the deployer AS or Nyx. When Nyx discovers the current path is congested using packet flow performance, we use our strategies to route around congestion and attempt to find with an evolutionary algorithm an alternate path with sufficient capacity, as we will discuss later in Section III-D. Finally, Nyx does not need to know whether individual traffic flows or packets are malicious or benign, since the Nyx deployer knows the critical AS a priori and treats all traffic from that AS as ”benign”. By forcing traffic from the critical AS onto a path outside of the sphere of influence of a DDoS event or other adverse network conditions, malicious traffic is completely irrelevant to the deployer assuming the deploying AS is multi-homed (has at least more than one connection to other ASes), which gives Nyx the property of botnet-size independence when mitigating both transit-link and traditional DDoS.

Beyond the information Nyx does and does not know, we make the following assumptions about the deployment of Nyx in practice, and validate these assumptions later in Section IV:

- Nyx should only require the defending AS to deploy Nyx. This means we do not rely on a full deployment of our system across the Internet to work. This further means that our critical AS will *not* provide our defender any assistance.
- Nyx should not negatively impact other ASes. Nyx should not alter any paths outside of routes to and from the defender.
- Nyx should not significantly impact other ASes normal activities. In order to utilize our techniques, the AS operator solely needs to be able to control the BGP speakers for the deployer AS.
- Nyx should function without any changes to BGP, since the techniques we have devised to adjust inbound traffic from known critical ASes can be performed only via adjustment of routing policies at the deployer.

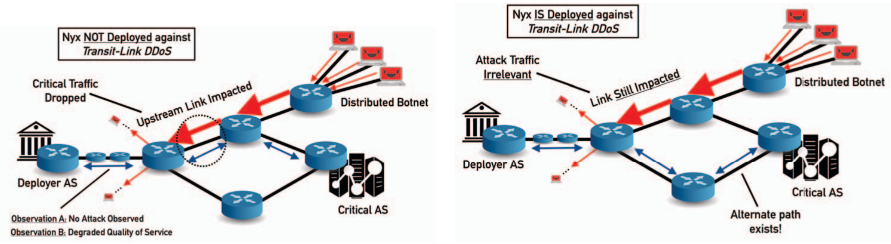
2) Adversarial Model: In accordance with how traditional DDoS and transit-link DDoS are typically controlled, our adversary does not control the underlying network structure and



(a) Nyx NOT Deployed

(b) Nyx IS Deployed

Fig. 2: Nyx deployment against **traditional DDoS**



(a) Nyx NOT Deployed

(b) Nyx IS Deployed

Fig. 3: Nyx deployment against **transit-link DDoS**

is *not routing-aware*, thus unable to make routing decisions. Instead, our threat model considers intelligent adversaries that control massive distributed botnets or a subset of hosts with the ability to generate multi-Tbps attack flows. With this restriction, the adversary can control the selection of bots for a particular attack, how much traffic the bots distributed across the Internet will send, and where in the entire Internet topology each bot should send its traffic. In our current adversarial model, we did not consider a global adversary in the design of Nyx; however, we will discuss in Section VI our ongoing work to address this issue. As mentioned earlier and shown in Tables I and II in the Appendix, Nyx does not know where the bot ASes live, how much traffic they are sending for a given attack, or the quantity of malicious bots in a given attack.

In the rest of this section, we will explore how Nyx achieves its three core goals, while ensuring Nyx falls within the design restrictions we have placed to ensure deployability and resiliency. The first challenge Nyx addresses is how to adjust inbound traffic from an upstream critical AS onto alternative paths, which is discussed next in Section III-B. The second challenge Nyx solves is discussed in Section III-C, where Nyx must reduce the disturbance caused by addressing the first challenge, where disturbance is defined as cases where ASes outside the the critical AS and ASes along the alternative path switch onto new paths to the deployer. The third challenge Nyx addresses is discussed in Section III-D, where we establish how Nyx attempts to maximize the number of instances that the new alternative path has sufficient capacity to handle the critical traffic.

B. Migrating Critical Traffic

Recall from earlier in Section II-B that outbound traffic from an AS is trivial to adjust via BGP local preferences; however, manipulating the paths inbound traffic takes to an AS would typically only be possible via coordination between the ASes on either end. Proposed systems such as SCION and SIBRA require this coordination in order to defend against DDoS effectively [15], [16]. Instead, Nyx assumes *no coordination* between the deployer AS and any other AS, specifically the critical AS. The deployer cannot directly adjust the local preferences of the critical AS to traverse links which avoid DDoS attacks and other adverse network conditions. This presents the first key challenge we address: How does Nyx route inbound traffic around DDoS traffic without cooperation?

By giving the deployer AS the ability to *restrict* the AS-level paths the critical AS can take to the deployer to only certain paths, Nyx can reap the same benefits Scion and SIBRA reap without redesigning the Internet routing infrastructure. To route around congestion, these paths must *not traverse the congested or attacked links* within the topology, such as those affected by traditional or transit-link DDoS attacks. Nyx does this without restricting the critical ASes connectivity to any other ASes, and without causing the critical AS to see any less BGP advertisements from ASes other than the deployer. At a high level, Nyx uses existing routing functionality to make attack traffic irrelevant, which is illustrated in Figures 2 and 3. Since critical traffic headed to the deployer is forced onto uncongested alternate paths, congestion due to any adverse conditions such as DDoS or broken links are no longer a problem.

To give the deployer this ability, we have developed a strategy called **Fraudulent Route Reverse Poisoning**, or FRRP. Nyx employs FRRP to ensure that any BGP advertisements which propagate to the critical AS, which are originated by the deployer AS, are guaranteed to *not* traverse links that are congested or under attack from DDoS or adverse conditions such as broken links or surges in bandwidth usage creating congestion. FRRP does not change the local preferences of the critical AS, rather, it **takes away the choice** of the critical AS to route outbound traffic headed to the deployer over the attacked links by ensuring advertisements which originate at the deployer do not contain the attacked links. By not allowing these congested links to be utilized for connectivity to the deployer, the critical AS chooses an alternate path by which to send traffic to the deployer. In this way, the deployer AS can control the inbound traffic from the critical AS via crafted advertisements combined with strategic lying to prevent congested links from being on the alternate path.

In detail, FRRP is illustrated in Figure 4 and works as follows: the normal traffic from the critical AS 3 to deployer AS 1 usually flows over AS 2 from 3, since the critical AS prefers using AS 2 over AS 4 (shown by Part 4a). However, attack traffic has congested the link from 3 to 2. In order to avoid this link and route the critical traffic over AS 4, the deployer lies about the path by appending AS 2 to it's BGP advertisements. The deployer also appends it's own AS number to the end of the path, which as we will discuss shortly, allows FRRP to function under deployed RPKI. When AS 4 receives this path, it advertises the path to AS 3 (as shown in Part 4b). When AS 2 sees that itself is in the path advertised from the deployer, BGP's built-in loop detection causes AS 2 to not forward to AS 3 the route advertised by AS 1 (shown by Part 4c). Thus, the critical AS 3 will no longer see the path to 1 over 2, and it will use it's only other available path, which is over AS 4, entirely avoiding AS 2. Nyx utilizes FRRP to route incoming traffic from a chosen critical AS onto alternate paths in situations where at least one or more alternate paths exist.

By using FRRP, we achieve over 98% success for the ability to move traffic off of links under DDoS. Figure 2 shows Nyx both deployed and not deployed against a traditional DDoS attack, and Figure 3 shows Nyx both deployed and not deployed against transit-link DDoS. In both cases, Nyx utilizes FRRP to achieve reactive route selection and subvert attacked links, rather than relying on filtering or prioritization of traffic from the critical AS.

1) FRRP under RPKI: When utilizing FRRP, properly deployed resource public key infrastructure (RPKI), also known as Resource Certification, would typically prevent advertising false routes [25]. However, Nyx addresses RPKI's effects on FRRP by ensuring that strategic lying as used by Nyx does not interfere with the route origination process. In detail, given an originating autonomous system, AS_{orig} and a set of ASes to avoid via loop detection, $AV_{AS} = \{\ AS_{AV_1}, AS_{AV_2}, \ldots, AS_{AV_N}\}$ where $AS_{orig} \notin AV_{AS}$, the deployer (the originator in this case) advertises the following path when using FRRP:

$$\{\ AS_{orig},\ AS_{AV_1},\ AS_{AV_2}, \ldots,\ AS_{AV_N}, \underbrace{AV_{orig}}_{\text{For RPKI}}\} \qquad (1)$$

The new path then propagates through the network along from AS_1 to AS_3, beginning at the destination, AS_{orig}, with the avoided ASes appended to the end followed by the originating or deployer AS again:

$$\{\ \underbrace{AS_3,\ AS_2,\ AS_1,}_{\text{Actual Path}}\ \overbrace{AS_{orig}}^{\text{Packet at Dest}}\ , \underbrace{AS_{AV_1}, \ldots,\ AS_{AV_N}, AS_{orig}}_{\text{Irrelevant for Forwarding}}\} \qquad (2)$$

This means that when routes are advertised by the originator in Equation 1, RPKI will treat the route as valid since RPKI only checks to ensure the AS that who originated the route is the last AS in the path. As the path propagates or grows throughout the network in Equation 2, ASes along the path will continue to forward the route as long as the originator remains in the path. The avoided ASes after the originator are irrelevant to forwarding since they lie after the destination AS, yet these additional ASes will not use the new path when receiving the path due to the mechanics of BGP loop detection. This is true because a BGP router will simply scan the entire path for it's own AS number and upon finding itself in the path, it will drop the path.

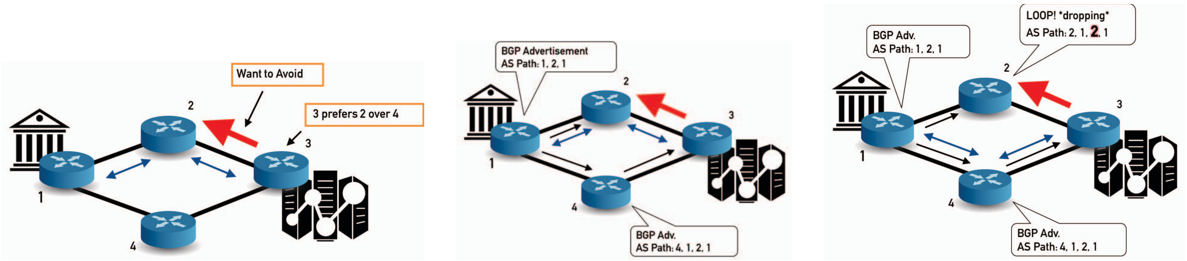
2) FRRP and Network Connectivity: In order to maintain network connectivity, the deployer still advertises its normal paths, but the FRRP paths will be hole-punched prefixes as discussed earlier in Section II-B. The deployer will advertise normal aggregates to maintain connectivity to ASes other than the critical, and will utilize de-aggregate advertisements for FRRP via hole punching. FRRP coupled with hole-punching ensures that the deployer running Nyx can successfully manipulate traffic inbound from the critical AS without losing connectivity to any other ASes.

As discussed in this section, FRRP gives Nyx the ability to route around DDoS attacks and adverse network conditions. Whether the alternate paths can handle the added load is discussed later in Section III-D. Before exploring this issue, we first examine the ability of Nyx to reduce the side-effects of utilizing FRRP.

C. Reducing Disturbance

By utilizing FRRP, we may unintentionally alter the preferred paths to the deployer of ASes other than the critical AS. In the worst case, we alter the path utilized by ASes containing large numbers of bots, potentially causing DDoS traffic to now flow over the alternate path. We term this effect *disturbance*. To address disturbance, we have implemented two techniques that modify the process of FRRP:

- **Selective Advertisement**: We first advertise the FRRP path, taking note of the most preferable alternative path from the critical AS to the deployer. We then withdraw the FRRP path and re-advertise it only to the first AS on the preferred alternative path, which is directly connected



(a) Critical links are congested

(b) Lying about paths and appending ASes to avoid

(c) Loop detection triggered and now the critical AS traverses the alternate path

Fig. 4: Fraudulent Route Reverse Poisoning (FRRP)

to the deployer. Ideally, this will prevent the spread of advertisements to other irrelevant paths in the deployer's routing table.

- **Path Lining**: Using the preferred alternative path, we utilize FRRP to trigger loop detection at every AS adjacent to the path and their customer cone, but not the ASes along the path. By halting the propagation of our alternate path by causing the irrelevant ASes to drop the path only meant for the critical AS, disturbance should be reduced. Keep in mind, path lining requires *no outside cooperation* or coordination from ASes outside of the deployer, since the deployer simply includes the ASes it desires to trigger loop detection for in it's fraudulent advertisements, such that when these ASes receive the route, they drop the route just as if they were along a congested link.

In scenarios where the deployer wishes to protect traffic from multiple critical ASes, the path lining mechanism would form a tree-like structure, rather than a straight path through the topology. We discuss protecting multiple critical ASes in Section VI as ongoing future work.

In our evaluation, selective advertisement alone actually increases the disturbance caused by FRRP, a byproduct of how the path propagates through the topology. Path lining *does* prevent disturbance however, since we are able to add ASes which we do not want our FRRP-advertised routes to propagate beyond to our list of ASes to drop the path due to BGP loop detection. When employing path lining, we see on average less than 10 ASes disturbed as a result of the deployer's actions, which will be discussed further in Section IV-B2.

D. Finding Performant Paths

Even when our system finds paths around ASes we want to avoid, the new paths may not be optimal with respect to the available bandwidth along the new path's links. When we move traffic from one path to another path, we force the alternate path to carry it's original traffic in addition to traffic from the critical AS and any disturbed ASes. If the new links cannot support the amount of added bandwidth we are placing on them, we will still experience congestion, and can even put the deployer in a worse situation than not using Nyx at all.

To counter the problem of moving traffic onto new links without enough bandwidth capacity, we have developed a *searching* algorithm to find the most performant paths when alternate paths exist, which when deployed, will repeatedly use FRRP and path lining to migrate critical traffic to an alternative path. As new alternative paths are used, each is evaluated to discover if congestion was alleviated.

The searching algorithm applied by Nyx is an *evolutionary algorithm*, where the fitness function is the packet loss performance based on the critical ASes traffic over each alternate path. When searching, if the alternate path is experiencing congestion, Nyx withdraws the alternative route, then repeats the FRRP and path lining process, but additionally treats the links along the former alternative path as if they are experiencing DDoS as well. This causes the critical AS to not route traffic from the deployer AS over the insufficient alternate paths. Essentially, Nyx repeats the alternative path generating process, routing around links experiencing DDoS as well as links that have failed to provide a performant alternative path.

IV. Evaluation

There are many questions that need to be asked of the effectiveness of Nyx. Can Nyx migrate inbound critical traffic around congestion onto any alternative path? Will Nyx be able to do so without affecting the BGP decisions and state of ASes not relevant to the deployer and critical AS? What is the increase, if any, in path length of the alternative routes? Can Nyx go beyond traffic migration and actually route critical traffic onto links that are totally uncongested, or at least have less congestion? Will Nyx's ability to route around congestion be insensitive to the adversary's choice of botnet, the link capacity of paths between the deployer and critical AS, and the varying topology of the Internet? Finally, can Nyx do everything mentioned while *only* deployed at a single multi-homed AS with *no* outside cooperation?

To answer these questions, we built a discrete, event-driven network simulator modeling the properties and functionality of routers and traffic flow on the Internet. Due to the high-level of complexity of evaluating Nyx, the design of the simulator presented a challenge. Not only do the claimed properties of Nyx need to be evaluated, but many distinct entities needed to be modeled: ASes, BGP router policies, BGP routers, the links in the actual topology between routers, the botnets used by the adversary, and the bandwidth model used.

A. Simulator Design

In this section, we explore several design choices made when building our simulator, which has been used in prior work by Schuchard et. al [1] [26], [27]. Many of the properties of the Internet needed for completely accurate simulation are closely-guarded secrets: actual traffic flow over links between ASes, bandwidth capacity of these links, monetary cost of using one link over another, etc. However, other properties such as an approximate, up-to-date network topology and botnet distributions have trusted public sources. We will now discuss some of the tradeoffs we made between accuracy and efficiency, including the botnet and bandwidth models used by the simulator and our methodology to evaluate Nyx. As we discuss information used by the simulator, Table IV-A in the Appendix summarized the information the *simulator* knows that Nyx does *not* know when routing around congestion.

1) Network Topology: To model the topology of the Internet we rely on the vast amount of information provided by CAIDA, specifically their AS inferred relationships dataset [24]. Using the inferred topology from CAIDA between late 2016 and mid-2017, the simulator models each AS as a software router which speaks BGP with a realistic configuration taken from routing policies used in practice. The BGP policies used by the simulated routers match the current best practices used by operators and rely on the standardized BGP decision process [22]. As described earlier, each AS is a diverse network in and of itself. Since Nyx is concerned with routing around congestion at an *AS level*, we can largely ignore an ASes internal routing dynamics. Even if we simulated inside an AS, transit-link DDoS attacks will work effectively no matter what end hosts within AS do to defend themselves, at least in the modern Internet. Existing systems such as Scion and SIBRA as mentioned earlier allow an end host to control bandwidth reservation, though these systems require the Internet routing infrastructure to be redesigned and redeployed to provide benefits to everyone. As we will demonstrate in the rest of this section, Nyx provides the same guarantees as bandwidth-reservation systems such as SIBRA without requiring changes to core Internet infrastructure or participation from other ASes.

2) The Bandwidth Models: We recognize that establishing an irrefutable bandwidth model for the modern Internet is an unsolved problem worth several papers on its own; therefore, we have developed and tested what we believe is an accurate and general model that effectively allows us to assign bandwidth capacities to links on the Internet. This in turn let's us simulate attacks against these links and model latency and packet loss. We call this model our *Inferred model*. In addition to this model, we have tested our system with two simpler models, one based on the degree or connectedness of ASes and one on the total number of IPs associated with ASes. We show that Nyx works effectively with simpler models later in Section IV-B6.

To model traffic flow across the internet, we need to know where traffic originates from, where its destination is, and how much of it there is. We base our model on existing work, specifically that of Gill *et al.* [28], supported by the measurements of Labovitz *et al.* [29], the World Bank [30], PeeringDB [31], and Sandvine [32]. Sandvine provides the amount of bandwidth consumption from an "average" user in various regions. This information was combined with the World Bank's estimation of the number of Internet users in each country to get relative inbound and outbound bandwidth on a per nation state basis. In order to assign that bandwidth to ASes, we first assigned each AS to the nation state it primarily resides in using IANA's assigned AS numbers [33], and then consulted PeeringDB, which is a system that allows ASes to advertise their willingness to peer with other ASes [31] along with average amount of inbound and outbound traffic handled. Of the roughly 58,000 ASes on the Internet, just over 8,000 reported bandwidth estimates exist in this dataset.

In order to establish relative bandwidth values between all ASes, a Decision Tree classifier using Scikit-Learn [34] was trained based on the data described above and other AS features including AS degree, the AS customer cone size, the AS's primary country of operation, and the size of IP space advertised by the AS. The resulting classifier had a misclassification error of under 10%, showing that our inferred model has roughly 90% accuracy using all data available.

Again, we recognize that our inferred bandwidth model is not perfect, but currently no literature has established a model for bandwidth sufficient for completely approximating traffic levels across the entire Internet. Regardless of the model used, the bandwidth of links in the simulator are *never* shown to the deployer AS or Nyx, and are only used by the simulator for evaluation.

3) The Botnet Models: Along with the network topology, bot placement may also affect simulation results. In this paper, we have *three* botnet datasets. Recent research has been done to enumerate botnet IPs, which we rely on here. The first dataset comprises 2.9 million unique Mirai [1] hosts, observed throughout 2016 and 2017 [35]. Mirai represents an ideal distribution to model the recent transit-link attacks against entities such as Dyn and Liberia [19], [2] and is largely clustered in IoT devices. The second botnet used is composed of 23 botnet families collectively known as the Conficker botnet with a total of 2.8 million unique hosts observed between 2012 and 2013 [36]. Both the Mirai and Conficker botnets are clustered in a relatively small number of ASes, as shown in Figure 12 in the Appendix, with less than 50 bots in over 97% of ASes, for a total of roughly 2.8 million unique bots in each dataset. The IP addresses of each bot were mapped to their parent AS by associating IPs with their CIDRS and tying CIDRS to an ASN using RouteViews data [37], providing a rough count of bots per AS.

What about adversaries utilizing a far more distributed

[1] Simulator source code: https://github.com/VolSec/chaos

botnet? For this purpose, our third and final dataset is a **fully distributed botnet** where every AS in the topology, except for the deployer and critical AS, is a bot AS with the ability to send malicious traffic.

4) Attack Scenarios: Nyx attempts to protect the deployer AS when it is affected by both *transit-link DDoS* and *traditional DDoS*. For traditional DDoS, instead of measuring our routing success in terms of distance to the deployer, we measure the routing success against **attacked segments** starting with the deployer AS. In this scenario, bots not only address their traffic to the deployer AS directly, but to every segment of hops between the deployer and critical AS; therefore, it is worth noting that less bot ASes have paths to long segments of ASes within the default-free zone of the Internet, as opposed to a given transit core AS. As we will show in the rest of this section, we are largely insensitive to the scenario chosen, which illustrates that we are able to defend against the two major forms of DDoS attacks seen today.

5) Simulation Methodology: Our event-driven network simulator allows us to evaluate the effectiveness of Nyx. At the beginning of the simulation, BGP routers connect to their peers and form a stable network state. These simulated routers use inferred AS relationships described earlier as well as valley-free routing policies. Once the network reaches a stable state, the simulator repeatedly chooses a deployer and critical AS from the topology, and then proceeds to simulate both traditional and transit-link DDoS for the current pair. For each deployer-critical pair, normal traffic flow is sent over the entire topology using the bandwidth models described earlier in Section IV-A2. Then, the simulator pulls bots from the bot-carrying ASes, depending on the botnet model being tested, and seeks to iteratively congest each link along the best path between the deployer and critical AS based on the stable network state.

Nyx then uses the techniques described earlier in Section III-B and Section III-D to route inbound critical traffic onto alternative paths and around congested links. Before and after Nyx routes around congestion, the simulator measures packet loss performance at the deployer based on critical traffic, which is represented as the *subscription factor*. This value is between 0.0 and 5.0, where 1.0 represents each link being at capacity and any more or less is under or over capacity. The simulator measures the congestion via a *congestion factor*, which in our simulation is either 2.0 or 5.0, where the simulator will send enough bot traffic to congest a link at two times it's capacity or five times it's capacity. In this way, the congestion factor determines the first setting of the intensity of an attack. The second is modeled by a value we call the *bandwidth tolerance*. The bandwidth tolerance is a constant value between 1.0 and 2.0 that describes how much additional capacity each link has based on a normal capacity of 1.0. For example, if the bandwidth tolerance is 1.5, then the AS can handle 50% more traffic than it's normal capacity of 1.0, where any higher than 1.0 means that the link is congested and may drop traffic flowing over it. By measuring the subscription factors of each link along the original best path and along the new alternative path, the simulator determines the *worst subscription factor* along the new path and evaluates how effectively Nyx was able to route around congestion.

Nyx follows this evaluation process for all bandwidth models, botnet models, both attack scenarios, and various settings of attack intensity. Out of these simulations, we can answer the questions described at the start of this section. Nyx's ability to route around congestion onto any alternative path is referred to as *routing success*. Nyx's ability to use the path lining techniques described in Section IV-B2 is referred to as *disturbance mitigation*. Nyx's ability to route around congestion onto alternative paths where every link is totally uncongested, such that they have a subscription factor of less than 1.0, is called *strong performance success*. Finally, Nyx's ability to route around congestion onto alternative paths where the most congested link is less congested than the current congestion factor is called *weak performance success*. Nyx's ability to do all of this and not route onto longer paths is also explored.

B. Simulation Results

With our simulator, Nyx was tested against adversaries controlling each of the botnets discussed, under various bandwidth models and attack intensity parameters, and for both transit-link and traditional DDoS scenarios.

1) Can Nyx Migrate Traffic Onto Links Not Impacted by DDoS Attacks?: Nyx is able to find valid paths and *move incoming traffic onto around congested links* with nearly complete success, which is the first step in mitigating transit-link and traditional DDoS of the volumes where current systems fail. Our simulator evaluates Nyx for routing success for both types of DDoS scenarios, and we label this result as **routing success**.

As shown in Figure 5, our system achieved nearly 100% routing success when using FRRP to influence the incoming traffic from ASes between 2 and 8 hops out from the deployer. This means that when transit-links upstream of the deployer AS are being targeted, the deployer AS can successfully cause incoming traffic from a chosen critical AS to move around the impacted links.

Not only can we do so with very high success when transit-links are attacked, but when we are under a traditional DDoS attack, our success in routing incoming traffic was above 78% for only 1 hop out, and nearly 100% when migrating traffic off links 2 hops or greater away from the deployer itself. This means that as an attacking botnet targets the two links closest to the deployer AS on the path from the deployer to critical AS, the deployer can migrate traffic from that critical AS around the two impacted links with nearly 100% success.

In Figure 14 in the Appendix, we show that we can route around congestion onto an alternative path in the Conficker model. In this case, our success is also above 98% for hops between 2 and 8 out from the deployer, both when upstream transit-links are under attack and when the deployer is directly under attack. Finally, we see that Nyx can migrate incoming traffic from the critical AS nearly 100% of the time when

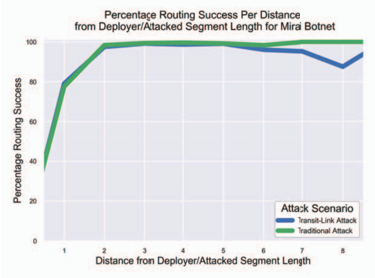


Fig. 5: Percentage routing success for both attack scenarios for Mirai

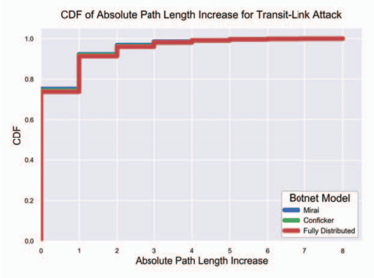


Fig. 6: Absolute path length increase for the transit-link attack scenario

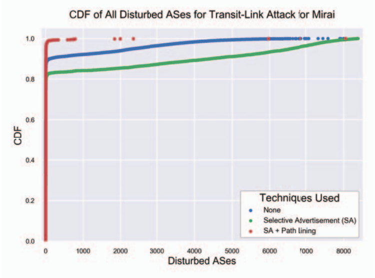


Fig. 7: Disturbed ASes for the transit-link attack scenario and Mirai

under attack from a *globally distributed botnet*, as shown by Figure 15 in the Appendix, which reveals that Nyx can adjust incoming traffic onto alternate paths even when every AS other than the deployer and critical AS is sending attack traffic either directly to the deployer or at upstream links.

Modeling Latency: Before we discuss disturbance mitigation, we first explore how the simulator evaluates Nyx's effects on latency. It is widely accepted that modeling latency is extremely difficult for massively distributed systems; therefore, we adopt the common notion of using path length as a proxy metric for latency. In practice, measuring the latency of chosen, alternate paths on the Internet depends heavily on the layer 1 technologies uses, such as the physical cables between ASes, as well as geographical distance between ASes and the quality of the hardware running the BGP daemons.

When routing around DDoS, alternative paths have length increases of greater than 5 hops in only 2% of runs, and for 94% of cases, we see *no path length increase*, which is shown in Figure 6 for transit-link DDoS, where traditional DDoS shows nearly the same results in Figure 16 in the Appendix. Figure 6 also shows the path length increase when using the Conficker and Fully Distributed botnet models. These results are nearly equivalent with over 94% of cases seeing no increase in path length, illustrating that Nyx is insensitive to changes in the botnet model with respect to path length increases, even for a *globally distributed botnet*.

Routing success can be achieved independent of whether the new path is actually more congested than the original path, and routing success where the network congestion is alleviated on the new path is discussed later in Section IV-B3. In the next section, we discuss how we address the second challenge described earlier in Section III, disturbance mitigation.

2) Can Nyx Migrate Traffic Without Disturbing Other ASes?: Despite being able to migrate incoming traffic onto new paths outside of the influence of a major DDoS attack, we discovered that the FRRP technique used by Nyx disturbed significant numbers of ASes. To overcome the problem of disturbance, we introduced two strategies in Section III-C to be utilized by Nyx: selective advertisement and path lining. When utilizing those strategies in unison, Nyx significantly lessened the disturbance to the ASes close to the deployer AS when Nyx was employed to migrate incoming traffic.

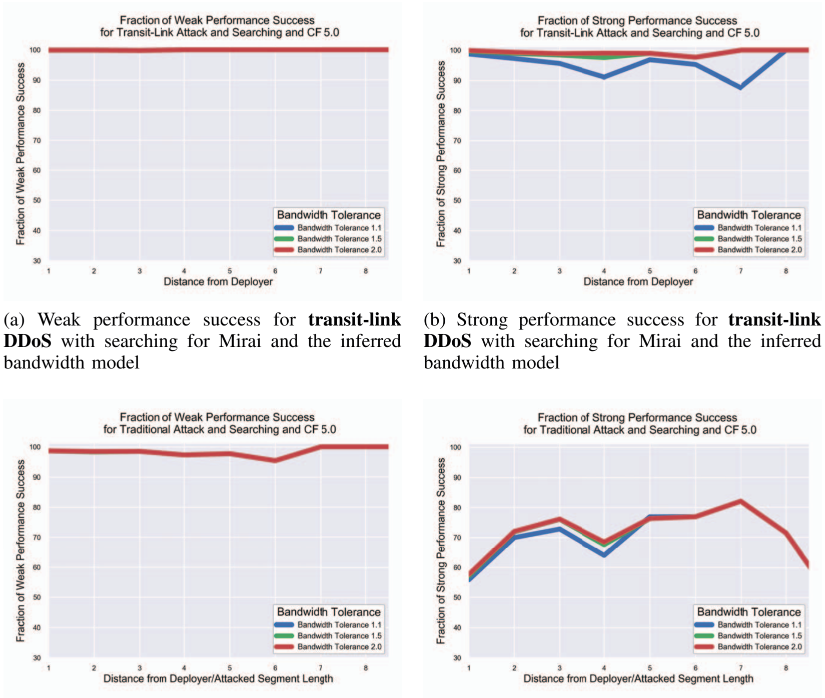
As shown in Figure 7 for transit-link DDoS, before employing any strategies to mitigate disturbance, Nyx disturbed between 1,000 and 6,000 ASes nearly 90% of the time, which in the modern Internet is roughly 10% of all existing ASes [2]. This is true when under either attack scenario: transit-link DDoS or traditional DDoS. In Figure 7, only the transit-link DDoS scenario for the Mirai botnet model is shown, with the traditional DDoS and other botnet models shown in Figure 17 in the Appendix.

However, the disturbance mitigation strategies employed by Nyx are highly effective *independent of the botnet model or attack scenario*. When only selective advertisement was used, disturbance was not reduced. But when combined with path lining, the number of disturbed ASes dropped from on average 5,000 to less than 10 on average, a 500% decrease in disturbance. Using path lining and selective advertisement, Nyx effectively mitigated the disturbance of ASes in the default-free zone, thus reducing the deployment costs of Nyx when both upstream transit-links are attacked and when the deployer AS is targeted directly. Furthermore, for each of those ASes, Nyx also disturbed less than 100 IPs residing within them for all botnets, though these results are not shown here.

Are There Any Local Preference Changes?: Beyond changing ASes best paths or the deployer's taken path lengths to the critical AS, we addressed the monetary cost of link usage from one provider to another provider. Since the actual costs associated with using one link over another within the modern Internet are closely guarded secrets, we used the act of switching onto a peer- or provider-learned path as a proxy for added monetary cost. In our simulations, the deployer AS using Nyx *never switches* from a customer learned path to peer- or provider-learned path, or a peer-learned path to a provider-learned path. We are currently working on explaining this behavior through further simulations.

3) Do the Alternate Paths Have Enough Capacity?: Now that we have shown that Nyx can successfully migrate incoming traffic and do so with little to no disturbance, we now show that Nyx can route around congested paths and onto uncongested paths successfully in nearly all cases for transit-link DDoS and a majority of cases for traditional DDoS. In order to measure performant paths, we use several settings of bandwidth tolerances (1.1, 1.5, 2.0) and congestion factors

[2] As of October 2017, the number of ASes according to CAIDA's Internet topology was roughly 58,000.



(a) Weak performance success for **transit-link DDoS** with searching for Mirai and the inferred bandwidth model

(b) Strong performance success for **transit-link DDoS** with searching for Mirai and the inferred bandwidth model

(c) Weak performance success for **traditional DDoS** with searching for Mirai and the inferred bandwidth model

(d) Strong performance success for **traditional DDoS** with searching for Mirai and the inferred bandwidth model

Fig. 8: Performance success metrics for both the transit-link and traditional DDoS attack scenarios with searching

(2.0 and 5.0), as discussed in Section IV-A. We additionally show that our system has the ability to search for performant paths using an evolutionary algorithm as described in Section III-D, which greatly enhances the success of migrating onto uncongested paths after a DDoS attack. Recall that strong performance success is when the subscription factor after Nyx acts is less than 1.0 for all links along the alternative path, and weak performance success is when the subscription factor is less than the congestion factor.

As shown in Figure 8a, the deployer AS can utilize Nyx with searching to achieve weak performance success of essentially 100% for all distances from the deployer AS in the case of transit-link DDoS. This means that no matter how far out a transit-link is being attacked, we can alleviate *some amount of congestion* in nearly 100% of cases. But what about alleviating *all* of the congestion? We show this in Figure 8b again with searching, where we are still able to find performant paths that are completely uncongested as compared to an original congestion of *5 times more than the capacity* in over *95% of cases* on average. When we do not employ searching, as shown in Figure 18b in the Appendix, we see total success in roughly 89% of cases. These results are for the hardest setting of bandwidth tolerance and congestion factor, illustrating the ability of Nyx to route around congestion under extremely adverse conditions, again *with no outside cooperation*.

Not only can we protect the deployer AS when it is under transit-link DDoS, but we show we can protect the deployer AS when it is targeted directly also for the *hardest settings* of bandwidth tolerance and congestion factor. As we show in Figure 8c, we are able to migrate traffic onto links that are more performant than the original paths in on average 93% of cases, and for strong performance success we can migrate traffic onto paths that are on average completely uncongested in 75% of cases. When employing searching, though we see a higher weak performance success in Figure 8c, with an average of nearly 98% success in alleviating some amount of congestion, we do not see searching helping to as great an extent for strong performance success, as shown in Figure 8d.

Why is traditional DDoS the harder case to protect against? The answer lies in how Nyx utilizes FRRP. When we advertise out our hole-punched paths from the deployer AS while under traditional DDoS attacks, we can end up dragging along large amounts of bot traffic that is being addressed directly to the deployer AS, whereas the bot traffic in transit-link DDoS is never addressed to the deployer AS. In this case, the bot traffic will not be dragged towards the deployer AS. Regardless of this side effect, we have still demonstrated that our system can protect a significant amount of traffic from a chosen critical AS known ahead of time, which usually cannot be done in any capacity for infrastructure based attacks using existing DDoS defense methods.

Furthermore, simulations show that when our system utilizes searching, the depth to which we search is small except in greater distances from the deployer, illustrated in Figure 13 in the Appendix. This means that the deployer does not have to force the BGP speakers implementing Nyx to waste precious time finding more performant paths around impacted links, and that it can be done in the case of transit-link DDoS in nearly 0 iterations on average and 14 iterations on average in the worst case for distances in excess of 8 hops from the deployer.

With these results, we demonstrate that by using Nyx, even when under a traditional DDoS attack, Nyx can route incoming traffic from a known critical AS onto links that are not impacted by the DDoS on average 75% of the time. Furthermore, the deployer can utilize Nyx with *no outside cooperation*, unlike existing work.

4) Is Nyx Insensitive to Attack Intensity?: We have discussed our results for performance success in the case of bandwidth tolerances and congestion factors, but how do we show that these values are not chosen simply to guarantee the success of our system? We show in Figure 9, that for the Mirai botnet model, once our bandwidth tolerance is at 1.1 or higher, the gains received by increasing the tolerance stabilize and do not increase further. This indicates that regardless of how much room you give the link capacities around a DDoS attack, strong performance success does not increase; therefore, our chosen values in the simulation are not in place to guarantee we have greater success.

For congestion factors, we see only slightly higher performance success for smaller congestion factors, such as our other tested factor of 2.0, but not by significant amounts. Though not shown here, the smaller congestion factor of 2.0 has little effect on the strong performance success, adding less than 5% more successful cases on average. This is the case when either transit-links are attacked or when the deployer is attacked directly. Given these results, our simulation's choice of congestion factor indicates that an attacker can continue to congest links on the normal path between the deployer and critical AS and Nyx will still be able to successfully route around the impacted links and alleviate congestion.

5) Is Nyx Insensitive to the Botnet Model?: In Section IV-A3, we described three botnet models: Mirai, Conficker, and a fully distributed botnet. In the previous section, we showed that Nyx significantly mitigates the effects of traditional DDoS and nearly defeats any congestion due to transit-link DDoS when the adversary controls a botnet with the size and topology of Mirai. However, Nyx performs as well with other models, including Conficker, which has a distribution and cardinality similar to Mirai, see Figure 12 in the Appendix, as well as a fully distributed botnet distribution. For Conficker, the results are similar in success to Mirai and are shown in the Appendix in Figure 19. For the fully distributed botnet, Nyx achieves strong performance success in 99% of cases on average for transit-link DDoS for the hardest settings of bandwidth tolerance and congestion factor. Nyx achieves 78% strong performance success on average for traditional DDoS as shown in the Appendix in Figure 20. This means that a globally distributed adversary, such that essentially *every AS in the modern Internet* possesses bots that can send attack traffic upon command, can be subverted by routing around the DDoS events with Nyx deployed at a single AS and without outside cooperation from other ASes.

6) Is Nyx Insensitive to the Choice of Bandwidth Model?: In Section IV-A2, we described the main bandwidth model used in our simulator. This model is fairly complex, but approximates the typical traffic levels through existing ASes using a variety of trusted data sources. We now evaluate our system's ability to perform well with simpler models and show that Nyx is insensitive to the choice of link capacities on the Internet. The additional chosen models were based on the AS Degree and the total number of IPs within each AS.

Figures 10 and 11 show that our system still achieves nearly identical strong performance success for all tested bandwidth models, with our most complex and general inferred model performing the worst overall in terms of alleviating congestion. For the transit-link attack scenario, our models averaged around 95% strong performance success, and for the traditional attack scenario, our models averaged around 70% to 75% success. Therefore, by modeling the link capacities on the Internet as a function of the AS Degree and AS Total IP count, Nyx achieves similar results to the more complex bandwidth model.

V. Related Work

Traditional and current DDoS defense systems attempt to mitigate packet loss and increased latency at the victim through a variety of means; however, no systems exist that defend against DDoS via route-altering techniques such as those used by Nyx. In this section, we will discuss several classes of DDoS defense systems in recent literature, then we will discuss why these systems fail to protect against recent transit-link DDoS attacks and massive traditional DDoS attacks leveraging botnets such as Mirai [19], [20], [2] and future adversaries, then we will discuss why our system does not suffer from the same flaws as these existing systems.

Traditional DDoS defense systems that attempt to alleviate DDoS attacks via packet filtering [38] using techniques such as packet marking [6], [7], [8], [9], [10], [11] and push-back techniques [39], [40], [41], [42] filter traffic at ingress and egress points on the network, but are incapable of withstanding DDoS attacks of the size of the Mirai botnet used in our evaluation. In general, no existing system with minimal deployment requirements provides **botnet-size independence**, the ability to defend against attacks regardless of the size of the malicious botnet sending attack traffic. Beyond defending against massive botnets, transit-link DDoS does not send attack traffic directly to the victim AS, and instead sends attack traffic to upstream links or wanted locations on the Internet; therefore, filtering on attack traffic would not be feasible since the victim would not see the traffic to be filtered. Nyx can handle massive inbound flows sent from distributed botnets because not only do we not physically have to handle malicious traffic in the case of transit-link attacks, but we can

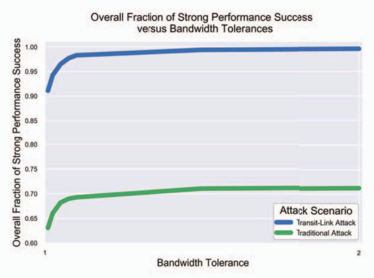


Fig. 9: Strong performance success over varying bandwidth tolerances

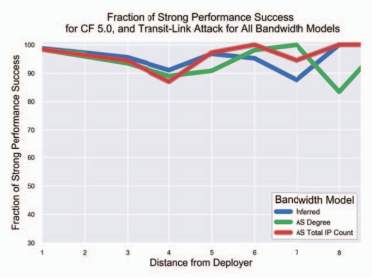


Fig. 10: Strong performance success for the transit-link attack scenario for all bandwidth models

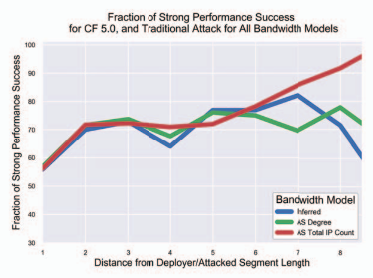


Fig. 11: Strong performance success for the traditional attack scenario for all bandwidth models

arbitrarily alter the paths that critical traffic takes, thus routing around congested links, and by doing so, spread the incoming attack traffic across links upstream of the victim AS.

Other techniques that filter traffic targeted at specific services [43], [44], [45] such as HTTP or DNS are ineffective against DDoS attacks that attack other services or even the underlying control plane. Because all internetwork traffic must be sent over paths determined by BGP routers, Nyx is able to reactively alter advertised paths such that no matter the type of traffic being sent by the adversary, the victim AS will move traffic from a chosen critical AS, known a priori, onto paths not impacted by the malicious traffic.

Strategies using game-theoretic approaches model the defender's best case strategy to maximize cost for an attacker [46], [47], but these approaches are ineffective when massive DDoS attacks can be launched with the click of a button at little cost to the attacker. Zhou et. al.'s work to protect the Internet's backbone and highly connected ASes [48] also fails to defend against transit-link DDoS, since the proposed system only handles traffic once it reaches the deployed system within the victim AS. Other recent works take this same deployment approach, where an attempt to detect and model botnet traffic is done at the victim AS using statistical methods [49], [50], which is not possible in the case of transit-link DDoS.

VI. Conclusion

In this paper, we presented Nyx, a novel system that can significantly reduce the impact of transit-link DDoS, which is previously unsolved form of DDoS being used to take down entire countries, and traditional DDoS. More importantly, Nyx requires only being deployed at the border of a targeted victim AS, *without any cooperation needed* from other ASes to guarantee higher quality of service for inbound traffic. First, we showed that an AS deploying Nyx, which we call the deployer AS, can manipulate not only outbound traffic from an AS, but also the paths which inbound traffic takes. This ability allows our system to intelligently route traffic coming from chosen critical ASes known a priori where the deployer always wants traffic to reach us, around links impacted by DDoS attacks with nearly 100% success. Second, we demonstrated that Nyx can route incoming critical traffic around impacted links without disturbing significant numbers of ASes in close proximity to the AS utilizing our system, with less than 10 ASes on average disturbed by our techniques, as opposed to 1000 to 5000 disturbed ASes on average before employing reduction methods. Third and most importantly, we demonstrated that Nyx can migrate traffic off impacted links onto links that are less congested than the original path in over 98% of cases regardless of the DDoS attack scenario, and to **completely uncongested paths with over 98% success for transit-link DDoS** and on average 75% success for traditional DDoS, thus causing *no traffic* from critical ASes to be dropped even while under a massive attack against the Internet's transit core. Ultimately, this work presents a realistically deployable and demonstrably successful alternative to ineffective filtering and prioritization methods used without success against recent DDoS attacks [1], [2], [51], and furthermore we have contributed the *first scalable and easily deployable* solution to transit-link attacks such as Crossfire and Coremelt [3], [4] without needing to redesign the Internet backbone to ensure bandwidth guarantees as explored by SCION and SIBRA [16].

Future Work The system we have developed creates many interesting opportunities for future work. Currently, our system works only for protecting traffic from a single chosen critical AS, and protecting traffic originating in multiple critical ASes from network congestion would often be necessary in some operational environments. Secondly, our adversarial model does not consider a global adversary that is routing-aware. This adversarial model becomes important when defenders want to protect their networks against an adversary controlling a significant amount of ASes, such as nation-states or major groups of ISPs. Finally, we are actively exploring the effectiveness of our system when there are multiple deployers.

Acknowledgments

We are extremely grateful to the anonymous symposium reviewers for their insightful comments and suggestions. We would also like to thank Aziz Mohaisen from the University of Central Florida for providing the Conficker data and the Network Security Research Lab from Qihoo 360 (netlab.360.com) for the Mirai data.

References

[1] Symantec, "Mirai: what you need to know about the botnet behind recent major DDoS attacks," https://tiny.utk.edu/orVeO, 2016, accessed: 17 January 2017.

[2] "Mirai iot botnet blamed for smashing liberia off the internet," https://www.theregister.co.uk/2016/11/04/liberia_ddos, 2016.

[3] M. S. Kang, S. B. Lee, and V. D. Gligor, "The Crossfire Attack." *IEEE Symposium on Security and Privacy*, 2013.

[4] A. Studer and A. Perrig, "The Coremelt Attack." *ESORICS*, 2009.

[5] Akamai, "State of the internet security report for q3 2017," https://www.akamai.com/us/en/about/our-thinking/state-of-the-internet-report/global-state-of-the-internet-security-ddos-attack-reports.jsp.

[6] V. A. Siris and I. Stavrakis, "Provider-based deterministic packet marking against distributed DoS attacks." *J. Network and Computer Applications*, 2007.

[7] A. Belenky and N. Ansari, "On deterministic packet marking." *Computer Networks*, 2007.

[8] Y. Xiang, W. Zhou, and M. Guo, "Flexible Deterministic Packet Marking - An IP Traceback System to Find the Real Source of Attacks." *IEEE Trans. Parallel Distrib. Syst.*, 2009.

[9] M. Muthuprasanna and G. Manimaran, "Distributed Divide-and-Conquer Techniques for Effective DDoS Attack Defenses." *ICDCS*, 2008.

[10] M. Ma, "Tabu marking scheme for ip traceback," *Parallel and Distributed Processing Symposium*, 2005.

[11] Y. Xiang and W. Zhou, "Protecting information infrastructure from ddos attacks by madf," *International Journal of High . . .*, 2006.

[12] B. Biggio, "Machine learning under attack: Vulnerability exploitation and security measures," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, ser. IH&#38;MMSec '16. New York, NY, USA: ACM, 2016, pp. 1–2. [Online]. Available: http://doi.acm.org/10.1145/2909827.2930784

[13] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: ACM, 2017, pp. 506–519. [Online]. Available: http://doi.acm.org/10.1145/3052973.3053009

[14] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic ddos defense." in *Usenix Security*, 2015, pp. 817–832.

[15] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen, "SCION - Scalability, Control, and Isolation on Next-Generation Networks." *IEEE Symposium on Security and Privacy*, 2011.

[16] C. Basescu, R. M. Reischuk, P. Szalachowski, A. Perrig, Y. Zhang, H.-C. Hsiao, A. Kubota, and J. Urakawa, "SIBRA: Scalable internet bandwidth reservation architecture," in *Proceedings of Symposium on Network and Distributed System Security (NDSS)*, Feb. 2016. [Online]. Available: http://www.scion-architecture.net/pdf/2016-SIBRA.pdf

[17] S. Shin, G. Gu, N. Reddy, and C. P. Lee, "A Large-Scale Empirical Study of Conficker," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 676–690, 2012.

[18] A. Khalimonenko and O. Kupreev, "Kaspersky labs q1 2017 ddos report," https://securelist.com/ddos-attacks-in-q1-2017/78285.

[19] "Dyn analysis summary of friday october 21 attack," https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack, 2016.

[20] "Dyn analysis summary of friday october 21 attack," https://nakedsecurity.sophos.com/2016/09/29/why-a-massive-ddos-attack-on-a-blogger-has-internet-experts-worried, 2016.

[21] T. B. J. Hawkinson, "Guidelines for creation, selection, and registration of an autonomous system (as)," United States, 1996. [Online]. Available: https://tools.ietf.org/html/rfc1930

[22] Y. Rekhter and T. Li, "A border gateway protocol 4 (bgp-4)," United States, 1995.

[23] G. Huston, "Bgp more specifics: routing vandalism or useful?" https://blog.apnic.net/2017/06/26/bgp-specifics-routing-vandalism-useful, 2017.

[24] "CAIDA AS relationship dataset," http://www.caida.org/data/active/as-relationships/index.xml, 2017.

[25] M. Lepinski and S. Kent, "An infrastructure to support secure internet routing," https://tools.ietf.org/html/rfc6480, United States, 2012.

[26] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper, "Routing around decoys," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 85–96. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382209

[27] M. Schuchard, A. Mohaisen, D. Foo Kune, N. Hopper, Y. Kim, and E. Y. Vasserman, "Losing control of the internet," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*. New York, New York, USA: ACM Press, 2010.

[28] P. Gill, M. Schapira, and S. Goldberg, "Let the market drive deployment: A strategy for transitioning to bgp security," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 14–25.

[29] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 75–86, 2011.

[30] "World bank global indicators," http://data.worldbank.org/indicator.

[31] "Peeringdb," https://www.peeringdb.com.

[32] "Global internet phenomena report," https://www.sandvine.com/trends/global-internet-phenomena.

[33] "Iana autonomous system (as) numbers," https://www.iana.org/assignments/as-numbers/as-numbers.xhtml.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[35] Netlab360, "Mirai Scanner," http://data.netlab.360.com/mirai-scanner/, 2017.

[36] M. Thomas and A. Mohaisen, "Kindred domains: Detecting and clustering botnet domains using dns traffic," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14 Companion. New York, NY, USA: ACM, 2014, pp. 707–712. [Online]. Available: http://doi.acm.org/10.1145/2567948.2579359

[37] RouteViews, "RouteViews Dataset," http://www.routeviews.org/.

[38] Mirkovic, J and Reiher, P, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication . . .*, 2004.

[39] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles." *IEEE/ACM Trans. Netw.*, 2005.

[40] Liu, X, Yang, X, and Lu, Y, "StopIt: Mitigating DoS flooding attacks from multi-million botnets," 2008.

[41] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network." *Computer Communication Review*, 2002.

[42] J. Ioannidis and S. M. Bellovin, "Implementing Pushback - Router-Based Defense Against DDoS Attacks." *NDSS*, 2002.

[43] C. Dixon, T. E. Anderson, and A. Krishnamurthy, "Phalanx - Withstanding Multimillion-Node Botnets." *NSDI*, 2008.

[44] J. C. Y. Chou, B. Lin, S. Sen, and O. Spatscheck, "Proactive Surge Protection: A Defense Mechanism for Bandwidth-Based Attacks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1711–1723, 2009.

[45] L. Von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Communications of the ACM*, vol. 47, no. 2, pp. 56–60, 2004.

[46] T. Spyridopoulos, G. Karanikas, T. Tryfonas, and G. Oikonomou, "A game theoretic defence framework against DoS/DDoS cyber attacks." *Computers & Security*, 2013.

[47] H. S. Bedi, S. Roy, and S. Shiva, "Game theory-based defense mechanisms against DDoS attacks on TCP/TCP-friendly flows," *. . . in Cyber Security (CICS)*, 2011.

[48] W. Zhou, W. Jia, S. Wen, Y. Xiang, and W. Zhou, "Detection and defense of application-layer DDoS attacks in backbone web traffic," *Future Generation Computer . . .*, vol. 38, pp. 36–46, Sep. 2014.

[49] D. Zhao, I. Traore, B. Sayed, W. Lu, and S. Saad, "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & . . .*, vol. 39, pp. 2–16, 2013.

[50] P. Senthilmahesh, S. Hemalatha, P. Rodrigues, and A. Shanthakumar, "Ddos attacks defense system using information metrics," 2013.

[51] IBTimes, "Why Microsoft and Sony couldnt stop Lizard Squad attack despite warnings," https://tiny.utk.edu/PutqC, 2014, accessed: 17 January 2017.

[52] ONF, "Open datapath," https://www.opennetworking.org/projects/open-datapath.

Appendix

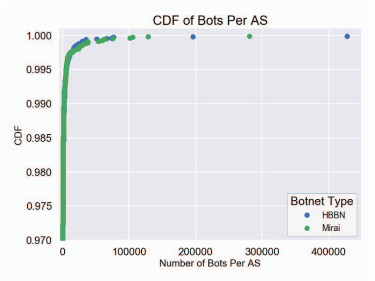


Fig. 12: Bot count per AS in the Mirai botnet and the Conficker botnet

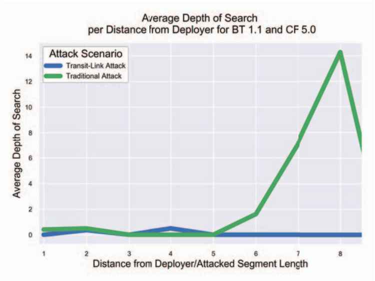


Fig. 13: Average depth of search for the hardest setting of bandwidth tolerance and congestion factor

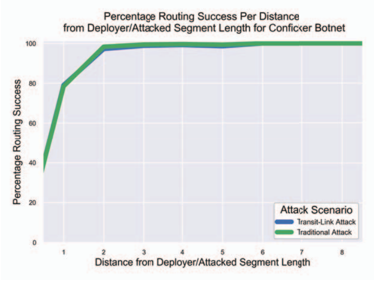


Fig. 14: Percentage routing success for both attack scenarios for the conficker botnet

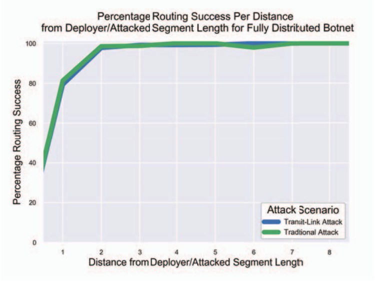


Fig. 15: Percentage routing success for both attack scenarios for the fully distributed botnet

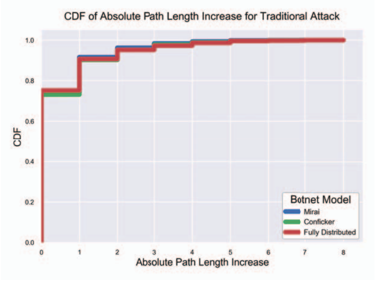


Fig. 16: Absolute path length increase for traditional attack

TABLE I: Information Needed by Nyx

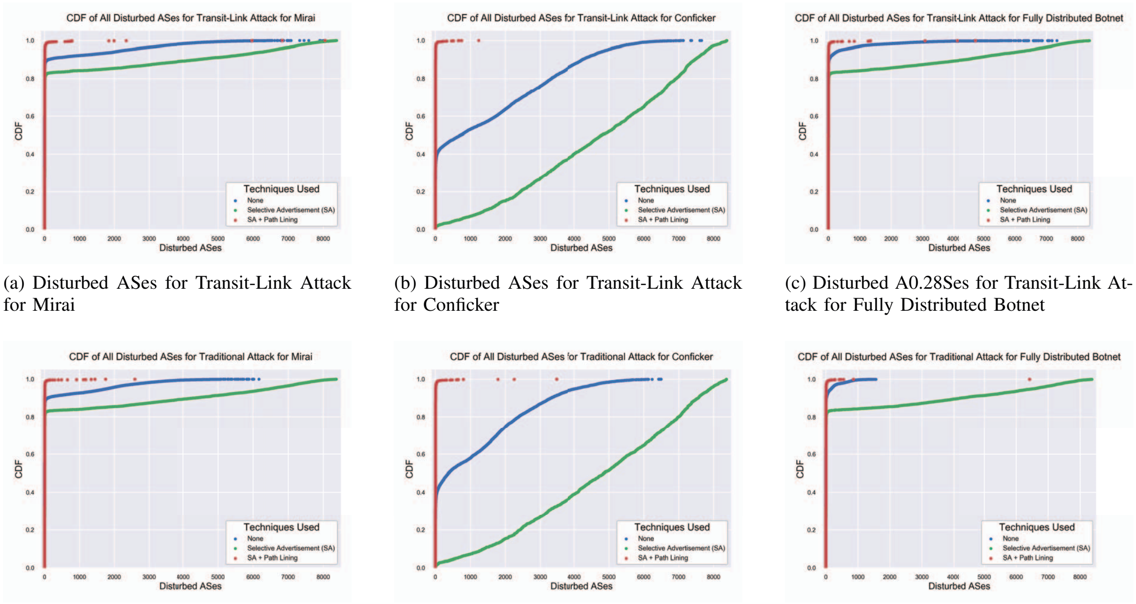
Information Needed	**How Nyx Uses Information**	**Information Source**
Critical AS	Traffic from Critical AS moved around degraded or attacked links	Chosen by Deployer AS
Paths between Deployer AS and Critical AS	Alternate, non-degraded paths between Critical AS and Deployer AS chosen based on any known paths	Deployer BGP speaker's Routing Information Base (RIB)
Packet flow performance	Used to detect service degradation due to DDoS event or adverse network conditions over alternate paths	OpenFlow [52]
ASes bordering alternate paths between Deployer AS and Critical AS	BGP loop detection is used during FRRP to reduce disturbance by appending ASes bordering alternate paths	Deployer BGP speaker's Routing Information Base (RIB) and Inferred AS Relationships Data from CAIDA [24]

TABLE II: Information **Not** Needed by Nyx

Information Not Needed	**How Nyx Works Without**
Bandwidth/Capacity of links in the Internet	Packet flow performance used as a proxy for congestion
Location of malicious bots and botnets in the Internet	Nyx continually discovers alternate paths until a path with sufficient capacity is found, without ever knowing the attack sources
Malicious and benign traffic	Nyx considers traffic from critical AS, known ahead of time as, ”benign”, without needing to know malicious traffic

Information Used by Simulator	**Use of Information**	**Revealed to Nyx**	**Information Source**
AS Relationships	Simulator needs to model the interaction of all known ASes, and Nyx needs to know ASes bordering the chosen alternate paths during path lining	YES	CAIDA AS Relationships [24], Route Views Project [37]
Inferred Bandwidth Model	Simulator uses as the primary bandwidth model to calculate congestion factors for links in the topology during simulation, contains mapping of AS to a ”traffic factor” for how much traffic that AS sends	NO	CAIDA AS Relationships [24], PeeringDB [31], IANA [33], World Bank [30], Sandvine [32], Labovitz *et al.* [29], Gill *et al.* [28]
AS Degree Bandwidth Model	Used as secondary bandwidth model for validation, contains a mapping between every AS to it's degree (number of connected ASes)	NO	CAIDA AS Relationships [24]
AS Total IP Count Bandwidth Model	Used as secondary bandwidth model for validation, contains a mapping of every AS to the number of total IPs known to live inside that AS based on traceroutes from RIPE Atlas	NO	Route Views Project [37]
Mirai Botnet Model	Botnet model used for attack traffic origination based on the Mirai botnet between August 2016 and June 2017, contains ASes with the number of Mirai infections within them	NO	Netlab360 [35]
Conficker Botnet Model	Conficker model used for attack traffic origination based on the Conficker botnet as measured between 2012 and 2013, contains ASes with the number of Conficker infections within them	NO	Thomas *et al.* [36]
Fully Distributed Botnet Model	Botnet model used for attack traffic origination where every AS except the current deployer and critical AS contain bots	NO	CAIDA AS Relationships [24]
Malicious Traffic	Traffic from bot ASes is sent from the originating bot ASes to other ASes such that their traffic flows over the simulator's currently attacked link (upstream of the Deployer AS), or in the traditional DDoS scenario, targets the Deployer AS directly	NO	Botnet Models

TABLE III: Information needed by the simulator



(a) Disturbed ASes for Transit-Link Attack for Mirai

(b) Disturbed ASes for Transit-Link Attack for Conficker

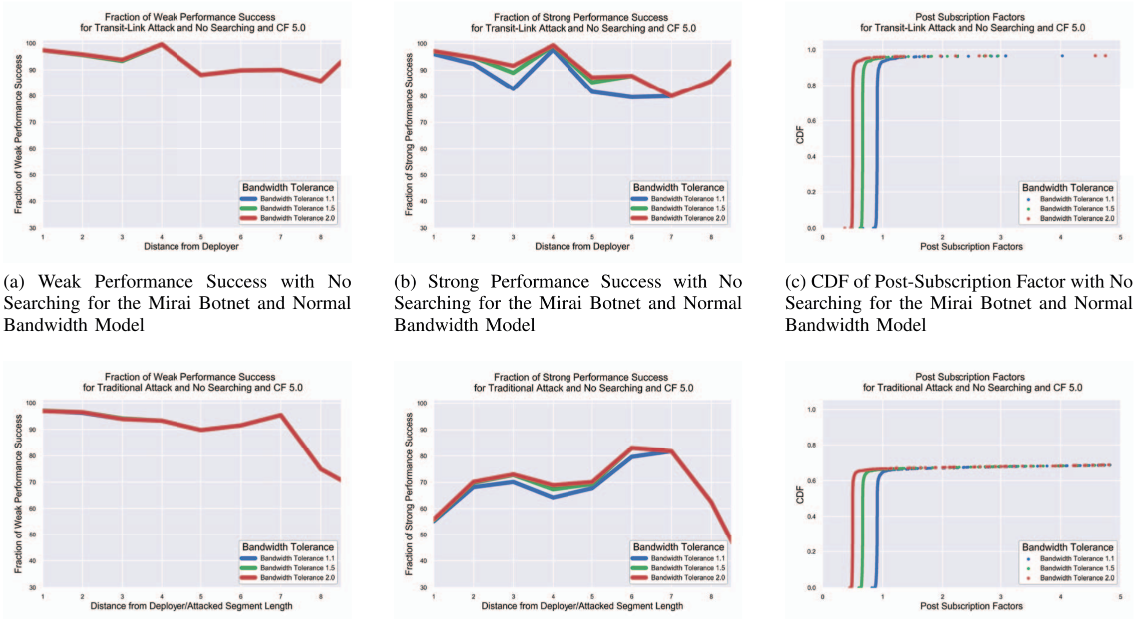
(c) Disturbed A0.28Ses for Transit-Link Attack for Fully Distributed Botnet

(d) Disturbed ASes for Traditional Attack for Mirai

(e) Disturbed ASes for Traditional Attack for Conficker

(f) Disturbed ASes for Traditional Attack for Fully Distributed Botnet

Fig. 17: Disturbed ASes with and without disturbance mitigation for all botnet models for the traditional attack scenario



(a) Weak Performance Success with No Searching for the Mirai Botnet and Normal Bandwidth Model

(b) Strong Performance Success with No Searching for the Mirai Botnet and Normal Bandwidth Model

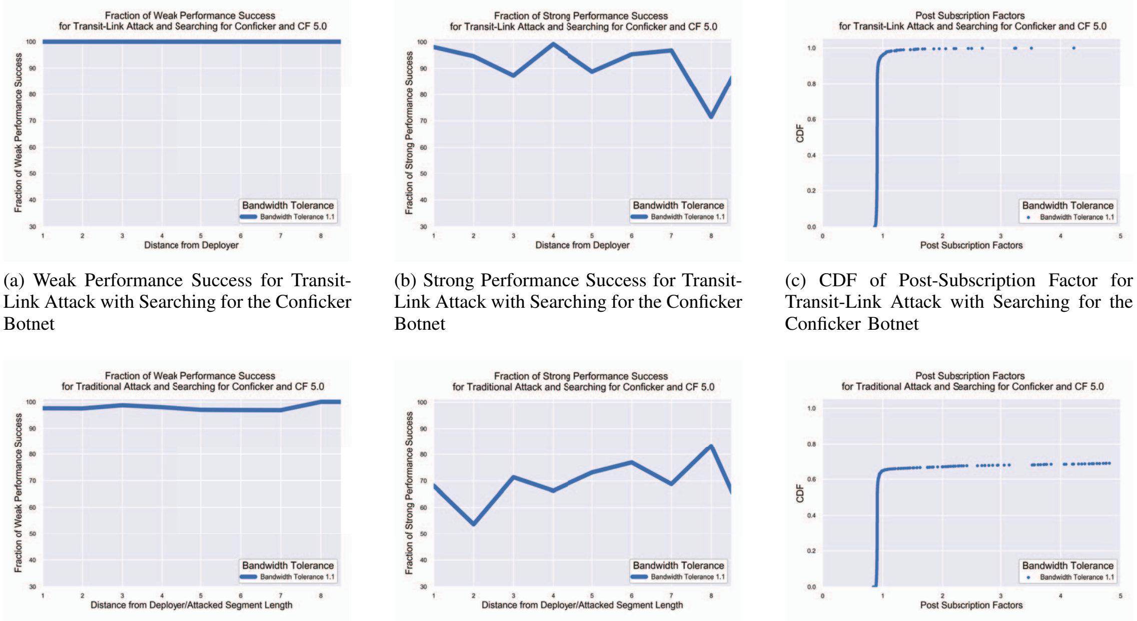
(c) CDF of Post-Subscription Factor with No Searching for the Mirai Botnet and Normal Bandwidth Model

(d) Weak Performance Success with No Searching for the Mirai Botnet and Normal Bandwidth Model

(e) Strong Performance Success with No Searching for the Mirai Botnet and Normal Bandwidth Model

(f) CDF of Post-Subscription Factor with No Searching for the Mirai Botnet and Normal Bandwidth Model

Fig. 18: Performance success metrics for the traditional and transit-link attack scenarios for Mirai without searching



(a) Weak Performance Success for Transit-Link Attack with Searching for the Conficker Botnet

(b) Strong Performance Success for Transit-Link Attack with Searching for the Conficker Botnet

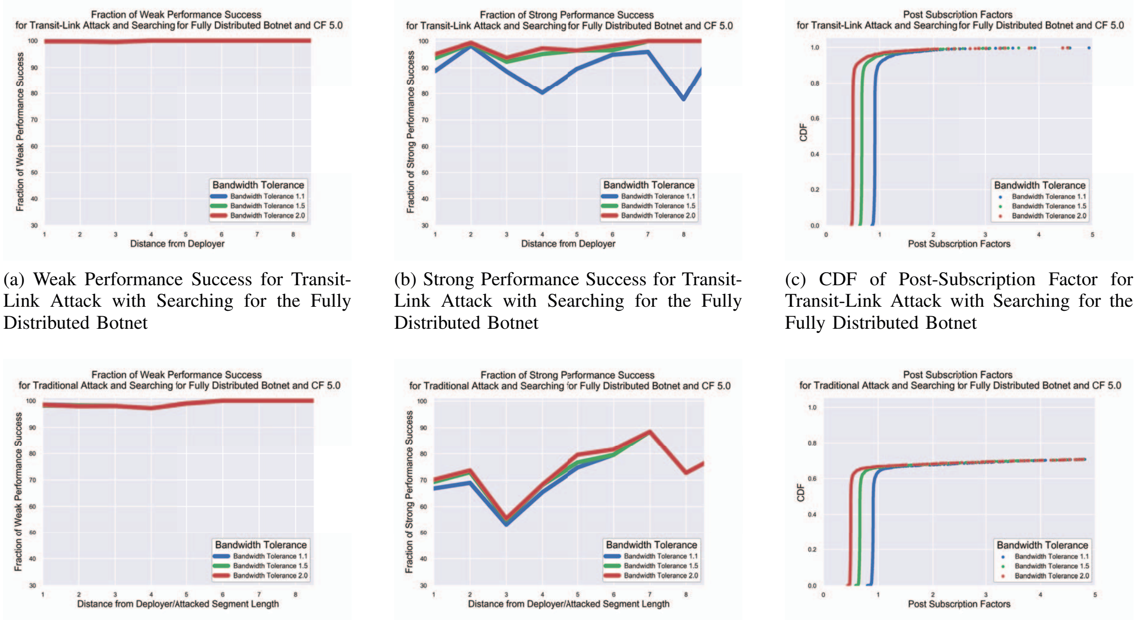
(c) CDF of Post-Subscription Factor for Transit-Link Attack with Searching for the Conficker Botnet

(d) Weak Performance Success for Traditional Attack with Searching for the Conficker Botnet

(e) Strong Performance Success for Traditional Attack with Searching for the Conficker Botnet

(f) CDF of Post-Subscription Factor for Traditional Attack with Searching for the Conficker Botnet

Fig. 19: Performance success metrics for the traditional and transit-link attack scenarios, normal bandwidth model, with searching for the **Conficker** botnet



(a) Weak Performance Success for Transit-Link Attack with Searching for the Fully Distributed Botnet

(b) Strong Performance Success for Transit-Link Attack with Searching for the Fully Distributed Botnet

(c) CDF of Post-Subscription Factor for Transit-Link Attack with Searching for the Fully Distributed Botnet

(d) Weak Performance Success for Traditional Attack with Searching for the Fully Distributed Botnet

(e) Strong Performance Success for Traditional Attack with Searching for the Fully Distributed Botnet

(f) CDF of Post-Subscription Factor for Traditional Attack with Searching for the Fully Distributed Botnet

Fig. 20: Performance success metrics for the traditional and transit-link attack scenarios, normal bandwidth model, with searching for the **fully distributed** botnet