12-2017

Taking Back the Internet: Defeating DDoS and Adverse Network Conditions via Reactive BGP Routing

Jared Michael Smith
University of Tennessee

Taking Back the Internet: Defeating DDoS and Adverse Network Conditions via Reactive BGP Routing

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Jared Michael Smith

December 2017

For Mom, Dad, Noah, and Kaleigh

I owe everything to them

Acknowledgments

I would like to thank my advisor, Dr. Max Schuchard, for pushing me to always go further, try harder, and solve hard problems. I would like to thank the VolSec research group at UTK, including Joseph, Jordan, Parker, and Savannah for always being willing to listen to me voice my frustration. I would like to thank the members of my committee: Dr. Bryan Lyles at ORNL for his mentorship and guidance in my work at ORNL and UTK, Dr. Audris Mockus for supporting me in my early research efforts as an undergraduate, and Dr. Mark Dean. I would like to thank Aziz Mohaisen from the University of Central Florida for providing the Conficker botnet data, and 360.cn and the Netlab360 team for providing the data on the Mirai botnet. Finally, I would like to thank Dr. Leon Tolbert, Dr. Wayne Davis, and the graduate school for providing me with the Tennessee Fellowship for Graduate Excellence to focus on this work as much as possible.

Abstract

In this work, we present Nyx, a system for mitigating Distributed Denial of Service (DDoS) attacks by routing critical traffic from known benign networks around links under attack from a massively distributed botnet. Nyx alters how Autonomous Systems (ASes) handle route selection and advertisement in the Border Gateway Protocol (BGP) in order to achieve isolation of critical traffic away from congested links onto alternative, less congested paths. Our system controls outbound paths through the normal process of BGP path selection, while return paths from critical ASes are controlled through the use of existing traffic engineering techniques. To prevent alternative paths from including attacked network links, Nyx employs strategic lying in a manner that is functional in the presence of RPKI. Our system only exposes the alternate path to the networks needed for forwarding and those networks' customer cones, thus strategically reducing the number of ASes outside of the critical AS that receive the alternative path. By leaving the path taken by malicious traffic unchanged and limiting the amount of added traffic load placed on the alternate path, our system causes less than 10 ASes on average to be disturbed by our inbound traffic migration.

Nyx is the first system that scalably and effectively mitigates transit-link DDoS attacks that cannot be handled by existing and costly traffic filtering or prioritization techniques. Unlike the prior state of the art, Nyx is highly deployable, requiring only minor changes to router policies at the deployer, and requires no assistance from external networks. Using our own Internet-scale simulator, we find that in more than 98% of cases our system can successfully migrate critical traffic off of the network segments under transit-link DDoS. In over 98% of cases, the alternate path provides some degree of relief over the original path. Finally, in over 70% of cases where Nyx can migrate critical traffic off attacked segments, the new path has sufficient capacity to handle the entire traffic load without congestion.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Do to their high level of impact, combined with low degree of technical complexity, Distributed Denial of Service, or DDoS, attacks continue to represent one of the largest unsolved persistent threats on the Internet. Recent successful DDoS attacks by the Mirai botnet against root DNS providers [39] and core transit links [3] highlight both the lack of an effective deployed solution to DDoS attacks and the impact such attacks have on critical network infrastructure. To make matters worse, increased botnet bandwidth has allowed adversaries to launch attacks against shared transit links located *outside* of the intended victim, rather than directly against the victim's end hosts, an attack methodology proposed in academic research by Kang [18] and Studer [38], which we call *transit-link DDoS*.

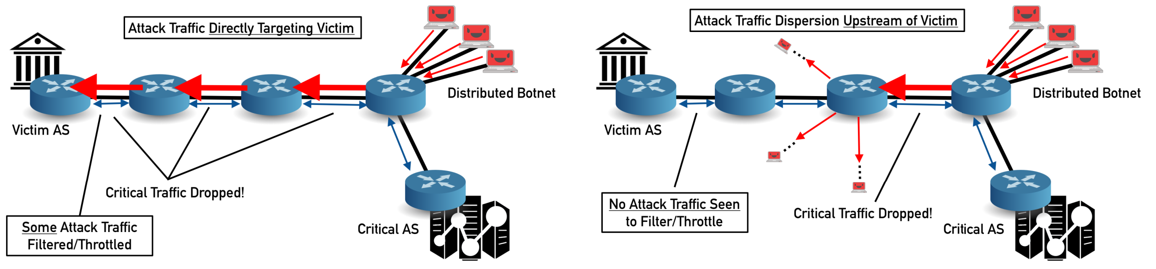
While DDoS represents one of the oldest and most well known security problems facing the Internet, research has yet to propose a solution that both provides effective mitigation against transit-link DDoS attacks **and** has a realistic deployment scenario. For example, filtering and prioritization techniques [36, 8, 44, 26, 23, 43] require costly per-stream calculations, presenting scalability concerns with modern DDoS attacks. Load balancing and CDN backed solutions [11] become a test of who possesses more bandwidth, the defender or the adversary, a tenuous proposition in an era of multi-Tbps attack flows, something the Mirai botnet and it's variants have repeatedly achieved. More importantly, *none of these defenses are capable to mitigate DDoS attacks launched against the Internet's transit infrastructure.* Attacks such as Kang's Crossfire are outside of the threat model considered by current DDoS defenses, which focused on protecting the last-mile links and provide no protection for transit links.

Systems such as SCION and SIBRA [46, 6], which integrate DDoS defense into the transit fabric of the Internet, present promise but require a complete redesign of the Internet, raising doubt about their deployability in the foreseeable future.

In this work, rather than considering DDoS mitigation as a filtering or prioritization problem, we approach DDoS mitigation as a **routing problem**. In our system, called **Nyx**, the defending or deploying network, specifically a multi-homed Autonomous System (AS), isolates 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 the DDoS attack. The inbound critical traffic will be routed to non-attacked paths with sufficient capacity using *currently deployed routing protocols*, specifically the Border Gateway Protocol (BGP). This approach to DDoS mitigation has several advantages over existing approaches. Instead of filtering, Nyx instead operates at the route selection level, avoiding costly per-stream decisions. Our system functions independently of the location of the link actually being attacked, even if that link is outside of the deployer's directly connected links, allowing our system to mitigate the impact of DDoS attacks against transit providers the deployer depends on that would normally be outside of the deployer's control. Since our system prevents malicious traffic from DDoS attacks and benign traffic from Critical ASes from being co-located, our capacity to successfully mitigate a DDoS attack in *not dependent* on the volume of malicious traffic, allowing our system to succeed against today's large-scale DDoS attacks, often reaching sustained traffic levels of 1 Tbps or more, which no known filtering mechanisms can handle. **Lastly, our system functions using existing routing protocols and protects traffic to and from the deployer without outside assistance**, allowing for a realistic deployment scenario of our system, unlike prior work presented to combat the problem of transit-link DDoS.

In order to realize Nyx, we address three key challenges. First, we address how the AS deploying Nyx, which we also call the *reactor or deployer AS*, can successfully maneuver both outgoing and *incoming* traffic off of attacked links. While altering outgoing paths is trivial, BGP provides the destination no direct way to control incoming paths. We overcoming this limitation through the use of currently existing traffic engineering mechanisms, while controlling path propagation via strategic lying about the networks on a path in an effort to trigger loop-detection. Our traffic engineering technique, which we call *Fradulent Route Reverse Poisoning*, works in the presence of deployed RPKI and is discussed later in Section 3. Our solution causes more preferable paths, with respect to packet forwarding, to propagate around, but never actually reaching, the links under a DDoS attack, which we can detect via observing loss of quality service on the links connected to the deployer when the deployer does not actually observe attack traffic directly.

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 or ASes; a property we term *disturbance*. Disturbance can result in two undesired scenarios. First, disturbance can result in malicious traffic also flowing along the alternate path, resulting in the alternate path itself suffering a DDoS attack. Second, even if the disturbed networks are not sources of attack traffic, too much traffic from ASes other than the chosen critical ASes 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 propagation of the path to all networks outside of the critical network and the networks appearing along the alternative path.


Attack Traffic Directly Targeting Victim
Victim AS
Distributed Botnet
Critical Traffic Dropped!
Some Attack Traffic Filtered/Throttled
Critical AS
Attack Traffic Dispersion Upstream of Victim
Victim AS
Distributed Botnet
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, thus nullify the effects of filtering and throttling techniques employed by the victim

Figure 1.1: Traditional and Transit-Link DDoS

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, it will attempt to search for a different alternative path. It accomplishes this by withdrawing the alternative path and attempting to re-propagate it, avoiding propagating the route to both links under DDoS attack and the bottleneck links in the previous alternative path. Nyx **does not require** knowledge of either the malicious traffic sources (i.e. the ASes containing malicious bots) or the actual capacity of upstream links to find alternate paths not under attack.

We demonstrate the ability of Nyx to accomplish all three of these tasks using Internet scale simulations in which our system attempts to mitigate a variety of DDoS attack scenarios. We find that it is possible to move critical traffic off attacked links and onto functional paths in 78% of cases where the primary link connecting the deploying AS to the Internet is attacked, which we call *Traditional DDoS* and greater than 98% of all other cases where the attacked link is upstream of the deployer, which we call **Transit-Link DDoS** as illustrated by Kang and Studher with Crossfire and Coremelt [18, 38]. 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 out of the our alternative paths **provide some degree of relief from the DDoS attack in 98% of cases**, and we find that we can move the critical traffic impacted by DDoS attacks onto **completely uncongested paths with in at least 70% of the time**.

The rest of this paper is laid out as follows. Section 2 will provide relevant background on DDoS, our threat model, and BGP. Section 3 will present details of our system design, including design constraints, our approach to DDoS mitigation, and the mechanisms by which we realize our mitigation strategy. Section 4 will cover details of our simulation methodology and the results of simulations testing the viability of our system. Lastly, in Section 5 we will compare our system to other DDoS mitigation systems.

Chapter 2

Background

2.1 DDoS and Botnets

Volumetric Distributed Denial of Service attacks degrade the availability of a victim host by saturating links the host utilizes to send and receive network traffic. DDoS attack traffic is commonly generated by botnets, collections of compromised end hosts scattered across the Internet. To carry out such attacks in practice, botnets of unforeseen scales need to be available to adversaries. Previously, worms such as Conficker [35] spread via malware. However, recent attacks have leveraged the Mirai botnet [39], which largely consists of IoT-based devices.

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 of attacks. Monitoring organizations have reported an increase in overall DDoS incidents of 83% from 2015 to 2016 [19]. 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 [19] from 2015 to 2016, with Mirai generating over *1 Tbps* of malicious traffic on multiple occasions. Historically, traditional DDoS attacks are where the adversary directs bot traffic directly at the victim network, forcing traffic at the edge of the victim's network to be dropped and significantly degrade quality of service and is illustrated in Figure 1.1a.

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 1.1b. In the wild, transit-link DDoS has been seen in recent attacks on the major DNS provider Dyn [1], the prominent security journalist Bryan Krebs with KrebsOnSecurity [2], and the country of Liberia [3]. With transit-link DDoS, the adversary directs bot traffic upstream of the network that is the ultimate target, which causes traffic 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 [38] and the Crossfire attack [18]. 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, and in doing so can bring down connections to selected critical servers in the transit-core simply by congesting their available capacity. Our system, Nyx, addresses transit-link DDoS directly, alleviating congestion from attacks for a single critical AS in the majority of cases, and will be discussed in depth in the next section, Section 3 with evaluation in Section 4.

2.2 Inter-Autonomous System Routing via BGP

Before we discuss how Nyx operates, we must review the Internet routing infrastructure, which despite holding up well for decades, is showing flaws not seen or mitigated when first designed.Today, the Internet is composed of many autonomous systems (or ASes), sets of routers and IP addresses each under singular administrative control [17]. Between ASes on the Internet, the Border Gateway Protocol [30] (BGP) is the de facto routing protocol. It 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 used to forward packets. These policies frequently extend beyond simply choosing the fastest or shortest routes: they 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" [30, 13]. 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. Additionally, there is no currently deployed mechanism to prevent a router from falsifying route properties, and we discuss getting around deployed RPKI in Section 3.2.

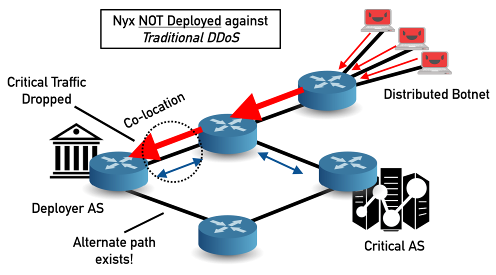
Chapter 3

System Design

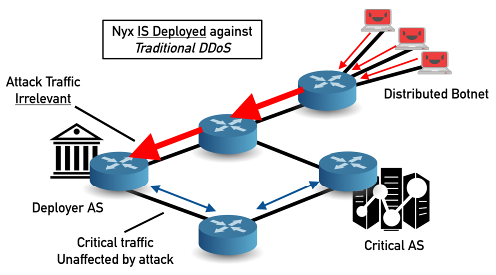
3.1 Routing Around DDoS

To combat the unmitigated threat posed by transit-link DDoS, we have designed **Nyx**, a system that mitigates DDoS attacks by routing traffic between a Nyx deployer and chosen critical AS known ahead of time, around links degraded by a DDoS attack or other adverse network conditions. By operating on a per-route basis, rather than a costly per-stream basis, Nyx utilizes BGP at the deployer to route around DDoS without adversely affecting the routing information of other ASes *and* by moving traffic inbound for the critical AS onto new paths with sufficient capacity to handle the added load. At a high level, Nyx makes attack traffic from botnets *irrelevant* achieving the property of **botnet-level independence**. The ability of Nyx to route around DDoS and make attack flows irrelevant is illustrated in Figure 3.1 for Traditional DDoS and Figure 3.2 for Transit-Link DDoS.

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, Crossfire and Coremelt [18, 38], as well as real-world attacks seen against Liberia [3], do not send their attack traffic directly to the targeted AS, thus eliminating the possibility of applying any filtering or prioritization technique to incoming traffic since critical traffic is dropped upstream and typically outside the control of the victim AS. Nyx approaches the problem differently, by focusing on the problem of *route selection*, utilizing normal BGP and traffic manipulation techniques to **route around DDoS**. By continually

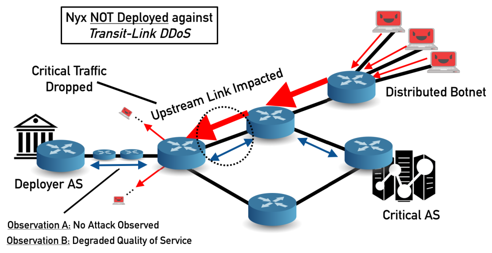


(a) Nyx NOT Deployed

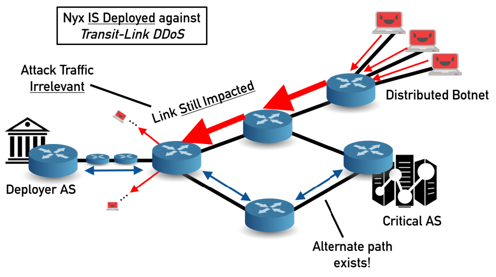


(b) Nyx IS Deployed

Figure 3.1: Nyx Deployment Against **Traditional DDoS**



(a) Nyx NOT Deployed



(b) Nyx IS Deployed

Figure 3.2: Nyx Deployment Against **Transit-Link DDoS**

selecting alternate paths with the ability to handle traffic otherwise due to be dropped on congested links, Nyx does not rely on existing filtering or prioritization techniques.

3.1.1 Realistic Deployment

Unlike prior systems which mitigate transit-link DDoS via bandwidth contracts [6], 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 [4]. In Tables 3.1 and 3.2 we show the information required and not required by Nyx. In detail, 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. Our system also does not have knowledge about the location of bots, which ASes have bots, and where in the internet bots live; instead, Nyx continues to use packet flow performance as an indicator that the current path between the critical AS and the deployer AS is congested. When Nyx discovers the current path is congested, we use our strategies to route around DDoS and attempt to find via an evolutionary algorithm an alternate path with sufficient capacity, as we will discuss later in Section 3.4. Finally, Nyx does not need to know what traffic is malicious or benign, since our system 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 due to Nyx's ability to route *around* links impacted by malicious botnet traffic, which gives Nyx the property of botnet-size independence when mitigating DDoS.

Beyond the information Nyx does and does not know, we make the following assumptions about the deployment of Nyx in practice:

- 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 means that our critical AS will **not** provide our defender any assistance, nor will any other ASes on the

Internet, which is a key feature of Nyx that distinguishes our system from any prior work proposing to mitigate transit-link DDoS.

- 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 advertisements on the routers that are BGP speakers for the deployer AS.
- Nyx should function without any changes to BGP, since the technique we have devised to manipulate inbound traffic from known critical ASes can be performed only via adjustment of routing policies at the deployer.

3.1.2 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 is *not routing-aware*, thus unable to make routing decisions. Instead, our threat model considers adversaries which control massive distributed botnets or a subset of hosts with the ability to generate massive 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 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 6 future work to address this issue. As mentioned earlier and shown in Table 3.2, 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.

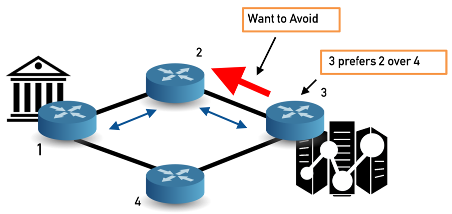
In the rest of this section, we will explore how Nyx achieves its three core goals within the design restrictions we have placed to ensure deployability and resistance to adversaries.

In Section 3.2 we will examine how Nyx adjusts incoming traffic to alternative paths, which is a functionality not controllable directly within BGP. When Nyx successfully migrates critical traffic off of links suffering DDoS we call this **routing success**, and will discuss our evaluation of routing success later in Section 4.5. Next, in Section 3.3 we look at how Nyx reduces **disturbance**, where ASes outside the the critical AS and those along the alternative path switch to the alternative path. Finally, in Section 3.4, we establish how Nyx attempts to maximize the number of instances of where the new link has sufficient capacity to handle the critical traffic.

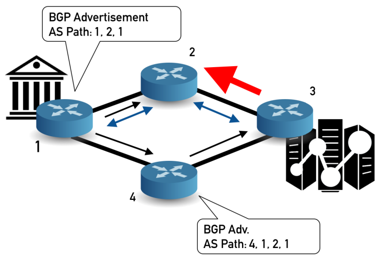
3.2 Migrating Critical Traffic

Recall from earlier in Section 2.2 that outbound traffic from an AS is trivial to adjust via local preferences at the external BGP router; however, manipulating the paths inbound traffic takes to an AS would typically only be possible via coordination between the ASes on either end, as existing systems such as SCION and SIBRA do to route around DDoS [46, 6]. Nyx, however 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. We address this issue by giving the deployer AS the ability to *restrict* the AS-level paths the critical AS can take to the deployer to only paths which *do not traverse the congested or attacked links* within the topology, such as those affected by traditional or transit-link DDoS attacks. We do 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 strives to *route around DDoS* as illustrated in Figures 3.1 and 3.2, where we show how Nyx makes attack events and congested links irrelevant, as critical traffic headed to the deployer is forced onto uncongested, alternate paths.

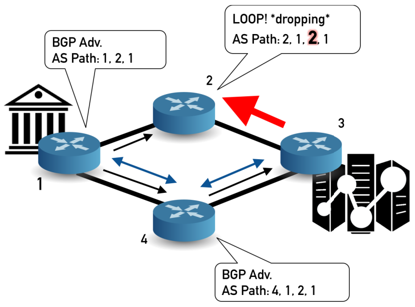
To give the deployer this ability, we have developed a strategy used by Nyx called **Fraudulent Route Reverse Poisoning** (FRRP). Nyx employs FRRP to ensure that any BGP advertisements which propagate to the critical AS, originated by the deployer AS, are guaranteed to *not* traverse links that are congested or under attack from DDoS or adverse conditions such as broken links or surges in bandwidth usage creating congestion. FRRP takes away the choice of the critical AS to route outbound traffic headed to the critical over the attacked links by ensuring advertisements which originate at the deployer do not reveal the paths with attacked links.



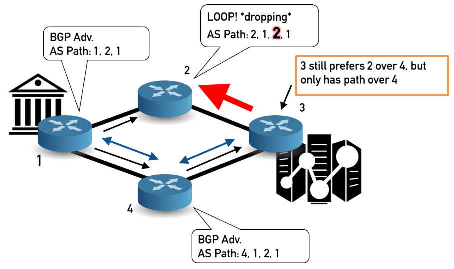
(a) Critical links are congested



(b) Lying about paths and appending ASes to avoid



(c) Loop detection



(d) Critical AS now traverses alternate path

Figure 3.3: Fraudulent Route Reverse Poisoning

In detail, FRRP is illustrated in Figure 3.3 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 3.3a). 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 it to AS 3 (as shown in Part 3.3b). 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 it's route to AS 1 (shown by Part 3.3c). 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 (shown in Part 3.3d).

Nyx utilizes FRRP at an *Internet-scale* to migrate incoming traffic from a chosen critical AS onto alternate paths in situations where many alternate paths exist.

By using FRRP, we achieve over 98% success for the ability to move traffic off of links under DDoS. Figure 3.1 shows Nyx both deployed and not deployed against a traditional DDoS attack, and Figure 3.2 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.

FRRP under RPKI

When utilizing FRRP, properly deployed resource public key infrastructure (RPKI), also known as Resource Certification, would typically prevent advertising false routes [21]. However, Nyx addresses RPKI's effects on FRRP by ensuring that strategic lying in order to trigger loop detection does not interfere with the route origination process. In detail, given an originating autonomous system, AS_{orig} and a set of ASes to blacklist, $BL_{AS} = \{\ AS_{BL_1}, AS_{BL_2}, \ldots, AS_{BL_N}\}$ where $AS_{orig} \notin BL_{AS}$, the deployer (the originator in this case) advertises the following path when using FRRP:

$$\{\ AS_{orig},\ AS_{BL_1},\ AS_{BL_2}, \ldots,\ AS_{BL_N},\ \underbrace{AS_{orig}}_{\text{For RPKI}}\} \tag{3.1}$$

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

$$\{\ \underbrace{AS_3,\ AS_2,\ AS_1,}_{\text{Actual Path}}\ \overbrace{AS_{orig}}^{\text{Packet at Dest}}\ , \underbrace{AS_{BL_1}, \ldots,\ AS_{BL_N}, AS_{orig}}_{\text{Irrelevant for Forwarding}}\} \tag{3.2}$$

This means that when routes are advertised by the originator in Equation 3.1, 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 3.2, ASes along the path will continue to forward the route as long as the originator remains in the path. The blacklisted 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. Since an AS will simply scan the entire path for it's own AS number and upon finding itself in the path, it will drop the path.

FRRP and Network Connectivity

In order to maintain network connectivity, the deployer still advertises it's normal paths, but the FRRP paths will be hole-punched prefixes as discussed earlier in Section 2.2. 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 any connectivity to 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 3.4. Before exploring this issue, we first examine the ability of Nyx to reduce the side-effects of utilizing FRRP in the next section.

3.3 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, observing what the most preferable alternative path from critical AS to the deployer is. We then withdraw the FRRP path and re-advertise it only to the AS directly connected to the deployer AS on the preferred alternative path.

- Path Lining: Using the preferred alternative path, we utilize FRRP to blacklist every AS adjacent to the path and their customer cone, but not the ASes along the path. When the blacklisted ASes see the FRRP-originated advertisement, they drop the new path due to loop detection in the same way that FRRP was used to avoid the attacked links due to DDoS. By halting the propagation of our alternate path, disturbance is 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 wishes to blacklist in it's fraudulent advertisements.

In our evaluation, to be discussed in Section 4, selective advertisement alone actually increases the disturbance caused by FRRP, a byproduct of how the path propagates through the topology. Path lining *does*, however prevent disturbance, 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 blacklist via 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 4.6.

3.4 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 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 its 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 end up 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* method 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, and then evaluate if congestion is being experienced. This searching is an *evolutionary algorithm*, where the fitness function is packet flow performance over each alternate path. When searching, if the alternative path is experiencing congestion, Nyx withdraws the alternative route and repeats the FRRP and path lining process, but additionally treats the hops along the former alternative path as if they are experiencing DDoS as well, thus blacklisting them and causing the critical AS to not route traffic ot the deployer AS over the insufficient alternate paths. In other words, Nyx repeats the alternative path generating process, avoiding ASes experiencing DDoS and those who have failed to provide a performant alternative path.

Table 3.1: 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 [1]
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 [4]

Table 3.2: 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

Chapter 4

Evaluation

4.1 Simulation Methodology

To evaluate the effectiveness of our system, we built our own BGP simulator, which has been used in prior work by Schuchard et. al [1] [33]. The simulator is essentially a collection of software routers who speak BGP configured in a realistic topology. The topology used in the simulation is from CAIDA's AS relationships dataset taken from December of 2016 [4]. The BGP policies used by the simulated routes match the current best practices used by operators. Additionally, we have used three bandwidth models, covered in Section 4.2, which are used to calculate link capacities, and two botnet models, which are used to calculate attack volumes available to each attacking AS throughout our simulation. We will show later in Section 4.7.2 that our system is resilient to changes in these models.

Using our simulator, we can examine both the effectiveness and cost of a deployer using Nyx, which we refer to as the *deployer AS*, to migrate critical traffic off of links suffering from DDoS or adverse network conditions. Our experiment repeatedly picks two random ASes from the Internet's default-free zone, which are ASes that are not stub ASes, and fixes one of the ASes as the deployer AS and the other as an AS generating critical traffic (i.e. the critical AS mentioned earlier). We then simulate the deployer attempting to respond with Nyx to a DDoS attack that is impacting links on the current best known path between the deployer and critical AS.

[1] Source code at `https://volsec.eecs.utk.edu/nyx`

When simulating a DDoS attack, we measure the used capacity of links on the best path between the deployer and critical AS in two cases by simulating traffic flow through the Internet: (1) we measure the used link capacity after the attacks effects have congested a link but before we have used Nyx to migrate traffic off links, (2) we measure after we use Nyx to migrate traffic onto an ideally more performant path. We call the used link capacity for a given link the **subscription factor** of that link, and we calculate these values using a combination of our bandwidth model, which we will discuss in the next section, Section 4.2, two constant fixed values varied between simulations that we call the "Bandwidth Tolerance" and "Congestion Factor", and the number of IPs in bot ASes that are attacking the deployer AS per run. The number of IPs per bot AS is determined via the earlier mentioned botnet models we use, which we will discuss further in Section 4.3.

The aforementioned bandwidth tolerance and congestion factors for inform us of whether our system can hold up under varying attack strengths. The **bandwidth tolerance** for the link between any given AS pair is a constant value between 1 and 2 that describes how much additional capacity the 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 link is congested and may drop traffic flowing over it.

The **congestion factor** is a value that is specific to a simulation instance. The congestion factor informs the simulator to send an amount of traffic to the current link we are simulating an attack on that would put the traffic flowing over that link at such a congestion factor. In our simulation, a value of 1.0 for the amount of traffic on a link is the max capacity, and anything over 1.0 means it is congested. We vary our congestion factors between 2.0 and 5.0 over runs, in order to simulate a moderate amount of congestion and a significant amount of congestion.

In order to calculate the pre- and post-traffic migration subscription factors, we need to calculate normal traffic levels flowing over every link in the topology. Using our bandwidth model, we get a predicted level of traffic that should flow over each link, which is then multiplied by the bandwidth tolerance to give us the normal traffic values for that link. Using our botnet model to determine the magnitude of bots per AS, which we will discuss in in Section 4.3, we then direct bot traffic at the links between the critical and deployer AS in the case of transit-link DDoS, and at the deployer AS itself for traditional DDoS, by allocating traffic first to the ASes with the most bots. With the combination of the normal traffic over the deployer-critical links and the bot traffic from the DDoS attack impacting them, we calculate our pre-subscription factor as a value above 0.0, where less than 1.0 means the link is uncongested, and above 1.0 means the link is congested. After we utilize Nyx to move traffic off the impacted links and ideally onto more performant paths, we flow traffic again in our simulator and calculate the post-subscription factor, which we use to determine our **performance success** metric.

We use our congestion factor as a proxy for packet loss, and we use path length as a proxy for latency. Modeling latency on the Internet itself is extremely difficult for massively distributed systems; therefore, we adopt the common notion of using path length as a proxy metric for latency, since we can measure path length easily within our simulator since the simulator knows the current Internet topology. Nyx must also know the topology to find alternate paths via our evolutionary algorithm for capacity alleviation, where an individual AS can gather this data from known open source datasets updated frequently via organizations such as CAIDA [4] or gather this on its own via targeted traceroutes. Table A.1 in the Appendix shows a summary the information visible to our Internet simulator, and illustrates that Nyx and the deployer AS know very little in practice, which is shown in Tables 3.1 and 3.2 earlier in Section 3.

4.2 Bandwidth Model

We recognize that establishing a complete and irrefutable bandwidth model for the modern Internet is an unsolved problem without wielding large-scale collaboration from nearly all existing ASes; therefore, we have developed what we believe to be an accurate and generalized model that effectively allows us to assign bandwidth capacities to links on the Internet. In addition to this model, we have tested our system with two simpler models, one based on the degree of ASes and one on the total IPs associated with ASes, and we show that it works effectively with simpler models later in Section 4.7.2.

To achieve this, we need an Internet scale model of 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.* [12], supported by the measurements of Labovitz *et al.* [20], the World Bank [42], PeeringDB [29], and Sandvine [32]. We call this model the *Inferred Model*, as we have used known and reputable Internet-wide data to assign approximate traffic constraints to links in the Internet known solely to the *simulator*, and *not* by the Nyx deployer.

In order to establish the relative values of traffic leaving and entering ASes three data sets were combined. Sandvine provides the amount of bandwidth consumption from an "average" user in various regions [32]. 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 [42]. 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 [14]. We then consulted PeeringDB, which is a system that allows ASes to advertise their willingness to peer with other ASes [29]. ASes which elect to participate in PeeringDB have the ability to optionally disclose the average amount of inbound and outbound bandwidth from their AS that peers should expect. Of the roughly 55,000 ASes which exist in our , where our topology is built based on CAIDA's AS relationship dataset [4], just over 8,000 report bandwidth estimates exist. In order to establish relative bandwidth values between all ASes, a Decision Tree classifier was trained based on 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 using Scikit-Learn, a popular machine-learning framework [28]. The resulting classifier had a correlation coefficient of 0.89, indicating that the PeeringDB data combined with additional AS information models bandwidth estimates with high accuracy.

Again, we recognize that our inferred bandwidth model is not perfect, but currently no literature has established a model for bandwidth sufficient for approximating traffic levels across the entire Internet.

4.3 Botnet Model

For simulating attacks on links in our topology, we have *three* botnet datasets. The first dataset comprises 2.9 million unique Mirai [39] hosts, observed between August 2016 and June 2017, which we call our **Mirai Botnet** model. This model was gathered by a Chinese CDN with a passive scanner setup to detect connections from IPs on known Mirai ports [27]. Given that the Mirai botnet caused massive failures of systems on the Internet in transit-link attacks, as seen in the DynDNS attack and in Liberia being knocked offline [1, 3], we use this botnet so that we can simulate Nyx standing up against a DDoS attack generated by the same model in which nearly all modern DDoS defenses have recently failed to protect against. For Mirai, the distribution of bots among ASes reveals that the majority of bots are clustered in a relatively small number of ASes, as see in Figure A.1 as seen in the Appendix, with less than 50 bots in over 97% of ASes with at least one bot. Note, that in Figure A.1, the y-axis is trimmed to only show ASes with bot quantities of over 97% of the total botnet size.

The second is a dataset of 23 botnet families collectively known as Conficker, which were observed launching DDoS attacks between late August 2012 and March 2013 with a total of 2.2 million unique hosts [40]. This data was gathered by enumerating all of the botnets Command and Control domains in advance based on exposed code from the botnet variant, and then track the hosts contacting the domains. The distribution of the host-based botnet is nearly identical to Mirai as shown by Figure A.1 in the Appendix, with again less than 50 bots in over 97% of ASes.

The third and final botnet dataset is a *fully distributed botnet* where every AS in the topology, except for the current deployer and critical AS, is a bot AS with the ability to send malicious traffic. We use this final botnet model to prove that Nyx is able to mitigate transit-link and traditional DDoS even when facing a fully distributed adversary.

In our simulator, when we iterate over each link of the original path between the deployer and critical ASes, we direct the traffic of the bot ASes to the deployer side of the current link if and only if the bot ASes best paths to every other AS in the Internet travels over the current link, thus modeling the Crossfire attack by Kang *et. al.* [18] popularizing transit-link

DDoS by setting up an N^N amount of connections between bots, where N is the number of bots in the current botnet model that can flow over the currently attacked link. This ensures that out of all the bot ASes we have in each bot dataset, we only use the subset of bots that can direct traffic either flowing over the attacked transit-link, in the case of transit-link DDoS, or the deployer AS, in the case of traditional DDoS. In our evaluation, we show graphs from both datasets, illustrating Nyx's resiliency to choice of botnet model.

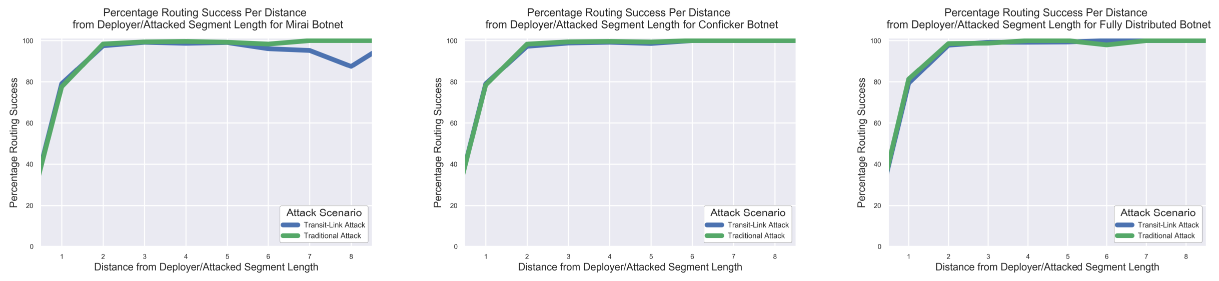
4.4 Attack Scenarios

In our simulation, we aim to mitigate attacks and protect the deployer AS in two distinct DDoS attack scenarios:

- **Transit-Link DDoS**: In this scenario, the bots in our dataset target links upstream of the deployer AS as described above. This is the primary scenario and most closely represents transit-link DDoS, where attacking ASes do not directly address attack traffic to the victim AS, and instead try to block up links outside of their direct sphere of influence. Success in moving traffic to less congested links in this scenario means that our system can effectively mitigate transit-link DDoS. Recall, the deploying AS *cannot filter or prioritize traffic when under transit-link DDoS*, as shown earlier in Figure 1.1b.
- **Traditional DDoS**: In this scenario the bots in our dataset directly target the deployer AS using the bots that actually have paths to the deployer. This scenario is the more difficult of the two scenarios, and success here is a major improvement to current DDoS defenses. Here, instead of measuring our routing success in terms of distance to the deployer, we are actually measuring 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. Now, we will explore our ability to migrate incoming traffic off their original paths, then show how we can migrate incoming traffic without disturbing significant numbers of neighboring ASes along the deployer-critical AS path, and finally reveal that we can migrate traffic off impacted links onto links that are less congested or totally uncongested relative to the original best path.

4.5 Can Nyx Migrate Traffic Onto Links Not Impacted by DDoS Attacks?



(a) Percentage routing success for both attack scenarios for the Mirai botnet.

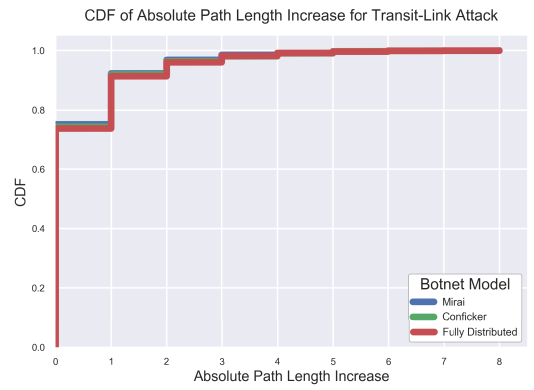
(b) Percentage routing success for both attack scenarios for the Conficker botnet.

(c) Percentage routing success for both attack scenarios for the Fully Distributed botnet.

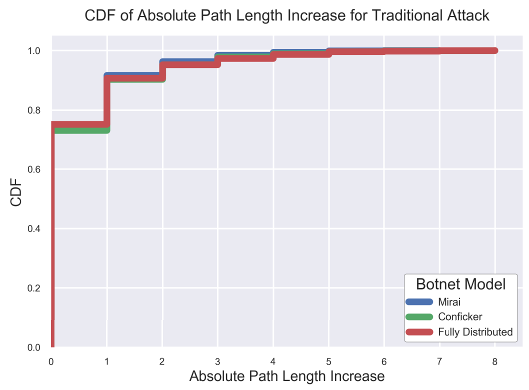
Figure 4.1: Routing Success for the Mirai, Conficker, and Fully Distributed Botnet models.

Nyx is able to find valid paths and *move incoming traffic onto around impacted links* with a great degree of success, which is the first step in mitigating transit-link and traditional DDoS of the volumes where current systems fail. We use our simulator to measure this result for both types of DDoS scenarios, and we label this result as **routing success**.

As shown in Figure 4.1a, our system achieved nearly 100% routing success over all simulation runs when using FRRP with selective advertisement and path lining 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



(a) Path Length Increase for Transit-Link Attack



(b) Path Length Increase for Traditional Attack

Figure 4.2: Path length increase for both attack scenarios for Mirai, Conficker, and Fully Distributed botnets.

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% at the extremely low end, 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 4.1b, we show that we can migrate incoming traffic off of nearly any arbitrary link in the Conficker model, and not only when under attack from the Mirai botnet. 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 under attack from a *globally distributed botnet*, as shown by Figure 4.1c.

4.5.1 Modeling Latency

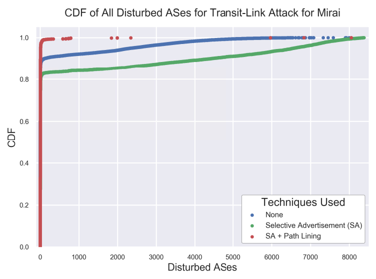
Recall that we use the widely accepted notion of path length increase as a proxy for increased latency while modeling our system. In practice, measuring the latency of chosen, alternate paths on the Internet depends heavily on the layer 1 technologies uses, such as the physical cables between ASes, as well as geographical distance between ASes, and the quality of the hardware running the BGP daemons.

We see path length increases of greater than 5 hops in only 2% of runs, and for 94% of runs, we see *no path length increase*, which is shown in Figure 4.2 regardless of the botnet model used. This is significant as well, because now we additionally do not cause incoming traffic to take longer paths to the deployer AS when traveling around impacted links due to the influence of Nyx. Figure 4.2 also shows the path length increase when using the Conficker and Fully Distributed botnet models, which is roughly equivalent and illustrates that Nyx is resilient to changes in the botnet model with respect to path length increases, even for a *globally distributed botnet*.

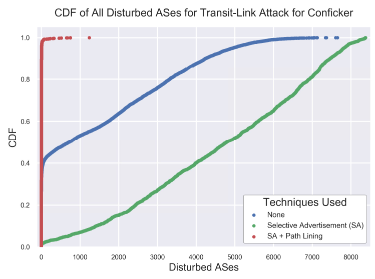
Routing success can be achieved independent of whether the new path is actually more congested than the original path, and routing success where the network congestion is alleviated on the new path is discussed later in Section 4.7. In the next section, we discuss how we address the second challenge described earlier in Section 3, disturbance mitigation.

4.6 Can Nyx Migrate Traffic Without Disturbing Other ASes?

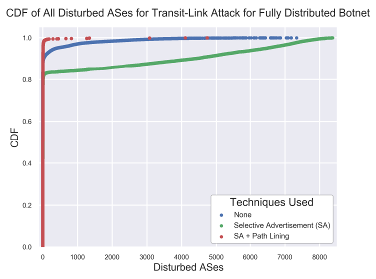
Despite being able to migrate incoming traffic onto new paths outside of the influence of a major DDoS attack, we discovered that the FRRP technique used by Nyx disturbed significant numbers of ASes. To overcome the problem of disturbance, we introduced two strategies in Section 3.3: selective advertisement and path lining. When utilizing those strategies in unison, we significantly lessened the disturbance to the ASes close to the deployer AS when Nyx was employed to migrate incoming traffic.



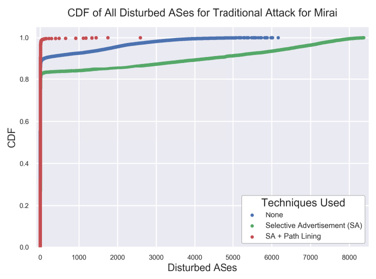
(a) Disturbed ASes for Transit-Link Attack for Mirai



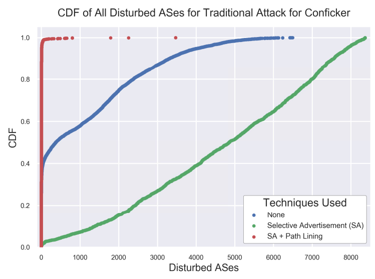
(b) Disturbed ASes for Transit-Link Attack for Conficker



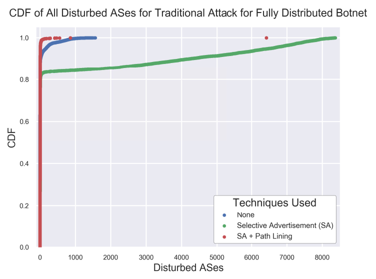
(c) Disturbed ASes for Transit-Link Attack for Fully Distributed Botnet



(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

Figure 4.3: Disturbed ASes with and without disturbance mitigation for all botnet models for Traditional and Transit-Link DDoS

As shown in Figure 4.3, before employing any strategies to mitigate disturbance, we 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. When we implemented selective advertisement alone, we did not see the disturbance drop, which indicated we needed to try another strategy. Then, we implemented path lining as described in Section 4.6, and brought the number of disturbed ASes to less than 10 disturbed ASes on average. Using path lining and selective advertisement, we 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 the IPs residing within them, as they may see new routes taken for traffic being sent. This is shown in Figure A.2 in the Appendix, since the story is roughly the same as the disturbance in ASes.

In summary, by employing our disturbance mitigation techniques, we are able bring the *costs of disturbance to virtually zero* in nearly 90% of cases. In the next section, we will discuss local preference disturbance.

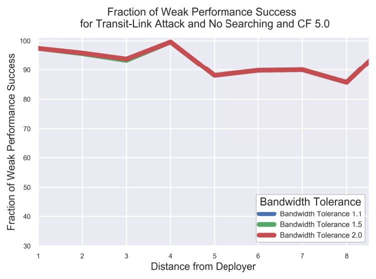
4.6.1 Are There Any Local Preference Changes?

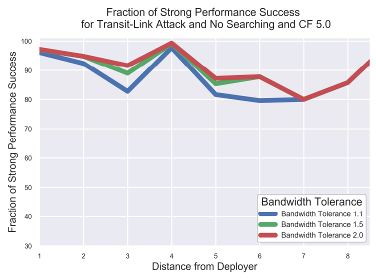
The cost of link usage from one provider to another provider in the actual Internet are closely guarded secrets; therefore, we use the act of switching onto a peer- or provider-learned path as a proxy for added monetary cost. In our simulations, the deployer AS using Nyx *never switches* from a customer learned path to peer- or provider-learned path, or a peer-learned path to a provider-learned path.

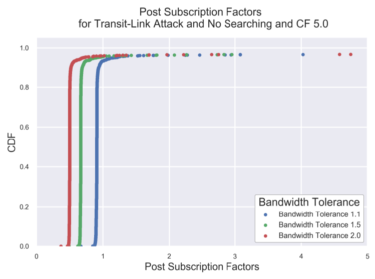
4.7 Do the Alternate Paths Have Enough Capacity?

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 migrate traffic onto more performant and uncongested paths in nearly all cases for transit-link DDoS and a majority of cases in

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



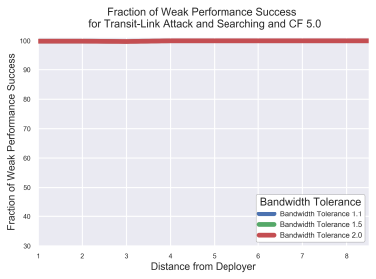


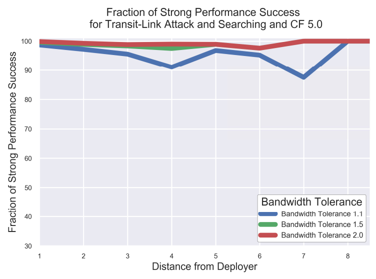


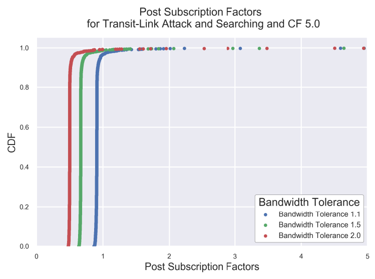
(a) Weak Performance Success with No Searching for the Mirai Botnet and Normal Bandwidth Model

(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 Searching for the Mirai Botnet and Normal Bandwidth Model

(e) Strong Performance Success with Searching for the Mirai Botnet and Normal Bandwidth Model

(f) CDF of Post-Subscription Factor with Searching for the Mirai Botnet and Normal Bandwidth Model

Figure 4.4: Performance success metrics for the transit-link attack scenario with and without searching.

traditional DDoS. In order to measure performant paths, we use several bandwidth tolerances and congestion factors, as discussed in Section 4.1. We additionally show that our system has the ability to search for performant paths as described in Section 3.4, which greatly enhances the success of migrating onto uncongested paths after a DDoS attack.

Our system achieves performance success in two distinct ways: when the post-subscription factor is less than the original subscription factor, and when the post-subscription factor is less than 1.0 (indicating that we have completely alleviated congestion from either an original subscription factor of 2.0 or 5.0 to less than 1.0). The bandwidth tolerances are between the range of 1.0 and 2.0, which model links where the capacity might actually have more room than our bandwidth model dictates, and this tolerance is applied across all of our simulations.

As shown in Figure 4.4, when the deployer AS is under transit-link DDoS, we are able to find paths more performant than the pre-attack path on average in over 90% of cases without searching for the hardest setting of bandwidth tolerance and congestion factor, but with searching as shown in Figure 4.4d, our performance success is essentially 100% for all distances from the deployer AS. 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 4.4b without 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 89% of cases on average. When we employ searching in Figure 4.4e, we bring that average up to over 95% on average for the hardest setting of bandwidth tolerance at 1.1.

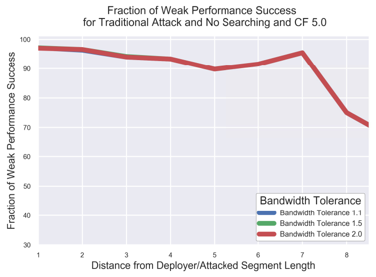
These results indicate that when transit-links are under a DDoS attack upstream of the deployer AS, we can guarantee *no* traffic loss for a particular critical AS in over 95% of cases. To state it in another way, we give the deployer the ability to operate under normal conditions for traffic being delivered from some critical network despite the transit-links between the deployer and critical network sustaining attack traffic loads of arbitrary amounts.

For transit-link DDoS with and without searching, we also show the post-attack subscription factors in Figure 4.4c and 4.4f, which indicates that for over 95% of cases we can migrate traffic onto uncongested paths out of the way of any DDoS attacks on upstream transit-links.

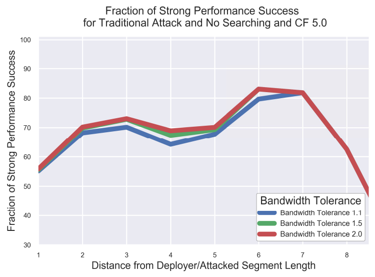
Not only can we protect the deployer AS when it is under transit-link DDoS, but we show we can protect the deployer AS when it is targeted directly in a traditional DDoS scenario for the *hardest settings* of bandwidth tolerance and congestion factor. As we show in Figure 4.5, we are able to migrate 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 4.5d, with an average of nearly 98% success in alleviating some amount of congestion, we do not see searching helping nearly as much for strong performance success, as shown in Figure 4.5e.

Despite this, we demonstrate that using Nyx, even when under a traditional DDoS attack, with little cost of deployment to an AS and no outside cooperation, we can migrate incoming traffic onto links that are not impacted by the DDoS at the ASes *doorstep*, targeted directly at the deployer, on average 75% of the time, as shown in the CDF of the post-subscription factors in Figure 4.5f. 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, and will 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 often cannot be done in any capacity with traditional DDoS defense methods that we will discuss in Section 5.

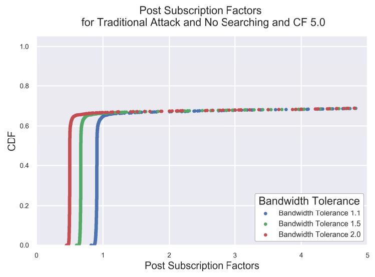
We show in Figure 4.6, that when our system utilizes searching, the depth to which we search is small except in greater distances from the deployer. This means that the deployer does not have to force the BGP speakers implementing Nyx to waste precious time finding more performant paths around impacted 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.



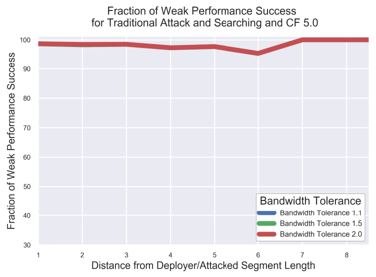
(a) Weak Performance Success with No Searching for the Mirai Botnet and Normal Bandwidth Model



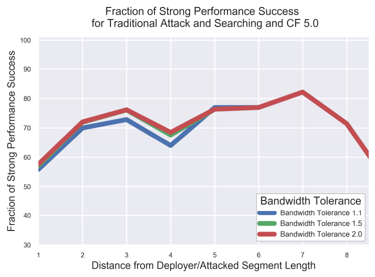
(b) Strong Performance Success with No Searching for the Mirai Botnet and Normal Bandwidth Model



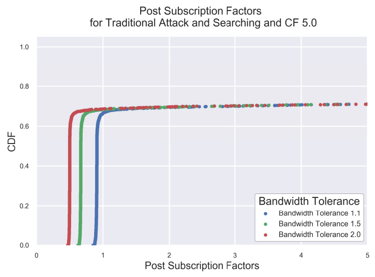
(c) CDF of Post-Subscription Factor with No Searching for the Mirai Botnet and Normal Bandwidth Model



(d) Weak Performance Success with Searching for the Mirai Botnet and Normal Bandwidth Model



(e) Strong Performance Success with Searching for the Mirai Botnet and Normal Bandwidth Model



(f) CDF of Post-Subscription Factor with Searching for the Mirai Botnet and Normal Bandwidth Model

Figure 4.5: Performance success metrics for the traditional attack scenario with and without searching.

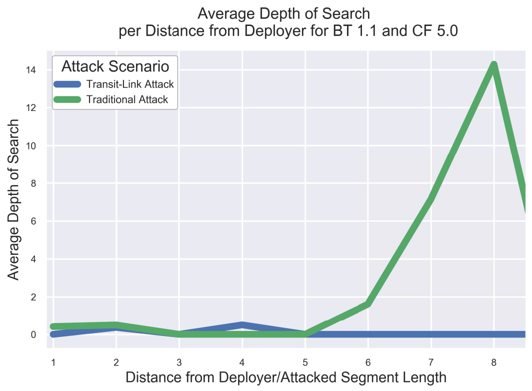


Figure 4.6: Average depth of search for the hardest setting of bandwidth tolerance and congestion factor.

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 4.7, 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, the strong performance success does not increase; therefore, our chosen values in the simulation are not in place to guarantee we have greater success.

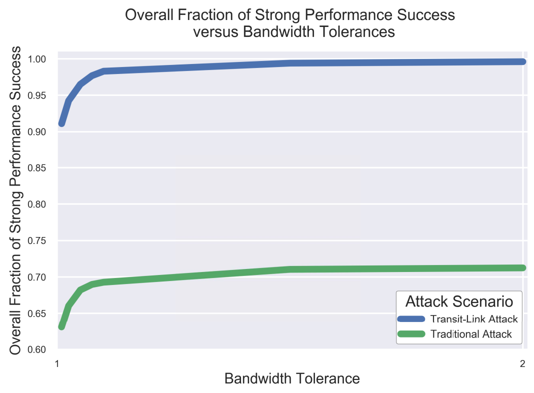
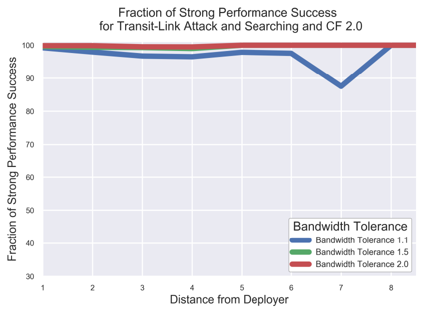
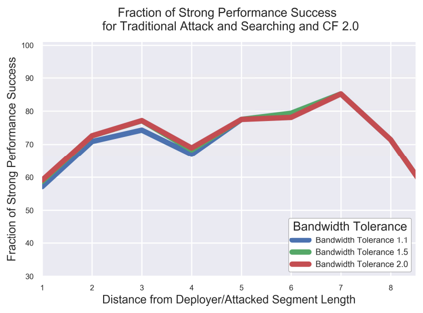


Figure 4.7: Strong performance success over varying bandwidth tolerances. Notice that once the bandwidth tolerance is greater than 1.1, the overall strong performance success stabilizes.

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. As shown in Figure 4.8, the smaller congestion factor of 2.0 has little effect on the strong performance success, where we must migrate traffic onto links that are uncongested. 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 you can continue to congest links on the normal path between the deployer and critical AS and still be able to successfully migrate traffic around the impacted links, while still alleviating congestion.



(a) Strong Performance Success for Transit-Link Attack for CF of 2.0



(b) Strong Performance Success for Traditional Attack for CF of 2.0

Figure 4.8: Strong performance success with searching for both attack scenarios for congestion factor of 2.0.

4.7.1 Is the Performance Success of Nyx Insensitive to the Botnet Model?

In Section 4.3, 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 A.1), and a fully distributed botnet. For Conficker, the results are similar in success to Mirai and are shown in the Appendix in Figure B.1. 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, and 78% strong performance success on average for Traditional DDoS as shown in the Appendix in Figure B.2. 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.

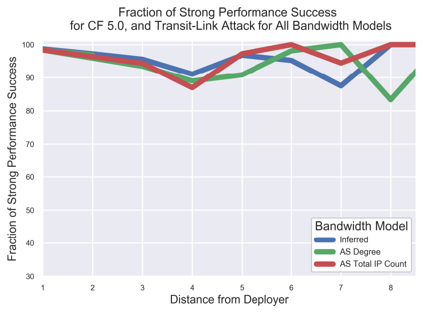
4.7.2 Is Nyx Insensitive to the Choice of Bandwidth Model?

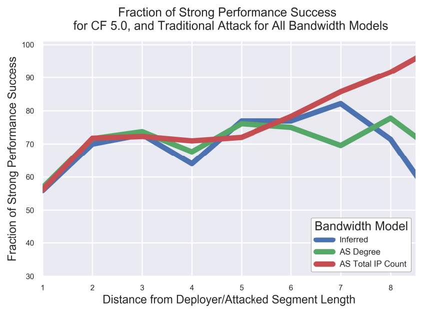
In Section 4.2, we described the main bandwidth model used in our evaluation. This model is fairly complex, but approximates the typical traffic levels on existing ASes through the application of several well-known datasets of AS-level and geographic data, which is needed because the real link capacities between major ASes are a in many cases a closely guarded secret. To evaluate our system's ability to still work with simpler models, we have built two other bandwidth models that influence our system's performance success (i.e. the ability to reduce congestion on the new routes chosen after successfully migrating traffic):

1. AS Degree Model: This model chooses the degree of each AS as the traffic factor value, or the approximation of the traffic sent by a given AS with arbitrary traffic units, described in Section 4.1. The AS Degree for any given AS is defined as the number of ASes where the given AS has a direct connection, as inferred from CAIDA's AS Relationship dataset [4].

2. AS IP Count Model: This model chooses the number of IPs associated with a given AS as it's traffic factor value. The number of IPs associated with a given AS is determined by the RIPE NCC RouteViews dataset [31].

In Figure 4.9, we show that our system still achieves nearly identical strong performance success for all tested bandwidth models, with our most complex and standard inferred model performing the worst overall. For the transit-link DDoS scenario, our models all averaged around 95% strong performance success, and for the traditional DDoS 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, we have achieved similar results as when we model the link capacities with our more complex, inferred bandwidth model that attempts to model the true link capacities in the Internet.





(a) Strong Performance Success for Transit-Link Attack Scenario for All Bandwidth Models

(b) Strong Performance Success for Traditional Attack Scenario for All Bandwidth Models

Figure 4.9: Strong performance success for both attack scenarios over all bandwidth models.

Chapter 5

Related Work

Traditional and current defense systems attempt to mitigate the negative effects of DDoS attacks through a variety of means; however, no systems we found defend against DDoS via route-altering techniques such as ours. 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 the Mirai botnet [1, 2, 3], 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 [25] using techniques such as packet marking [36, 8, 44, 26, 23, 43] and push-back techniques [45, 22, 24, 16, 5] 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 to test Nyx. Additionally, transit-link DDoS typically does not send attack traffic directly to the reactor AS, and instead to upstream links; therefore, filtering on attack traffic would not be feasible since the victim would not see the traffic. Furthermore, our system 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 arbitrarily manipulate the paths that attack traffic takes, and by doing so spread the incoming attack traffic across links upstream of the deployer AS such that no traffic from the critical AS is dropped along the way.

Other techniques that filter traffic targeted at specific services [10, 9, 41] are ineffective against DDoS attacks that attack different services or even the underlying routing infrastructure. Because all internetwork traffic must be sent over paths determined by BGP speakers, as simulated in this work, our system 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 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 [37, 7], 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] 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 [47, 34], which cannot be done in the case of transit-link DDoS.

Chapter 6

Conclusion

In this work we presented a system that can significantly reduce the impact of transit-link (targeting links upstream of the victim) and traditional (targeting the victim directly) DDoS when deployed only onto the deployer AS's BGP speakers. First, we showed that we can manipulate not only outbound traffic from an AS, but also the paths which inbound traffic takes. This ability allows our system to intelligently migrate traffic, coming from chosen critical ASes where we want traffic to always reach us, off links impacted by DDoS attacks with nearly 100% success. Second, we demonstrated that we can migrate incoming traffic off of 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 path manipulation techniques, as opposed to 1000 to 5000 disturbed ASes on average before employing reduction methods. Third and most importantly, we demonstrated that we can migrate traffic off impacted links onto links that are uncongested with over 98% success for transit-link DDoS and on average 70% 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 an alternative to ineffective filtering and prioritization methods used against recent DDoS attacks [39, 3, 15], and finally presents the *first scalable and easily deployable* solution to transit-link attacks such as Crossfire and Coremelt [18, 38].

6.1 Future Work

The system we have developed creates many interesting opportunities for future work. First, our system has demonstrated success with the bandwidth models and botnet models described earlier, though with our open source simulator, new bandwidth and botnet models can be rapidly tested. 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. Furthermore, our system relies on searching for alternate paths until an uncongested or non-degraded path is found; however, if the deployer AS had the ability to detect degraded quality of service along upstream links, our system could make more informed decisions of where to migrate critical traffic.

Finally, as mentioned in Section 2, 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 an adversary controlling a significant amount of ASes. In general, combating a global adversary would require our system to combine all prior-mentioned future work: protecting multiple critical ASes, the ability to detect upstream adverse network conditions and congestion without assistance, and utilizing bandwidth and botnet models representative of routing-aware adversaries.

Bibliography

[1] (2016). Dyn analysis summary of friday october 21 attack. https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/. 6, 23, 38

[2] (2016). 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/. 6, 38

[3] (2016). Mirai iot botnet blamed for smashing liberia off the internet. https://www.theregister.co.uk/2016/11/04/liberia_ddos/. 1, 6, 8, 23, 38, 40

[4] (2017). CAIDA AS relationship dataset. http://www.caida.org/data/active/as-relationships/index.xml. 10, 18, 19, 21, 22, 36, 50

[5] Argyraki, K. J. and Cheriton, D. R. (2005). Active Internet Traffic Filtering - Real-Time Response to Denial-of-Service Attacks. *USENIX Annual Technical Conference, General Track*. 38

[6] Basescu, C., Reischuk, R. M., Szalachowski, P., Perrig, A., Zhang, Y., Hsiao, H.-C., Kubota, A., and Urakawa, J. (2016). SIBRA: Scalable internet bandwidth reservation architecture. In *Proceedings of Symposium on Network and Distributed System Security (NDSS)*. 2, 10, 12

[7] Bedi, H. S., Roy, S., and Shiva, S. (2011). Game theory-based defense mechanisms against DDoS attacks on TCP/TCP-friendly flows. *... in Cyber Security (CICS)*. 39

[8] Belenky, A. and Ansari, N. (2007). On deterministic packet marking. *Computer Networks*. 1, 38

[9] Chou, J. C. Y., Lin, B., Sen, S., and Spatscheck, O. (2009). Proactive Surge Protection: A Defense Mechanism for Bandwidth-Based Attacks. *IEEE/ACM Transactions on Networking*, 17(6):1711–1723. 39

[10] Dixon, C., Anderson, T. E., and Krishnamurthy, A. (2008). Phalanx - Withstanding Multimillion-Node Botnets. *NSDI*. 39

[11] Fayaz, S. K., Tobioka, Y., Sekar, V., and Bailey, M. (2015). Bohatei: Flexible and elastic ddos defense. In *Usenix Security*, pages 817–832. 1

[12] Gill, P., Schapira, M., and Goldberg, S. (2011). Let the market drive deployment: A strategy for transitioning to bgp security. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 14–25. ACM. 22, 50

[13] Huston, G. (2017). Bgp more specifics: routing vandalism or useful? `https://blog.apnic.net/2017/06/26/bgp-specifics-routing-vandalism-useful/`. 7

[14] IANA (2017). Iana autonomous system (as) numbers. `https://www.iana.org/assignments/as-numbers/as-numbers.xhtml`. 22, 50

[15] IBTimes (2014). Why Microsoft and Sony couldnt stop Lizard Squad attack despite warnings. `http://www.ibtimes.com/why-microsoft-sony-couldnt-stop-lizard-squad-attack-despite-warnings-1769174`. Accessed: 17 January 2017. 40

[16] Ioannidis, J. and Bellovin, S. M. (2002). Implementing Pushback - Router-Based Defense Against DDoS Attacks. *NDSS*. 38

[17] J. Hawkinson, T. B. (1996). Guidelines for creation, selection, and registration of an autonomous system (as). 6

[18] Kang, M. S., Lee, S. B., and Gligor, V. D. (2013). The Crossfire Attack. *IEEE Symposium on Security and Privacy*. 1, 4, 6, 8, 23, 40

[19] Khalimonenko, A. and Kupreev, O. (2017). Kaspersky labs q1 2017 ddos report. `https://securelist.com/ddos-attacks-in-q1-2017/78285/`. 5

[20] Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J., and Jahanian, F. (2011). Internet inter-domain traffic. *ACM SIGCOMM Computer Communication Review*, 41(4):75–86. 22, 50

[21] Lepinski, M. and Kent, S. (2012). An infrastructure to support secure internet routing. `https://tools.ietf.org/html/rfc6480`. 14

[22] Liu, X, Yang, X, and Lu, Y (2008). StopIt: Mitigating DoS flooding attacks from multi-million botnets. 38

[23] Ma, M. (2005). Tabu marking scheme for ip traceback. *Parallel and Distributed Processing Symposium*. 1, 38

[24] Mahajan, R., Bellovin, S. M., Floyd, S., Ioannidis, J., Paxson, V., and Shenker, S. (2002). Controlling high bandwidth aggregates in the network. *Computer Communication Review*. 38

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

[26] Muthuprasanna, M. and Manimaran, G. (2008). Distributed Divide-and-Conquer Techniques for Effective DDoS Attack Defenses. *ICDCS*. 1, 38

[27] Netlab360 (2017). Mirai Scanner. http://data.netlab.360.com/mirai-scanner/. 23, 50

[28] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. 22

[29] PeeringDB (2017). Bgp peering information. https://www.peeringdb.com/. 22, 50

[30] Rekhter, Y. and Li, T. (1995). A border gateway protocol 4 (bgp-4). 6, 7

[31] RouteViews (2017). Routeviews dataset. http://www.routeviews.org/. 36, 50

[32] Sandvine (2017). Global internet phenomena report. https://www.sandvine.com/trends/global-internet-phenomena/. 22, 50

[33] Schuchard, M., Geddes, J., Thompson, C., and Hopper, N. (2012). Routing around decoys. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 85–96, New York, NY, USA. ACM. 19

[34] Senthilmahesh, P., Hemalatha, S., Rodrigues, P., and Shanthakumar, A. (2013). Ddos attacks defense system using information metrics. 39

[35] Shin, S., Gu, G., Reddy, N., and Lee, C. P. (2012). A Large-Scale Empirical Study of Conficker. *IEEE Transactions on Information Forensics and Security*, 7(2):676–690. 5

[36] Siris, V. A. and Stavrakis, I. (2007). Provider-based deterministic packet marking against distributed DoS attacks. *J. Network and Computer Applications*. 1, 38

[37] Spyridopoulos, T., Karanikas, G., Tryfonas, T., and Oikonomou, G. (2013). A game theoretic defence framework against DoS/DDoS cyber attacks. *Computers & Security*. 39

[38] Studer, A. and Perrig, A. (2009). The Coremelt Attack. *ESORICS*. 1, 4, 6, 8, 40

[39] Symantec (2016). Mirai: what you need to know about the botnet behind recent major DDoS attacks. `https://www.symantec.com/connect/blogs/mirai-what-you-need-know-about-botnet-behind-recent-major-ddos-attacks`. Accessed: 17 January 2017. 1, 5, 23, 40

[40] Thomas, M. and Mohaisen, A. (2014). Kindred domains: Detecting and clustering botnet domains using dns traffic. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 707–712, New York, NY, USA. ACM. 23, 50

[41] Von Ahn, L., Blum, M., and Langford, J. (2004). Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60. 39

[42] WorldBank (2017). World bank global indicators. `http://data.worldbank.org/indicator/`. 22, 50

[43] Xiang, Y. and Zhou, W. (2006). Protecting information infrastructure from ddos attacks by madf. *International Journal of High ...* . 1, 38

[44] Xiang, Y., Zhou, W., and Guo, M. (2009). Flexible Deterministic Packet Marking - An IP Traceback System to Find the Real Source of Attacks. *IEEE Trans. Parallel Distrib. Syst.* 1, 38

[45] Yau, D. K. Y., Lui, J. C. S., Liang, F., and Yam, Y. (2005). Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Trans. Netw.* 38

[46] Zhang, X., Hsiao, H.-C., Hasker, G., Chan, H., Perrig, A., and Andersen, D. G. (2011). SCION - Scalability, Control, and Isolation on Next-Generation Networks. *IEEE Symposium on Security and Privacy.* 2, 12

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

[48] Zhou, W., Jia, W., Wen, S., Xiang, Y., and Zhou, W. (2014). Detection and defense of application-layer DDoS attacks in backbone web traffic. *Future Generation Computer ...*, 38:36–46. 39

Appendix

A Additional Simulation Details

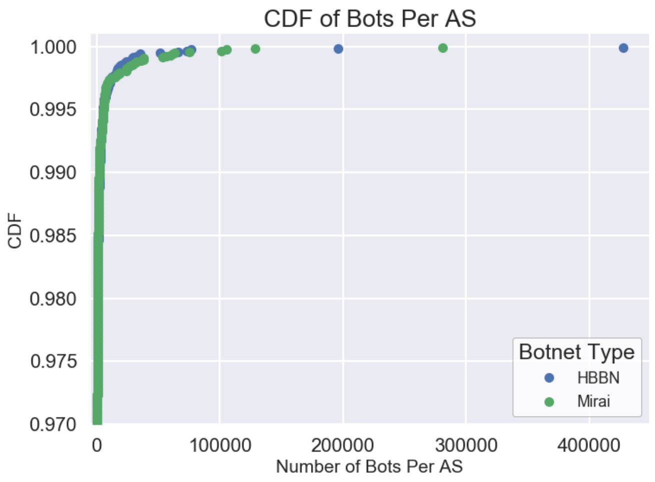
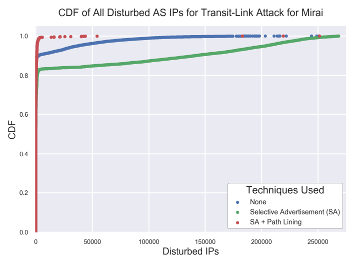
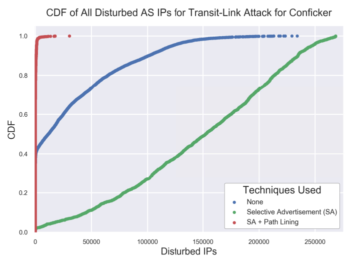


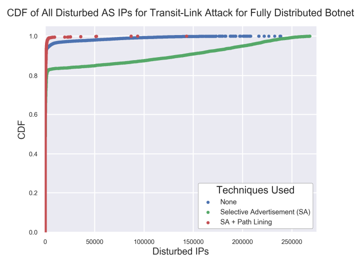
Figure A.1: Bot Count per AS in the Mirai Botnet and the Host-Based Botnet.



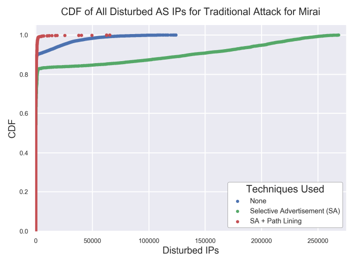
(a) Disturbed IPs for Transit-Link Attack for Mirai



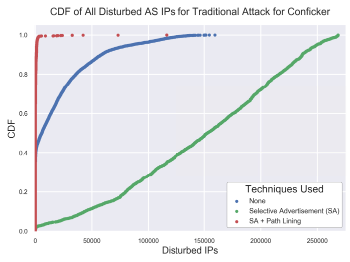
(b) Disturbed IPs for Transit-Link Attack for Conficker



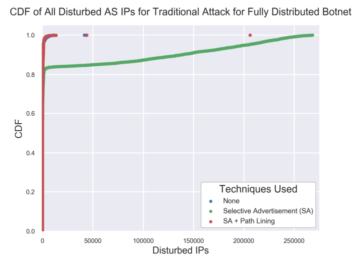
(c) Disturbed IPs for Transit-Link Attack for Fully Distributed Botnet



(d) Disturbed IPs for Traditional Attack for Mirai



(e) Disturbed IPs for Traditional Attack for Conficker



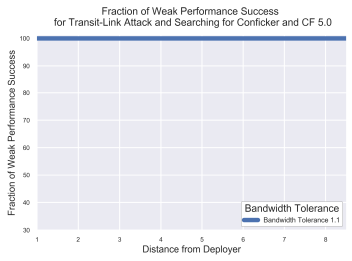
(f) Disturbed IPs for Traditional Attack for Fully Distributed Botnet

Figure A.2: Disturbed IPs with and without disturbance mitigation for all botnet models for Traditional and Transit-Link DDoS

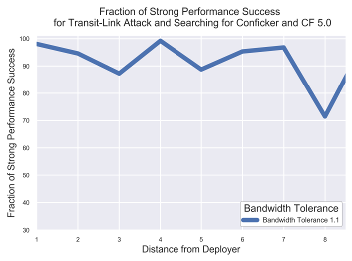
Table A.1: Information needed by the simulator

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 [4], Route Views Project [31]
Inferred (with machine learning) 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 [4], PeeringDB [29], IANA [14], World Bank [42], Sandvine [32], Labovitz *et al.* [20], Gill *et al.* [12]
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 [4]
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 [31]
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 [27]
Conficker Botnet Model (Conficker)	Host-based botnet (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.* [40]
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

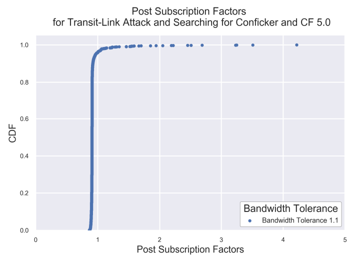
B Performance Success of Conficker and Fully Distributed Botnets



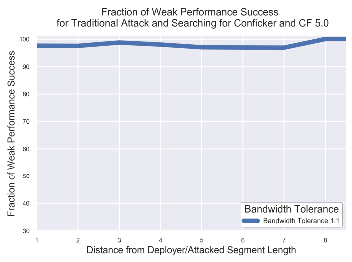
(a) Weak Performance Success for Transit-Link Attack with Searching for the Conficker Botnet



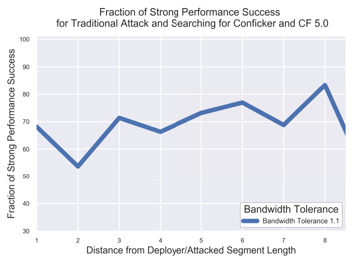
(b) Strong Performance Success for Transit-Link Attack with Searching for the Conficker Botnet



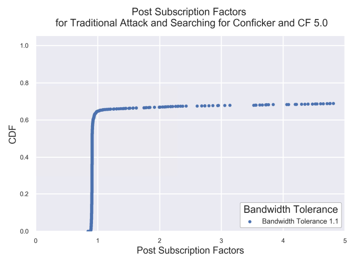
(c) CDF of Post-Subscription Factor for Transit-Link Attack with Searching for the Conficker Botnet



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

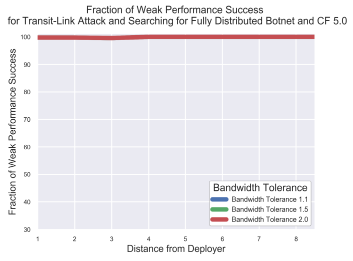


(e) Strong Performance Success for Traditional Attack with Searching for the Conficker Botnet

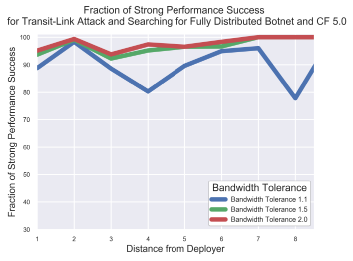


(f) CDF of Post-Subscription Factor for Traditional Attack with Searching for the Conficker Botnet

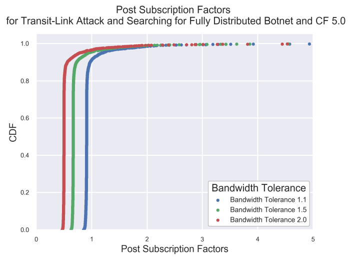
Figure B.1: Performance success metrics for both Traditional and Transit-Link attack scenario, normal bandwidth model, with searching for the **Conficker** botnet.



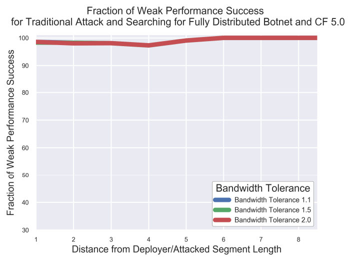
(a) Weak Performance Success for Transit-Link Attack with Searching for the Fully Distributed Botnet



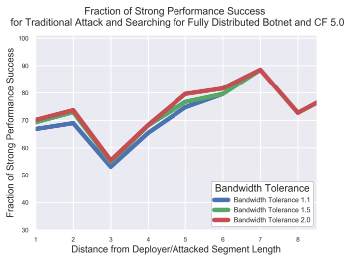
(b) Strong Performance Success for Transit-Link Attack with Searching for the Fully Distributed Botnet



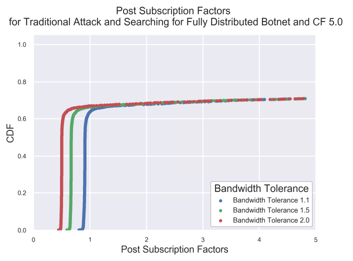
(c) CDF of Post-Subscription Factor for Transit-Link Attack with Searching for the Fully Distributed Botnet



(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

Figure B.2: Performance success metrics for both Traditional and Transit-Link attack scenario, normal bandwidth model, with searching for the **Fully Distributed** botnet.

Vita

Jared is a Tennessee Top 100 Graduate Research Fellow currently working on his concurrent MS/PhD in Computer Science at the University of Tennessee with a focus on Internet infrastructure resiliency and privacy-preserving methods of communication in VolSec, the UT Computer Security Lab. Jared is also a Cyber Security Researcher at Oak Ridge National Laboratory, where he leads a combined team of ORNL staff researchers and student interns on R&D projects in the Digital Forensics and Incident Response space. He also works on other projects in collaboration with researchers from MIT, Stanford Research Institute, U.S. Department of Homeland Security, FBI, NSA, GE, and many other organizations. While at ORNL, Jared has received over 1.2 million dollars in award or grant money for proposals he has written and now leads or has led in the past, funded by agencies such as the U.S. Department of Energy Cybersecurity for Energy Delivery Systems (CEDS) program, the U.S. Department of Homeland Security Transition to Practice (TTP) program, and the ORNL Global Security Directorate.

Jared also received his BS in Computer Science from the University of Tennessee. While he was an undergrad at UT, Jared founded and led the student-organized computer security organization, HackUTK, growing it from four founders to over 200 members. He also founded and led the first VolHacks, UTs annual software, hardware, and VR hackathon, bringing in 23 sponsors and 200+ students in its first year. Jared has held research assistantships or fellowships with UTK's High Energy Particle Physics Group, the Center for Intelligent Systems and Machine Learning, with Dr. Bruce MacLennan and Dr. Jeremy Holleman, Dr. Audris Mockus, and the UT Computer Security Laboratory under the supervison of Dr. Max Schuchard.

Throughout the course of his undergraduate and master's program at the University of Tennessee, Jared has published several papers and posters at major ACM and IEEE conferences or workshops, including the ACM Conference on Computer and Communications Security and the ACM International Conference on Software Engineering, has given over 20 different talks at major industry (non-academic) conferences and user groups, and competed in and won several pitch competitions for a total of over 35 thousand dollars in venture-capital funding for startups.

In his spare clock cycles, Jared teaches courses at Treehouse (a popular e-learning platform with over 200,000 subscribers), advises or has helding the acting CTO role for several companies ranging from media and healthcare startups to Fortune 500 computer security vendors on a wide array of technical matters, gives conference talks across North America, organizes technology conferences, and hikes in the Great Smoky Mountains.