8-2020

A Privacy Evaluation of Nyx

Savannah A. Norem
University of Tennessee, Knoxville, snorem1@vols.utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Savannah A. Norem entitled "A Privacy Evaluation of Nyx." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Maxfield J. Schuchard, Major Professor

We have read this thesis and recommend its acceptance:

Maxfield J. Schuchard, Scott I. Ruoti, Michael R. Jantz

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

A Privacy Evaluation of Nyx

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Savannah Austin Norem

August 2020

To my parents. Thank you for your constant love and support.

Acknowledgments

There are countless people that have helped me get to this point, but there are a few that have enormously helped push me across the finish line. Tyler McDaniel has been a wealth of knowledge. Jared Smith has been a constant source of encouragement. Last but not least, Dr. Schuchard has advised me on research just as much as he has on life and I would not be writing this if not for him.

Abstract

For this project, I will be analyzing the privacy leakage in a certain DDoS mitigation system. Nyx [11] has been shown both in simulation and over live internet traffic to mitigate the effects of DDoS without any cooperation from downstream ASes and without any modifications to current routing protocols. However it does this through BPG-poisoning, which can unintentionally advertise information. This project explores what the traffic from Nyx looks like and what information can be gathered from it. Specifically, Nyx works by defining a deployer/critical relationship whose traffic is moved to maintain even under DDoS circumstances, and I will be evaluating how often that relationship can be discovered.

This project will analyze the privacy leakage in the Nyx DDoS mitigation system [11]. Nyx's effectiveness in rerouting critical traffic around congestion has been demonstrated both in simulation and in practice. Importantly, Nyx functions without cooperation from downstream ASes or modifications to current routing protocols. However, Nyx achieves routing based DDoS mitigation through BGP poisoning, which can unintentionally advertise information. This project will analyze Nyx's BPG advertisements to evaluate its privacy implications. Specifically, this work studies whether an adversary can determine the critical relationship that the AS deploying Nyx has defined. We find that in the authors initial naive approach, finding this relationship is essentially trivial and an adversary can narrow down the critical relationship to a maximum of 4 out of 9,767 autonomous systems in the active internet topology. In their more complex approach found in [10] we find that the critical relationship is more difficult to determine with significant accuracy, with our anonymity sets ranging from 3 to 7,788. This project then explores why that range is so large in an attempt to highlight how Nyx could become more privacy focused.

Table of Contents

List of Figures

Chapter 1

Introduction

Preventing information leakage and ensuring the privacy of sensitive information are core requirements for a computer system to be considered "secure". In the field of distributed systems security, mitigating Distributed Denial of Service, or DDoS, is increasingly necessary to prevent widespread loss of availability of online services. Yet, systems that mitigate DDoS *must not* compromise a victims privacy when attempting to prevent traffic loss. For the purposes of this work, our victim is someone who our adversary is trying to learn information about. In Nyx, these are the deployers, or the AS advertising poisoned paths to force their inbound traffic to take a certain path from a defined critical AS.

DDoS attacks remain one of the most serious and persistent threats on the Internet due to their increasing potency and decreasing cost. In October of 2016 the Mirai botnet brought down services like Netflix, Spotify and Twitter, with AWS also reporting outages for multiple hours [8]. Dedicated websites exist to allow anyone with the monetary resources the ability to control a sizeable botnet. Due to the growing threat, there are numerous novel techniques proposed that claim to alleviate this threat, such as categorizing benign and malicious traffic [9][1][15] and the use of CDNs to add bandwidth [3]. While there have been recent advances in distinguishing malicious from benign traffic, this approach can impact network performance and has been shown to have fatal flaws when adversarial machine learning is applied [2]. Utilizing CDNs to add bandwidth makes DDoS and DDoS prevention a bandwidth arms race. While often this comes down to monetary resources, botnets have been shown to be

incredibly powerful[5], and this approach simply will not prevent large-scale botnets from launching a successful DDoS attack.

Recently another DDoS mitigation approach has been presented, Nyx [11]. Nyx uses BGP poisoning to control traffic such that an AS using Nyx is able to route around congestion both on their outbound paths, and poison paths to force inbound traffic to comply with its routing decision. Nyx uses a BGP poisoning loop, called path-lining, to limit the propagation of the critical AS's new re-routing path, isolating inbound traffic from the critical from malicious traffic. This isolation is necessary to prevent malicious as well as critical traffic from switching onto the new path.

In this work, we develop algorithms that ingest information that an adversary would be able to see on the live Internet and attempt to discover if Nyx leaks information about the critical relationship during the re-routing process. In a world where Nyx is routinely used, this critical relationship is likely to be a business relationship that an AS is willing to pay to maintain connectivity and communication to. While some of these relationships would likely have limited impact if they were made public, ASes do not even publish the relationships they have with other ASes now. The state of the art relationship data we use from CAIDA [7] is all inferred. Given that, it is unlikely ASes would want the critical relationships they define to be public knowledge. To see how often we are able to expose this relationship, we evaluate the two modes Nyx has, infinite and greedy path lining. To discover these relationships, we develop algorithms that take in what an adversary would be able to see on the live Internet and get out a set of potential critical relationships. In the infinite case, we find that we can correctly identify the critical relationship 100% in simulation. The next scenario we consider is a more realistic one, where there is a limit on how many ASNs can be added to the path-lining. In this scenario we correctly identify the critical relationship 998 out of 999 times.

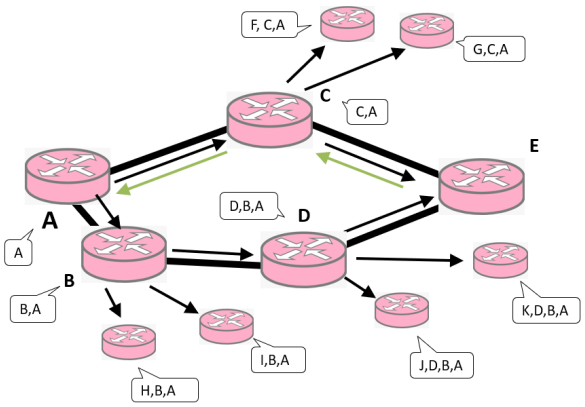
Chapter 2

Background and Related Work

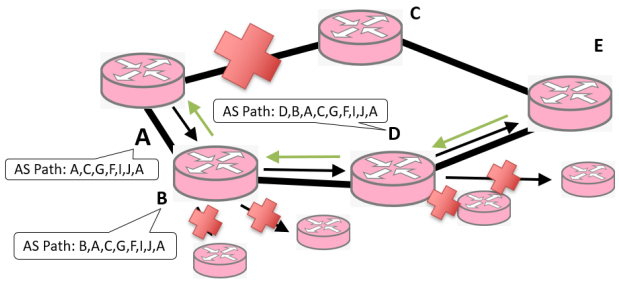
2.1 Background

2.1.1 Autonomous Systems

While many would like to think of the internet as one big cloud that gets traffic from point A to B, in reality it's made up of many smaller networks that pass traffic between one another to get traffic from one end host to another. These smaller clouds are called Autonomous Systems, or ASes. Each Autonomous System has a number assigned to it, or an ASN (Autonomous System Number). ASes are generally owned by entities such as internet service providers and universities. Within the cloud of a single AS, traffic patterns are generally opaque to outside observers; once the traffic is inside an AS we do not know much about it. However, to get traffic across the entirety of the Internet, ASes have to communicate with each other. Thus they have relationships with each other that help determine how traffic moves, mainly where larger ASes (e.g. AT&T and L3) provide trafficking services to smaller ASes (e.g. UT) for a fee. This is known as a customer / provider relationship. ASes also have relationships with "peers", e.g. an AS with whom they freely share traffic and information in a mutually beneficial way.



(a) Original Advertisement



(b) Poisoned Advertisement

Figure 2.1: Path advertisements with and without BGP Poisoning via loops.

2.1.2 BGP and BGP Poisoning

The way that these ASes communicate with each other is defined by the Border Gateway Protocol, or BGP. Figure 2.1 shows a simple example of BGP advertisements.In this figure, AS A originates a route to one of its prefixes. AS A's neighbors prepend themselves to the route and propagate the advertisement. This process is repeated to spread the advertisement throughout the topology. Per the protocol, AS E would be able to choose the path of its outbound traffic to AS A, either over C or over D. In this example they prefer to talk over C.

Figure 2.1 shows how the propagation process can be altered via BGP poisoning. A critical feature of BGP is loop detection; when an AS is advertising a path, if a different AS detects that their uniquely identifying number is already on this path, they will drop the advertised path to avoid a loop. Here we can see that AS A is able to "poison" AS C. The advertising AS controls the beginning of the route carried in the advertisement and can artificially create a "loop" by inserting select other ASNs into the advertisement. This new "poisoned" path to AS A (A,C,A) gets picked up by both B and D. AS C sees that it is already on the path, therefore filters the advertisement, and the propagation stops there. Because C was poisoned and filtered the advertisement, AS E and other remote ASes can no longer send their traffic to AS A over C. When prepending ASes to poison a path, the advertising AS places these ASNs between copies of its own ASN. This "bookending" ensures that traffic is routed properly and the advertisement has the correct origin. This holds regardless of how many ASes are being poisoned.

2.1.3 Valley Free Routing

Another critical property of BGP is valley-free routing. In short, the valley-free routing model states that will never transit a provider or a peer after passing a customer. While the protocol does not enforce valley-free routing, ASes typically minimize monetary costs by following valley-free routing. And while work has shown that not all BGP routes obey valley-free routing [4], most routes do follow this model. Nyx's authors acknowledge that

some of the divergence between simulated Nyx deployment and Nyx in practice could stem from valley-free routing violations [10].

Logically, ASes choose paths that limit their monetary costs. To do so, they both restrict the advertisement of paths that transit their providers and widely disseminate paths that transit their customers. This idea influences path propagation and advertisements. Figure 2.1 also shows that an AS learns about paths through BGP advertisements. ASes are selective about what paths they advertise to maximize their profit. An AS will transit traffic along any path that it advertises. Therefore if an AS learns about a path from a customer, they will advertise it to all their connections to increase the chances of transiting traffic for their customer (and making money). In a small example, we consider AS X, who has Customers A and B and Providers C and D. Customer A has a new path that they advertise to X. X then advertises that path to everyone since X will make money by sending traffic to A. At some point, Customer B needs to use it. Customer B then has to send traffic to X (X makes money) then X sends it to Customer A (X makes money again). If Provider C needs to use that path, X has to pay C, but then makes money off of A, and X has negotiated their rates to cover this cost. If Provider C advertises a path to X, X advertises that path to Customers A and B, since they will still have to pay X, but not to Provider D. This is because that would cost X money twice, once for getting traffic from C, and once for sending it to D. The same logic applies to peering relationships. AS X will advertise routes it has learned from customers to its peers, but not routes learned from other peers or providers. This restriction follows from AS X avoiding transiting traffic for free or at cost when possible.

These decisions provide the basis for valley free routing. For a path to be valley free, it goes up to providers 0 or more times, over to a peer 0 or 1 time(s), and then down to customers 0 or more times. Valley free routing "has been widely perceived as a universal property of the Internet BGP routing that is only violated due to transient configuration errors" [4]. This model is essential to our understanding because Nyx, while searching for alternate paths, strictly obeys valley free routing.

Customer Cones

A key way that Nyx attempts to minimize disturbance is by taking advantage of how paths propagate. Because ASes will propagate all paths they see to their customers and customers of customers will propagate paths to their own customers, until an AS with no customers is reached. Therefore to minimize disturbance, Nyx will prioritize poisoning ASes with larger "customer cones", which is the sum of their customers plus their customers customers and so on. Thus an AS's customer cone is one way to measure how many ASes will see an advertisement from that AS.

2.1.4 Nyx

The work that this project will evaluate is called Nyx [11]. Nyx works by exploiting the fact that ASes will drop a path that appears to form a loop. To reiterate, Figure 2.1 shows that by adding AS C to the path that AS A is advertising, AS A is able to control how AS E will send traffic back to it. This is essential because it means only one AS in the system has to have Nyx deployed for it to be effective.

We call the AS that is using Nyx the "deployer". This deployer is able to define "critical" ASes that it wants to maintain communication with in the event of DDoS affecting a link between them. Looking back to Figure 2.1, if AS C was unable to communicate to either AS A or AS E because of DDoS, but only AS A realizes this, they would want to tell AS E to adjust their path to use ASes B and D so that communication could continue uninterrupted. Unfortunately, There is no accommodation in the protocol for this kind of communication. Nyx provides this capability by forcing the critical AS to use a certain path to the deployer that avoids the attack.

To do this, Nyx first must find a new path between a deployer and one or more critical AS(es). Nyx then uses "path lining" to add a number of ASNs to the path that the deployer is advertising. This path lining controls the propagation of the poisoned advertisement to prevent bots or other ASes from using the new path. The deployer advertises a path with a loop, forcing traffic off of the link that is suffering from DDoS indicated by the red X. Each of their neighbors see this new path, and propagate the advertisement to their own

neighbors. Path-lining ensures that this neighbor-to-neighbor propagation is controlled and does not spread throughout the Internet, because otherwise the DDoS attack or other traffic could congest the new path.

2.2 Related Work

There are three pieces of work that are critical for this work to exist. The original Nyx paper [11], Tran et al's study on the feasibility of re-routing based DDoS mitigation [12], and the Nyx authors preprint further examination [10]. The original Nyx paper proposed the novel way of mitigating DDoS via current routing protocol. This work took the naive approach of using infinite path lining and measured its success strictly in simulation.

Tran et al's study of the feasibility of re-routing based DDoS mitigation [12] pointed some of the critical flaws in the naive approach of Nyx. Mainly that infinite path lining is infeasible. The authors argued that due to an average AS-level path of less that four [14], ASes would not accept these paths whose average length is over 1,800 ASes, citing Cisco routers that drop paths over 254 ASes long and that NANOG discourages paths over 75 ASes. They go on to show that the number of ASes that will filter paths longer than that significantly reduces Nyx's ability to find paths that would allow for these long advertisements.

In response, the authors of the Nyx paper used resources such as Caida, Ripe Atlas, PEERING, and their university's AS to ethically conduct Nyx experiments on the live Internet. While they found the claim that very long paths from [12] holds, they also found that when Nyx limits its poisoning it is still able to route around congestion [10]. The Nyx authors show that on the live Internet they are able to route around congestion with a 98% success rate. Because this is what a realistic adversary would be seeing, we focus more on these results.

Chapter 3

Experiment Design

3.1 Experiment

3.1.1 Threat Model

For this measurement, the main piece of information we have is the log files from Chaos, which is the simulator used to run [11] and used to capture data for [10] which give us all the information we need. From these files we get: 1 - the deployer, 2 - the critical AS, and 3 - the modified path between the two. The adversary we define would have access to the information in 3, the modified path, which contains the loop full of poisoned ASes. This is the path that gets advertised from the deployer once Nyx detects a problem (DDoS) and finds an alternate path to maintain the connection between the critical and the deployer. We use this modified path to build our suspicion(s) regarding the identity of the critical AS. Finding this relationship is the main goal for our adversary.

To get an understanding of the privacy concerns surrounding Nyx, we want to consider what type of adversary is able to obtain and utilize this information. While BGP advertisements are available in public router dumps [7], so *technically* anyone can view this information, the key thing an adversary needs to know is that Nyx caused the path to change because of a DDoS attack. We could be an adversary who simply hordes BGP advertisements and looks for changes when DDoS attacks happen, however many DDoS attacks are unnoticed due to higher resiliency in large networks, and many that are announced are not announced

until after the fact. So, this would be a very inefficient adversary. If, however, we were an adversary that could produce a DDoS attack, we would be able to control when we see advertisements change because of Nyx reacting to DDoS. It follows then that our primary adversary has the resources to procure a botnet, trigger a DDoS attack on a selected link, and observe the consequences.

3.1.2 Setup

To be able to use the information we get from simulated Nyx runs, we also have to set up what the Internet looks like. To find out what the Internet looks like, we use state-of-the-art inferred AS relationship data from CAIDA [13]. From this relationship file, we build multiple graphs of the topology of the Internet. We end up with a directed graph to check for valley-free-ness as well as an undirected graph to get neighbors for potential path lining. While Nyx got more complex with greedy path lining, it is still relatively naive in that path lining does not take into account whether or not a route would be generally propagated to the neighbors of the path, e.g. whether or not the propagation would be valley-free. Since Nyx does not take that into account, neither can we.

Once we have graphs that represent the same topology (built from the same relationship file) as the Nyx run we are pulling a log file from, and the customer cones associated with each AS, we prune the topology of stub ASes. The relationship file we have contains roughly 60,000 ASes, however the vast majority of those are basically insignificant. An AS is considered a stub if it does not provide transiting services to anyone, i.e. it has no customers. It is important these ASes be counted when calculating the customer cone sizes of their providers, however after that is complete we take them out of the topology since we know Nyx's topology for determining deployers, criticals, and new paths does not contain stub ASes. After this we have the active topology, which contains 9,767 ASes. As we go through the algorithms for both the infinite and greedy scenarios, we memoize for each AS some supplemental information. This supplemental information has the set of ASes they can lead us to, that is their neighbors, along with their neighbors supplemental information.

3.2 Algorithms

We evaluate Nyx's privacy implications over two scenarios, infinite and greedy path lining, as described in Section 2.1.4. For both scenarios, we take the log files from the simulated Nyx runs from [11] and [10] and get the path lining. This corresponds in practice to an adversary observing the Nyx advertisement, which contains all the path-lining ASes. Specifically, the path lining is all ASes between copies of the originator's ASN (i.e., all poisoned ASes) in the Nyx advertisement. We also know the deployers identity, since it is included in the Nyx advertisement. To begin our attempt at finding the critical AS we follow the general Algorithm 1. We start by getting each neighbor of our known deployer (D), an empty set of our current path, and an empty set of our current cover. Cover being everything that our current path can touch. We begin by asking each neighbor of D if their supplemental information can explain the path lining we know. If that neighbor (N) can eventually explain it, i.e. some combination of ASes we can get to through N can explain it, we add N to our potential path. We then check to see whether or not the path that we have can fully explain our path lining. To do this, we add N's supplemental information to our current cover, and use this to determine the current path lining. If our current path lining explains the path lining we know to be true, we add N to our potential critical set and keep going. We recursively check for these paths until we exhaust options that fully explain the path lining.

3.2.1 Infinite Path Lining

There are some key distinctions in how two lines of Algorithm 1 happen in the infinite scenario as opposed to the greedy scenario. The first being Line 10, "determine current path lining". For the infinite path lining scenario this is means getting all of the neighbors of everything on the path, and removing the things on the path we are walking, including the deployer. The next being line 11, "if current path lining explains path lining". For the infinite scenario this is a set comparison. It comes down to if every single thing in our current path lining match every single thing in the path lining we know to be true. For infinite path lining to get to a point where it "breaks" path lining and stops going down a path, we simply have to reach an AS with more than one neighbor not in the path or the path lining. Until

Algorithm 1: Path Lining Attack Algorithm

```
1  function FindCritical (Deployer, PathLining)
   Input  : The AS deploying Nyx Deployer, The set of ASes poisoned by Nyx
            PathLining
   Output: The set of potential critical ASes Criticals
2  Neighbors = Deployer.neighbors
3  CurrentPath = []
4  CurrentCover = ∅
5  foreach AS ∈Neighbors do
6  |  get AS supplemental information
7  |  if supplemental information covers path lining then
8  |  |  Add AS to CurrentPath
9  |  |  Add SupplementalInfo to CurrentCover
10 |  |  Determine current path lining
11 |  |  if CurrentPathLining explains PathLining then
12 |  |  |  Add AS to Criticals;
13 |  |  else
14 |  |  |  NewNeighbors = AS.neighbors - CurrentPath
15 |  |  end
16 |  else
17 |  |  Repeat ForEach with NewNeighbors
18 |  end
19 end
20
```

that happens, any potential critical could also be simply another hop along the path. In most (691/999) cases, the first potential critical is not completely path-lined, and therefore is definitively the critical. In some cases however it is, and so we identify an anonymity set of potential critical ASes.

3.2.2 Greedy Path Lining

Looking again at Algorithm 1 Line 10 and knowing that for this scenario our path lining information is limited to 245 ASNs, this path lining looks very different than it did for infinite path lining. We still start by getting all of the neighbors of everything on the current path. After that we have to rank them by the size of their customer cones, and then trim that to the top 245. However it turned out not to be quite that simple. If, for instance, the minimum sized customer cone in the known path lining is 2, we have to discard everything with a customer cone of the same size. This is because Nyx does not have a defined way to prioritize beyond customer cone size and since this is a set, there is no inherent order. To complete the check on Line 11, we therefore truncate the path lining from the log file to not include the AS with the minimum, then truncate our current path lining to get rid of any AS with the same sized customer cone and compare those. Since we have no way to break ties, we look at whether our path lining would explicitly break the 245 that were chosen, i.e. there is an AS in our potential path lining with a larger than minimum customer cone that is not in the path lining. If the largest AS in our potential path lining that is not in the path lining has the same size customer cone as the minimum, we consider it a valid possible path. To find the critical AS, we walk until we have a valid possible path, then add the next ASes to our anonymity set until adding one to the path forces it to no longer be a valid possible path. We find that in 513/999 instances of greedy path lining, at least one incorrect potential critical is in fact on the correct path.

Chapter 4

Evaluation

4.1 Evaluation

Overall this study finds that Nyx does leak privacy information. We consider our adversary as someone with the means to cause DDoS and observe the changes that Nyx forces into BGP advertisements. This adversary is able to obtain some amount of information about who the critical relationship almost 100% of the time. In the realistically infeasible case of infinite path lining, we are able to narrow the critical relationship to a max of 4 ASes, with 69% of our trials resulting in a single, correct, critical relationship. Due to feasibility issues with propagating paths longer 250, our more relevant result is how often we can correctly identify the critical when the path lining is limited to 245 ASNs. We know that the greedy path lining is performed by looking at the neighbors of the redirected path, and poisoning the top 245 neighbors in terms of customer cone size. This makes sense since the goal is to minimize disturbance i.e. how widely the poisoned path gets propagated. ASes with larger customer cones will spread the path more, so they are the highest priority for path lining. In this more feasible case, we are able to eliminate an average of 67.7% of the active topology, leaving our adversary with a sizeable but not meaningless anonymity set.

Table 4.1: Infinite Path Lining Results

# of Trials	# of Criticals
69.2	1
27.7	2
2.9	3
.2	4

4.1.1 Infinite Path Lining

Our infinite path lining algorithm correctly find the critical relationship 100% of the time with an average anonymity set of 1.34 potential critical relationships. A breakdown of the anonymity sets to come out of infinite path lining can be found in Table 4.1. Given that our active topology has 9,767 ASes in it, narrowing our anonymity set down to a maximum of 4 was considered enough of a success and we did not feel it necessary to attempt to discover more information about this scenario.

4.1.2 Greedy Path Lining

Our greedy path lining algorithm found the correct critical in 998/999 trials. As shown in Figure 4.1, we have a wide range of anonymity sets, with a minimum of 1, a maximum of 7,788, an average of 3,153.13 and a median of 2,538. Less than 10% of trials result in anonymity sets of over 7,000 while just over 19% of trials result in anonymity sets smaller than 1,000.

We then attempted to determine what affects the size of these anonymity sets. We considered the following:

- The length of the modified path Figure 4.2a
- The length of the original path Figure 4.2b
- The number of poisons added Figure 4.3b
- The fraction of the modified path that is a "large" AS Figure 4.4b
- The fraction of the path lining that is a "large" AS Figure 4.4a

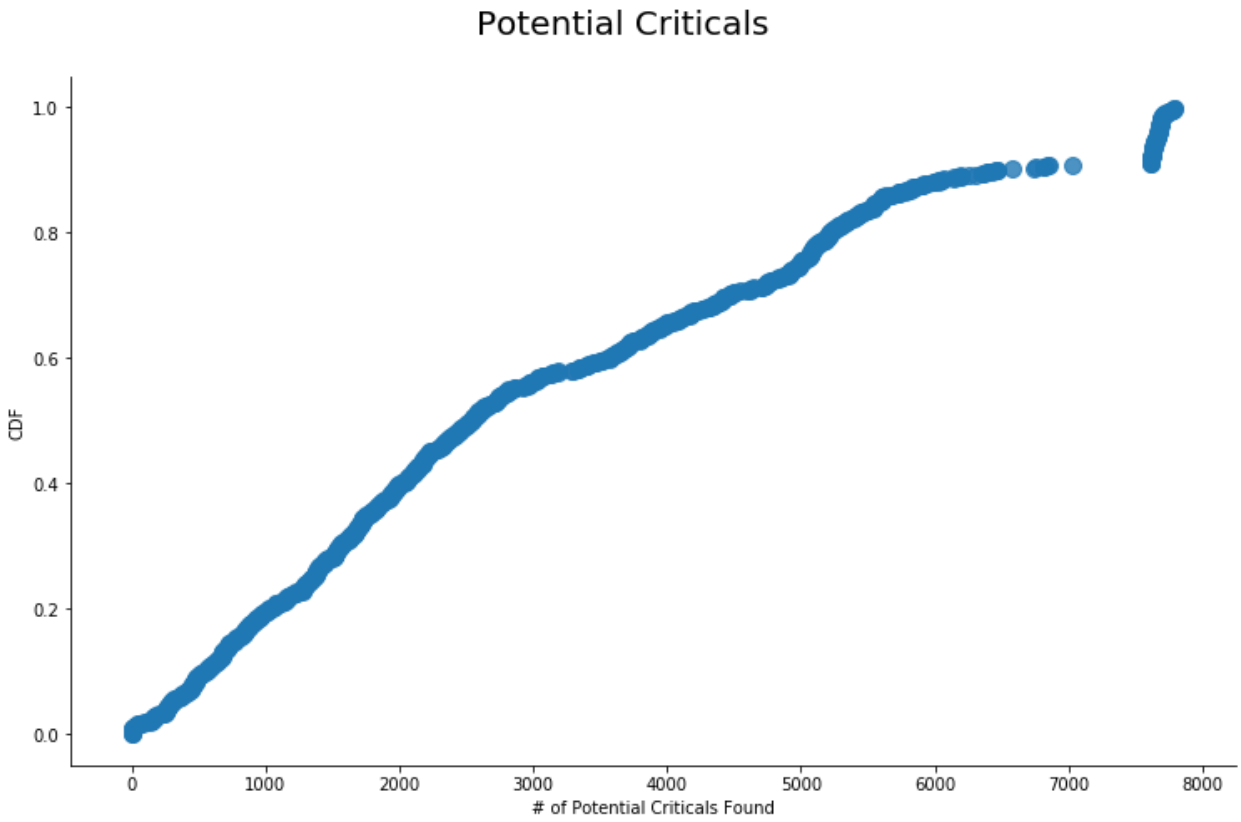


Figure 4.1: Distribution (CDF) of the Sizes of Anonymity Sets Across All Greedy Scenarios

- The average customer cone size in the path lining Figure 4.5
- The fraction of how much information we lose going from infinite to greedy Figure 4.6

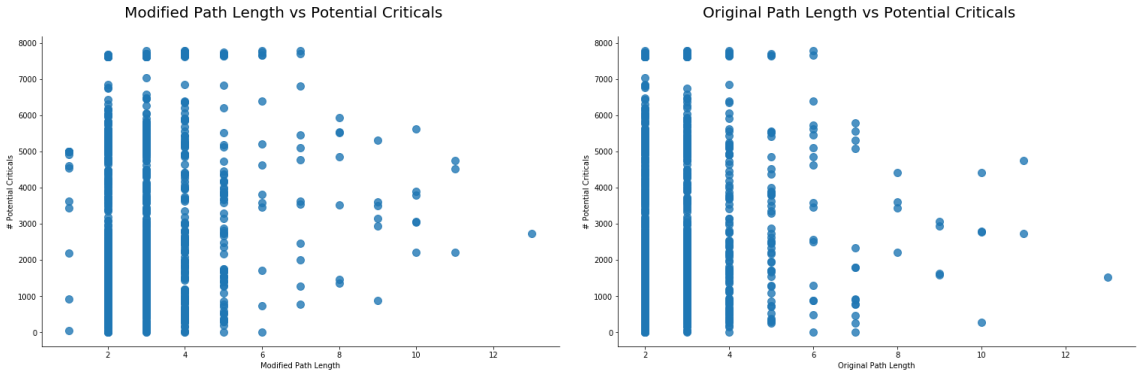
We first examined the lengths of the paths and found virtually no correlation in Figure 4.2, with some of the data points going against our suspicion that longer paths would lead to more ambiguity. In fact, once we get beyond 6 ASes in the original path and 7 in the modified, we see none of the largest anonymity sets.

In our next Figure 4.3 we see the number of poisons added as well as the minimum customer cone sizes found in the path lining. In Figure 4.3a we see that for every instance of narrowing down the anonymity set size to less than 2,000 there was a minimum customer cone size of 2. While going beyond that does not tell us much, we do see that having an AS with a small customer cone in the path lining limits the number of potential criticals we identify. While most paths in the greedy path lining scenario had all 245 poison spots filled, not quite all of them did. Figure 4.3b shows that for the 13 trials that do not have 245 ASes path-lined, 12 of them have less than 1,000 potential criticals.

After noting that small customer cones limit our potential criticals, we then looked to see how our anonymity set size relates to how many large ASes are in the path lining. Note that large ASes are defined as those with over 50 direct customers [6]. We looked at how often these ASes are on the modified path as well as how often they are in the path lining in Figure 4.4. Our initial assumption was that large ASes would be on more paths, leading to more ambiguity. As far as how often they are on the path, this did not hold particularly true as shown in Figure 4.4b. However for how often they are in the path lining in Figure 4.4a, we find an R^2 value of .55 which tells us this is an actual factor.

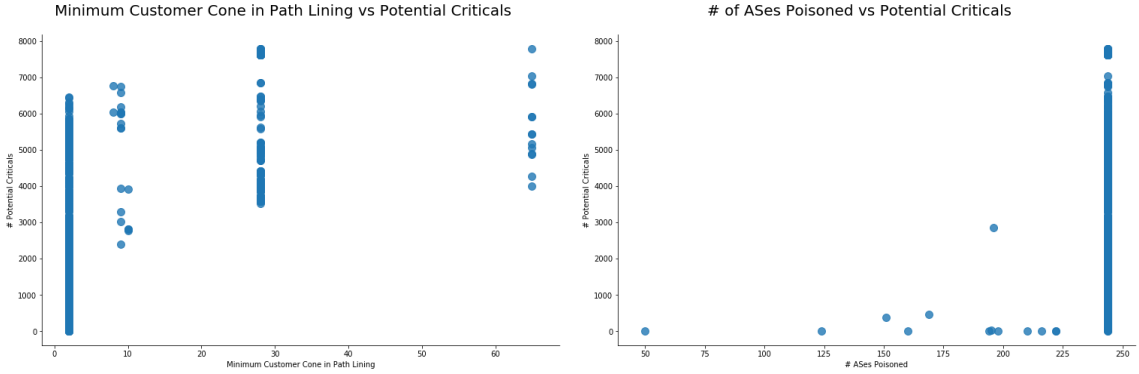
After seeing that the number of large ASes in the path lining had a notable correlation, we then examined the average customer cone size found in the path lining shown in Figure 4.5. Since this is how these ASes were sorted to be used in the path lining, we expect these averages to be high. With this data we find an R^2 value of .5.

Due to our significantly smaller anonymity sets from the infinite path lining combined with the knowledge of how greedy path lining happens, we noted that both Figure 4.4a and Figure 4.5 indicate that how much information we lose could be a factor. Since we



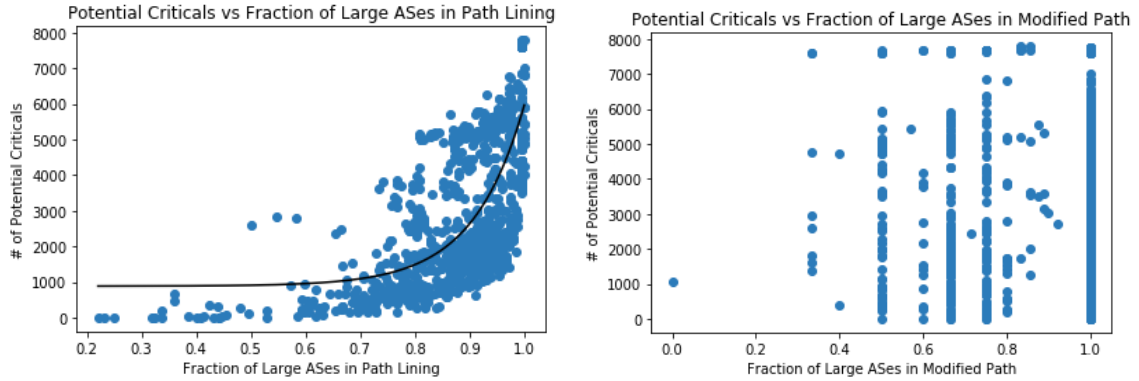
(a) Modified Path Length **(b)** Original Path Length

Figure 4.2: Path Lengths vs Potential Criticals Found in Greedy Path Lining Scenarios



(a) Minimum Customer Cone in Path Lining **(b)** Number of ASes Poisoned

Figure 4.3: Minimum Customer Cone and Number of ASes Poisoned vs Potential Criticals Found in Greedy Path Lining Scenarios



(a) Fraction Large ASes in Path Lining **(b)** Fraction Large AS in Modified Path

Figure 4.4: Large ASes Found in Greedy Path Lining Scenarios

rank ASes by their customer cones, we know we lose a significant amount of information, especially about paths with small ASes neighboring them. Pointing back to Figure 4.3a, we know that only instances where the path lining contains an AS with a customer cone size of 2 have the potential to have less than 2,000 potential criticals via our algorithm. We also know that 789/999 instances had at least one such AS and that all 999 runs in infinite path lining had such an AS, with an average of over 285 such ASes in each instance of infinite path lining. This led us to investigate what the ratio of how many ASes were in infinite path lining to how many are in the greedy path lining is, and whether it affects our anonymity set size.

This led to our strongest correlation, an R^2 value of .6, shown in Figure 4.6. Since most (98.7%) of the greedy path lining scenarios use all 245 poisons available, this graph really shows how much information we lose from infinite path lining. Infinite path lining resulted in an average of over 1,800 poisons being added, with a maximum of 4,533 poisons. Showing us that how many ASes line that path and do not make it into the path lining is the largest factor we discovered to explain why some anonymity sets are larger than others.

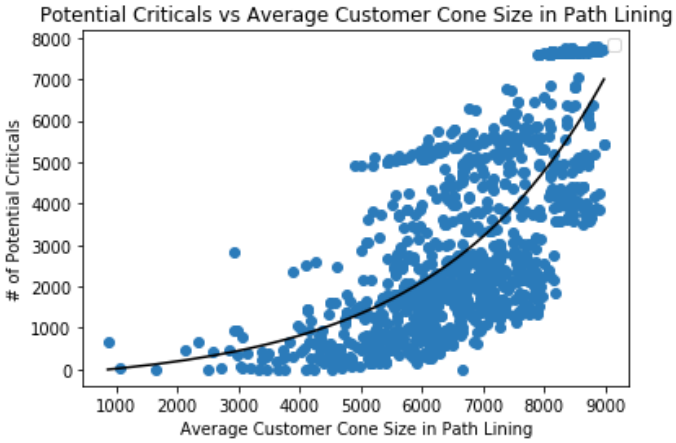


Figure 4.5: Average Size of Customer Cone in Path Lining vs Potential Criticals Found

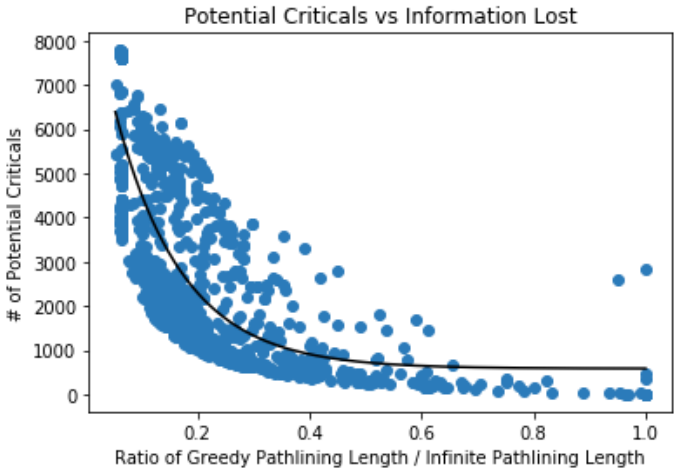


Figure 4.6: Greedy Path Line / Infinite Path Line

Chapter 5

Conclusions and Future Work

5.1 Conclusion

Overall we find that Nyx does leak information. Our adversary almost always is able to get the correct critical into the set of potential criticals. On average we are able to eliminate roughly $\frac{2}{3}$ of the active topology from the anonymity set. We find that the 3 factors that had a notable correlation were the fraction of large ASes in the path lining, the average customer cone size of the ASes in the path lining, and how much information is lost from infinite to greedy path lining.

5.2 Future Work

5.2.1 The Rogue One

While 99.89% correctness is good, this result begs the question of why our algorithm is incorrect in this single instance. This path has nothing observably unique about it. Further digging into this instance and closer examination of where in our path building algorithm it does not get that it should are required to find out why this instance does not behave like the rest of them.

5.2.2 Editing Nyx

The logical next step is to edit the way Nyx creates the path lining, with privacy in mind. This could be done a number of ways, but the way we discussed was taking off a number of the smaller ASes from path lining and adding in larger ASes. This would be an interesting problem to see how much randomness can be added to path lining while keeping the disturbance at a minimum. However, doing this would beg the question of how well it actually affects Nyx's privacy leaks, which would require re-doing our algorithms to allow for a certain amount of randomness, and this was determined to be out of scope. As the focus of this was to evaluate Nyx's privacy, editing Nyx did not make it into this work.

5.2.3 Paths

During our experiments to discover the correct relationship with infinite path lining, we also maintained how often we were able to find the correct path (roughly 95% of the time). However when we expanded our algorithm to look for a path with the greedy path lining, maintaining each path found to compare to the correct path became more memory intensive and drove performance down significantly. We made the decision that the correct path was not significantly valuable and took that measurement out. However with more time one could modify where some information is being stored and determine how often the correct path is found with greedy path lining.

5.2.4 Machine Learning

Because we are looking at 999 instances of a systematic way to reroute traffic, there is a decent chance we missed patterns. There are potentially factors that we did not even think to consider. For those reasons, we could definitely apply some different machine learning models to our data set and see if there are patterns we have yet to detect.

Bibliography

[1] Belenky, A. and Ansari, N. (2003). Ip traceback with deterministic packet marking. *IEEE Communications Letters*, 7(4):162–164. 1

[2] Biggio, B. (2016). Machine learning under attack: Vulnerability exploitation and security measures. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, IHamp;MMSec '16, page 1–2, New York, NY, USA. Association for Computing Machinery. 1

[3] Fayaz, S. K., Tobioka, Y., Sekar, V., and Bailey, M. (2015). Bohatei: Flexible and elastic ddos defense. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 817–832, Washington, D.C. USENIX Association. 1

[4] Giotsas, V. and Zhou, S. (2012). Valley-free violation in internet routing — analysis based on bgp community data. In *2012 IEEE International Conference on Communications (ICC)*, pages 1193–1197. 5, 6

[5] Kang, M. S., Lee, S. B., and Gligor, V. D. (2013). The crossfire attack. In *2013 IEEE Symposium on Security and Privacy*, pages 127–141. 2

[6] Oliveira, R., Willinger, W., Zhang, B., and et al. (2008). Quantifying the completeness of the observed internet as-level structure. 17

[7] Orsini, C., King, A., Giordano, D., Giotsas, V., and Dainotti, A. (2016). BGPStream: a software framework for live and historical BGP data analysis. In *ACM Internet Measurement Conference (IMC)*. 2, 9

[8] Seaman, C. (2016). Threat Advisory: Mirai Botnet. https://www.akamai.com/us/en/resources/our-thinking/threat-advisories/akamai-mirai-botnet-threat-advisory.jsp. 1

[9] Siris, V. A. and Stavrakis, I. (2005). Provider-based deterministic packet marking against distributed dos attacks. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 8 pp.–. 1

[10] Smith, J. M., Birkeland, K., McDaniel, T., and Schuchard, M. (2018). Withdrawing the BGP Re-Routing Curtain: Understanding the Security Impact of BGP Poisoning via Real-World Measurements. *arXiv e-prints*, page arXiv:1811.03716. v, 6, 8, 9, 11

[11] Smith, J. M. and Schuchard, M. (2018). Routing around congestion: Defeating ddos attacks and adverse network conditions via reactive bgp routing. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 599–617. v, 2, 7, 8, 9, 11

[12] Tran, M., Kang, M. S., Hsiao, H., Chiang, W., Tung, S., and Wang, Y. (2019). On the feasibility of rerouting-based ddos defenses. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1169–1184. 8

[13] UCSD-CAIDA (2019). CAIDA AS Relationship dataset. https://bit.ly/2RpRWuv. 10

[14] Wang, C., Li, Z., Huang, X., and Zhang, P. (2016). Inferring the average as path length of the internet. In *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pages 391–395. 8

[15] Xiang, Y., Zhou, W., and Guo, M. (2009). Flexible deterministic packet marking: An ip traceback system to find the real source of attacks. *IEEE Transactions on Parallel and Distributed Systems*, 20(4):567–580. 1

Vita

After completing my BS in Computer Science in spring of 2018, I went to work in NYC and quickly discovered the big apple was not for me. I found a role as a software developer at a local Knoxville company. After some time there, I really felt that my education just wasn't complete yet. Since at that point they didn't have room for a part-time employee, I wound up working part-time at Oak Ridge National Lab while working on my MS.