Poisonous Defense: Circumventing BGP Level Censorship

Arizona State University CSE 548

Leyba, Kirtus
kleyba@asu.edu

Schmidt, Ryan
raschmi4@asu.edu

Derbique, Austin
aderbiqu@asu.edu

February 21, 2021

Abstract

Internet security at the Border Gateway Protocol (BGP) level is a tug-of-war between BGP attacks and BGP defense mechanisms, often relying on similar technologies. We propose to take this back and forth struggle in a new direction by using *path poisoning* as a line of defense against censorship and surveillance actions on the BGP network. We propose to implement a technique for path engineering on BGP networks called Fraudulent Route Reverse Poisoning (FRRP) by designing poisoned path announcements on a test network. FRRP was used previously in the Nyx framework to mitigate DDoS attacks [11]. We will test our implementation and verify that FRRP can successfully mitigate DDoS attacks. Additionally, we will extend this work by investigating how FRRP can route around censorship and surveillance on the network. In order to determine if this tool can be used to avoid censoring nodes on the autonomous system (AS) network and prevent surveillance of packets as they move across the network we propose an experimental evaluation of our solution on a realistic network test-bed.

Keywords— SDN, BGP, FRRP, Censorship, Path Poisoning

I. Introduction

In the Border Gateway Protocol (BGP), autonomous systems (ASes) autonomously gather routes to regions of IP space known as prefixes. These routes are stored within BGP routers in *routing tables* [9]. Together, the routing tables of ASes around the world represent the global routing topology of the Internet. For a variety of reasons network operators are interested in controlling the routes that other ASes adopt. Sometimes this is for economic reasons [6], for repairing outages [6], or as part of a man in the middle attack [7]. Naively, this isn't possible because each AS can only choose where to send packets next, and cannot control the routes that other ASes adopt. Despite this, network operators have discovered a method to conduct traffic engineering on BGP networks: *path poisoning.*

Path poisoning is a method by which an AS announces false paths with a particular structure that influences other ASes to drop routes from their routing tables. This exploits a protection in BGP known as loop detection, where an AS will discard any received routes it receives that already contain that AS [9]. An AS will drop routes containing cycles and will replace them with other valid routes to the same destinations as they are announced.

While path poisoning is an attack on BGP, it can be used in a defensive manner, such as routing around congested links due to an ongoing DDoS attack [11]. The authors of [11] propose a system known as **Nyx** to detect and perform BGP path poisoning to eliminate ASes from routing considerations should they become congested. We aim to implement Fraudulent Route Reverse Poisoning (FRRP), the primary algorithm in Nyx, and apply it to the domain of censorship, where some node is known *a priori* to be monitoring or censoring traffic that flows through it. With application of path poisoning, we can cause incoming and outgoing traffic between us and a target prefix to avoid these censorship nodes.

The internet is increasingly used to express political opinions and organize protests [2], and as such the control of the Internet has become a focus of national actors. Many nation-states make use of BGP and ASes to capture traffic and censor it by injecting spurious TCP RST packets or by returning block pages [2]. On other occasions, entire national networks are shut down by the deactivation of key infrastructure [10]. By routing around such ASes by utilizing path poisoning,

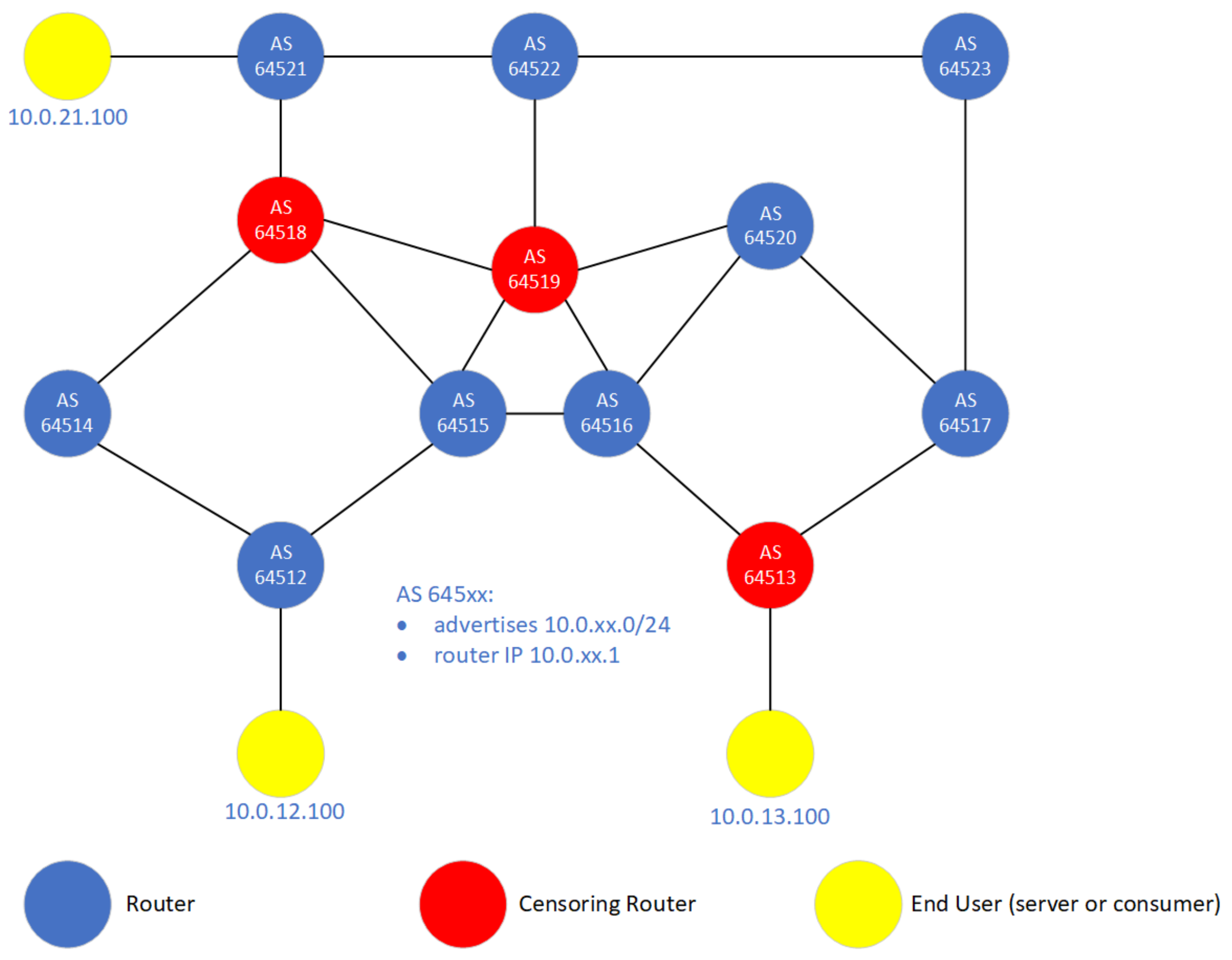


Figure 1: The network topology that we implemented to conduct tests on. Each node is a separate virtual machine connected together according to the edges shown.

we can bypass such censorship and even maintain connectivity when critical ASes are shutdown to prevent communication.

By conducting this project, we expect to demonstrate the effectiveness of BGP path poisoning strategies in avoiding censorship ASes where possible, and detecting cases where avoiding such ASes is not possible. The poisoning method does not require coordination between us and any other AS to be effective. We have built a test network consisting of multiple simulated ASes, each connected together via a topology representing actual internet topologies between core routers. BGP was be used to establish routes between these simulated ASes, and we then conducted our path poisoning attacks in this network.

II. System Models

A. System Model

Due to BGP's autonomous nature, given a particular network structure it is not trivial to estimate the routing tables for each AS, or even if the routing tables will converge to a stable state. Thus, we require a real test environment to conduct useful experiments. Our test network consists of 15 virtual machines (VMs) connected together via a virtual switch. Each VM was running Ubuntu Server 20.04 with 2 CPU cores at 2.1 GHz, 3 GB of RAM, and 16 GB of disk space. Networking between the VMs was handled by a VMWare vSphere vSwitch; each VM was assigned an IP address according to Figure 1 for communication between the VMs. All autonomous system VMs additionally had IP addresses assigned to their network interface for each outgoing connection to another AS; these were each in small independent subnets so that the ASes could communicate with each other to establish BGP routes. Finally, each VM had an IP address in a different subnet assigned so that it could communicate with our ansible server for automated provisioning and external accessibility.

To provision these VMs, we used ansible which allows us to store "playbooks" of configuration and deploy them to all or a subset of the VMs. This automation allowed us to install and deploy the BGP configuration across the ASes as well as web servers on the end user nodes. We additionally used ansible to perform testing of the BGP poisoning attack and to extract routes and other data from each node to determine if the attack was successful and how BGP modified the routing tables of each AS. For BGP support, each AS VM had FRRouting installed with its BGP daemon enabled.

To conduct tests, we installed an nginx web server on each end user VM, and determined whether or not it was reachable before and after performing BGP poisoning. Our censorship routers would examine the packets going through them and we would determine attack success rate based on whether or not the number of packets going through the censorship routers dropped to 0.

Originally, we had planned to build up the system infrastructure on GENI or CloudLab, both of which are ways to spin up capacity for academic projects. The main roadblock we ran into with CloudLab was that our project was not approved until well into the project timeline, necessitating an alternative plan in order to ensure the project was finished on time. While the CloudLab project was eventually approved, we already had infrastructure set up on the VMWare vSphere server and spending time rebuilding that on CloudLab did not seem necessary given that vSphere was suiting our needs.

B. Software

Various tools we have used are as follows:

- VMWare vSphere and ESXi as the hypervisor hosting the test network virtual machines. The VMs themselves ran Ubuntu Server 20.04.
- FRRouting to provide an implementation of BGP for the test network [8].
- Another software framework worthy of exploration for a test network is Mininet [14]. This software creates a virtual network capable of creating ASes with links and other resources required for testing our algorithm. Mininet was used in developing the poisoning utility while the main test network was being built out.
- Multiple Autonomous Systems, along with a critical AS which will be the AS to be poisoned. These can be virtualized using FRRouting.
- A traffic generator to send packets to a destination for ASes to route around the critical AS. One such solution is SolarWinds WAN Killer Network Traffic Generator [13].
- A monitor to observe traffic as it passes from AS to AS using BGP such as BGP Stream [15].
- We will write the implementation of FRRP using Python.

C. Threat Model

A variety of censorship and surveillance can be done at the BGP level of the Internet. We can define two broad categories: *First*, ASes can be directly removed from the network by shutting down routers or even cutting network cables. This has been done historically to prevent communication during times of unrest [10]. *Second*, network operators might install packet sniffing, DNS redirecting, or keyword filtering devices along paths in the AS network [3]. In the first case, our intention will be to ensure connectivity despite some ASes being strategically removed from the network. In the second case, we seek to conduct path engineering such that ASes with censorship or surveillance devices are avoided by packets forwarded from a host AS.

With these types of threats in mind, we define our threat model as follows: Some ASes are known *a priori* to conduct censorship or surveillance. We will refer to this set of ASes as *Censored Routers*. In our threat model, the censored routers each possess an oracle which can halt traffic and inspect packets. For the purposes of our experiments the exact details of this oracle are irrelevant. The set of censored routers is constant, and all non-censored routers are assumed to not possess the censoring oracle. The entity conducting the censorship can only interact with packets that traverse censoring routers and with the connections of those routers. We are not investigating the ability of the censoring entity to conduct its own FRRP, which we leave as future work.

III. Project Description

The first phase of the project involves setting up a test network. The test network will contain multiple links between simulated ASes, some of which are pre-designated as "censoring" ASes. In some cases, alternative routes between a source prefix and destination prefix that do not travel through a censoring AS exist. In other cases, it is not possible for packets to route between the source and destination without travelling through a censoring AS. This will allow us to examine the performance of our utility in a variety of realistic cases.

Next, we will build a utility to implement the FRRP framework described in [11]. This framework allows for BGP path poisoning in both the incoming and outgoing direction from our source AS, without requiring coordination from other ASes on the network. We will verify that after the poisoning is performed, that no packets between the source and destination route through a censoring AS.

A. Project Overview

Broadly speaking, there are three areas of tasks in this project: writing the paper, setting up the test network, and creating the BGP path poisoning framework. These areas can all be started independently, although we cannot test the path poisoning framework without first having completed the test network. In the task breakdown below, dependencies on previous tasks will be highlighted.

B. Task 1: Create Proposal Document

This task involves researching related literature to BGP path poisoning and means of internet censorship. We have identified [11] as the paper to implement after consulting with Dr. Huang, and will be taking that paper in a new direction with avoiding censoring ASes instead of routing around congestion caused by DDoS.

C. Task 2: Provision Network Resources

Using research clouds such as CloudLab or GENI, or by using SDN infrastructure elsewhere, we will need to spin up enough nodes to provide a good test-bed for our simulated network. Each node will represent one AS (router), responsible for some /24 prefix on the 10.0.0.0/8 subnet. We will also provision nodes on some ASes to represent users or servers running on those networks.

D. Task 3: Create Test Network

We will establish static routes between certain nodes to represent physical interconnects between ASes, and then use FRRouting or similar software to implement the BGP protocol on these nodes to propagate information of these interconnects. This will define a simulated internet between the ASes we set up. Then, we will nominate certain nodes as “censoring” ASes, and install some sort of network monitoring software on them to determine whether or not data is flowing through those nodes at any given point in time. These “censoring” ASes are assumed to be known *a priori* in the actual internet. On the internet, they can be detected using methods described in [2], which is outside of the scope of this project. We will explore using Mininet as a way of deploying said test network [14].

This task depends on Task 2 being completed first.

E. Task 4: Run Services on Test Network

Before implementing any BGP path poisoning attacks, we will first test various user and server nodes to ensure that traffic flows between them as expected, and the users can talk to the services. Some of these flows will travel through censoring ASes, and we will be able to see that traffic logged in our traffic monitoring tool to verify that our “censorship” is working. Example servers would include a webserver providing content over HTTP, a DNS server, and an SSH server. This provides a mixture of TCP and UDP traffic flows to show that our test network is robust and a decent simulation of the real internet.

This task depends on Task 3 being completed first.

F. Task 5: Create Path Poisoning Utility

We will follow the framework given in [11] to create a program that performs BGP Path Poisoning from a given source AS to a given target AS, while avoiding the known censoring ASes. The paper is only concerned with one “critical network” to connect to our source AS, so this would represent the user attempting to bypass censoring for a single website or other online resource rather than a general bypass.

G. Task 6: Test Path Poisoning Utility

To ensure the poisoning attack is successful, we will then test the service from the user to the target server, and verify that the censoring ASes do not log anything in their traffic monitoring tool. If the censoring AS was able to log packets before the poisoning but not afterwards, this indicates the poisoning was successful and the censoring AS is no longer part of the route between the user and server.

This task depends on tasks 4 and 5 being completed first.

H. Task 7: Enhance Path Poisoning Utility

As an enhancement over the paper, we will extend the utility to allow bypassing the censoring ASes for more than just one target AS. When a user wishes to not be censored or monitored, they likely wish that to apply to all of their internet traffic instead of just one destination. The authors of [11] describe extending their framework to multiple critical networks as future work, and we will provide an implementation of that.

This task depends on Task 6 being completed first.

I. Task 8: Create Midterm Report

With our current timeline, we anticipate that the midterm report will be due after we complete Tasks 2, 3, and 5. We will see if we are still on track and what modifications we needed to make to this proposal for the project to move forwards.

This task depends on Tasks 1, 2, 3, and 5 to be completed first.

J. Task 9: Create Final Report and Demo

Upon completion of Task 7, our project is complete and we will work on making the final report and demonstration video.

This task depends on all other tasks being completed first.

K. Project Task Allocation

Our team will work together on all tasks, however, we will allocate the following tasks to task leaders to lead the effort and find solutions to obstacles in the way of getting the task completed. The overall workload breakdown for each task is given in 1. We have elected **Kirtus Leyba** as the project lead.

Task	Task Leader
Create Proposal Document	Austin Derbique
Provision Network Resources	Ryan Schmidt
Create Test Network	Austin Derbique
Run Services on Test Network	Kirtus Leyba
Create Path Poisoning Utility	Kirtus Leyba
Test Path Poisoning Utility	Ryan Schmidt
Enhance Path Poisoning Utility	Austin Derbique
Create Midterm Report	Kirtus Leyba
Create Final Report and Demo	Ryan Schmidt

Table 1: The workload distribution of the project

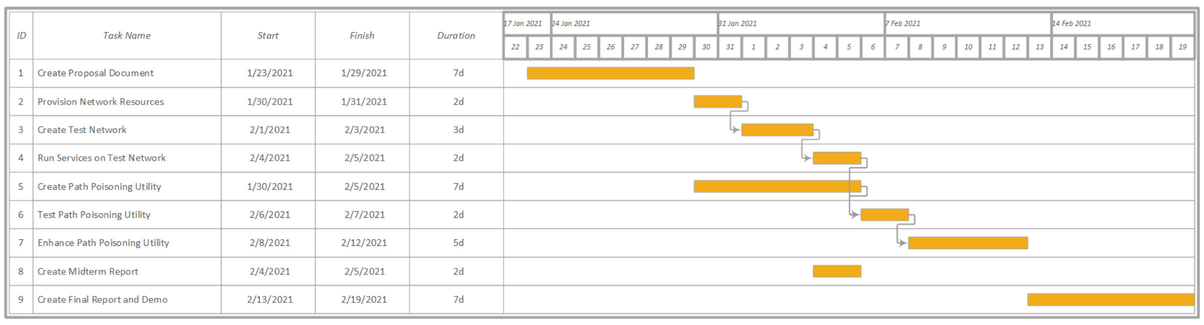
L. Deliverables

The main deliverables of the project will be the utility to perform BGP path poisoning given known censoring ASes, a demonstration video showing this utility working on our test network, and a final report detailing our work and the research that went into creating the utility. All of these deliverables will be accessible via the class GitLab repository and our presentation and demo video is available at YouTube [4].

At the time of the midterm report, we are currently working on provisioning the test resources and creating the path poisoning utility. Work on each proceeds independently. CloudLab has not yet approved our account request, so we are looking into alternatives to host our test network should they not come through in the next few days.

At project completion, we have successfully created all deliverables. We did not use CloudLab due to the request being approved far too late, instead opting for a local VMWare vSphere instance to spin up virtual machines on. Our poisoning utility as well as the ansible configuration playbooks for the test network are all available on GitLab.

M. *Project Timeline*



ID	Task Name	Start	Finish	Duration
1	Create Proposal Document	1/23/2021	1/29/2021	7d
2	Provision Network Resources	1/30/2021	1/31/2021	2d
3	Create Test Network	2/1/2021	2/3/2021	3d
4	Run Services on Test Network	2/4/2021	2/5/2021	2d
5	Create Path Poisoning Utility	1/30/2021	2/5/2021	7d
6	Test Path Poisoning Utility	2/6/2021	2/7/2021	2d
7	Enhance Path Poisoning Utility	2/8/2021	2/12/2021	5d
8	Create Midterm Report	2/4/2021	2/5/2021	2d
9	Create Final Report and Demo	2/13/2021	2/19/2021	7d

Figure 2: Gantt chart of project timeline

IV. Risk Management of the Project

The main risk we are facing right now is related the provisioning of network resources. Our account request at CloudLab is still pending approval, and we cannot provision anything on CloudLab until that is approved. To mitigate this risk and unblock future tasks that depend on the test network being provisioned, we are evaluating other places we can host the network. Ryan Schmidt has a server at home that could additionally be used to host virtual machines for all of these nodes.

Another risk relating to the network setup is if the path poisoning attack disables network connectivity to the point that we are no longer able to log into nodes to determine if packets are flowing properly. This can be mitigated by using out-of-band management such as virtual machine console access to log into nodes and restore proper connectivity.

The risk preparation in the midterm report helped us complete our test network on time by using the alternative infrastructure to host the nodes. Additionally, to mitigate the risk of the attack disabling network connectivity, we designed the test network in a way to be resilient against this. Ansible connected to the test nodes using a completely different subnet than was being used for BGP routing, ensuring that connectivity existed regardless of what BGP attacks were performed. Furthermore, the BGP network was designed to revert back to a clean slate on node reboot, allowing us to reproduce our tests and try other methods without lots of manual work required to reset the network. By taking these precautions, we were able to finish on time and create all of our deliverables.

V. Discussion

Network censorship is another element of network security that isn't often the first concern of network security approaches. Anonymity and data privacy are subjects that are both controversial and of large interest to powerful Internet organizations. These include national networks and Internet companies such as social networks. Our approach proposed here reflects the growing interest in the Internet research community in anonymity and Internet censorship.

A. *Implementation of Proposed methodology*

Test Network The network itself was developed on top of a VMware ESXi server, providing compute, networking, and storage to our virtualized nodes. Two networks were created, one for management and one for the BGP routers themselves. When referring to the *test network*, we refer to the 10.0.0.0/8 network comprising a different subnet for each node. In Figure *A.*, we see a combination of yellow, blue, and red nodes. The only way nodes on the test work are capable of communicating with each other is with defined routes configured in their route tables. These route tables are configured in such a way that the topology is identical to what is displayed in the topology figure.

Deployment Mechanism Using Ansible [1], as a configuration management tool, we built tooling to automatically create and initialize our environment as well as run our experiment and report back on the data captured. As seen in Figure *A.*, a controller node with IP 192.168.80.27/20 is used to orchestrate a series of commands sent out to all nodes on the network. More information regarding how to run and deploy software can be found in the repository readme file.

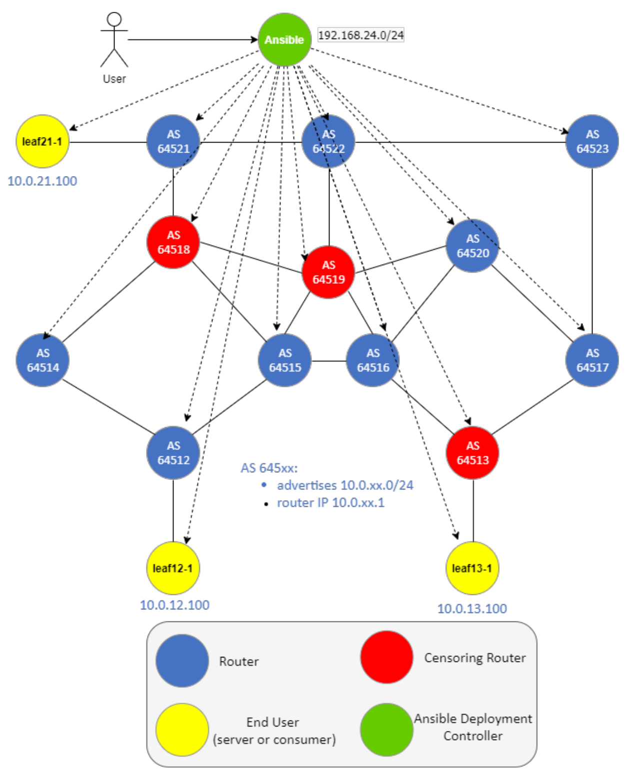


Figure 3: Two networks are used in our environment. The first is a control network used for pushing configuration updates from our Ansible deployment node. The second is the test network used for BGP routing experiments.

Algorithm 1: Path Poisoning Pseudo Code. The source AS S searches its routing table for paths including the target AS T. The target ASN is prepended to a duplicate path p′ and then that path is added to a changed set of paths ΔP. Finally, the source AS announces the new paths to the target AS.

```
input  : A source AS $S$, a target AS $T$
$\Delta P \longleftarrow$ new path set;
for path $p$ in $S.routingtable$ do
    if $T \in p$ then
        p′ $\longleftarrow$ p.prepend($T$);
        add($\Delta P$, p′);
    end
end
announcePaths($\Delta P$, $T$);
```

Path Poisoning To conduct path poisoning in our final simulation we utilized the BGP configuration option in many popular BGP daemons ”BGP as-path prepend”. The AS responsible for poisoning modified its configuration to strategically append the poisoned ASes own ASN to paths it announced to the target AS. This triggers the built-in loop detection algorithm, causing the poisoned AS to drop the poisoned routes. Effectively, this allowed us to conduct path engineering and remove selected edges from the network without direct control of the censoring nodes. A pseudo-code formulation for our path poisoning procedure is given in algorithm 1.

Path Analysis We simulated a BGP enabled network in order to investigate the effectiveness of path poisoning to circumvent censorship and to explore the method’d impact on the network. The results of these simulation experiments are detailed in table 2. A major result is that the path poisoning was effective at altering paths taken on the network. We show that by poisoning select ASes we can change which packets are intercepted at which AS.

Our threat model described in section *C.* explains the types of censorship we planned to test our method against. By decreasing the number of times packets encounter censorship nodes in our network we reduce the chances that some actor could perform censorship or surveillance on the traffic. An instance of this in our results is show in experiments 1 and 2, where we reduce the number of packets that are intercepted by the censoring node AS65418.

Our results also show the impact of topology on the effectiveness of path poisoning as a means to get around censorship. Note that mitigating the censorship power of AS64513 was impossible in our simulation. This is because in the network topology AS64513 is effectively a border node,

Exp. Number	Poisoning ASes	Target ASes	Packets Intercepted AS64518	Packets Intercepted by AS64519	Packets Intercepted by AS64513
0	–	–	4	0	4
1	64521	64518	2	2	4
2	64522	64519	4	0	4
3	64521, 64522	64518, 64519	2	0	5
4	64521, 64522, 64516	64518, 64519, 64513	4	0	4

Table 2: Table of path poisoning experiments to circumvent censorship. We tabulate the packets that encounter censoring ASes with traceroute. For each experiment we have a different set of ASes conducting path poisoing (Poisoning ASes column) and target ASes to be poisoined (Target ASes column). They are listed respectively for each experiment.

preventing access for the hosts behind it to the rest of the network. This is a realistic scenario, and somewhat reflective of real life national AS networks [5].

B. Challenges

There were several challenges faced in this project, some of them overcome, others forcing us to pivot into an alternate direction. First and foremost, the time constraints given for the project meant that we needed to have a virtual network to test with as soon as possible. After waiting over a week for approvals from the research public cloud vendor Cloudlab, the team made a decision to use VMWare ESXi instead. It is worth noting that well into our development of a test network on VMware were we notified of approval for use on Cloudlab's infrastructure. With time being of the essence, we pressed forward with our development on VMware.

Additionally, a challenge faced by this project was the design of an experimental BGP network topology. We simultaneously needed to maintain a static topology for recreating experiments and have a variety of simulation situations for interesting results. With the resources at hand we designed a diverse topology that represented both of these qualities. In the future, large scale experiments for path poisoning and censorship mitigation could take place on very large BGP networks. Research progress into path poisoning on the Internet scale is an important next step for this censorship mitigation strategy.

C. Review and Considerations

A major question raised by this censorship mitigation technique is that of practicality. Can path poisoning really allow users to access blocked Internet services? One requirement is that there are censored networks with gaps where censoring devices can be avoided. Evidence of this exists in reality. For instance, the authors of [3] show that traceroutes could reach entirely past the great firewall of China in some rare cases, particularly when utilizing the Chinese Education and Research Network (CERNET). If a particular network is more prone to letting packets through without inspection or interception then that can be seen as a censorship gap that could theoretically be exploited by the method presented in this paper.

There is also experimental evidence that our technique would face new challenges if deployed on the real Internet. Particularly important is that path poisoning is less likely to succeed in the wild [12]. BGP network operators may employ defensive strategies against path poisoning, or network dynamics may prevent routers from dropping the targeted routes.

D. Looking Forward

Our approach isn't necessarily limited to BGP and the AS network. A potential future research outcome would be developing similar technologies for circumventing nefarious nodes in a variety of networks. Small scale autonomous networks formed by Internet-of-things (IoT) devices have the potential to create a large amount of useful functionality while also presenting new security challenges. Can networks be robust to the style of path engineering proposed here? This is a potential future research interest for the IoT community.

VI. Conclusion

In the paper, we have demonstrated path poisoning as a defense mechanism against Internet censorship. Path poisoning has traditionally been used for traffic engineering attacks or for DDos prevention. Using fraudulant reverse route poisoning as our platform, we worked to extend the functionalities to a new domain: avoiding censorship and surveillance on the AS network. We described the test network in detail and explained deployment procedures for implementign BGP path poisoning in simulated test environment. We confirmed past work showing that path engineering was possible with path poisoning. In our simulation study of AS level censorship we confirmed that BGP path poisoning was successful in broadcasting false routes to neighboring ASes, and censoring ASes altered their paths, allowing packets to circumvent censorship devices.

Acknowledgement

We thank Dr. Dijiang Huang for his guidance on selecting a novel research topic.

References

[1] Red Hat Ansible. URL: `https://www.ansible.com/`.

[2] Shinyoung Cho et al. "A Churn for the Better: Localizing Censorship Using Network-Level Path Churn and Network Tomography". In: *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '17. Incheon, Republic of Korea: Association for Computing Machinery, 2017, pp. 81–87. ISBN: 9781450354226. DOI: `10.1145/3143361.3143386`. URL: `https://doi.org/10.1145/3143361.3143386`.

[3] Roya Ensafi et al. "Analyzing the Great Firewall of China over space and time". In: *Proceedings on privacy enhancing technologies* 2015.1 (2015), pp. 61–76.

[4] Kirtus Leyba, Ryan Schmidt, and Austin Derbique. *Poisonous Defense: Circumventing BGP Level Censorship*. Feb. 2021. URL: `https://youtu.be/PmgrjoqBPKg` (visited on 02/21/2021).

[5] Kirtus G Leyba et al. "Borders and gateways: measuring and analyzing national as chokepoints". In: *Proceedings of the 2nd ACM SIGCAS Conference on Computing and Sustainable Societies*. 2019, pp. 184–194.

[6] Matthew Luckie. "Spurious Routes in Public BGP Data". In: *SIGCOMM Comput. Commun. Rev.* 44.3 (July 2014), pp. 14–21. ISSN: 0146-4833. DOI: `10.1145/2656877.2656880`. URL: `https://doi.org/10.1145/2656877.2656880`.

[7] Alex Pilosov and Tony Kapela. "Stealing the Internet: An Internet-scale man in the middle attack". In: *NANOG-44, Los Angeles, October* (2008), pp. 12–15.

[8] FRRouting Project. *FRRouting*. 2021. URL: `https://frrouting.org/` (visited on 02/06/2021).

[9] Y. Rekhter, T. Li, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271. `http://www.rfc-editor.org/rfc/rfc4271.txt`. RFC Editor, Jan. 2006. URL: `http://www.rfc-editor.org/rfc/rfc4271.txt`.

[10] Matt Richtel. "Egypt cuts off most internet and cell service". In: *New York Times* 28 (2011).

[11] James Smith and Max Schuchard. "Routing Around Congestion: Defeating DDoS Attacks and Adverse Network Conditions via Reactive BGP Routing". In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 599–617. DOI: `10.1109/SP.2018.00032`.

[12] Jared M Smith et al. "Withdrawing the BGP Re-Routing Curtain". In: NDSS. 2020.

[13] LLC SolarWinds Worldwide. *Best Network Traffic Generator and Simulator Stress Test Tools*. 2020. URL: `https://www.dnsstuff.com/network-traffic-generator-software` (visited on 01/28/2021).

[14] Mininet Team. *Mininet*. 2018. URL: `http://mininet.org/` (visited on 02/06/2021).

[15] The Regents of the University of California. *BGPStream: A Software Framework for Live and Historical BGP Data Analysis*. 2016. URL: `https://bgpstream.caida.org/pubs#bgpstream-tech-rep` (visited on 01/28/2021).