DDoS Detective: A Collaborative Telemetry System For DDoS Identification and Mitigation

Hirsh Guha
Princeton University
hguha@princeton.edu

Josh Gardner
Princeton University
jg41@princeton.edu

Rushi Shah
Princeton University
rushis@princeton.edu

Abstract

Distributed Denial-of-Service (DDoS) is a broad class of network attacks that overwhelm a server with a high volume of traffic [11]. Common DDoS attacks such as Memcached DDoS [15] and Syn Flood DDoS [2] rely on IP spoofing to achieve attack amplification. This paper presents a system called "DDoS Detective" for cooperation between telemetry systems across ASes to detect and mitigate amplification based DDoS attacks that use IP spoofing. Many recent systems have used techniques from statistics and machine learning to achieve successful attack detection [5][3][8]. In contrast, DDoS Detective uses no statistical techniques. Low level operations in the system are formulated and tested using Sonata [6] queries and can be combined across ASes to map an attack. In running these queries on Sonata, we unintentionally discover and present implementation issues with the underlying Sonata code base.

1 Introduction

Network operators may wish to detect that their system is experiencing a distributed denial of service (DDoS) [11] attack. DDoS attacks can have devastating effects on applications by overloading the network to deny access to legitimate users. These attacks can involve multiple distinct ASes, as depicted in Figure 1. This paper aims to describe how DDoS packets traverse network domains, and to what extent collaboration between network administrators can identify and mitigate ongoing DDoS attacks. This paper leverages a network telemetry tool called Sonata [6] that allows network administrators to collect the necessary forensic evidence, which, when viewed as a whole, may provide a useful overview of the relevant players in an amplification attack. Although our discussion applies to packet spoofing based amplification attacks generally, we will often reference the memcached DDoS attack [15] for the purposes of illustration.

2 The Distinct Parties in An Amplification Attack

Adversarial servers perform amplification attacks by utilizing botnets to attack and compromise target servers. The path of an attack packet through the internet can be abstracted into three general parties: the attacker AS, the intermediate ASes, the victim AS. Each of these three sections will likely be administered by distinct parties and would thus be running independent telemetry systems. The set of intermediate ASes can be further subdivided into botnet ASes (any AS that contains one of the bots in the botnet), and transport ASes (any AS involved in transporting packets between the adversary and the bot, or any AS involved in transporting packets between the bot and the victim). Because each party has a different viewpoint of the attack, each would run different queries for detection and play different roles in mitigation. The following subsections will address the information each party has for detection, provide specific Sonata queries for detection, and discuss coordination strategies for mitigation.

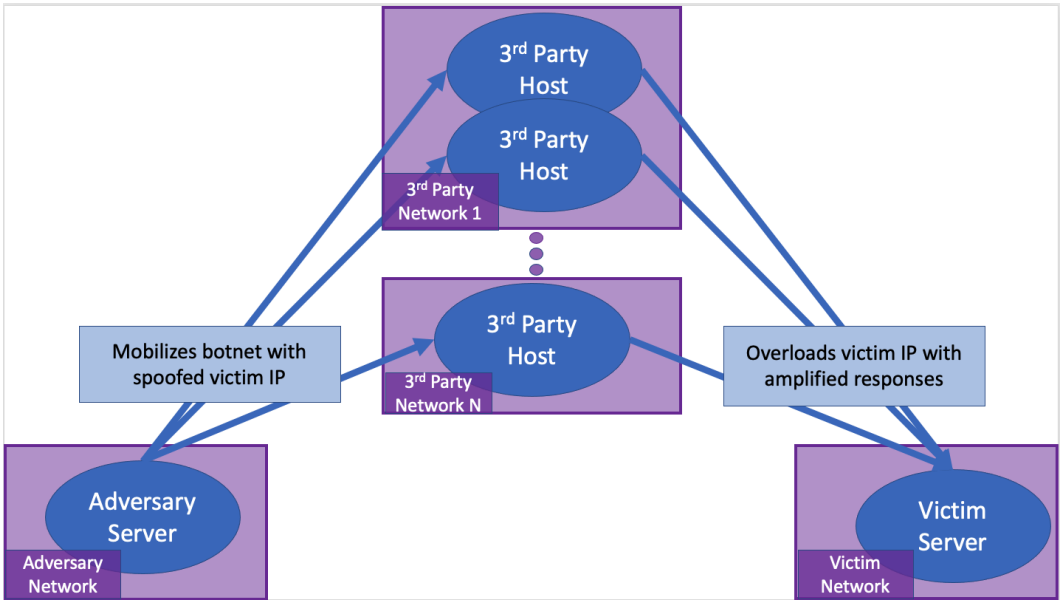

Figure 1: Network topology of standard amplification attack

2.1 Victim AS

The identification of an amplification attack on a victim network is relatively straightforward. There is an asymmetry between the number of incoming response packets and the number of outgoing request packets. This can be detected easily with minor modifications to an existing set of Sonata queries provided by its authors:

Listing 1: Modified Version of the Sonata DDoS Query from Gupta et. al. 2018 [7]

```
n_resp = (PacketStream(1)
    .filter(filter_keys=('ipv4.protocol',), func=('eq', 17))
    .map(keys=('ipv4.dstIP', 'udp.sport'),
         map_values=('count',),
         func=('set', 1,)
        )
    .reduce(keys=('ipv4.dstIP', 'udp.sport'), func=('sum',))
    .filter(filter_vals=('count',), func=('geq', T))
    )

# Confirm the fin flag number here
n_req = (PacketStream(2)
    .filter(filter_keys=('ipv4.protocol',), func=('eq', 17))
    .map(keys=('ipv4.srcIP', 'udp.dport'),
         map_values=('count',),
         func=('set', 1,)
        )
    .reduce(keys=('ipv4.srcIP', 'udp.dport'), func=('sum',))
    .filter(filter_vals=('count',), func=('geq', T))
    )

victim_ips = (n_resp
    .join(query=n_req, new_qid=3)
    .map(keys=('ipv4.dstIP', 'ipv4.srcIP',),
         map_values=('count1', 'count2',),
         func=('diff',)
```

```
        )
    .filter(filter_vals=('diff3',), func=('geq', T))
    .map(keys=('ipv4.dstIP'))
    )
```

Note that we have slightly modified the original code for the sake of semantic clarity. The code works by computing the difference between the number of response and request packets for each destination/source pair. When the difference is greater than a threshold T, this indicates that a DDoS attack is taking place since the destination is receiving responses that it appears to have never requested, suggested that someone else is spoofing their IP and making the requests.

2.2 Intermediate ASes

The set of intermediate ASes can be further subdivided into botnet ASes (any AS that contains one of the bots in the botnet), and transport ASes (any AS involved in transporting packets between the adversary and the bot, or any AS involved in transporting packets between the bot and the victim). Note that a transport AS can be involved in one or both of the path between the adversary and the bot, and the path between the bot and the victim.

The transport ASes simply receive a packet from one network and are directed to pass it along to another network. This gives them limited information, because they do not know much about the sender or the receiver.

However, ASes containing botnet participants have slightly more information than transport ASes. In particular, they have access to the request packet that, for example in a memcached DDoS attack, requests an IoT device's memcache, and they have access to the response packet that delivers said memcache. This allows ASes containing botnet participants to detect packet amplifications in a way that intermediary transport ASes can not.

Furthermore, an AS may contain multiple bots being used in the same DDoS attack, as demonstrated by 3rd Party Network 1 in Figure 1. If many botnet devices are on a single AS, this leads us to a potential amplification DDoS detection telemetry mechanism for botnet ASes. Botnet ASes can run a telemetry query to look out for may distinct IP addresses within their network originating large responses to a single IP address. This would indicate that many bots within the network are being leveraged to amplify an attack on the victim server.

Note that a large number of distinct IP addresses sending normal sized packets is not noteworthy in and of itself. Consider the case where the destination may be a large tech company data center. Similarly, a small number of IP addresses sending large responses to the same source would not be noteworthy. Maybe they are uploading videos or other large files to the destination host. However, a high number of distinct large responses to the same IP destination may help a AS detect that it contains devices participating in an ongoing DDoS attack. Moreover, this can be detected with existing telemetry systems (see Sonata code below). Nevertheless, this detection scheme depends on the botnet being concentrated on particular ASes. Significantly, if the attacker makes sure the bots are thinly distributed across ASes, this detection scheme will fail completely.

Listing 2: Query for External DDOS Target Identification

```
num_senders_threshold = 40
packet_len_threshold = 40
likely_ddos_targets = (PacketStream(1)
    .filter(filter_keys=('ipv4.totalLen',),
            func=('geq', packet_len_threshold)
           )
    .map(keys=('ipv4.dstIP', 'ipv4.srcIP'))
    .distinct(keys=('ipv4.dstIP', 'ipv4.srcIP'))
    .map(keys=('ipv4.dstIP', ),
         map_values=('count',),
         func=('set', 1,)
        )
    .reduce(keys=('ipv4.dstIP', ), func=('sum',))
```

```
    .filter(filter_vals=('count',),
            func=('geq', num_senders_threshold)
           )
    .map(keys=('ipv4.dstIP',))
    )
```

Note that this detection method could be greatly improved by using statistical measures to determine what constitutes anomalous traffic. Our heuristic is nevertheless useful in so far as it can add information to the broader detection system. As an example, on detection of a potential amplification attack coming from inside its networks, an AS could send a signal to the target IP and AS indicating that it may be under attack. Notably, if an AS identifies that it is experiencing a DDOS attack using the UDP request/response asymmetry detector shown earlier and additionally receives warnings from other ASes, it can be highly confident it is under attack. Furthermore, if on receiving a warning message, the victim currently undergoing an attack sends a reply that it is in fact being attacked, the AS containing botnet participants can potentially take mitigating actions. However, this requires an existing set of shared communications protocols between network administration systems. We will discuss this concept in more depth in following sections.

2.3 Attacker AS

The AS containing the attacker is not likely to be interested in identifying the attacker. With that being said, it may be able to use telemetry to detect the start of an outgoing attack if interested.

We recognize that the paths that requests and responses take through the network have no reason to be symmetric, and in practice would likely be asymmetric. However, **for the purposes of easier analysis we make the unrealistic simplifying assumption that the ratio of request and responses one AS sees remains roughly even for legitimate traffic. We use this assumption throughout the remainder of this section.** We attempt to use this heuristic to identify the attacker, and we will see that even under this greatly simplified model, identifying the AS of an attacker is often infeasible.

Notably, the invariant used earlier to identify DDoS victims is insightful for identifying DDoS perpetrators (or at least which AS they are located in). While the AS containing the victim should see vastly more responses than requests, the AS containing the attacker should see vastly more requests than responses. Identifying the location of attackers requires only minor modifications to the query we used to identify victims:

Listing 3: Sonata DDoS Attacker Detector

```
victim_ip_at_attacker = (n_req
    .join(query=n_resp, new_qid=3)
    .map(keys=('ipv4.dstIP', 'ipv4.srcIP',),
         map_values=('count1', 'count2',),
         func=('diff',)
        )
    .filter(filter_vals=('diff3',), func=('geq', T))
    .map(keys=('ipv4.srcIP','diff3'))
    )

botnet_ips_at_attacker = (n_req
    .join(query=n_resp, new_qid=3)
    .map(keys=('ipv4.dstIP', 'ipv4.srcIP',),
         map_values=('count1', 'count2',),
         func=('diff',)
        )
    .filter(filter_vals=('diff3',), func=('geq', T))
    .map(keys=('ipv4.srcIP', 'diff3'))
    )
```

Note that the n_req and n_resp queries are omitted here because they were shown earlier. Moreover, observe that the only difference between these queries and the ones shown earlier is the reversal

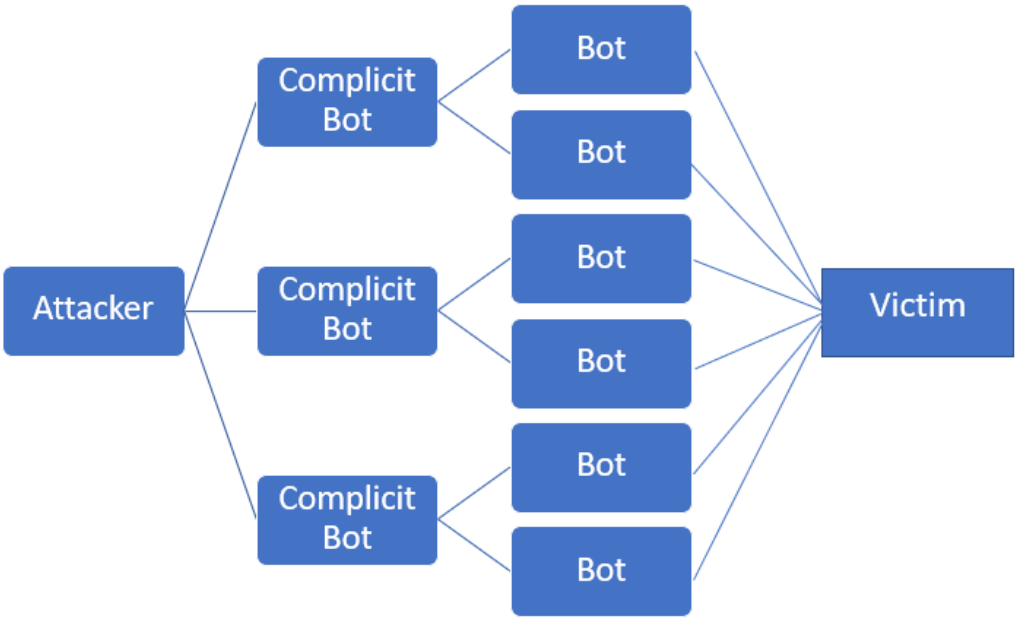

Figure 2: An Attack Using Multiple Layers of Amplification

of the order of subtraction and that we want the source IP rather than the destination IP. Note that this query also returns the victim IP and botnet IPs, but for an appropriate threshold T, will only be triggered at the attacker AS or on a bottleneck AS on the path between it and its botnet ASes. While there are still likely to be a disproportionate number of request packets relative to response packets for the victim IP at intermediate nodes between the attacker and its botnet ASes, the fanning out from the attacker AS means that, in general, the difference (but not necessarily the ratio) should be higher at the attacker AS itself (barring intermediary bottlenecks). This may allow mitigating actions to be taken before the attack does substantial damage to the victim.

It is important, however, to recognize that even if all ASes were cooperative and used our telemetry scheme, there are still easy ways for the attacker to remain undetectable (even under our strong and unrealistic assumption that all legitimate traffic is symmetric). First, the attacker may initiate contact with the botnet through an intermediary bot (i.e. the attacker would only be sending one packet to its bot, which would in turn be sending hundreds of packets or more to the broader botnet) . Consequently, our detection scheme would be identifying the AS the amplifying bot resides in, rather than the AS of the actual attacker themselves. Regardless, it would still be possible to use the information to mitigate the attack by filtering out suspicious packets from the botnet's AS. In this case, the attacker may notice that their attack has been stopped and may contact a bot in another AS to continue it.

In an even more pernicious case, the attacker launches amplification attacks from multiple intermediate bots (called complicit bots in our diagrams) in distinct ASes. In this sense, there are multiple layers of amplification, and the scheme described above would not only fail for the primary attacker but would likely also fail for the amplifying bots because each one could send fewer outbound signals to packets to the next layer of bots, making the traffic appear less anomalous. Consequently, we would only be able to identify the victim and the final layer of bots that are directly sending it attack traffic. Notably, our Sonata queries all assume that amplification attacks are typically performed by the attacker themselves directly making requests using spoofed IPs. However, the multi-layer attack shown in Figure 2 would be more sophisticated in so far as the complicit intermediary bots are performing the IP spoofing rather than the attacker. The attacker would simply be requesting that the complicit bots launch attacks on the victim IP. This requires that the complicit bots either be compromised machines the attacker can remotely command or that they at least be willing to operate

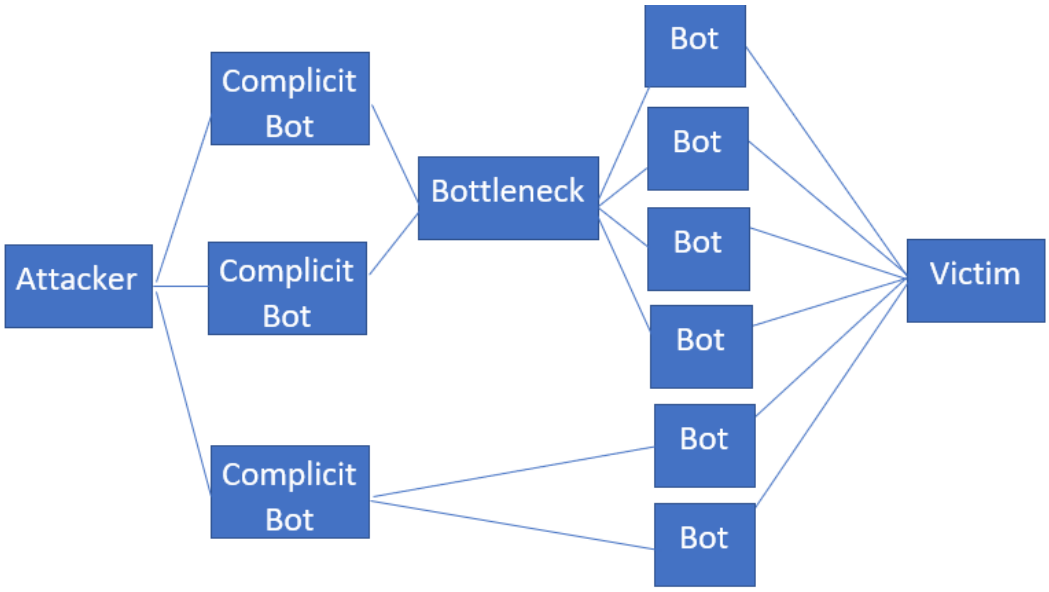

Figure 3: An Attack Using Multiple Layers of Amplification with an Intermediate Bottleneck

on the attacker's behalf. Assuming all machines in Figure 2 are on different ASes, our queries would fail to identify the attacker. Moreover, they may fail to identify the complicit bots as well because each one could issue a smaller number of spoofed request packets to avoid going above our detection threshold. By using multiple layers of amplification, the attacker could remain well concealed even in the event that all parties reside in cooperative ASes actively using our queries.

However, we could also consider each of the complicit bots a separate attacker since they should display a dampened version of the behavior we are looking for in our Sonata query. If we are able to filter out spoofed request packets at the ASes of the complicit bots, this would still significantly reduce the severity of the downstream attack. Because of the fact that the true attacker's AS is undetectable to our queries, we treat each of the complicit bots as a separate attacker, and thus only concern ourselves with one layer of amplification in later sections.

There is yet another complication in identifying attackers (once again, even under our immensely simplified model where legitimate traffic is symmetric and illegitimate traffic is asymmetric). Consider Figure 3. The highest difference between number of requests and responses to the victim IP should occur at the AS we have labelled Bottleneck. Our query would likely identify the bottleneck as the primary attacker. The extreme difficulty in identifying an attacker even under a highly simplified model suggests that the identification and mitigation schemes developed in later sections should focus on the victim, their immediate neighbors, and the last layer of the botnet, rather than trying to address the problem closer to the source.

3 Collaborative Detection

As noted earlier, each of the distinct parties in the attack sees different information at different times. Moreover, there is a risk of false positives at each of the points in the attack topology. However, the risk of a false positive may be decreased if multiple parties signal anomalous traffic. In particular, we envision a system in which disparate AS administration systems can communicate with each other over a secure protocol to share information useful in identifying and mitigating attacks. For the purpose of simplification, we envision all ASes as communicating with a abstracted detection and mitigation system we refer to as the "DDoS Detective".

Whenever anomalous traffic is detected through the methods presented in the previous section, the information is shared with the detective, which must in turn determine whether an attack is occurring

and help all parties best respond. Notice that every party in the attack topology knows its own AS number, the victim's IP, and at least some subset of botnet IPs (the victim should know all in theory but the ASes containing the botnet devices only know a subset). Under our scheme, each party at the minimum sends the detective a standardized "threat alert" tuple containing information (ROLE, AS_ID, VICTIM_IP, BOTNET_IPS). We envision the detective initially seeing the following during an idealized attack in which all ASes are cooperative:

1. Multiple Botnet ASes send (BOTNET, AS_ID, VICTIM_IP, BOTNET_IPS)

2. Victim AS sends (VICTIM, AS_ID, VICTIM_IP, BOTNET_IPS)

Note that it is entirely possible and even likely that a large subset of the botnet ASes will not be participating in our collaborative telemetry scheme and thus will not send information. Moreover, although in theory the above is the order in which the attack should flow through the network topology, there are no guarantees that the Detective will actually receive the tuples in the appropriate order. Due to network delay, differences in routes, etc. the packets signalling the impending attack from the botnet may arrive at the detective after the victim has already signalled that the attack is occurring. Moreover, if the attack is very well distributed, it may be the case that the attack traffic remains below our detection thresholds until it actually reaches the victim's AS.

It is much more likely that the first signal to the detective would come from the victim. The detective could in turn request that ASes containing botnet IPs identified by the victim lower their anomaly detection thresholds or begin filtering out packets from botnet IPs.

Depending on the number of botnet devices, it may even be feasible to ask cooperating ASes to drop packets which have the victim IP as their source IP and one of the botnet IPs identified by the victim as their destination IP. Note that depending on the number of botnet IPs detected by the victim, it may not be feasible to quickly send this data. Furthermore, a sophisticated adversary could change the botnet devices they employ throughout the course of the attack to render defense strategies like this useless. However, in the event the attacker is not sophisticated enough to switch up its botnet and that the data overhead is manageable, preemptively dropping the spoofed request packets would save the network the hassle of having to deal with the much larger reply packets later on.

4 Towards Implementing a "DDoS Detective"

4.1 Robustness Against Attacks

The detective is merely an abstraction, and it is arbitrary who plays the role of the Detective. Logically, the victim has the most incentive to devote computational and storage resources towards the detective work. However, if its buffers are actively being overwhelmed by a DDoS attack, it may not make the most effective detective. Moreover, the need to store substantial state presents difficulties for preforming the detective work directly on routers and switches. It may make more sense to outsource the detective work to separate systems within an AS. Furthermore, the system should be relatively robust against attacks, otherwise an attacker could just DDoS the detective as well. There are two obvious solutions to this:

1. Hide the identity and location of the detective

2. Use a modified version of a distributed consensus protocol such as Paxos [10] or RAFT [12] to make the detective robust against attacks and network partitions

The first solution would be simple but easily compromised. The second solution would be harder and more expensive to implement but much more difficult to compromise. In essence, there would be a detective server at some IP on each of the cooperating ASes and they would run Paxos to attain consistency across the ASes' detectives. When an AS is attacked, its detective server can be treated as if it is "down" due to a network partition, however the broader "DDoS Detective" system across the cooperating ASes will continue running without issue.

4.2 Secure Communication Scheme

A critical component of inter-AS coordination, whether involving our "DDoS Detective" system or not, is a secure and authenticated line of communication between the collaborating parties. This is not built-in feature of the TCP and UDP protocols, for good reason. Such a line of communication involves too much overhead to be useful for everyday internet uses. However, there are a number of higher level protocols built on top of traditional transport protocols to send authenticated messages over insecure lines. The proposed system requires that AS administrators use a standard secure communication protocol in sharing information for attack identification and mitigation. Notably, messages sent over this protocol may sacrifice performance in favor of traditional security concerns like authenticity and integrity.

Notably, this sort of secure communication can be implemented with message authentication codes [1], much in the same way as S-BGP [9] or DNSSEC [4]. We propose that the cooperating ASes secure their messages using public key cryptography (for implementation options see [14]). Then, authenticated messages can be distributed over traditional transport protocols (like UDP/TCP) without the concern that messages have been spoofed (or modified by adversaries). Network administrators (or their detective servers) can periodically or automatically check for authenticated messages from the other ASes, and adjust their network configurations and telemetry queries accordingly. The asymmetric key infrastructure is computationally expensive, especially when compared to typical non-authenticated UDP/TCP protocols, but disallows various forms of attacks, including the sort of spoofing involved in the DDoS amplification attacks we are trying to mitigate [14].

Notice that our proposed security measures differ from Transport Layer Security (TLS) in that the content of the messages themselves are authenticated, rather than just the server being authenticated [13]. This means that users can be assured that messages are not tampered with in transit, for example.

There are multiple use cases for such a secure line of communication. For example, if a victim recognizes through their Sonata queries that they are experiencing an attack, and identifies a property they can use to distinguish legitimate packets from malicious packets (such as packet size, for example), they can securely tell upstream collaborating intermediate ASes to preemptively drop the malicious packets. Notice that the authentication is critical here, because if these messages could be spoofed, then an attacker could implement an easy denial-of-service attack by sending a message to upstream ASes to drop all packets to a victim. Similarly, authenticating the message content, rather than just the server is critical, because modification of the contents of the message would enable additional denial-of-service attacks.

Additionally, this line of communication can be used to adaptively set thresholds in telemetry queries that different network operators are running. As an example, an AS containing botnet participants may detect anomalous outgoing traffic all to a single destination IP, it could then then alert the victim AS that it may be under attack and that it should use lower detection thresholds than usual. This way, it may be possible for the victim AS to identify the attack before the victim server is compromised.

Notice that the overhead of secure cooperation is substantial. Even distributing and maintaining a set of up to date public keys is expensive. However, it would enable collaborative attack detection and mitigation across ASes. Without such security measures, collaboration would simply create more vulnerabilities for attacks.

5 Discussion

5.1 Lessons for Attackers

Notably, even though this project sought to design novel DDoS defense strategies that make use of network telemetry systems and cooperation between ASes, in the process it revealed a lot about what makes an attack effective. While our goal is not to give attackers a better understanding of how to be adversarial, it is important to understand what further ideas can be employed.

First, DDoS attackers can increase the difficulty of detecting them by using multiple layers of amplification. Under the unrealistic simplification that legitimate traffic is symmetric whereas illegitimate traffic is not, it is often possible to identify a DDoS attacker's AS, and harder with multiple layers

```
    runtime = Runtime(config, queries)
TypeError: __init__() takes exactly 4 arguments (3 given)
```

Figure 4: Number of Arguments Compilation Error

```
# Incorrect
runtime = Runtime(config, queries)
# Correct
runtime = Runtime(
            config,
            queries,
            os.path.dirname(os.path.realpath(__file__))
          )
```

Figure 5: Argument error code change

and methodologies of amplification. While more realistic conditions are already in favor of allowing attackers obscurity, they may nevertheless better conceal themselves by using multiple levels of attack amplification.

Next, switching which bots are used to attack and which ASes they are located in will make it much more difficult to map the attack and mount an effective defense. If an attacker frequently switches their bots, our system will be essentially helpless against their attack. Notably, defense systems which make heavy use of statistics and machine learning may be more robust against these sorts of attacks (for examples of such systems see [5] [3] [8]). Our scheme relies on the formalization of and analysis of high level attack topologies, whereas other schemes can use subtler and often semantically vaguer properties of packets to identify attack traffic more specifically.

5.2 Challenges

One of our largest obstacles in completing this project was the steep learning curve for working with Sonata. Although the tool is conceptually useful and easy to understand at a high level, we encountered significant engineering difficulties in implementing our project on top of it.

Sonata includes a cleanup script to stop processes, clear log files, and free resources. Executing this cleanup script between runs is critical. Without it, the VM provided with the Sonata repository crashes on successive runs of Sonata queries. We ran Sonata once successfully, and tried to run the cleanup script. However, we encountered the following syntax error with the script `Syntax error: word unexpected (expecting "do")`

We reached out to some of the original code base authors, Jennifer Rexford, Arpit Gupta, and Rob Harrison. Ultimately, we found that there were UNIX/DOS compatibility issues in the cleanup script itself. After fixing the script, we were able to consistently run Sonata.

After successfully running Sonata, we worked through each of the thirteen example programs. Five of the thirteen worked with no further issues. Five threw an error based on the number of arguments passed to one of the functions in the example code, as shown in Figure 4.

We were able to fix the error by cross-referencing that function call with the corresponding function call in the working examples. The necessary code change is shown in Listing 5.2.

After fixing that error, two of the five programs successfully ran to completion, the other three threw a different compilation error with their P4 code, that we were not able to debug, shown in Figure 6. We were also unable to debug the remaining three programs: two of them threw key errors on the query (Figure 7), and one of the emitted debug messages that only said “WEIRD”, before crashing with a syntax error.

Table 8 summarizes what happens when each example program is run.

```
2020-12-06 19:45:26,727 - P4Target - INFO - run
2020-12-06 19:45:26,727 - P4Target - INFO - init P4 application object
2020-12-06 19:45:26,727 - P4Application - INFO - init
2020-12-06 19:45:26,728 - P4Application - DEBUG - create query pipeline for qid: 10032
2020-12-06 19:45:26,731 - P4Target - INFO - generate p4 code and commands
2020-12-06 19:45:26,737 - P4Target - INFO - compile p4 code to json
2020-12-06 19:45:26,737 - P4DataPlane - INFO - compile p4 to json
**************************************************

ERROR! YOUR P4 CODE COULD NOT COMPILE SUCCESSFULLY.

**************************************************
```

Figure 6: P4 Compilation error in Sonata's Malicious Domain Detector

```
vagrant@vighata:~/dev$ sudo $SPARK_HOME/bin/spark-submit sonata/examples/dns_tunneling/test_ap
p.py
Traceback (most recent call last):
  File "/home/vagrant/dev/sonata/examples/dns_tunneling/test_app.py", line 39, in <module>
    q3 = (q1.join(q2)
  File "/home/vagrant/dev/sonata/query_engine/sonata_queries.py", line 144, in join
    left_query = map_dict['query']
KeyError: 'query'
```

Figure 7: Key error in Sonata's DNS Tunneling Detector

Example	**Works?**
completed_flow	No (key error, query)
dns_ttl	No (args issue, then p4 code does not compile)
dns_tunnelling	No (key error, query)
heavy_hitter	Yes
malicious_domain	No (args issue, then p4 code does not compile)
newly_opened_connections	Yes
port_scan	Yes
reflection_dns	No (args issue, then p4 code does not compile)
slowloris_attack	Yes
ssh_brute	Yes (after fixing args issue)
superspreader	Yes
syn_flood	No ("WEIRD" debug messages, then syntax error)
udp_traffic_asymetry	Yes (after fixing args issue)

Figure 8: All thirteen examples provided and their status when ran.

We also encountered substantial issues in trying to compose our own Sonata queries. As an example, consider the following three queries. Respectively, they represent filtering by packet lengths that are equal to 50, greater than 45, and less than 55.

Listing 4: Three queries to identify packets of length 50

```
equal = (PacketStream(1)
    .filter(filter_keys=('ipv4.totalLen',), func=('eq', 50))
    .map(keys=('ipv4.srcIP', 'ipv4.totalLen'))
    )

gequal = (PacketStream(1)
    .filter(filter_keys=('ipv4.totalLen',), func=('geq', 45))
    .map(keys=('ipv4.srcIP', 'ipv4.totalLen'))
    )

lequal = (PacketStream(1)
    .filter(filter_keys=('ipv4.totalLen',), func=('leq', 55))
    .map(keys=('ipv4.srcIP', 'ipv4.totalLen'))
    )
```

For attack packets of exactly length 50, they should all be logically equivalent and should display the source IPs and lengths for those packets. However, when ran, only the "equal" query actually displays matches. This suggests that there may be deprecation issues somewhere in the code base. The lack of ability to use even basic filter operators made it extremely difficult to write working code. If we have a fundamental misunderstanding of Sonata's syntax, documentation to clarify the nuances of the code would be helpful. From what we can tell, the victim query (the Sonata authors' UDP asymmetry example) runs and works well on simulated attack traffic. Additionally, the botnet AS query we developed earlier in the paper runs and detects attack traffic effectively (excluding the packet size filter).

Our difficulties with Sonata made it very difficult to test our ideas empirically and verify our lines of reasoning, nevertheless working with the codebase also challenged us to reason about and attempt to debug an unfamiliar system by reading the source code, all while learning network telemetry concepts.

6 Conclusion

This paper reasoned about what a DDoS attack look like to various parties along a network topology, designed telemetry queries to extract useful information about attacks, and outlined a high level proposal for a distributed defense system called "DDoS Detective." Just as importantly importantly, the design schemes made in the process reveled the value of sharing information between ASes involved in a DDoS attack and presented novel mitigation strategies which involve cooperation and information sharing between cooperating ASes. The ideas outlined may have implications for later defense system designs.

References

[1] Marina Blanton. *Message Authentication Codes*, pages 1715–1716. Springer US, Boston, MA, 2009.

[2] Mitko Bogdanoski, Tomislav Suminoski, and Aleksandar Risteski. Analysis of the syn flood dos attack. *International Journal of Computer Network and Information Security (IJCNIS)*, 5(8):1–11, 2013.

[3] Rohan Doshi, Noah Apthorpe, and Nick Feamster. Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 29–35. IEEE, 2018.

[4] Donald Eastlake and C Kaufman. Domain name system security extensions. Technical report, rfc 2535, March, 1999.

[5] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. Statistical approaches to ddos attack detection and response. In *Proceedings DARPA information survivability conference and exposition*, volume 1, pages 303–314. IEEE, 2003.

[6] Arpit Gupta, Rob Harrison, Marco Canini, Nick Feamster, Jennifer Rexford, and Walter Willinger. Sonata: Query-driven streaming network telemetry. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 357–371, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Arpit Gupta, Rob Harrison, Marco Canini, Nick Feamster, Jennifer Rexford, and Walter Willinger. Sonata: Query-driven streaming network telemetry. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 357–371, 2018.

[8] Shuyuan Jin and Daniel S Yeung. A covariance analysis model for ddos attack detection. In *2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577)*, volume 4, pages 1882–1886. IEEE, 2004.

[9] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (s-bgp). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, 2000.

[10] Leslie Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[11] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[12] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, pages 305–319, 2014.

[13] Rolf Oppliger. *SSL and TLS: Theory and Practice*. Artech House, Inc., USA, 2009.

[14] Arto Salomaa. *Public-Key Cryptography*. Springer-Verlag, Berlin, Heidelberg, 1991.

[15] Kulvinder Singh and Ajit Singh. Memcached ddos exploits: Operations, vulnerabilities, preventions and mitigations. In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pages 171–179. IEEE, 2018.