Exploring Modern DoS Mitigation using Client Puzzles

Andrew Walkowski and Theodore Yin

Advised by Professor Noureddine

1 Introduction

The Internet's design gives it the ability to send anything to anyone at any time. This very design creates one of the Internet's biggest threats in Denial of Service (DoS) attacks. The Volumetric Distributed Denial of Service Attacks (DDoS) is one of the most common problems in network security. Volumetric DDoS Attacks occur when a server is flooded with so much fake traffic that the server can not serve requests from legitimate clients. Attackers can flood a server with noise, consuming all the available resources. A simple example of a DDoS attack is asking a printer to print hundreds of copies of a large textbook. The printer will be so busy printing the requested textbooks that it will not get to legitimate print requests until much later, if ever. Figure 1 illustrates how attack traffic overwhelms a system. Not only filling up all current resources, but also creating somewhat of a waiting line of traffic. When a legitimate client eventually makes a request, their packets will typically time out or if lucky just take forever to receive a response.

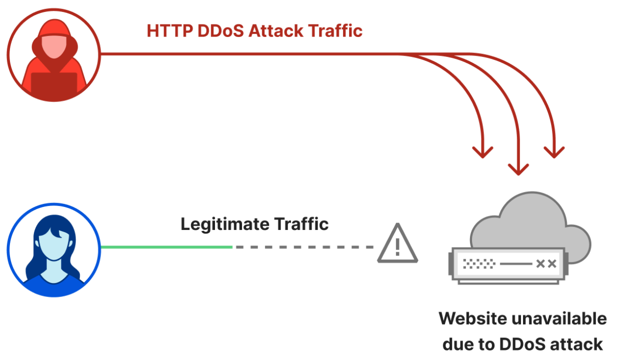

Figure 1: Illustration of a Distributed Denial of Service Attack [1]

The characteristics of these attacks make them difficult to defend against because the attackers disguise their malicious requests as normal internet traffic. This means that traditional security measures can struggle to identify and block these attacks. Attackers often take control of small electronic devices, such as smart TVs, smart speakers, or other Internet of Things (IoT) devices, to create a large volume of requests sent to the victim. This can overwhelm the targeted system, leading to service disruptions and even system failures. Since IoT devices are typically cheaper than functional computers, malicious users can deploy an attack with serious consequences at low cost. Any person with moderate coding skills would be able to perform similar attacks. Another issue is IP spoofing. With more and more incidences of attacks containing the above characteristics, advanced defense techniques can prevent too many connections originating from the same source (IP address). Spoofing an IP address is a technique used to alter the source address of a packet. This is done to hide the identity of the sender or to impersonate other people on the network. This is an easy process and can be done with little technical knowledge. There are countless different methods available today to help alter an IP packet's source addresses. IP address spoofing is a very common technique used by attackers in a variety of attacks. An attacker can change the source address in the ip packet so that when the victim gets the packet the victim can only tell that the packet is from a legitimate unique source address. By constantly changing the source address an attacker can anonymously send packets to a victim and the victim can not identify the source [2]. Spoofing adds and makes defending DDoS attacks much harder. An attacker can use a botnet of devices that spoof their address and now you have virtually unlimited unique source addresses that firewalls on the victims side can't properly blacklist or filter. With spoofing technology, an attacker can run an attack from thousands of unique IP addresses leaving traditional defense mechanisms useless. The lack of uniqueness of such attacks also makes it hard for a system to distinguish between legitimate and malicious connections. [3]. This is why it is so difficult to put in place effective measures to reduce the negative impact of the internet, while still keeping it open and fair.

A variety of strategies to prevent DDoS attacks range from sophisticated machine learning and scheduling algorithms to simply provisioning more resources for attackers [4] [5] [?]. Due to this wide variety, there is no clear solution that can be made to prevent these attacks. An attack with endless resources and a large enough botnet could hold some of the most secure systems today hostage for hours or days. The issue of DDoS is not just a problem for web servers. Almost all Virtual Private Network (VPN) protocols utilize some type of DDoS defense system to prevent floods on endpoint access requests. However, research suggests that endpoints with advanced security in VPN systems could be brought down by an attack of just a few gigabytes [6]. Performing an attack of that size is quite easy with on-demand attacks being available for online purchase. The growing effects and decreasing cost of these attacks have only added to the popularity. Figure 1 shows how common these attacks are, with some months having more than 20

Since DDoS attacks are becoming cheaper to run, the effectiveness of these attacks are also improving. It is estimated that one of these attacks can cost a small business 120,000 USD if they were to fall victim to a DoS or DDoS attack; furthermore, for a larger enterprise it can cost them more than 2 million USD [7]. This problem will only get worse with the increasing popularity of remote Virtual Private Networks (VPN) protocols, the expansion of Internet of Things (IoT) devices, and the overall movement of everything to the internet.

The most common industry strategies to mitigate the effects of DoS attacks are called absorption strategies. Absorption strategies aim to maintain normal traffic service to legitimate clients during an attack. These strategies build upon the idea that there is no perfect way to stop a DDoS attack, so they don't try to stop the attack, just prepare to absorb it. Absorption strategies can be thought of as a kitchen sink. The faucet is the traffic and the system under attack is the drain. If the drain is too small the sink will flood and new traffic will never get to the system. The absorption strategy would suggest to just expand the size of the drain. In this example the implementation sounds really simple, but using more resources to handle more traffic is not cheap and requires a scalable infrastructure. This strategy really does nothing to prevent DDoS attacks, and in some cases this may motivate attackers to attack a victim simply in hopes of running up their bill. However, absorption is one of the most common strategies for DDoS effect mitigation because of its ability to handle all the legitimate traffic from clients.

A common reactive strategy, filtering, does a lot more to actually block a DDoS attack from straining the victim system. Filtering techniques sift through all traffic and determine what is a legitimate user and what is not. The complexity of these filter systems may vary from network to network. Some Networks utilize machine learning to analyze traffic and isolate key features of an attack and block similar traffic [8]. Other implementations like StopIt send a request to the initial sender asking them to temporarily stop and if they refuse the block that IP address [5]. Figure 3 explains how a Domain Name Server would filter traffic during a DDoS attack. This

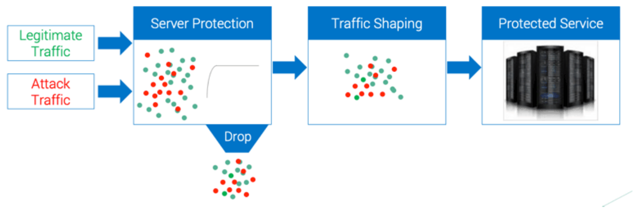

Figure 2: DNS filter system diagram filtering attack and legitimate traffic [9]

diagram illustrates very well how both legitimate and attack traffic can get dropped by the server protection strategy. While this will indeed mitigate the risk of the victim service being flooded with just attack traffic, it also drops some traffic from legitimate clients. That is another form of a DDoS where now legitimate traffic is being denied access to the service because attack traffic looks similar to some normal traffic. This also illustrates a vulnerability of filters and other bouncer strategies: resource exhaustion on the filter itself. If the filtering system receives more traffic that it can process, it will have to delay packets or start dropping packets at random, thus opening up the field for a new type of DDoS attack.

The third solution would be capabilities, a type of strategy that aims to prevent traffic from having any strain on the system. The basics of capabilities act almost like an admission ticket to the network. Since the incoming traffic may be malicious, why not verify them with appropriate authentication? A simple form of capabilities are the word or picture puzzles

Figure 3: A snippet of a puzzle that must be solved before a user can log into their account [9]

like Figure 3 seen commonly on login pages. These types of capabilities require the user to manually solve a puzzle. Some researchers have done the math on average time it takes to solve these interactive puzzles and estimate that 17 years of human life is wasted every day solving these puzzles [10]. The same research found almost all participants in a study said they would prefer not to use interactive puzzles like Figure 3 . These Capabilities serve as a stamp for each incoming packet so that the system would validate each packet with their unique stamp. In this strategy both the sender and the routing system communicate back and forth to validate the sender. The initial request from the sender enters the routing system and is sent back along with a stamp. The routing system temporarily saves this stamp and for a time being any packet that comes into the routing system with that stamp will get passed on to the application. This style of DDoS defense is rarely implemented since it forces a change on the clients end and the network infrastructure. Unlike filtering where the routing system can handle the filtering for the system, capabilities require senders to handle a stamp response and know to stamp their future packets with it. Finally having to validate each packet puts more burden on the routers and even makes them the victim for DoS attacks similar to filtering.

2 Related Work

Papers like *Fabio Streun's evaluation of VPN susceptibilities* [6] emphasized not only the effectiveness of DDoS attacks, but also how these attacks will only get worse and more impactful if nothing is done [6]. Therefore we began to investigate different solutions for volumetric Distributed Denial-of-Service attacks in the field.

In the paper [5], the researchers focused on using both filtering techniques and capability authentications. Researchers developed their filtering models and did a comparison on the performance of both filtering and authorizing. The results show the average effectiveness for either one is relatively similar for the majority of DDoS attacks. As there is no obvious advantage, we decided to combine both capabilities and filtering.

However, a common challenge presented by filtering and client puzzles, which will be discussed in the next section, is that they create a new attack vector on the filtering or authorization system. This is where the puzzles get introduced. The portcullis architecture [11] describes how computational effort per bandwidth requests can stop attacks from spamming authorization requests. A user requesting a small amount of system bandwidth will first get sent a puzzle to solve. Any requests received by the portcullis architecture are hashed with a newly generated seed and sent back to the sender with a segment of the seed to solve. This process can happen so quickly that illegitimate traffic with no intention of solving the puzzle basically gets dropped.

Other works utilize client puzzles as their main focus. Noureddine et al. implemented their TCP client puzzle modeling Stackelberg game [12]. The TCP implementation utilizes python scripts in a test bed to send packets, generate puzzles, solve puzzles and collect data for the simulation. Some researchers designed a system that accepts incoming requests if they match outgoing requests it has already made and refuses them otherwise [13]." These two papers set the foundation of our work.

3 Puzzle Overview

Among the capability solutions, there are client puzzles. Client puzzles are a measure that asks each client to solve unique computational puzzles for their requests. When the endpoints receive a request from a client, it will prompt the client with a puzzle before forwarding the request to the server's resources. The client then sends back the solution and once the solution is verified the endpoint will send the request to the server. The goal of such a technique is to stall malicious activities that require a large portion of the server's resources by exhausting malicious hosts.

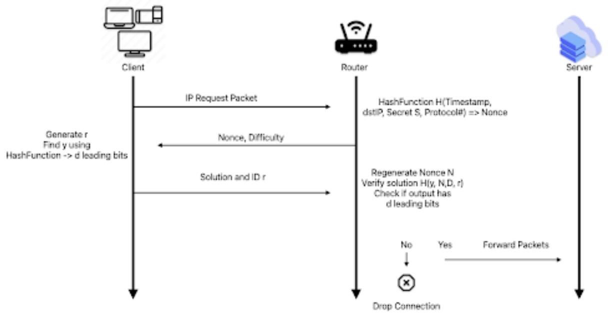

Figure 4: Client Puzzle Implementation

In Figure 4, the clients first send in IP Request Packet to the routing system, then the router creates a Nonce N that includes a hash function H using the Timestamp, destination IP address, a Secret S and protocol number from the packet. The routing system then responds to the request with the Nonce N and Difficulty d. After receiving the packet, the clients generate a random number r as unique identification and find the answer y by using H to find d leading zero bits. Clients then send back the solution and ID r back to the routing system. The routing system regenerates Nonce N and verifies that the solution using H with y generates d leading bits. Solving a difficult puzzle on the clients' side would take an exponential amount of hashes, while the server, on the other hand, could verify the answer with one single hash. Such a difference in terms of the number of hashes indicates the clients has been slowed down, and their computational power has been consumed.

Puzzles like these require computation directly from the clients' devices. It may seem to be easy, but the difficulty of the puzzle increases exponentially with an increased number of bits required. This leaves anyone who wants to DDoS the server spending most of the devices' computational power solving puzzles, and thus DDoS attack is mitigated.

It has been demonstrated that puzzle implementation can be used to successfully mitigate DDoS attacks [12]. Puzzles were used, and prevented an attack launched with IoTs, which usually have a low computation ability, therefore they struggled to solve the puzzles. However, it raises the issue of selecting the right difficulty for every client, aka fairness. Since clients use devices ranging from cell phones to high-end workstations, devices with weaker computational power will struggle to solve their puzzles for each packet request. This fails to meet the fairness principle that the internet was built on [12]. Every device, no matter its computational power should be able to access a website. The tradeoff becomes if client puzzles in an implementation are too weak, then attackers will face very little resistance. However, if the puzzles are too difficult to solve, then regular clients on IoT or mobile devices will lose access to the site. The proposed idea is one where the puzzles' difficulty are dynamically adjusted similarly to per-bandwidth puzzles proposed in [11]. The difference is instead of a certain amount of bandwidth being allocated per solution to a puzzle, the puzzle given to a client is reflexive to the payload size of the packet request. The assumption is that devices with weaker computational power are less likely to make large packet requests repeatedly. Therefore, devices that make smaller requests will solve easier puzzles and vise-versa.

4 Simulation Methodology

In this section we explain how we will design and run simulations to compare the different implementation strategies. We are expecting to collect data from our 5-stage simulations which will then help us depict the tradeoffs of current DDoS mitigation strategies and our experimental strategies.

Stage 1: Normal Traffic

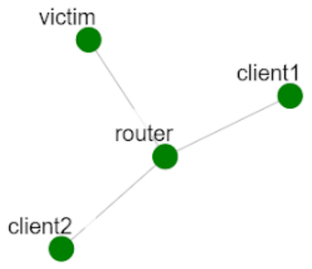

Figure 5: Normal Traffic simulation Diagram

To establish a baseline for normal traffic from legitimate connections, we will conduct a simulation involving two clients sending packets with different sizes and at various rates to the server. The routers will forward the packets to the server in a manner consistent with a standard cloud routing system. This stage will demonstrate how the system should treat legitimate clients. This will create a baseline for future simulation comparisons, which is critical for understanding how the system responds to anomalous traffic. Figure 5 illustrates how two different clients will communicate with the network, providing insight into the communication between different clients and a cloud network. During this simulation, we will primarily record the average round-trip time (RTT) for each packet sent, enabling us to gain insight into the system's performance under normal circumstances and establish a benchmark for future testing.

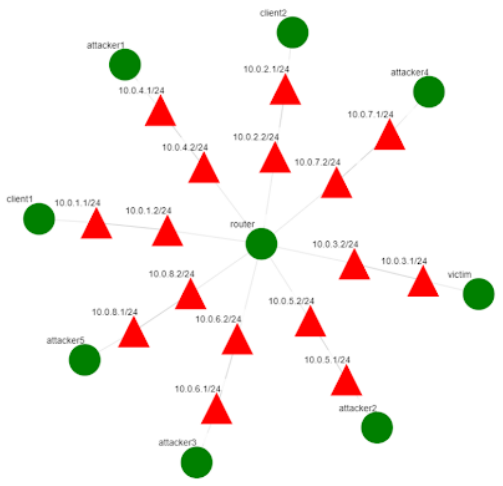

Figure 6: Topology of simulation generated by Merge Test Bed

textbfStage 2: Unprotected system under Attack On top of the stage 1 simulation, stage 2 simulation includes attackers performing DDoS attacks on the victim (server). Stage 2 will contain two normal clients, one victim (server) and five attackers as shown in Figure 6. By adding these attackers to the network the router and victim should be flooded with attack traffic. This attack will be ran using hping3's ping flood command on all five attackers. This will send roughly 50,000 packets a minute to a destination. Meanwhile the clients will run the same script run in the previous simulation. The time it takes to get response from the server and the number of dropped packets are the key measurement schemes at this stage. We are expecting to see a drastic increase in response time or complete denial of service.

Stage 3: Filter implementation under Attack In this stage, we are implementing a basic filtering technique to mitigate DDoS attacks. We are using the unprotected network topology under attack from stage 2, but with an added logic at to the router for filtering. Figure 7 describes the filter logic that will enable the router to detect and filter incoming IP

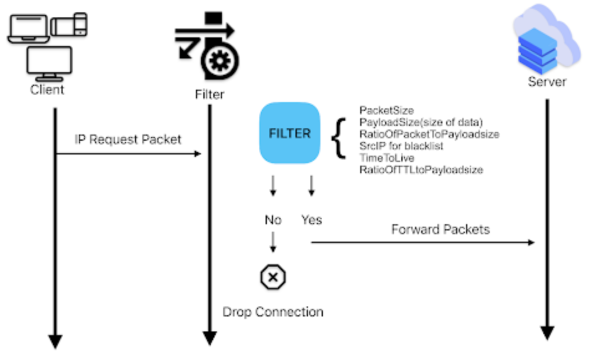

Figure 7: Diagram of Filter simulation Implementation

requests, determining whether or not to drop certain requests. To accomplish this, we utilized the IPtables tool to create a basic filtering defense. IPtables is a command-line utility in Linux used for configuring and managing firewall rules to control network traffic by allowing or blocking specific types of connections based on different criteria. Our general IPtables rules refer to DDoS protection with IPtables [14]. We also added additional rules that examine the source IP addresses and establish a blacklist of IPs that are known to be malicious. Additionally, we set a threshold for the maximum packet transfer within a specified time interval. We expect this simulation to simulate how many legitimate packets get dropped versus how many attack packets get dropped.

Stage 4: Puzzle implementation under Attack Stage 4 is the simulation of a puzzle strategy. The system will stay the same as before, but now the routing logic will be modified. Using DPDK a sniffer script will be ran that catches all incoming traffic. Figure 8 shows how the router will the generate a puzzle at kernel level and then send it back to the source. The clients and 1 attacker will solve the puzzle and then send their solutions back. The puzzles be implemented according to the description in the Puzzle Overview section. We want to examine the time it takes to get a response back from the time an initial request is sent. We also want to see how long each step takes. In this simulation there could be improved Round trip times by not recording the time at each step and potential optimizing the code, but for exploratory purpose we use Mohammad Noureddine's [12] python puzzle code and converted it to c scripts.

5 Results

In this section we discuss how the results we gathered show the strength and weakness of attack mitigation strategies. We also highlight how client puzzles using new technology can better mitigate Denial Of Service attacks than previously explored by other researchers.

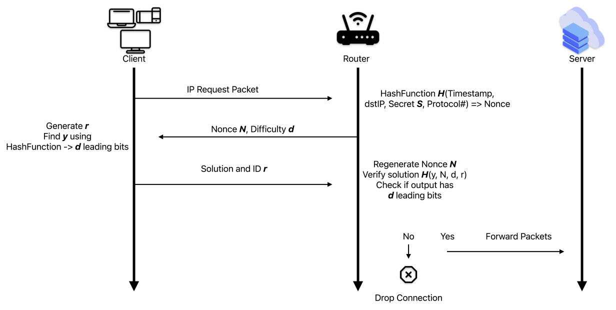

Figure 8: Diagram of Client puzzle simulation Implementation

Normal Traffic When conducting experiments to establish a baseline for research we got efficient rates of throughput for our two clients communicating. Client 1 got an average response time of 1.17ms over 50 requests, and Client 2 got 1.27 ms. This produces a 1.23ms response per request rate for clients. This is both extremely fast and efficient since no packets get dropped.

Traffic under attack with no defense Our attack simulation compared to our normal traffic highlights how impactful Distributed Denial of Service attacks can be. By flooding the router with 5 devices for only 50 seconds the client experience is drastically altered. Notably, Client 1 received back 31 of 50 attacks, while Client 2 received back 33 responses to the 50 requests. Both Clients also faced a drastic slow down in throughput during the attack. In figure 9 you can see the time per response jumps from the normal traffic averages of about 1.23 ms to hovering around 160 ms per 2 responses.

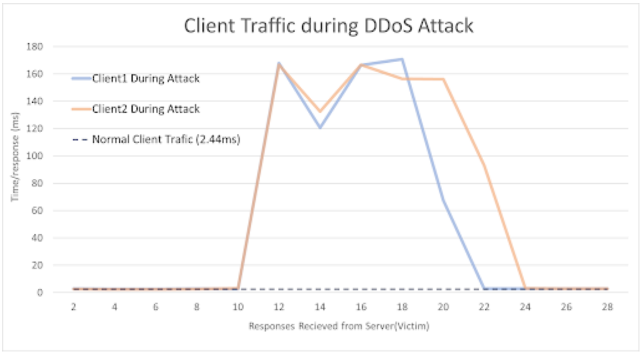

Figure 9: Graph of time it takes a client to receive two responses for every two requests

This simulation also explains how absorption could fail. The absorption strategy, if an attack like this started, would be to just increase the number of routers and scale up the ability of the victim so that additional request from clients get handled like normal traffic. If this is done, clients would essentially see no change from the normal traffic scenario.

The issue arises if an additional attacker is added to the equation. Now the network must expand again to handle the new attackers request. As an attack adds attack devices the network will slowly consume all of their additional resources. The network could run out of these resources and then the clients are back to where they started. Additionally, the victim is now having to pay to use all of their additional resources on wasted-malicious traffic. Absorption can work as long as you have access to enough additional resources, and the money to pay for these resources.

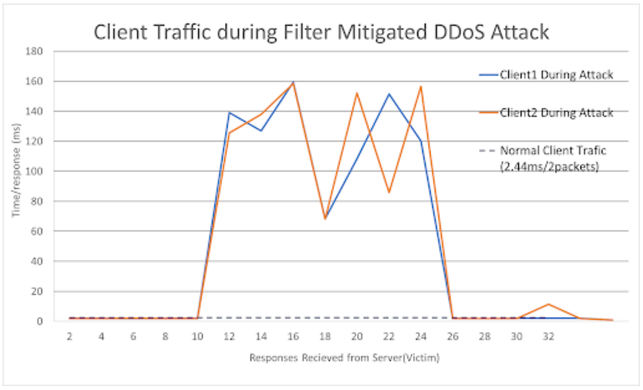

Figure 10: Graph of filter defense under attack

Filter Defense Result Our simulations of a filter mitigation implementation produce some fascinating results that highlight the strategies strengths and weaknesses when used in a network. The common misunderstanding is that filters can blacklist malicious actors. When we constructed this filter it performed efficiently. The filter would block the five attackers, so the clients could operate once again at normal traffic times averaging 1.29ms. These results show that filters can be effective when attacks are run by mindless attack devices in a botnet. However, when giving our attackers even the simplest form of logic the filter fails. By spoofing IP addresses every 2 seconds, the filter started blocking accounts that were not responsible for the attack. In this situation of smart attackers, the clients return to an experience similar to an attack with no defense. To combat this spoofing issue, we decided to make a smarter filter that didn't blacklist, but instead regulated throughput per IP. This minorly improves the statistics from the no defense attack. In Figure 10 you can see that clients experience roughly 140 ms response wait per packet. Which is better than 160 ms found in the previous simulation. The clients with filtering mitigation also got more responses back, with both getting 35 responses for the 50 requests sent. While this is not the greatest improvement, it is worth noting there is some effectiveness using a filter and there could be even more promise with a machine learning base filter that blocks and regulates more efficiently.

Puzzle Defense Result The puzzle data we collected was based on a slightly different test topology. In a simple metrics

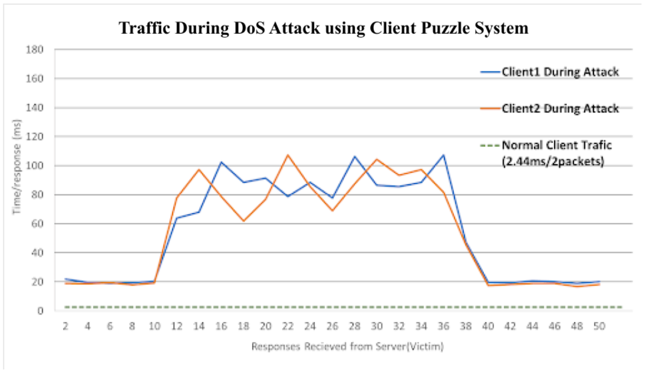

Figure 11: Graph of puzzle defense under attack

simulation, one client communicated with the router and the client experienced a slight slow down in response. With no attackers during a puzzle simulation client1 took 9.49ms to get a response back. The breakdown of times to roughly 0.8ms to send, 1.26ms for the router to generate puzzles, 5.7 ms for the client to solve a puzzle, and then 1.43ms for the router to confirm the solution. When adding a flood from just one attacker, both clients performed better. Both clients received responses for all 50 of their requests. As show in Figure 11, the clients during the attacks only saw their times to increase about 60-80 ms for receiving 2 responses. These numbers highlight the strength of puzzles by showing that the puzzle solving response rates for both the clients and the attackers are both slowed due to the puzzles.

6 Conclusion

Based on our results and research we believe that there is definitely a case to bring Client Puzzles back into the discussion for mitigating DoS attacks. When client puzzles were first introduced, the technology to efficiently implement them as a proper mitigation tool were not readily available. Today, with Software defined networks and cloud scalable networks the true power that puzzles possess can be used. In our simulations, puzzles outperformed all other strategic forms of mitigation. This matches up with previously designed client puzzle strategies, but the difference is our implementation could be easily installed by users. The code running on the router and client could be installed to run at both device's kernel level. Using DPDK, like done in our simulations, real world networks can seamlessly transfer over to a client puzzle mitigation system.

The statistics for the benefits of implementing client puzzles are clear. Client puzzle mitigation strategies will slightly slow down response rates for all traffic, but efficiently helps to flatten the curve of impact from DDoS attacks. Attackers that don't solve puzzles will make no impact, but attackers who do have their bonet solve puzzles will have to pay for

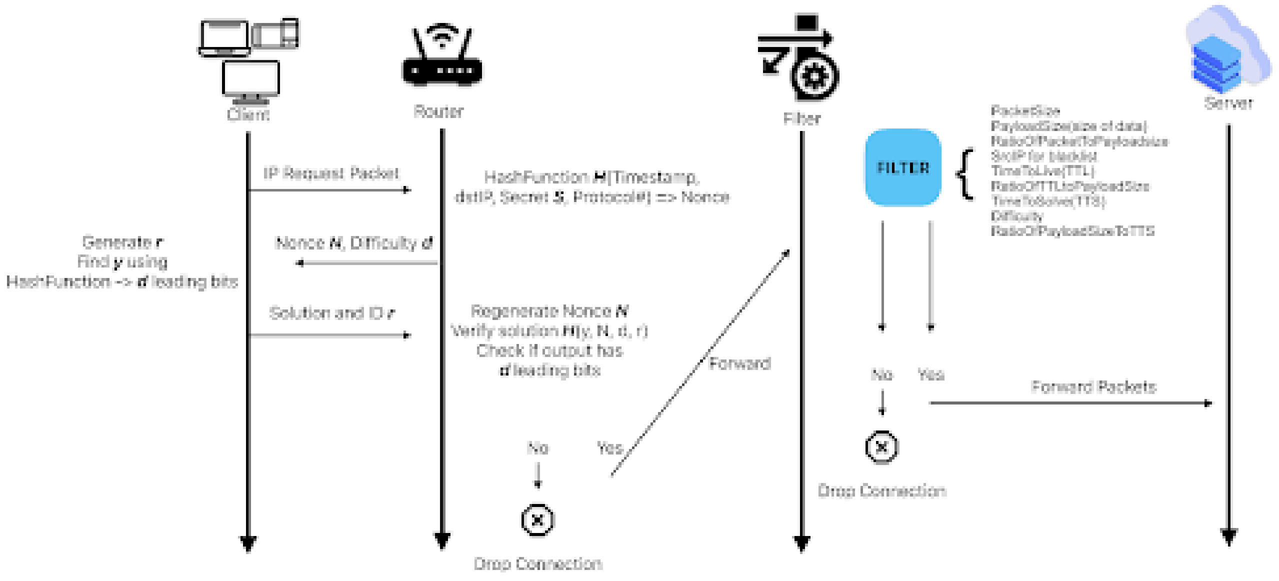

Figure 12: Diagram of a possible implementation client puzzles in future work

that computational load.

Client puzzles alone can help mitigate DDoS attacks, but when paired with other DDoS mitigation strategies a network can build a stronger and more robust defense strategy. One way to pair these strategies is to put a client puzzle system in front of a filter to prevent DDoS attacks on filters. In figure 11 we lay out how a puzzle filter system would work. The puzzles work normally to force clients to prove their legitimacy with a solution. After a solution gets verified the request along with the information about the solution gets sent to the filter router. This combination of defense strategies is beneficial because the filter is now getting additional information about the client. The solution information the client sends back can help a filter determine how computationally strong the source of the request is. Using that information filters can compare computational power to size of requests being made. This combination in theory would perform well against IoT device attacks such as the 2016 Dyn attack [15]. Figure 11 shows how the filter could use the additional information from the client to better filter traffic before it reaches the victim. Future work could be done to implement machine learning in a filter that uses client puzzle information. Additionally, absorption could and should be added to a network with this strategy in figure 11. This way even if a sophisticated expensive attack comes through you still have resources on stand by to absorb that traffic so the client experience stays the same.

Future work could be done to further improve the application of client puzzles. We note a few things that could be further research to improve client puzzles; scaling puzzle difficulty, machine learning driven filter behind puzzle system, deploying a multi router system for, and more accurate statistics using real web page requests. Our results demonstrated that the use of client puzzles reduces the efficiency of attacks by making it more difficult for them to launch DoS attacks. And, the introduction of client puzzles not only improves the security of the system, but also the experience for clients.

References

[1] O. Yoachimik, F. Alex, and D. Julien, "Cloudflare mitigates record-breaking 71 million request-per-second ddos attack," Feb 2023.

[2] R. Beverly and S. Bauer, "The spoofer project: Inferring the extent of source address filtering on the internet," in *Usenix Sruti*, vol. 5, pp. 53–59, 2005.

[3] M. Anagnostopoulos, G. Kambourakis, P. Kopanos, G. Louloudakis, and S. Gritzalis, "Dns amplification attack revisited," *Computers & Security*, vol. 39, pp. 475–485, 2013.

[4] A. Shameli-Sendi, M. Pourzandi, M. Fekih-Ahmed, and M. Cheriet, "Taxonomy of distributed denial of service mitigation approaches for cloud computing," *Journal of Network and Computer Applications*, vol. 58, pp. 165–179, 2015.

[5] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: Network-layer dos defense against multimillion-node botnets," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pp. 195–206, 2008.

[6] F. Streun, J. Wanner, and A. Perrig, "Evaluating susceptibility of vpn implementations to dos attacks using adversarial testing," *arXiv preprint arXiv:2110.00407*, 2021.

[7] "Bulletproof - annual cyber security report 2019."

[8] C. Tseung, K. Chow, and X. Zhang, "Anti-ddos technique using self-learning bloom filter," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 204–204, IEEE, 2017.

[9] "How to defend dns services from all types of ddos attacks," May 2022.

[10] K. Krol, S. Parkin, and M. A. Sasse, "Better the devil you know: A user study of two captchas and a possible replacement technology," in *NDSS Workshop on Usable Security (USEC)*, vol. 10, 2016.

[11] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: Protecting connection setup from denial-of-capability attacks," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 289–300, 2007.

[12] M. Noureddine, *Achieving network resiliency using sound theoretical and practical methods*. PhD thesis, University of Illinois at Urbana-Champaign, 2020.

[13] X. Yang, D. Wetherall, and T. Anderson, "Tva: A dos-limiting network architecture," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1267–1280, 2008.

[14] "Ddos protection with iptables: The ultimate guide."

[15] J. Scott Sr and W. Summit, "Rise of the machines: The dyn attack was just a practice run december 2016," *Institute for Critical Infrastructure Technology, Washington, DC, USA*, pp. 16–18, 2016.