AI_Adaptive_POW: An AI assisted Proof Of Work (POW) framework for DDoS defense

Trisha Chakraborty (Department of Computer Science and Engineering, Mississippi State University, tc2006@msstate.edu),
Shaswata Mitra (Department of Computer Science and Engineering, Mississippi State University, sm3843@msstate.edu),
Sudip Mittal (Department of Computer Science and Engineering, Mississippi State University, mittal@cse.msstate.edu),
Maxwell Young (Department of Computer Science and Engineering, Mississippi State University, myoung@cse.msstate.edu)

Abstract
To protect the client-server architecture from a Distributed Denial of Service (DDoS) attack we present AI_Adaptive_POW. AI_Adaptive_POW protects an organization by injecting latency during communication by generating client reputation adaptive puzzles, which need to be solved by a client before the server begins processing a request. The framework adaptively tunes the difficulty of a puzzle based on a reputation score calculated by an AI model. This slows down the volume of incoming adversarial traffic. Additionally, the framework compels the adversary to incur a cost per connection, hence making it expensive for an adversary to sustain a volumetric DDoS attack.
Keywords
Cybersecurity, Proof of work, Artificial Intelligence, Distributed Denial of Service

1. Introduction

In client-server architecture a *load balancer* is responsible for validating and distributing incoming client requests among various server instances. This prevents individual server instances from getting overwhelmed. A client begins the interaction by initiating a request. After a successful connection has been established, the load balancer places the request on a server queue. Given a server has bounded queue size, a flood of malicious traffic can exhaust this queue, making it unavailable for genuine client requests. For the scope of this paper, we revisited the *volumetric Distributed Denial of Service* (DDoS) attacks. Here an adversary pretends to be a genuine client, thereby consuming a considerable quantity of server resources, and leaves little to no resources for the genuine client. One possible defense strategy is to force all connecting clients to solve a *proof of work computational puzzle* as part of the initial client-server connection establishment phase.

A generic *Proof of work* (POW) framework consists of a *puzzle generator*, *puzzle solver*, and *puzzle verifier*. The puzzle generator issues the puzzle to the solver, which solves them and sends the solution to the verifier. These puzzles have different levels of difficulty, i.e., each puzzle requires a different amount of computational resources to solve. The task of solving puzzles introduces latency during the interaction, and this latency duration is directly proportional to puzzle difficulty. In this paper, we build a POW based DDoS defense framework using the Java programming language called AI_Adaptive_POW. The framework is assisted by an Artificial Intelligence (AI) to adaptively slow down the adversarial traffic by assigning appropriate POW puzzles and thus improve the availability of a server during an ongoing DDoS attack. Our AI_Adaptive_POW framework utilizes *reputation scores* to guide the decision of how hard of a puzzle should each client solve. A reputation score is a heuristic that guides a system in distinguishing between genuine and malicious clients. This heuristic is computed using AI algorithms that inspect the features of the incoming client requests.

2. Description

AI_Adaptive_POW is an open-source framework implemented using Java Springboot and Python Flask Framework. The framework consists of four customizable modules, i.e., depending on the security needs of the industry, various configurable defense postures can be incorporated [1]. Figure 1 illustrates the overall architecture of the framework. AI_Adaptive_POW operates at the application layer of the networking stack and interacts with the transport layer, specifically operating during the *TCP handshake phase*. When a client wants to initiate

Nr.	Code metadata description	Please fill in this column
C1	Current code version	*V1.1*
C2	Permanent link to code/repository used for this code version	*https : //github.com/trishac97/AI_Adaptive_POW*
C3	Permanent link to Reproducible Capsule	N/A
C4	Legal Code License	*Creative Commons Zero v1.0 Universal License*
C5	Code versioning system used	*Git*
C6	Software code languages, tools, and services used	*Java, Springboot 2.4.0, Python v3.7.6 & Flask 2.1.2*
C7	Compilation requirements, operating environments & dependencies	flask, flask_restful, csv, pickle, rgensim, numpy, requests, spring-boot-starter-web, lombok, spring-boot-starter-tomcat, springfox-swagger2, spring-boot-starter-validation, okhttp, gson
C8	If available Link to developer documentation/manual	*https://github.com/trishac97/AI_Adaptive_POW/blob/main/README.md*
C9	Support email for questions	tc2006@msstate.edu

Table 1: Code Metadata.

communication with the server, it sends a SYN packet to the load balancer. The framework fetches the IP address associated with the SYN packet which serves as an input to an AI model. The AI model then computes a reputation score. The score is converted into corresponding level of puzzle difficulty as translated by a policy module. AI_Adaptive_POW then generates puzzle parameters which is sent back to the client attached to the SYN-ACK packet. To prevent the adversary from solving easier puzzle, the load balancer can use a small memory to cache the client IP addresses and currently dispatched puzzle parameters. The client-side employs a puzzle solving module and computes the puzzle solution. The solution is sent back to the load balancer attached to the ACK packet. On receiving correct solution, AI_Adaptive_POW notifies the resource allocation unit of the load balancer, which forwards the client's request to the server queue.

An important component of our framework is the *POW puzzle* [1]. To solve a POW puzzle, the client repeatedly performs a cryptographic hash function evaluation on the input parameters with an aim of converging to a constrained solution. The constrained solution governs the *level of difficulty* of a puzzle. A d-difficulty puzzle requires d consecutive zeros as the prefix in the output from the hash function. Our framework uses *Java v14 MessageDigest SHA-256* library for hash function evaluation. SHA-256 is a natural choice as it is collision resistant and bandwidth-efficient, producing only 256 bits of solution.

Next, we describe each of the four customizable modules part of AI_Adaptive_POW.

2.1 AI Module

The AI module is responsible for computing the reputation score for a client. Our framework employs an Euclidean distance-based system called DAbR [3] to produce reputation scores. We trained the AI model using a dataset provided by Cisco Talos [5]. The dataset contains 70,636 known malicious IPs and attributes (autonomous system number (ASN), internet service provider (ISP), country where the IP is registered, usertype (residential or commercial user), country where the IP is located, subdivision in the country where the IP is located, city in the country where the IP is located).

The AI module consists of two phases. First, the module calculates a feature vector from the IP attributes and represents them in an euclidean space. Collection of such representations form a cluster in the euclidean space. The cluster origin and feature vectors can be found here [2]. In the second phase, when a client sends a SYN packet, the IP address x.x.x.x alongside its attributes are captured. The module then calculates the feature vector of the client IP using pre-calculated feature vector and then generates a reputation score based on feature similarities between the malicious IP cluster origin and the client IP. The output of the AI module is

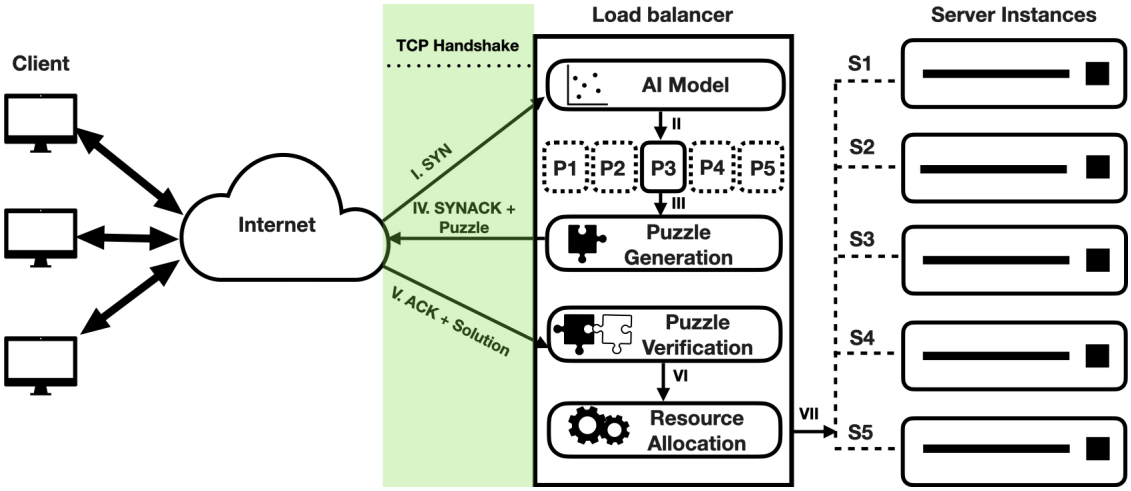

Fig. 1. Figure illustrates our AI_Adaptive_POW framework. I. Client initiates a TCP handshake with the load balancer and sends a SYN packet. II. The load balancer fetches the TCP IP address from the packet and computes a reputation score. III. Depending on the policy enforced (P1, P2, P3, P4, P5), a d-difficulty puzzle is assigned to r-reputation score. IV. The puzzle generation module generates a d-difficulty puzzle and sends it along with the ACK packet. V. The puzzle verification module verifies the puzzle solution returned along with the SYNACK packet. VI. On solving the puzzle correctly, the load balancer informs the resource allocation unit. VII. The request is then sent to a server instances (S1, S2, S3, S4, S5).

a reputation score r that can take a value in $[0, 10]$, where 0 is considered the worst possible reputation score.

2.2 Policy Module

The policy module is responsible for translating a computed reputation score to a puzzle of corresponding difficulty. The module takes reputation r as input and produces a value d that refers to the difficulty of a puzzle. For instance, an IP address with 1 reputation score receives a 0-difficulty puzzle, i.e., the acceptable hash function solution must contain atleast 1 zero as prefix. Figure 1, depicts a load balancer containing five possible policies P1, P2, P3, P4 and P5 where the policy P3 is currently in effect.

2.3 Puzzle Generation

The puzzle generation module is a lightweight module responsible for assembling the parameters required to solve a puzzle. The assembled parameter are attached along with the SYN-ACK packet. The module takes random seed s, timestamp t and puzzle difficulty d as input and the output is a concatenated bitstring $s||t||d$. Note that the load balancer may use a small cache to store corresponding IP address with their corresponding puzzle parameters. The random seed s is alphanumeric randomly generated string at most 32 bits in length and is periodically updated as an routine function within our framework. The random string prevents the adversary from launching a pre-computation attack on the server. Timestamp t prevents the adversary from using one solution and using it several times.

2.4 Puzzle Verification

The puzzle verification module is responsible for verifying the returned solution from the client. The input is a 256 bits solution bitstring sent by the client and the output is a boolean 0 or 1, where 1 indicates that the solution is correct and 0 otherwise.

3. Impact Overview

AI_Adaptive_POW is a prototype framework which operates at the application layer to strengthen organizational defensive posture against transport layer volumetric DDoS attacks. The main utility of this framework is to introduce delay on the adversary's side. This delay can slow down the volume of adversarial requests sent to the server at a given time and prevents the adversary from consuming majority of the server resources. The framework's impact can be summarized in two areas: (1) design impact, and (2) functional impact.

3.1 Design Impact

AI_Adaptive_POW consists of four main modules and each module is lightweight and customizable. Our baseline framework uses an AI module called DaBR [3], to produce reputation scores which can be replaced by even more sophisticated reputation score calculation techniques (for example, see [4]). The AI model is trained using a list of known malicious IPs provided by Cisco Talos[5] which can be replaced by any other third party IP list service or an amalgamation of more than one lists. When deployed, the framework can leverage any Cyber Threat Intelligence source available to an organization. Alternatively, the framework can use integer square root or cuckoo cycle variant as computational puzzle instead of using hash function evalutions. Different policies can be incorporated which maps the reputation score to difficulty level of the computational puzzle. These customizable design decisions make the framework ideal for use by cyber defense practitioners.

3.2 Functional Impact

Hash function evaluation ultimately translates into a monetary cost. Faster a machine can compute hash solution, higher is the power of the CPU. When using AI_Adaptive_POW, there is a cost associated with each client to establish a TCP connection. The cost per client increases if the client's reputation decreases. For an adversary controlling large number of machines and commandeering each one of them to DDoS an AI_Adaptive_POW defended server, each of these client machines needs to establish a TCP connection by solving puzzles as contingent to their reputation, and only then get on the server queue. This property of AI_Adaptive_POW imposes a monetary cost on the adversary for launching a DDoS attack. Additionally, due to our POW module, the computational puzzle introduces a delay in the system. Assuming the reputation score of an IP is accurate, the adversarial traffic can be reduced due to the puzzle solving phase. As a result the server can accommodate genuine requests in its queue, hence improving the availability of the server resources.

4. Conclusion and Future Work

In this work, we implemented AI_Adaptive_POW framework to defend against DDoS attacks by reducing the adversarial traffic. AI_Adaptive_POW accomplishes this by generating POW puzzles with the assistance of an AI model. Our framework is open-source and can be customized based on the security needs of an organization. For future work, we aim to implement more robust AI models that produce higher accuracy reputation scores. Additionally, we plan to explore efficient policy designs.

5. Acknowledgement

This work was supported in part by NSF grant CNS-1816076.

References

[1] Trisha Chakraborty et al. "A Policy Driven AI-Assisted PoW Framework". In: *52nd IEEE/IFIP International Conference on Dependable Systems and Networks (IEEE/IFIP DSN 2022)* (2022).

[2] DAbR Implementation. https://github.com/shaswata09/DabR. 2022.

[3] Arya Renjan et al. "Dabr: Dynamic attribute-based reputation scoring for malicious ip address detection". In: *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE. 2018, pp. 64–69.

[4] Henanksha Sainani et al. "IP Reputation Scoring with Geo-Contextual Feature Augmentation". In: 11.4 (Oct. 2020). URL: https://doi.org/10.1145/3419373.

[5] Cisco Talos. *Talos Threat Source*. https://www.talosintelligence.com/. 2022.