International Journal of Engineering & Technology, 7 (1.1) (2018) 230-233

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Reducing distributed denial of service (DDoS) attacks using client puzzle mechanism

C. Vasan Sai Krishna[1] *, Y. Bhuvana[1], P. Pavan Kumar[1], R. Murugan[2]

[1]Student, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation
[2]Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation
**Corresponding author E-mail: vasanchanumolu@gmail.com*

Abstract

In a typical DoS attack, the attacker tries to bring the server down. In this case, the attacker sends a lot of bogus queries to the server to consume its computing power and bandwidth. As the server's bandwidth and computing power are always greater than attacker's client machine, He seeks help from a group of connected computers. DDoS attack involves a lot of client machines which are hijacked by the attacker (together called as botnet). As the server handles all these requests sent by the attacker, all its resources get consumed and it cannot provide services. In this project, we are more concerned about reducing the computing power on the server side by giving the client a puzzle to solve. To prevent such attacks, we use client puzzle mechanism. In this mechanism, we introduce a client-side puzzle which demands the machine to perform tasks that require more resources (computation power). The client's request is not directly sent to the server. Moreover, there will be an Intermediate Server to monitor all the requests that are being sent to the main server. Before the client's request is sent to the server, it must solve a puzzle and send the answer. Intermediate Server is used to validate the answer and give access to the client or block the client from accessing the server.

Keywords: *Distributed Denial of Service Attacks (DDoS); Client Puzzle Mechanism; Cryptographic Puzzles; Authentication.*

1. Introduction

In a typical DoS Attack, the attacker hacks a set of computers and uses them to flood the server. DoS attacks do not impose any threat to the server. But they prevent access to the server or the website. DoS attacks are classified into Volume based, Protocol, and Application layer attacks. DoS attacks crash the system to disrupt the access which is given to a real client who can't send or get the acknowledgement/response from server after the attack. The attacker consumes and uses the majority of the assets of the web server like computational power.

A counter measure is a process that avoids or mitigates the consequences of the DoS attack done on the server. In this paper we provide details regarding a project we have developed in light of the thoughts of Yongdong Wu, Zhigang Zhao. Denial of Service attacks are of three types: Smurf, UDP flood and SYN flood attacks. Regardless of the huge assortments of attacks there is a typical target among a wide range of DoS attacks. The attackers intend to disrupt the assets of the framework which contains CPU cycles, memory, computation power. The attacker usually produces a large number of requests or bogus queries so that he can consume all the computation power of the server and thereby prevent the server from being accessed by the legit users for whom the services are intended. Regularly their task is just to generate and send a lot of queries to the web server or the website. However, the attack can shift fundamentally in numerous angles. In this paper we propose cryptographic puzzles as a counter measure on the attacks which can save a lot of resources of the server and let it offer services to the intended users. In the puzzle scheme, the client has to do some work before the server responds to its requests. The client machine which cannot solve the puzzle within the required time will be blocked from accessing the server. This whole process requires a human to operate the client machine so that the process cannot be automated just like in the case of botnets. If the process is automated in a botnet, the puzzle cannot be solved and submitting the wrong answer or not solving or not submitting the answer within the specified time leads to blocking the client. In most normal cases, each puzzle requires a great number of cryptographic operations to be performed, for example, hashing to process the puzzle arrangement. So, in this case, the attacker need to possess and control a lot of resources like client machines, more bandwidth, and processing power than the server. Initially, the client won't be able to send request to the server directly.

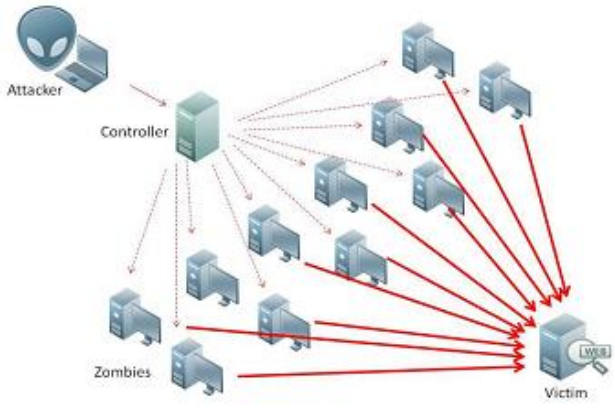


Fig. 1: A Typical DDoS Attack.

Copyright © 2018 C. Vasan Sai Krishna et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We introduce an intermediate server which lies in between the client and the server. The intermediate server looks after the puzzles, Answers to the puzzles and all. It processes the request and gives the client a puzzle to solve. The intermediate server can also specify the time period in which the puzzle is supposed to be solved. If the client machine is unable to solve, it blocks the access. The development of this scheme is to block access to the attackers or illegitimate users and provide services to the legit users for whom the services are intended.

2. Literature survey

Juels et.al presents a protocol where small crypto puzzles can be used to prevent or control resource depletion attacks [1]. This protocol is useful when attackers are capable of depleting connections at a fast rate i.e., attackers exploiting internal resources or SSL protocols.

Jeff Green et.al analyses different attack types in perspective of GPU-based attacks and distinguishes attributes that enable security systems to withstand attacks [2]. Specifically, they exhibit hash-reversal plans which adjust exclusively on server. He also presents about different Proof of Work techniques which require the client to put some computational effort to be able to solve the puzzle to gain access to the server. The main proposal is that the Hash-Reversal Proof of Work schemes are able to restrict by adjusting the difficulty of the puzzle by basing on the client's past behavior.

Kaiser et.al in their research work described an approach to mod_kaPoW system which has great efficiency and human transparency of PoW strategy [3]. The proposed mechanism has a software backwards compatibility. Still, there are a few places where CAPTCHAs do not work up to the mark. For example, visually challenged people cannot solve them.

Christos et.al presents different methods to deal with the DDoS and DoS attacks so that they can be prevented [4]. Besides, different specifications, effect of the attacks on the server, attack landscapes are presented in the paper. The objective of the paper is to give a brief about the different types of DoS and DDoS attacks, how DDoS attacks can be done and in this way more productive and viable calculations, strategies, and techniques to battle these attacks can be created. They have mentioned in their research work that not only wired networks, but also wireless networks are prone to DDoS attacks.

Qiang Tang et.al have proposed a security architecture where he concentrated on two properties, regarding the determinable difficulty and the parallel computation resistance [5].

Yves Igor et.al introduces a client puzzle mechanism where the generation of puzzle is dependent on calculating of square roots modulo with a prime [6]. This mechanism is a counter measure for solving the puzzles by using parallel computing i.e., running a single puzzle on different machines to solve it. According to the proposed method, it is impossible to distribute the puzzle solving task to different machines. This is an advantage because the solution cannot be obtained faster than scheduled. The puzzles can be implemented with feature rich UI in an interactive manner without compromising comfort with security. They've also introduced a bandwidth based cost factor for solving the puzzle. This uses client-side computation in order to reduce the consumption of server's computational resources.

3. Implementation

3.1. Time-lock mechanism

In a time-lock puzzle scheme the client machine will be demanded to give solution to the puzzle in a specific atime. Unlike in the other puzzle schemes where the solution can be found at any time, A time lock puzzle restricts to a specific amount of time. A time lock puzzle system hides the puzzle at the client side and allows the sender to solve the puzzle only within a particular amount of time. If the sender fails to solve it in the specific time, He will be blocked. The first objective was to guarantee that a customer can't decode a given message until the point that a given timeframe has passed. This is the notion of time-released crypto where we end up encrypting a message so that it cannot be decoded by others including the sender until a particular amount of time has passed. Here we are trying to send information into the future. These puzzles need a precise amount of time to solve. The solution reveals a key that can be used to decrypt the encrypted information.

3.2. Puzzle based authentication

Puzzle Based mechanisms are used to ensure that access to the server or the website is only given to a client for whom the service is intended. Puzzle based mechanisms usually try to reduce the burden on the server and will require the client machine to perform computationally expensive tasks so that we can ensure that the access is given to only the legitimate client but not the hacker. These mechanisms are used to maintain authenticity. Here a puzzle will be generated at the client side. The client puzzle is generated at the client side and the client has to solve it so that the server can be accessed.

3.3. Memory-bound client puzzles

A resource can be exploited by the attackers if the clients have no cost to use it. So, we require the client to do some computationally expensive tasks. This prevents DDoS attacks because the attacker has to spend a lot of resources to send requests to the server or the website. But there may be many legit users who have access to low computation powers. This prevents legit users from accessing the website or server. Abadi et. al. [7] and Dwork et. al. [8] addressed this limitation by showing that memory access times vary much less than CPU speeds, and hence offer a viable alternative.

We are proposing a software puzzle methodology in this paper in order to ensure authenticity and provide access to only those legitimate users. This can prevent botnets from exploiting the websites or the servers' resources. Here we introduce an intermediate server which lies in between the client and the server. This intermediate server can be used to look after the IPs that are doing malicious activities. The intermediate server can be used to monitor the traffic because the users are not actually provided access directly to the web server. There are intermediate servers in between to regulate and monitor the traffic. They can be used for blocking/unblocking the users too.

To access the web server the client has to solve the puzzle given by the server so that he can be recognized as the legitimate users.

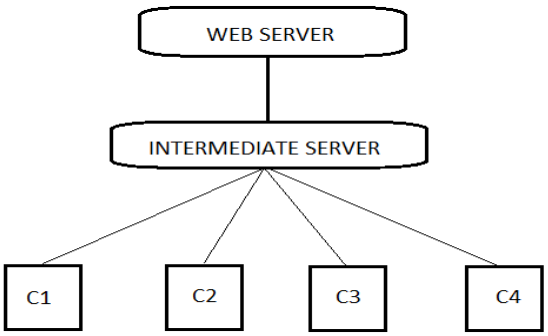


Fig. 2: Architecture of the Proposed System.

Unlike existing puzzle schemes, the puzzle will only be generated after the request is received from the client side. Here we protect the puzzle code so that it cannot be decoded by bots so easily. We also put time limit for solving it. The client will be demanded to produce a solution to the puzzle in a given specific time period that is given by the server. If he fails to solve the puzzle in the given time or provides a wrong answer, the message will not be sent to the server. At first the client is required to send the message. The message will not be received at the server side until the puzzle has been solved. The intermediate server will then have to

set some time limit to solve the puzzle. Upon successful solving of the puzzle, the message will be received at the server's end. Though we focus on GPU inflation techniques, it can be extended to prevent attackers from exploiting other resources like cloud computing too.

SSL protocol is most widely used across many websites. There is a limitation/drawback for this protocol i.e., the SSL server must perform expensive operations like RSA decryption. The server performs the decryption for every request or query from the client thereby processing a great number of requests. This makes the protocol vulnerable to DoS attack. In our work we mainly concentrate on protecting the web server against DoS attacks. We will always ensure that the process of solving the puzzle should always be done by physical presence of a human being and cannot be automated. The main motto is to secure the SSL server from the GPU inflated DoS attacks. SSL protocol usually involves several rounds which include RSA decryption round. The depth of the defence mechanism is dependent on time cost of the server. The proposed client puzzle mechanism is dynamic and is generated after a request is received from the client side. This method helps when the attacker can inflate the GPU and automate the puzzle solving process. This weakens the effectiveness of the client puzzles. Here we introduce a new web server called the Intermediate Server. The Intermediate Server can be a virtual and isolated space in the same web server or a different web server at another location. Unlike in traditional client-server model where both data and authentication mechanism are in the same web server, we use an intermediate server to authenticate users. Intermediate server also maintains a log of the IPs and can also be used to block suspicious users. We divide the whole thing into three parts namely the Client, Intermediate Server and Web Server.

3.4. Web server

A typical client-server model consists of a web server where we will have all the data in a centralized web server. In our work, puzzle generating code and the data will be in the web server. There will be different Intermediate servers deployed at different places. The main server will generate the puzzle algorithm and will validate the solution. The client will be sending requests to the web server.

Secret key	No of Packets	Secret keypath	Issued Time	Resolved Time
8 - 8 - 8 - 7	1	secret.dat	35secs	21secs
9 * 9 + 3 / 6	1	secret.dat	22secs	8secs
2 * 9 - 8 * 9	1	secret.dat	15secs	4secs
1 - 6 * 5 + 4	1	secret.dat	52secs	9secs
8 / 3 * 10 - 8	1	secret.dat	28secs	6secs

Fig. 3: Server-Side Log.

Upon receiving the requests from the client, the network server checks if the request is legit or not by going through the IP log (Fig 3). The database consists of the secret key, number of packets, keypath, issued time, resolved time. If the IP is supposed to be blocked, the Intermediate Server will block it. Else the request will be forwarded to the web server. The web server will then generate a puzzle which has to be solved by the client. The web server will also give some time to solve the puzzle. If the puzzle is solved by the client in the specified time, the web server accepts the request and a secure connection will be established. Request processing and resource sharing will be done post connection establishment.

3.5. Client/node

Client is nothing but the computer that belongs to the user from which he/she is trying to access the website. In order to gain access to the server, the client machine has to solve computationally expensive tasks.

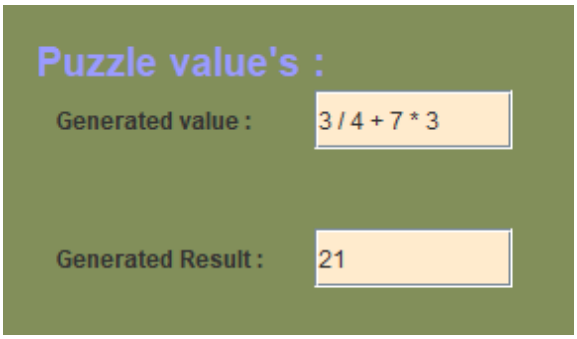


Fig. 4: Client-Side fields for Puzzle Solving.

The client puzzle needs to be solved before the client's message or query is sent to the server. Puzzle generation is done before receiving the query by the intermediate server. Then the client needs to solve the puzzle in the time specified by the intermediate server. This is done to authenticate client to the website or the server.

We will always ensure that the client machine cannot automate the puzzle solving process and human presence is required to solve the puzzle. The attackers usually hack botnets to automate the puzzle solving process. Thus, by taking proper security measures like "puzzle generation upon receiving the request", we can protect websites/servers from Denial of Service attacks.

3.6. Intermediate server

Source IP	Destination IP	Seq No	Status
192.168.0.103	127.0.0.1	1	UnBlock
192.168.0.103	127.0.0.1	1	UnBlock
192.168.0.103	127.0.0.1	1	UnBlock
192.168.0.103	127.0.0.1	1	unblock

Fig. 5: Log Maintained by the Intermediate Server.

Between the client and the server, there exists an intermediate server which is used to generate puzzles that are to be solved by the client to gain access to the server. The intermediate server is responsible for preventing the malicious activities by clients and abuse of the server's resources. It generates a puzzle upon receiving request from the client node and allocates some time to solve the puzzle. If the client fails to solve the puzzle within specified time, he cannot access the server or the website. Through intermediate server we can monitor traffic (Fig 6). It also maintains a log (Fig 5) of the IPs for which the access is given and those for which the access is not given. The log consists of Source IP, Destination IP, Seq Number, Status. Intermediate server can block and unblock the malicious clients from accessing the website.

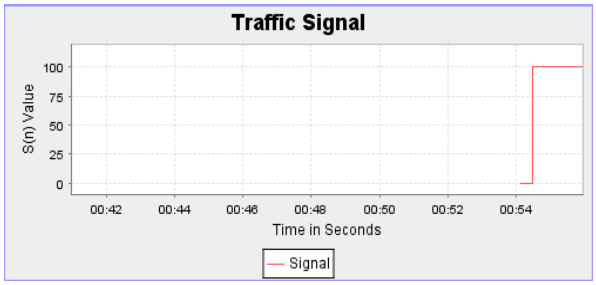


Fig. 6: Signal in Intermediate Server.

4. Conclusion

Denial of Service attacks are evolving day by day. The attackers are creating botnets with more resources. In this paper we have proposed a method to authenticate users. The users are required to prove that they're humans before they are given access to the web server. In this method, a client puzzle algorithm is generated only after the client sends request for resources to the web server. This

method reduces the burden on web server which is done by attackers to crash it. Access is given only to those legitimate users for whom the resources are intended. This demands the client to spend some computing resources than putting it all on the web server. We also introduced an intermediate server which receives the requests first and then forwards to the web server where the actual resources exist. The intermediate server is used to maintain a log of the IPs which are accessing the resources, to block/unblock IPs, to monitor the traffic. The scheme always ensures the physical presence of human being and blocks those client machines which belong to botnets.

References

[1] A. Juels and J. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in Proc. Netw. Distrib. Syst. Secur. Symp. 1999

[2] J. Green, J. Juen, O. Fatemieh, R. Shankesi, D. Jin, and C. A. Gunter "Reconstructing Hash Reversal based Proof of Work Schemes," in Proc. 4th USENIX Workshop Large-Scale Exploits Emergent Threats, 2011.

[3] E. Kaiser and W.-C. Feng "mod_kaPoW: Mitigating DoS with transparent proof-of-work,", in Proc. ACM CoNEXT Conf., 2007. https://doi.org/10.1145/1364654.1364737.

[4] Christos Douligeris, "DDoS attacks and defense mechanisms: classification and state-of-the-art," Department of Informatics, University of Piraeus, 80 Karaoli and Dimitriou Str, Piraeus 18534, 13 October 2003.

[5] Qiang Tang* and Arjan Jeckmans, "Towards a security model for computational puzzle schemes," International Journal of Computer Mathematics Vol. 88, No.11,pp. 2246–2257, July 2011. https://doi.org/10.1080/00207160.2010.543951.

[6] Yves Igor Jerschow Martin Mauve, "Non-Parallelizable and Non-Interactive Client Puzzles from Modular Square Roots", Institute of Computer Science, Heinrich Heine University, D¨usseldorf, Germany.

[7] Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. In: Proceedings of Network and Distributed Systems Security Symposium, San Diego, California, USA, 2003, pp. 107–121 (February 2003)

[8] Dwork, C., Goldberg, A., Naor, M.: On memory-bound functions for fighting spam. In: Proceedings of the 23rd Annual International Cryptology Conference, pp. 426–444 (2003) https://doi.org/10.1007/978-3-540-45146-4_25.

[9] T. J. McNevin, J.-M. Park, and R. Marchany, "A DoS limiting network architecture," Virginia Tech Univ., Dept. Elect. Comput. Eng., Blacksburg, VA, USA, Tech. Rep. TR-ECE-04-10, Oct. 2004.

[10] Sujata Doshi, Fabian Monrose, and Aviel D. Rubin Johns, "Efficient Memory Bound Puzzles Using Pattern Databases" J. Zhou, M. Yung, and F. Bao(Eds.): ACNS 2006, LNCS 3989, pp. 98–113, 2006. c Springer-Verlag Berlin Heidelberg 2006.

[11] Yongdong Wu, Zhigang Zhao, Feng Bao, and Robert H. Deng, " Software Puzzle: A Countermeasure to Resource-Inflated Denial-of-Service Attacks " Ieee Transactions On Information Forensics And Security, Vol. 10, No. 1, January 2015

[12] J. E. Smith and R. Nair, Virtual Machines: Versatile Platforms for Systems and Processes. San Mateo, CA, USA: Morgan Kaufmann, 2005, p. 19.

[13] J. Ansel et al., "Language-independent sandboxing of just-in-time compilation and self-modifying code," in Proc. ACM SIGPLAN Conf. Program. Lang. Design Implement. 2011, pp. 355–366. https://doi.org/10.1145/1993498.1993540.

[14] H.-Y. Tsai, Y.-L. Huang, and D. Wagner, "A graph approach to quantitative analysis of control-flow obfuscating transformations," IEEE Trans. Inf. Forensics Security, vol. 4, no. 2, pp. 257–267, Jun. 2009. https://doi.org/10.1109/TIFS.2008.2011077.

[15] D. Kahn, the Codebreakers: The Story of Secret Writing, 2nd ed. New York, NY, USA: Scribners, 1996, p. 235.

[16] X. Wang and M. K. Reiter, "Mitigating bandwidth-exhaustion attacks using congestion puzzles," in Proc. 11th ACM Conf. Comput. Commun. Secur, 2004, pp. 257–267. https://doi.org/10.1145/1030083.1030118.