Defending DoS Attacks Using a Puzzle-Based Approach and Reduction in Traceback Time towards the Attacker

Anup Mathew Abraham[1] and Shweta Vincent[2]

[1] St. Thomas Institute of Science and Technology, Kottayam, Kerala, India
[2] Karunya University, Coimbatore, Tamil Nadu, India

Abstract. In today's world Denial-of-Service attacks have huge impact on network security. DoS attacks are usually launched to make the service of a system unavailable to a person who is authorized to use it. Several methods were introduced for defending Denial-of-Service attacks earlier. In our paper we propose a client-puzzle mechanism approach to defend DoS attacks. Here, intermediate routers can be used for issuing and solving network puzzles of various difficulty levels depending on the intensity of the attack. The target server will be protected by using an intermediate firewall router for issuing the puzzles; this will reduce the load over the server. Likewise intermediate proxy routers can be used for solving the puzzle. But furthermore, this proxy can also be a target of attack. This problem can be overcome by using a hybrid traceback mechanism for the attacking client. This technique helps to find out the attacking node and the router through which the attack packet was forwarded.

Keywords: Client-Puzzle Approach, Flooding DoS Attack, Game Theory, Nash Equilibrium, IP Traceback.

1 Introduction

Security has become one of the primary concerns when an organization connects its private network to the Internet. In a DoS attack, an attacker attempts to prevent legitimate users from accessing information or services from a server. By targeting a computer an attacker may be able to prevent a user from accessing email, websites, online accounts, or other services that rely on the affected computer. The most common and obvious type of DoS attack occurs when an attacker floods a network with unnecessary packets of information.

Network Puzzles can be used for preventing a DoS attack to some extent. Network Puzzles make the attacker client to commit his resources for solving a puzzle before he gets access to the services provided by the server [1]. The service of the server is only provided for clients who are able to solve the puzzle and send the correct solution. The primary function of the server is to produce a puzzle with optimal difficulty. If the difficulty of the puzzle is less the attacker will be able to solve it and initiate an intense attack. If the difficulty of the puzzle is high then the attacker will respond by sending some random answers, the server will have to verify the solution of the puzzle which causes exhaustion of the defending server's resources. So the puzzle must be of an optimal difficulty.

The network puzzle used will be installed in the server or puzzle issuing router and also in the client machine or intermediate puzzle solving proxy. If the zombies are attacking through a particular proxy then the proxy will be throttled by packets and puzzles. This may cause the legitimate users to wait for long time till the proxy solves the puzzle for the sent packets.

Tracing back is the technique used to identify the attacking node after a DoS attack has occurred [7], [10]. By combining both the client-puzzle technique and traceback an attack over a particular proxy router can be stopped to a certain extent. The packet flow will be halted at the first router which is forwarding the packets. This will reduce the traffic over the proxy router who is solving the puzzle for the clients. The copies of packets used for attacking are sent to other routers in the network so that if similar packets are found then the packets can be dropped at that point itself.

In this paper the second section highlights about the concept of a client-puzzle. The third section elaborates on the system design. The next few sections explain in detail about the use of Game theory, Nash equilibrium in the creation of Network puzzles, and the concept of Traceback. The final section describes about our implementation results and scope of further enhancement.

2 Protocol for Generation of Puzzles

This section describes the concept used in generating puzzles by a server, for being issued to clients that request the server for a connection and its services [1]. The server, upon receiving a request from a client, generates a puzzle and its answer along with a hash of the answer, the server nonce, the puzzle expiration time, the puzzle maturity time, and the flow identifier (Figure 1). The server then sends back to the client: the client cookie, the puzzle and its parameters, the flow identifier, and a server cookie consisting of the above hash, the server's timestamp, the puzzle maturity and expiration times.

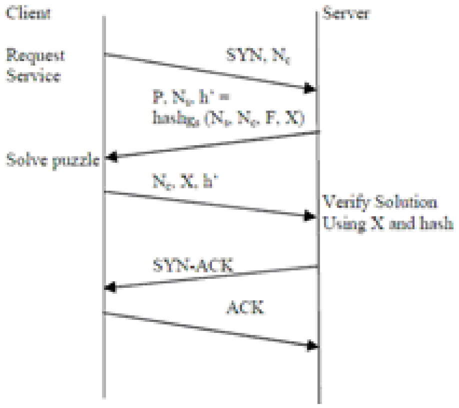
Client
Server
Request Service
SYN, N_c
P, N_p, h' = hash_Ks (N_p, N_c, F, X)
Solve puzzle
N_c, X, h'
Verify Solution Using X and hash
SYN-ACK
ACK

Fig. 1. Puzzle Mechanism (Adapted from [1])

The client, upon receiving the puzzle, calculates the solution and sends back the answer along with the server cookie. Upon receipt of this message, the server takes the server timestamp, uses it to index into the server nonce cache to obtain the server nonce, checks that the nonce has not expired, and verifies the answer by regenerating the hash and comparing it against what the client sent. The Client Nonce used will provide more authenticity to the puzzle protocol. Maturity time for the puzzle is also added to avoid eavesdropping of the puzzle. Expiry time is another feature added to avoid reuse of the same puzzle; it may cause the protection through puzzles.

3 System Design

This section highlights the design of the system worked upon for the implementation of the Client-puzzle mechanism and concept of traceback.

Upon receiving a request from a client for a connection, the defender server produces a puzzle and sends it to the requester. If it is answered by a correct solution, the corresponding resources are then allocated. As solving a puzzle is resource consuming, the attacker client is deterred from perpetrating the attack.

The defense involves a verification stage: every time a request for a service is received, the defender responds with a puzzle, which the attacker must allocate resources to solving before receiving the said services. These two players form the most basic realization of the game (in Game Theory) [1]. Both players have strategies that involve tradeoffs. The attacker must balance the severity of the attack against the amount of resources allocated to carrying it out. The, defender must balance the amount of resources spent defending against an attack and the quality of service it provides to legitimate users.

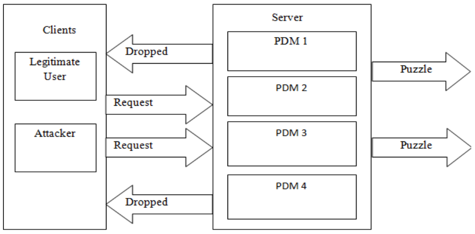

Fig. 2. System Architecture (1/2)

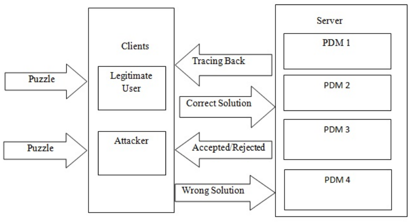

Fig. 3. System Architecture (2/2)

4 Game Theory and Nash Equilibrium

A game is a formal description of a strategic situation. Game theory is the formal study of decision-making where several players must make choices that potentially affect the interests of the other players [1]. A game consists of a set of players, a set of moves available to those players, and a specification of payoffs for each combination of strategies.

Nash equilibrium, also called strategic equilibrium, is a list of strategies, one for each player, which has the property that no player can unilaterally change his strategy and get a better payoff [1]. A payoff is a number, also called utility that reflects the desirability of an outcome to a player, for whatever reason. When the outcome is random, payoffs are usually weighted with their probabilities. In game theory, Nash equilibrium is a solution concept of a game involving two or more players, in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only his or her own strategy unilaterally. If each player has chosen a strategy and no player can benefit by changing his/her strategy while the other players keep their unchanged, then the current set of strategy choices and the corresponding payoffs constitute Nash equilibrium. Players are in equilibrium if a change in strategies by any one of them would lead that player to earn less than if she remained with her current strategy.

5 Network Puzzles

Until recent times, client puzzles [3] have been used only as an application layer defense against flooding attacks. Yet, client puzzles in the application layer can be easily thwarted if any adjacent or underlying protocol does not provide a similar defense [3]. Network layer puzzles do not preclude the use of higher layer defenses, but rather augment them.

Network puzzles [3] can be selectively applied to all communication from a client. Attacks which use seemingly innocent communication channels for coordination can be thwarted by having difficult puzzles applied to the coordination traffic in addition to the attack traffic. Being able to place puzzle issuers arbitrarily close to the client allows quenching undesirable traffic closer to the source, reducing the wastage of resources deeper within the network. Another issue with puzzles is that the work load required for solving each puzzle must be adjusted to meet the real-time needs of a network. For example, a congested server can throttle its clients using harder puzzles, or a server with a client who is acting maliciously can throttle the individual using more difficult puzzles [5]. Throttling a client cannot be done simply during the establishment of a connection; the need for puzzles may start and stop at any point in a flow's lifetime.

6 Tracing Back

Trace back is a name given to any method for reliably determining the origin of a packet on the Internet [7], [8]. Due to the trusting nature of the IP protocol, the source

IP address of a packet is not authenticated. As a result, the source address in an IP packet can be falsified causing DoS attacks. The problem of finding the source of a packet is called the IP trace back problem.

With router based approaches, the router is charged with maintaining information regarding packets that pass through it. The idea proposed in [8] is to generate a fingerprint of the packet, based upon the invariant portions of the packet and the first 8 bytes of payload.

The concept of packet marking and logging is used as in the Hybird IP traceback approach proposed in [8]. An attack packet is 'marked' by an ingress router as it enters the network as finds its path to the target server. At the next hop, relevant information of this 'marked' packet is 'logged' by the next hop router. This process continues until the target server is reached. Finally once the server goes down, the attacking node can be successfully traced back to using the marking and logging information present in various routers en route.

7 Implementation

There are basically two parts which have been followed in this paper for the implementation of the Network Puzzles. The first part is to create a puzzle issuer which resides in the server. This application is activated when a client tries to connect to the server. The second part is a solver which is on the client machine. The solver solves the puzzle created by the puzzle issuer and sends the solution to the server which is verified by the puzzle issuer. The puzzle solver and issuer are created using Netfilter in Linux. Further, results of time taken to solve network puzzles of various difficulty levels are simulated using NS-2 [13]-[16].

Netfilter and IP tables in Linux are used for creating the network puzzle. Netfilter is the set of hooks within the Linux kernel for intercepting and manipulating network packets. Netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook.

IP tables are a generic table structure for the definition of rule sets. Each rule within an IP table consists of a number of classifiers and one connected action.

In the implementation part two approaches are considered. In the first approach the code is installed in a Linux system. The time taken for creating the puzzle and solving it is studied at various difficulty levels. In the second approach the scenario is simulated in NS-2.

In the first approach, rules are added to the IP table. The rules will act as chains through which the packet has to flow. There is an issuer, a solver, puzzle manager and a nonce manager. The issuer is installed in the firewall router and it issues the puzzle to the client. The solver is installed at the proxy router and it is responsible for solving the puzzle. The puzzle manager controls the difficulty level of the puzzle. This is done based on the number of packets received and the resources left, for solving the puzzle, for the server or firewall router. The nonce manager produces the server nonce and also verifies the client nonce sent by the client.

In the second approach, the various readings of time for solving the puzzles of different difficulty levels are used for simulating the scenario in NS-2. Figure 4 clearly illustrates the scenario of the simulation. Here, node 5 is the end server and node 4 is its proxy. Node 4 is responsible for issuing puzzles to all the clients in the network. Those clients which send request packets at a higher rate are given puzzles of greater difficulty to solve than the rest.

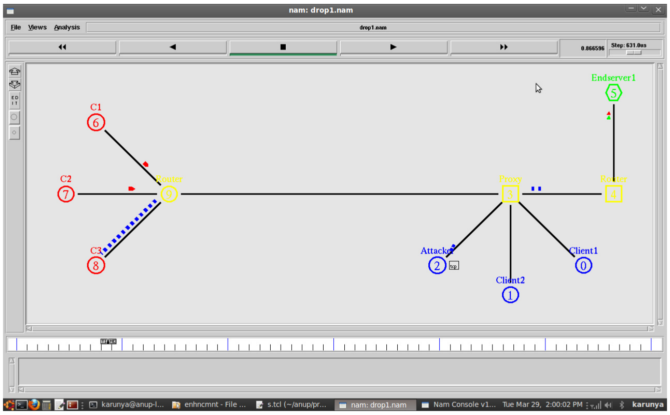

Fig. 4. Attack Scenario for Client-Puzzle Mechanism Simulation in NS-2

Further, an attempt was made to implement the Hybrid IP Traceback mechanism as proposed in [8] and to enhance its application. The enhanced version of HIT has been termed as Enhanced Hybrid IP Traceback mechanism. The simulation scenario consisted of 40 nodes serving as routers in the network. Figure 5 illustrates the scenario generated for the HIT and EHIT mechanism implementation.

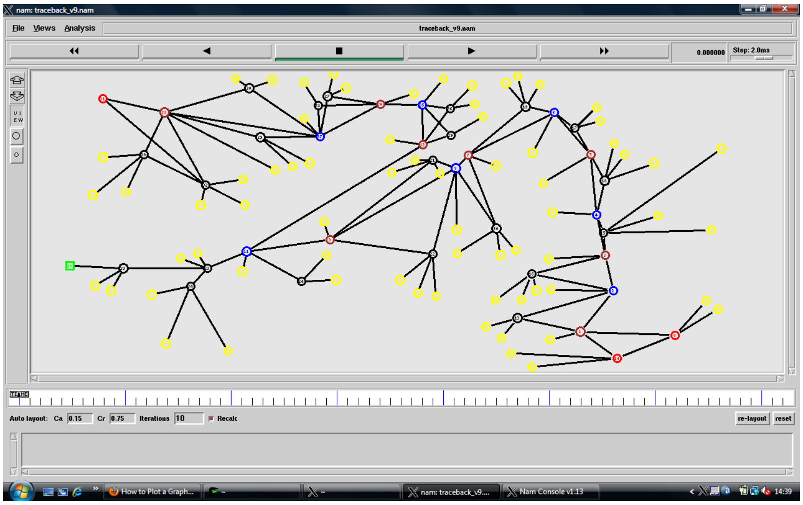

Fig. 5. HIT [8] and EHIT Simulation Scenario in NS-2

Each of the routers is either a marking or logging router. The implementation includes the calculation of time taken for traceback when simple HIT mechanism was used. The enhancement proposed to HIT i.e. EHIT includes sending copies of the attack packet (once the attacker has been identified) to the egress router of the network from which the attacker was detected, so that if the attacker tries to attack from a different host in the same network, the router will compare the packet with the copy of the attacking packet. If matched, the packets from the attacker will not be forwarded. Thus the attack can be thwarted earlier in the second phase of attackc.

8 Result Analysis

The results for Client-Puzzle mechanism were generated for a network that consisted of a set of four clients and a single puzzle-protected server connected on a single VLAN via a Cisco Catalyst 4006 Gigabit switch. Each client and server was a dual 1.8GHz Intel Xeon machine with Gigabit Ethernet interfaces. To measure the baseline performance of the system, clients were configured to flood the server with 64-byte UDP packets while solving all of the puzzles that were received.

The time taken to solve the puzzles of various difficulties was calculated. The response time obtained was plotted versus the difficulty level of the puzzle. Figure 6 clearly illustrates the relation between the response time of solving a puzzle with its difficulty level.

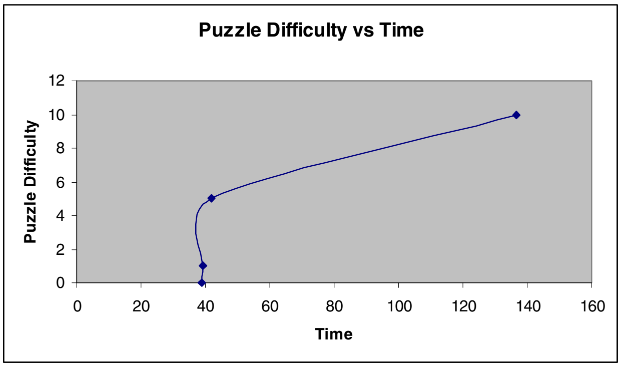

Fig. 6. Plot of Puzzle difficulty vs. Time

Further more, the simulation of HIT and EHIT was carried out and the reslults of simulation time in seconds is tabulated in Table 1. Two attackers, 1 and 2 were chosen belonging to the same network for both the scenario simulations. In case of HIT, the time for tracing back to both the attackers remains similar. In case of EHIT, the attack packet information was propagated to the egress router to whose network attacker 1 belonged; once attacker 1 had been identified. Hence, when attacker 2 from the same network as attacker 1, launched an attack, it could be thwarted faster. This is evident from the tabulated time in Table 1. The figure 7 illustrates the Traceback time requirement graph plotted for HIT and EHIT protocols. This figure clearly illustrates the reduction in tracek back time when EHIT protocol is used.

Table 1. Simulation results of HIT and EHIT in NS-2

PROTOCOL	ATTACKER NO.	TRACEBACK SIMULATION TIME IN SECONDS
HIT	Attacker 1	5.388
	Attacker 2	5.280
Enhanced HIT	Attacker 1	5.380
	Attacker 2	1.772

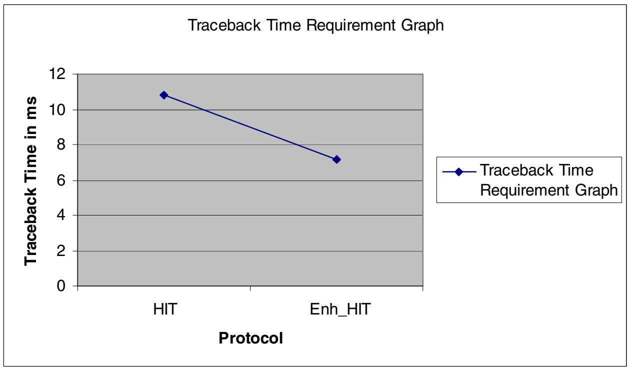

Fig. 7. Traceback time requirement graph for HIT and EHIT

9 Conclusion

A puzzle based mechanism is proposed in this paper and is implemented in Netfilter using IP tables. The server to be protected will issue the puzzles and the client has to solve the puzzle compulsorily to gain access to the server. The method is effective because the puzzle mechanism works at network level. Tracing back provides additional security to the system by helping the server to trace back the attacker to the last router. The puzzle solving proxy router can be protected by this method. By using this method optimal puzzle difficulty can be produced and the Denial of Service attack can be stopped.

Also, the Hybrid IP Traceback mechanism was simulated and a further enhancement for the same was proposed. The simulation results showed that trace back time taken in EHIT is lesser than that for HIT.

Future work would include sending copies of the attack packet to all the nearby routers so that if the attacker tries to attack from a different location the router will compare the packet with the copy of the attacking packet. If matched, the packets from the attacker will not be forwarded. Thus the attack can be solved in the network without even issuing the puzzle or packets reaching the firewall router or proxy router.

References

[1] Fallah, M.S.: A Puzzle-Based Defense Strategy Against Flooding Attacks Using Game Theory. IEEE Transactions on Dependable and Secure Computing 7(1) (January-March 2010)

[2] Waters, B., Jules, A., Halderman, J., Felten, E.: New Client Puzzle Outsourcing techniques for DoS Resistance. In: Proc. ACM Conf. Computer and Comm. Security, pp. 246–256 (2004)

[3] Feng, W., Kaiser, E., Feng, W., Luu, A.: The Design and Implementation of Network Puzzles. In: Proc. 24th Ann. Joint Conf. IEEE Computer and Comm. Societies, pp. 2372–2382 (2005)

[4] Wang, X., Reiter, M.: Defending Against Denial-of-Service Attacks with Puzzle Auctions. In: Proc. IEEE Security and Privacy, pp. 78–92 (2003)

[5] Mahimkar, A., Shmatikov, V.: Game-Based Analysis of Denial of Service Prevention Protocols. In: Proc. 18th Computer Security Foundations Workshop, pp. 287–301 (2005)

[6] Moore, D., Shannon, C., Brown, D.J., Voelker, G.M., Savage, S.: Inferring Internet Denial-of-Service Activity. ACM Trans. Computer Systems 24(2), 115–139 (2006)

[7] Savage, S., Wetherall, D., Karlin, A., Anderson, T.: Practical Network Support for IP Traceback. In: SIGCOMM 2000, Stockholm, Sweden (2000)

[8] Gong, C., Sarac, K.: A More Practical Approach for Single-Packet IP Traceback Using Packet Logging and Marking. IEEE Trasactions on Parallel and Distributed Sysems 19(10) (October 2008)

[9] Savage, S., Wetherall, D., Karlin, A., Anderson, T.: Network Support for IP Traceback. IEEE/ACM Trans. Networking 9(3), 226–237 (2001)

[10] Snoeren, A., Partridge, C., Sanchez, L., Jones, C., Tchakountio, F., Schwartz, B., Kent, S., Strayer, W.: Single-Packet IP Traceback. IEEE/ACM Trans. Networking 10(6), 721–734 (2002)

[11] Al-Duwairi, B., Manimaran, G.: Novel Hybrid Schemes Employing Packet Marking and Logging for IP Traceback. IEEE Trans. Parallel and Distributed Systems 17(5), 403–418 (2006)

[12] Snoeren, A., Partridge, C., Sanchez, L., Jones, C., Tchakountio, F., Kent, S., Strayer, W.: Hash-Based IP Traceback. In: Proc. ACM SIGCOMM 2001 (August 2001)

[13] Wang, J.: ns-2 Tutorial (1), Multimedia Networking Group, The Department of Computer Science, UVA

[14] Wang, J.: ns-2 Tutorial (2), Multimedia Networking Group, The Department of Computer Science, UVA

[15] Wang, G., Xia, Y., Harrison, D.: An NS-2 TCP Evaluation Tool: Installation Guide and Tutorial, NEC Laboratories China, BitTorrent (April 29, 2007)

[16] Wang, G., Xia, Y., Harrison, D.: An NS-2 TCP Evaluation Tool: Installation Guide and Tutorial, NEC Laboratories China, BitTorrent (November 24, 2008)