New Integrated Approach for Mitigating DDOS Attacks

Pranay Meshram
Information Technology
PJLCOE
Nagpur ,India

Ravindra Jogekar
Computer Sci.& Engineering
PJLCOE
Nagpur ,India

Pratibha Bhaisare
Computer Sci.& Engineering
AGPCOE
Nagpur ,India

ABSTRACT

In this paper we provide an integrated defense solution that enables filtering and admission challenges to be implemented in a distributed manner throughout the network on behalf of the target. The admission challenge is provided through the client puzzles employed at the target. This scuttles any attempt made by the attacker to flood the target because until the client solves the puzzle it isn't granted access to the targets resources. If the attack persists or worsens, then the target could propagate a distress signal upstream to its Internet Service Provider (ISP), who could deploy proxy defenses at the ingress points to the ISP's network on behalf of the target. In general, the target's ISP could request other upstream ISPs to also deploy the defenses for the target by using the pushback technique, so that the attack traffic is blocked as close as possible to the source of the traffic. A key advantage of this proposed approach is that it could enable the defenders to harness greater computational resources in order to counteract the growth in attack power that is becoming available to attackers.

Keywords

Client Puzzle, Pushback, Integrated Approach

1. INTRODUCTION

The Internet connects hundreds of millions of computers across the world running on multiple hardware and software platforms. It serves uncountable personal and professional needs for people and corporations. However, this interconnectivity among computers also enables malicious users to misuse resources and mount denial of service (DoS) attacks against arbitrary sites. In a denial of service attack, a malicious user exploits the connectivity of the Internet to cripple the services offered by a victim site, often simply by flooding a victim with many requests. A DoS attack can be either a single-source attack, originating at only one host, or a multi-source, where multiple hosts coordinate to flood the victim with a barrage of attack packets. The latter is called a distributed denial of service (DDoS) attack. The goal of a DDoS attack is to compromise scores of computer systems(known as zombies/slaves) spread over the Internet using a variety of methods and targeting a single system thereby causing denial of service for users of the targeted system. The flood of incoming messages to the target system essentially forces it to shut down, thereby denying service of the system to legitimate users. A Distributed Denial of Service Attack involves typically a master and many slaves/zombies. The master represents the host used by the attacker to coordinate the attacks, while the zombies represent the vulnerable systems exploited by the attacker. Prior to launching an attack, the attacker scans the systems and uploads the attacking software into the master and zombie hosts. This allows the attacker to remotely control the each zombie and master host without running the risk of being detected.

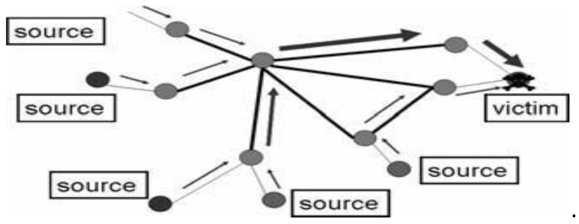

Fig 1: DDoS attack

Attackers constantly modify their tools to bypass these security systems, and researchers in turn modify their approaches to handle new attacks. The DDoS field is evolving quickly, and it is becoming increasingly hard to grasp a global view of the problem. In Feb. 2000, a string of DDoS attacks crippled popular wed sites including CNN.com, yahoo.com, eBay.com for several hours. In 2003, for example, one honey pot research project saw 15,164 unique zombies from a large botnet within days. In 2004, the witty worm created 12,000 zombies within 45min. IP spoofing has often been exploited by DDOS attack to 1) conceal flooding sources and dilute localities in flooding traffic 2) coax legitimate host into becoming reflectors redirecting and amplifying flooding traffic.

2. RELATED WORK

Denial of service attacks attempt to exhaust or disable access to resources at the victim. These resources are either network bandwidth, computing power, or operating system data structures. Attack detection identifies an ongoing attack using either anomaly-detection [30, 6, 14] or signature-scan techniques [32, 21]. Most response mechanisms attempt to alleviate the damage caused by the attack by taking reactive measures like reducing the intensity of the attack by blocking attack packets [22, 23, 6], or tracing the source of the attack using traceback techniques [26, 15, 28, 2, 11]. Besides the reactive techniques discussed above, some systems take proactive measures to discourage DoS activity, for example, both CenterTrack [25] and SOS [3] use overlay techniques with selective re-routing to prevent large flooding attacks. MULTOPS exploits the correlation of incoming and outgoing packet rates at different level of subnet prefix aggregation to identify attacks [30]. Wang provides a rigorous statistical model to detect abrupt changes in the number of TCP SYN packets as compared to the TCP SYN ACK packets [14]. All the above techniques are based on anomaly-detection which is faster than static signature-scan techniques used by Snort [21]. Snort has one main disadvantage; new attacks that do not have well-define signatures may go undetected until the signature is defined. Response to an attack consists of localizing the attackers and reducing the intensity of the attack. The SPIE systems can traceback individual packets within a domain using packet digests [2]. On the other hand, Burch and Cheswick propose a technique to traceback to the Client puzzles [4, 24, 18, 1, 10, 31, 17, 19, 20, 16, 35, 34] have been proposed as a mechanism for controlling undesirable network communication like virus and worm attacks [12, 29, 5]. This approach forces each client to solve a cryptographic puzzle thus imposing a formidable computational challenge to the attackers who aim to slow down the server by generating legitimate service requests and using up the servers resources. Such a mechanism gives devices the ability to selectively push back load to the source of an attack when overloaded. To apply a binary filter to traffic for preventing undesirable communication is difficult due to impact of false positives and the inability to differentiate good traffic from bad. Client puzzles provide an analog control against traffic that may potentially be deleterious and also limit an attacker's ability to send bad traffic to multiple victims concurrently by consuming their computational resources. To reduce the intensity of an attack, Mahajan proposed an aggregate congestion control and pushback technique to identify and throttle the attack flows [23]. Pushback is a cooperative technique that allows routers to block an aggregate upstream. Our approach combines filtering and admission challenges in a pushback scheme.

3. PROPOSED WORK

Please use a 9-point Times Roman font, or other Roman font with serifs, as close as possible in appearance to Times Roman in which these guidelines have been set. The goal is to have a 9-point text, as you see here. Please use sans-serif or non-proportional fonts only for special purposes, such as distinguishing source code text. If Times Roman is not available, try the font named Computer Modern Roman. On a Macintosh, use the font named Times. Right margins should be justified, not ragged. A. Client Puzzle

There are several important goals that must be achieved in order for client puzzles to be deployed effectively at the network layer. These goals include:

Lightweight: Puzzle creation and solution verification should be inexpensive and low on computation task on part of the server so that receiving a flood of requests will not exhaust its computational resources.

Adjustable: In order to provide graceful degradation of services, the server should be able to increase and decrease the computational resources required of the client as the server's load increases and decreases.

Universal Deployment: The protocol must be sufficiently flexible to support universal puzzle issuance at arbitrary points in the network. Furthermore it should be solvable on most types of client hardware.

Cheat Proof: First, it should not be possible for a client to cheat by pre-computing the solutions to puzzles. Puzzle answers should not be valid indefinitely and should not be usable by other clients. Stateless: The puzzle should also be stateless, so the server does not have to store the solution or any other information about the client while it is solving the puzzle. Otherwise, a flood of requests may exhaust the server's memory by filling it up with stored state information.

Reusable: The same puzzle may be given to several clients. Knowing the solution of one or more clients does not help a new client in solving the puzzle.

Minimal application impact: The use of the puzzle protocol should not break latency-sensitive applications such as interactive voice, streaming video and networked games. Clients who are able and willing to solve puzzles should be able to run all of their applications seamlessly.

4. PUZZLE PROTOCOL

The protocol starts with the client issuing a client nonce (Nc) to the server. The client nonce contains client specific information which helps the server to distinguish it from other clients. Client nonces also prevent a server from continually issuing puzzles indefinitely to a client that is no longer requesting service. Similar to client nonce even a server maintains a server nonce. Server nonces are kept secret and are used to efficiently verify answers. On receiving the client nonce the server generates a puzzle and an answer to it. It also calculates the hash of the corresponding answer and stores it accordingly in the server nonce table and then returns the client nonce, puzzle and hash to the client. The advantage of generating the hash is that it allows the server to discard the otherwise memory consuming information about the respective clients. After receiving the client nonce the client checks its nonce table to verify the validity of the nonce. If valid it goes ahead and solves the puzzle and presents the answer along with the hash to the server. The server uses the hash to generate a corresponding hash from the server nonce table. If it matches, the correct answer has been given and the server accepts the packet.

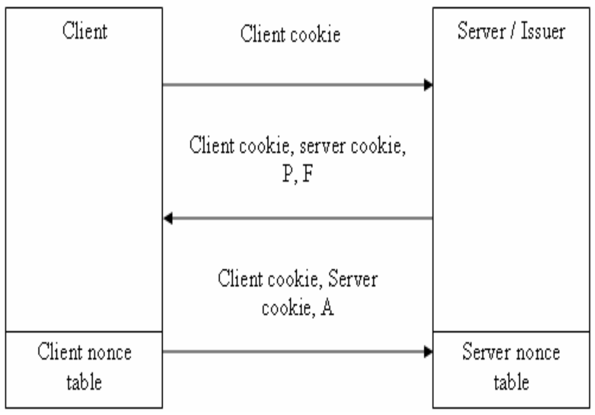

Fig 2: Full Client Puzzle Protocol

Some of the puzzles are, Time-lock puzzle [36], Hash reversal puzzle [6], multiple hash reversal puzzle [37], Hint Based hash reversal puzzle [38]. We here present a new way to use puzzle to mitigate spoofed DDoS attack. At first the client puzzles are first implemented centrally at the target. If the attack persists or worsens, then the target could propagate a distress signal upstream to its Internet Service Provider (ISP), who could deploy puzzles at the ingress points to the ISP's network on behalf of the target. In general, the target's ISP could request other upstream ISPs to also deploy the defenses for the target, so that the attack traffic is blocked as close as possible to the source of the traffic. The distress signal would be propagated using pushback techniques as discussed in the nextsection.

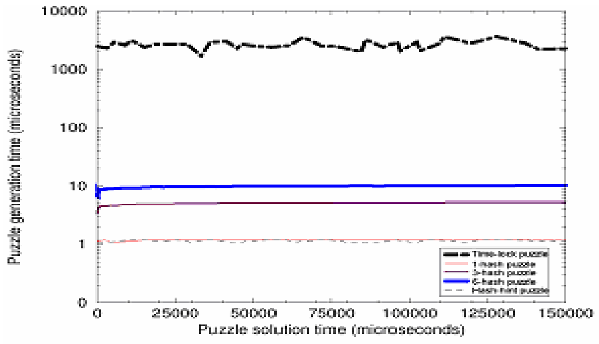

Fig 3: Comparison of various Client Puzzle Protocols

In this paper we are using the "Hint based hash reversal puzzle". As figure 3 shows that the generation time for time-lock puzzles is several orders of magnitude greater than that of the hint based hash-reversal puzzle hence it's our obvious choice. Its basic approach is to provide the client with a hash reversal puzzle and a hint which would give the client a chance to solve the puzzle. The difficulty level of the puzzle can be increased or decreased by suitably varying the hint. To generate a puzzle with difficulty level f(D) the server passes the client a hash and a hint, x- u(0,D) where x is a randomly generated number used as input to hash and u(0,D) is a randomly chosen number located between 0 and D. The client then uses the hint to search the range to get the answer. The number of hashes done by the client to find x varies probabilistically but the expected value is D/2. The creation of puzzle is outsourced to a secure entity we call a bastion. An arbitrary number of servers or routers can use the same bastion, and can safely share the same set of puzzles. Once constructed, the puzzles will be digitally signed by the bastion so that they can be redistributed by anyone. The client can solve the puzzles off-line, so that users don't have to wait for puzzles to be solved. Solving a puzzle gives a client access, for a time interval, to a channel on the server (i.e.) to a small slice of the servers' resources and the server ensures no virtual channel uses more than its fair share of available resources. The client must present their solution with the server cookie which was attached to the puzzle. To verify correctness, the server uses the timestamp to index into the nonce table and obtains the corresponding nonce, performs a hash of the client solution with the nonce and checks to see if it matches the echoed server cookie. C. Pushback Scheme

A network-based solution, Pushback, tries to solve the problem of spoofed DDoS attacks from within the network using the congestion level between different routers. When a link's congestion level reaches a certain threshold, the sending router starts dropping packets and tries to identify illegitimate traffic by counting the number of times packets are dropped for a certain destination IP address, since the attacker constantly changes the source IP address. The router then sends a pushback message to the routers connecting it to other congested links, asking them to limit the traffic arriving to this destination. To illustrate Pushback, consider the network in Figure 4. The server D is under attack; the routers Rn are the last few routers by which traffic reaches D. The thick lines show links through which attack traffic is flowing; the thin lines show links with no bad traffic.

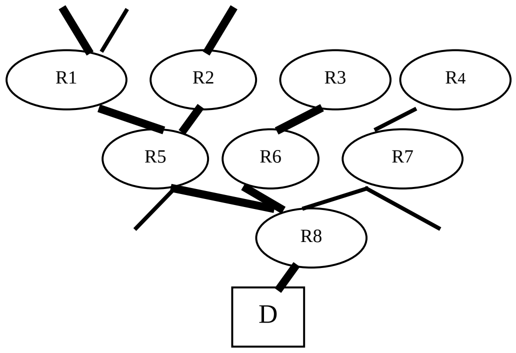

Fig 4: DDoS Attack in Progress

Only the last link is actually congested, as the inner part of the network is adequately provisioned. Without any special measure, hardly any non-attack traffic can reach the destination. Some of the non-attack traffic is flowing through the links between R2-R5, R3-R6, R5-R8, R6-R8, and from R8 to D, but most of it is dropped due to congestion in R8-D.

With Pushback, R8 sends messages to R5 and R6 telling them to rate-limit traffic for D. Even though the links downstream from R5 and R6 are not congested, when packets arrive at R8 they are going to be dropped anyway, so they may as well be dropped at R5 and R6. These two routers, in turn, propagate the request up to R1, R2, and R3, telling them to rate-limit the bad traffic, allowing some of the 'poor' traffic, and more of the good traffic, to flow through.

5. AGGREGATE DETECTION

When the attack persists or worsens, then the target tries to inform the upstream routers to block the traffic using pushback. The first step towards this is to detect and create an aggregate set (congestion signature). We present such an algorithm here. We start by considering the drop set, that is, the set of packets that are dropped by the target. A drop set should be exhaustive such that malicious packets don't slip away. On the other hand it must not also be resource intensive i.e. it shouldn't use a lot of resources on the server's part. Only the packets which most frequently satisfy the definition of "malicious" are included in the drop set. The important feature is that the algorithm should run in less time that it takes to collect the packets. In order to obtain the drop set it starts by deciding whether the congestion level is high enough, that is, the drop rate is high enough. If it is seen that the traffic on a particular input link (Wi) exceeds a certain threshold value of the traffic on the output link (Wo) say Wi> 1.5Wo then the algorithm checks each of the dropped packets for malicious signatures. The dropped packets can be compared according to their eventual destination addresses. The packets with the highest count are included in the drop set.Apart from the information of the characteristics of the packets dropped by the router it must also include information to distinguish between normal traffic and attack traffic which can be obtained by applying the algorithms as discussed in [30,6,14,31]. After including all the information regarding the dropped packets and characteristics of the traffic it gives us the drop set for a particular router. As the pushback signal is propagated the size of the drop set keeps increasing to make it more exhaustive and precise.

6. IMPLEMENTATION OF PUSHBACK

Once the router has identified the drop set(congestion signature), the next step is to communicate that information to its upstream links. The messages exchanged by routers implementing Pushback are described in detail in [23]. There are three such messages: request, response, and status. The pushback request is shown in Figure 5.

Various header fields
RLS-ID
Maximum depth
Depth of Requesting
Bandwidth Limit
Expiration Time
Congestion Signature

. **Fig 5: Pushback Request**

The header fields contain many fields like type of pushback (PType), type of feedback (SRMode) etc. Each request has a Rate-Limiting Session Identifier (RLS-ID), which is used to match responses to requests. If we were to consider the whole network as a tree with the originator of the pushback request as the root and all other routers as the child nodes then the maximum depth signifies the last level of the tree till which the pushback request will be propagated. The depth of the originator is considered 0. With each upstream router the depth is incremented by 1. The maximum depth of propagation is set by the originating router and passed along by each subsequent router. Depth information is useful in setting timers for sending feedback. The bandwidth limit is expressed in bytes per second and defines an upper bound for the bandwidth to be provided to an aggregate in case of congestion. When the depth becomes 0 before reaching its eventual destination the pushback request is discarded. However the router marks the request and makes an entry into its table for future use. If the pushback refresh message arrives after the expiration time then that entry is deleted. Thus the expiration time is the time period after which the pushback request expires if no REFRESH messages arrive. The congestion signature includes specific information about the aggregates which are to be rate limited. The aggregates are identified by the algorithm as explained above. All those traffic which come under the purview of the aggregate definition are rate limited by the upstream routers. Each of these upstream routers employs their own aggregate detection algorithms in order to make the congestion signature more exhaustive. In this way the attack traffic is slowly rate limited near its source.

7. CONCLUSION AND FUTURE WORK

In this paper, we presented a defending technique against spoofed DDoS traffic. This technique intends to complement, rather than replace existing schemes. For instance, the integrated solution combines filtering and admission challenges with a pushback scheme between the target and the upstream ISPs. In our approach, we place the client puzzle mechanism centrally at the router at which the attack is taking place by which most of the spoofed packets are discarded. When the attacker strengthens its attack volume such that it surpasses the defending capacity of the router under attack then it sends a distress signal using the pushback technique to the upstream ISP's to employ proxy defense mechanisms on its behalf so as to limit the attack traffic right near the source. In this way as the algorithm keeps working it continuously builds up an attack database which will eventually mitigate the attack traffic to a great extent. A key advantage of this proposed approach is that it enables the defenders to harness greater computational resources in order to counteract the growth in attack power that is becoming available to attackers. Many open issues still need to be addressed, both in terms of research and management. The first issue is how to ensure that the pushback signal can be trusted, so that it is not open to manipulation by attackers. The problem of managing trust in a distributed environment is a challenging issue for research. The final issue is how to ensure the scalability of the pushback approach when it involves multiple ISPs and targets with many simultaneous attacks.

8. REFERENCES

[1] A. Juels and J. Brainard, "Client Puzzles: A Cryptographic Defense against Connection Depletion," in NDSS, 1999, pp. 151–165.

[2] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio Stephen T. Kent, and W. Timothy Strayer. Hash-based ip traceback. In Proceedings of the ACM SIGCOMM, pages 3–14, San Deigo

[3] Angelos D. Keromytis, Vishal. Misra, and Dan. Rubenstein. SOS: Secure Overlay Services. In Proceedings of ACM SIGCOMM 2002, August 2002.

[4] C. Dwork and M. Naor, "Pricing via Processing or Combatting Junk Mail," in Crypto, 1992.

[5] CERT, "CERT Advisory CA-2004-02 Email-borne Viruses," http://www.cert.org/advisories/CA-2004-02.html, 2004.

[6] Christos Papadopoulos, Robert Lindell, John Mehringer, Alefiya Hussain, and Ramesh Govindan. COSSACK: Coordinated Suppression of Simultaneous Attacks. In Proceeding of Discex III, Washington, DC, USC, April 2003.

[7] Cisco Systems. Netflow services and applications. http://www.cisco.com/warp/public/732/netflowCisco Systems. Rmon. http://www.cisco.com/warp/public/614/4.html

[8] Drew Dean, Matt Franklin, and Adam Stubblefield. An algebraic approach to IP traceback. In Proceedings of Network and Distributed Systems Security Symposium, San Diego, CA, February 2001.

[9] D. Dean and A. Stubblefield, "Using Client Puzzles to Protect TLS," in 10th Annual USENIX Security Symposium, 2001.

[10] Dawn X. Song and Adrian Perrig. Advanced and authenticated marking schemes for IP traceback. In Proceedings of the IEEE Infocom, Anchorage, Alaska, April 2001.

[11] D. Moore, C. Shannon, and J. Brown, "Code-Red: A Case Study on the Spread and Victims of an InternetWorm," in Internet Measurement Workshop, November 2002.

[12] Fu- Yuan Lee, Shiuhpyng shieh. "Defending against spoofed DDOS attack with path fingerprint"- www.elsevier.com/locate/cose

[13] Haining Wang, Danlu Zhang, and Kang Shin. Detecting SYN flooding attacks. In Proceedings of the IEEE Infocom, New York, NY, June 2002. IEEE.

[14] Hal Burch and Bill Cheswick. Tracing anonymous packets totheir approximate source. In Proceedings of the USENIX LISA, pages 319–327, New Orleans, USA, Decemeber 2000. USENIX.

[15] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A Distributed anonymous Information Storage and Retrieval System," Lecture Notes in Computer Science, vol. 2009, pp. 46+, 2001.

[16] J. Leiwo, T. Aura, and P. Nikander, "Towards Network Denial of Service Resistant Protocols," in SEC, 2000, pp. 301–310.

[17] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using Hard AI Problems for Security," in Eurocrypt 2003., 2003.

[18] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.Wallach, "Security for Peer-to-Peer Routing Overlays," in Proceedings of OSDI, December 2002

[19] M. Abadi, M. Burrows, M. Manasse, and T.Wobber, "Moderately Hard, Memory-bound Functions," 2003.

[20] Martin Roesch. Snort - lightweight intrusion detection for networks.http://www.snort.org/docs/lisapaper.txt

[21] Peter Reiher Jelena Mirkovic, Greg Prier. Attacking DDoS at the source. In Proceedings of the IEEE International Conference on Network Protocols, Paris, France, November 2002.

[22] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. In ACM Computer Communication Review, July 2001

[23] R. Merkle, "Secure Communications Over Insecure Channels," Communications of the ACM, vol. 21, no. 4, April 1978.

[24] Robert Stone. Centertrack: An IP overlay network for tracking DoS floods. In Proceedings of the USENIX Security Symposium, pages 199–212, Denver, CO, USA, July 2000. USENIX.

[25] [Steven Bellovin. ICMP traceback messages.IETF draft-bellovin-itrace-00.txt

[26] S. Crosby and D. Wallach, "Denial of Service via Algorithmic Complexity Attacks," in USENIX Security Symposium, August 2003.

[27] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for IP traceback. In Proceedings of the ACM SIGCOMM Conference, pages 295–306, Stockholm, Sweeden, August 2000. ACM.

[28] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time," in 11th USENIX Security Symposium (Security '02), 2002.

[29] Thomer M. Gil and Massimiliano Poletto. MULTOPS: A Data-Structure for bandwidth attack detection. In Proceedings of the USENIX Security Symposium, pages 23–38, Washington, DC, July 2001.

[30] Vern Paxson. Bro: A system for detecting network intruders in real-time. Computer Networks, 31(23–24):2435–2463, Decemeber 1999.

[31] Vern Paxson. Bro: A system for detecting network intruders in real-time. Computer Networks, 31(23–24):2435–2463, Decemeber 1999.