ISSN 1512-1232

A NEW APPROACH TO DEFEND AGAINST DDOS

First N.Kiruthika

Institute of Technology,Coimbatore,Tamilnadu,India.
kiruthika.me@gmail.com

Abstract

We propose a mechanism which combines the advantage of cryptographic client puzzle and hop count filtering as a countermeasure to spoofed DOS attack. Once the communication channel has been established then HCF starts its work wherein an Internet server can easily infer the hop-count information from the Time-to-Live (TTL) field of the IP header. Using a mapping between IP addresses and their hop-counts, the server can distinguish spoofed IP packets from legitimate ones. We propose a technique that permit the outsourcing of puzzles—their distribution via a robust external service that we call a bastion. Our outsourcings techniques help eliminate puzzle distribution as a point of compromise. Our design has three main advantages over prior approaches. First, it is more resistant to DoS attacks aimed at the puzzle mechanism. Second, it is cheap enough to apply at the IP level, though it also works at higher levels of the protocol stack. Third, it allows clients to solve puzzles online, reducing the need for users to wait while their computers solve puzzles. We implement and evaluate this technique in the Linux kernel, demonstrating its effectiveness with experimental measurements.

***Keywords*:** DOS, HCF,IP,TTL

INTRODUCTION

The threat posed by network denial of service attacks is a growing concern as these types of attacks continue to successfully disable well-known Internet websites. In February 2000, a string of distributed denial of service attacks crippled popular websites including CNN.com, Amazon.com, eBay.com, and yahoo.com for several hours. Another large distributed denial of service attack in October 2002 took out nine of the thirteen root DNS servers. In 2003, for example, one honey pot research project saw 15,164 unique zombies form a large botnet within days. In 2004, the Witty worm created 12,000 zombies within 45 minutes.

IP spoofing has often been exploited by Distributed Denial of Service (DDoS) attacks to: 1) conceal flooding sources and dilute localities in flooding traffic, and 2) coax legitimate hosts into becoming reflectors, redirecting and amplifying flooding traffic. IP spoofing is commonly associated with malicious network activities, such as Distributed Denial of Service (DDoS) attacks, which block legitimate access by either exhausting victim servers' resources [1] or saturating stub networks' access links to the Internet [5]. Most DDoS attacking tools spoof IP addresses by randomizing the 32-bit source-address field in the IP header, which conceals attacking sources and dilutes localities in attacking traffic. The recent "backscatter" study, which quantifies DoS activities in the current Internet, has confirmed the widespread use of randomness in spoofing IP addresses. Moreover, some known DDoS attacks, such as smurf and more recent Distributed Reflection Denial of Service (DRDoS) attacks [5], are not possible without IP spoofing. Such attacks masquerade the source IP address of each spoofed packet with the victim's IP address. Overall, DDoS attacks with IP spoofing are much more difficult to defend.

To thwart DDoS attacks, researchers have taken two distinct approaches: *router-based* and

ISSN 1512-1232

host-based. The router-based approach installs defense mechanisms inside IP routers to trace the source(s) of attack, or detect and block attacking traffic. However, these router- based solutions require not only router support, but also coordination among different routers and networks, and wide-spread deployment to reach their potential. In contrast to the router-based approach, the host-based approach can be deployed immediately. Moreover, end systems should have a much stronger incentive to deploy defense mechanisms than network service providers.

The current host-based approaches protect an Internet server either by using sophisticated resource-management schemes or by significantly reducing the resource consumption of each request to withstand the flooding traffic such as SYN cookies and Client Puzzle. Without a mechanism to detect and discard spoofed IP traffic at the very beginning of network processing, spoofed packets will share the same resource principals and code paths as legitimate requests. Under heavy attacks, current approaches are unlikely to be able to sustain service availability due to resource depletion caused by spoofed IP packets. Furthermore, most of existing host-based solutions work at the transport-layer and above, and cannot prevent the victim server from consuming CPU resource in servicing interrupts from spoofed IP traffic. At high speed, incoming IP packets generate many interrupts and can drastically slow down the victim server. Therefore, the ability to detect and filter spoofed packets at the IP layer without any router support is essential to protection against DDoS attacks. Since filtering spoofed IP packets is orthogonal to the resource-protection mechanisms at higher layers, it can be used in conjunction with advanced resource-protection schemes.

HOP COUNT FILTERING

The fundamental idea is to utilize inherent network information—that each packet carries and an attacker cannot easily forge—to distinguish spoofed packets from legitimate ones. The inherent network information we use here is the number of hops a packet takes to reach its destination: although an attacker can forge any field in the IP header, he cannot falsify the number of hops an IP packet takes to reach its destination, which is solely determined by the Internet routing infrastructure. The hop-count information is indirectly reflected in the Time-to-Live (TTL) field of the IP header, since each intermediate router decrements the TTL value by one before forwarding a packet to the next hop.

Based on hop-count, *Hop-Count Filtering*, weed out spoofed IP packets at the very beginning of network processing, thus effectively protecting victim servers' resources from abuse. The rationale behind hop-count filtering (HCF) is that most randomly spoofed IP packets, when arriving at victims, do not carry hop-count values that are consistent with the IP addresses being spoofed. As a receiver, an Internet server can infer the hop-count information and check for consistency of source IP addresses. Exploiting this observation, HCF builds an accurate IP-to-hop count (IP2HC) mapping table, while using a moderate amount of storage, by clustering address prefixes based on hop-count. To capture hop-count changes under dynamic network conditions, we also devise a safe update procedure for the IP2HC mapping table that prevents pollution by attackers. The same pollution-proof method is used for both initializing IP2HC mapping table and inserting additional IP addresses into the table.

To minimize collateral damage, HCF has two running states, *learning* and *filtering*. Under normal conditions, HCF stays in the *learning* state, watching for abnormal TTL behaviors without discarding any packets. Even if a legitimate packet is incorrectly identified as spoofed, it will not be dropped. Therefore, there is *no* collateral damage in the *learning* state. Upon detection of an attack, HCF switches to the *filtering* state, in which HCF discards those IP packets with mismatching hop-counts.

ISSN 1512-1232

CLIENT PUZZLE

A good puzzle should have the following properties:

1. Creating a puzzle and verifying the solution is inexpensive for the server.
2. The cost of solving the puzzle is easy to adjust from zero to impossible.
3. The puzzle can be solved on most types of client hardware (although it may take longer with slow hardware).
4. It is not possible to precompute solutions to the puzzles.
5. While the client is solving the puzzle, the server does not need to store the solution or other client-specific data.
6. 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.
7. A client can reuse a puzzle by creating several instances of it.

New Client Puzzles:

We present a new way to use puzzles to mitigate denial-of-service attacks. It has three main attributes:

• The creation of puzzles is outsourced to a secure entity we call a bastion. An arbitrary number of servers can use the same bastion, and can safely share the same set of puzzles, due to special cryptographic properties of the puzzles. Once constructed, the puzzles will be digitally signed by the bastion so that they can be redistributed by anyone.

• Verifying a puzzle solution requires very little work for a server. In fact, it only requires a simple table lookup.

• Clients can solve puzzles off-line, so that users do not have to wait for puzzles to be solved.

• Solving a puzzle gives a client access, for a time interval, to a "virtual channel" on the server—i.e., to a small slice of the server's resources—and the server ensures no virtual channel uses more than its fair share of available resources.

Our approach is more coarse-grained in that it relies on virtual channels, which can be used as an abstraction to protect different types of resources. For example, a web server might limit the number of open TCP connections per channel or a database server could control the rate of database queries processed. When at high risk of DoS attack (or in the midst of an attack) a host in our system accepts communication only via a restricted collection of channels. To contact a host through one of these channels, a client must provide a valid token. A token consists of the solution to the client puzzle associated with a particular channel and time interval. A client can easily attach tokens to every packet it transmits. The host can enumerate in advance the set of valid tokens, so the host can verify tokens and filter channel traffic very efficiently. An adversary with limited computational resources can successfully attack only a limited number of channels, and the remaining channels will be available to support normal communications from benign clients. We note that multiple clients can use the same channel for communication. The primary purpose of channels is to segregate adversary requests from user requests. We justify the use of tokens by the following observation. In typical DoS attacks an attacker commandeers a cohort of "zombie" machines on the edge of the network, but generally does not compromise routers in the middle of the network. Based on this observation, we consider an attack model that assumes only limited eavesdropping by the adversary.

As we have explained, puzzle-based DoS solutions provide a newly attractive DoS target: the point of distribution of puzzles. To address this problem, we propose a novel approach to client-puzzle distribution. We show how to outsource puzzle distribution to a robust independent web service such as a highly distributed content-serving network like Akamai or a well-protected set of core servers like the root DNS system. We refer to this service as a bastion. A bastion will serve as

ISSN 1512-1232

a leverage point, reducing the robustness requirements needed to defend a server against DoS.

Properties:

Our preferred "D-H" construction, which is based on the Diffie-Hellman problem, has two important properties that allow it to avoid the shortcomings of previous client puzzle systems.

The first is that a server's puzzle solutions are made from a combination of the server's public key and the solution to a puzzle posed by the bastion. When publishing a puzzle, the bastion does not need to know which servers will use that puzzle. Since servers can effectively share puzzle challenges, only a constant number of puzzles need to be published for each time interval, and these puzzles can be distributed and replicated widely. This property, along with the ability to quickly check token solutions, insulates the puzzle distribution mechanism from attack.

The second property is that when a client solves a puzzle for a particular channel, the solution can be used at any server. The solution for a particular channel is combined with the public key of a server to produce a token solution specialized for that server. This means the client machine can compute solutions ahead of time and adapt them on the fly to whatever servers the user chooses to contact. The user will then experience no extra delay once he decides to go to a site.

We show our methods to be both theoretically sound and practical to implement using existing Internet protocols (with the addition of new client-side and server-side components). Our method also maintains compatibility for unmodified clients, although their traffic does not receive the benefit of DoS resistance.

IMPLEMENTATION

HOP-COUNT COMPUTATION

Since hop-count information is not directly stored in the IP header, one has to compute it based on the final TTL value. TTL is an 8-bit field in the IP header, originally introduced to specify the maximum lifetime of each packet in the Internet. Each intermediate router decrements the TTL value of an in-transit IP packet by one before forwarding it to the next-hop. The final TTL value when a packet reaches its destination is, therefore, the initial TTL decreased by the number of intermediate hops (or simply hop-count). The challenge in hop-count computation is that a destination only sees the final TTL value.

CONSTRUCTION OF IP2HC MAPPING TABLE

Building an accurate IP2HC mapping table is critical to detect the maximum number of spoofed IP packets. Our objectives in building a table are: (1) accurate IP2HC mapping, (2)up-to-date IP2HC mapping, and (3) moderate storage requirement. By clustering address prefixes based on hop-counts, we can build accurate IP2HC mapping tables and maximize the effectiveness of HCF without storing the hop-count for each IPaddress.

Ideally, the IP2HC mapping table has one entry for each valid IP address. However, this will consume a very large amount of memory, and it is unlikely that an Internet server will receive legitimate requests from all live IP addresses in the Internet. By aggregating IP address, we can reduce the space requirement of IP2HC mapping significantly. More importantly, with IP address aggregation, it is sufficient to capture the hop-count value of one IP address from each subnet in order to build a complete HCF mapping table.

ISSN 1512-1232

CAPTURING LEGITIMATE HOP-COUNT VALUES

To maintain an accurate IP2HC mapping table, we capture val7id hop-count mappings and legitimate changes in hopcount, while foiling any attempt to slowly pollute the mapping table. We can accomplish this through TCP connection establishment. The IP2HC mapping table should be updated only by packets belonging to TCP connections in the established state. The three-way TCP handshake for connection setup requires the active-open party to send an ACK to acknowledge the passive party's initial sequence number. The automaton (or flooding source2) that sends the SYN packet with a spoofed IP address will not receive the victim's SYN/ACK packet and thus cannot complete the three-way handshake. Using packets from established TCP connections ensures that an attacker cannot slowly pollute a table by spoofing source IP addresses.

IMPLEMENTATION OF INSPECTION AND VALIDATION ALGORITHM

The inspection algorithm extracts the source IP address and the final TTL value from each IP packet. The algorithm infers the initial TTL value and subtracts the final TTL value from it to obtain the hop-count. The source IP address serves as the index into the table to retrieve the correct hop-count for this IP address. If the computed hop-count matches the stored hop-count, the packet has been "authenticated"; otherwise, the packet is classified as spoofed. Note that a spoofed IP address may happen to have the same hop-count as the one from a zombie to the victim. In this case, HCF will not be able to identify the spoofed packet. However, even with a limited range of hop-count values, HCF is highly effective in identifying spoofed IP addresses.

```
for each packet:
  extract the final TTL Tf
          and the source IP address S;
  infer the initial TTL Ti;
  compute the hop-count Hc=Ti-Tf;
  index S to get the stored hop-count Hs;
  if (Hc != Hs)
      the packet is spoofed;
  else
      the packet is legitimate;
```

Fig. 1 Hop-count inspection algorithm.

PROTOCOL DESIGN - FULL PUZZLE PROTOCOL

The client puzzle protocol is developed to resist when DOS attacks arises in the Grid networks. In this module we are constructing a new client puzzle protocol between Grid client and server, Grid resource and grid server. While the puzzle protocol facilitates the efficient deployment of puzzles at the network layer, the puzzles themselves must be appropriately designed for use with our protocol.

PUZZLE GENERATION

Puzzle generation provides functionality of creating puzzle which supports constant-state operation at the client, resource and server . The only state required is a set of randomly-generated, periodically updated client nonces (Nc) and server nonces (Ns). In order to get the client to solve a puzzle, a server must echo a client nonce correctly, thus preventing spoofing attacks from third

ISSN 1512-1232

parties that are not along the path of communication.

Client nonces also prevent a server from continually issuing puzzles indefinitely to a client that is no longer requesting service. Server nonces are kept secret and are used to efficiently verify answers. Since attacks on pseudo-random number generators are possible, both client and server nonces should be generated using a "true" random number generator

ANSWER VERIFICATION

The denial-of-service-resistant authentication protocol proceeds as follows: a client or resource sends a request to the server. The server responds by signing and sending the puzzle to Client or resource. Client or Resource generates a random N_C, signs and sends its solution to the server.

The server verifies that Client or Resource has not already submitted a correct solution using N_S (the server's random nonce) and N_C Client random nonce), and if not, verifies the solution by computing h(C, NS, NC, X). If the solution is correct (the first k bits of the hash are zeros), the server stores (C, NS, NC) and now may commit computational resources toward verifying the signature.

EVALUATION OF FILTERING ACCURACY

We build a table based on the set of client IP addresses at each web server and evaluate the filtering accuracy under each aggregation method. We assume that the attacker generates packets by randomly selecting source IP addresses among legitimate clients. We further assume that the attacker knows the general hop-count distribution for each web server and uses it to randomly generate a hop-count for each spoofed packet. This is the most effective DDoS attack that an attacker can launch without learning the exact IP2HC mapping.

We use the percentages of false positives and false negatives to measure filtering accuracy. False positives are those legitimate client IP addresses that are incorrectly identified as spoofed. False negatives are spoofed IP addresses that go undetected by HCF.

Use either SI (MKS) or CGS as primary units. (SI units are strongly encouraged.) English units may be used as secondary units (in parentheses). **This applies to papers in data storage.** For example, write "15 Gb/cm^2 (100 Gb/in^2)." An exception is when English units are used as identifiers in trade, such as "3½ in disk drive." Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity in an equation.

The SI unit for magnetic field strength *H* is A/m. However, if you wish to use units of T, either refer to magnetic flux density *B* or magnetic field strength symbolized as $\mu_0 H$. Use the center dot to separate compound units, e.g., "A·m^2."

EXPERIMENTS AND RESULTS

The initial collection period should be long enough to ensure good filtering accuracy even at the very beginning, and the duration should depend on the amount of daily traffic the server is receiving. HCF will continue adding new entries to the mapping table when requests with previously unseen legitimate IP addresses are sighted. Thus, over time, the IP2HC mapping table will capture the correct mapping between IP address and hop-count for all clients of a server. This ensures that spoofed IP traffic can be detected, and then discarded with little collateral damage during a DDoS attack.

From an experiment for different Hop Count values, the mapping table is activated and

ISSN 1512-1232

observed for the false positive and false negative prediction percentage. This is shown in a graph.

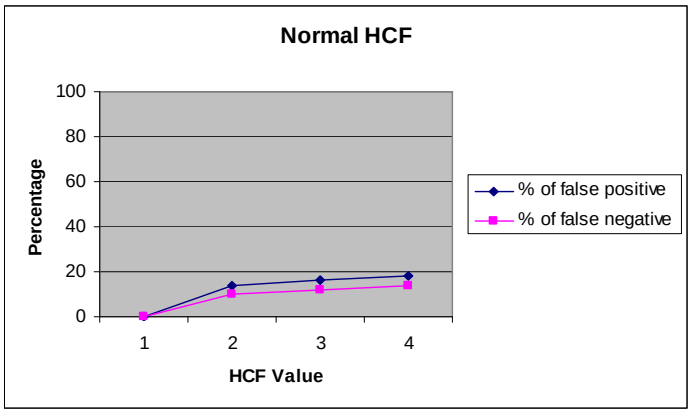

Fig 2. Percentage of false positives and false negatives in HCF only

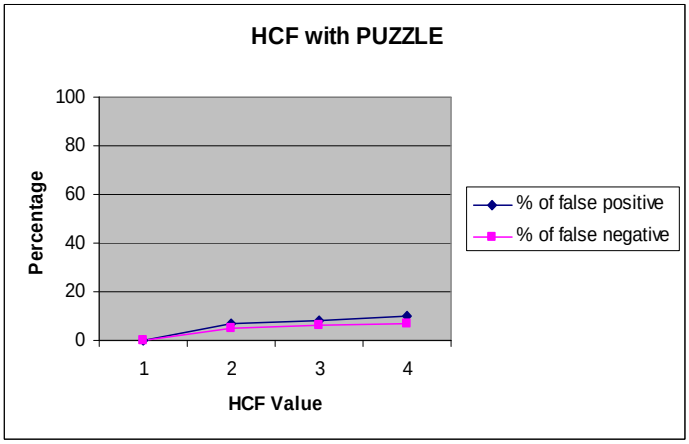

Fig 3. Percentage of false positives and false negatives in HCF with puzzle

Another important factor to be considered is the time complexity i.e. the time taken by the technique to categorize a client as an attacker or a valid client. The time taken for HCF combined with client puzzle is more than normal HCF. The server has to verify the client identity by resolving the encrypted puzzle and only when it proves to be same allow the client to establish connection. So the time taken by this procedure is comparatively more than hop count filtering. This technique serves to stop the client at the beginning of connection establishment thereby reducing the number of attackers at the beginning. The graph shown previously proves that the HCF with client puzzle reduces the percentage of false positive and false negatives. Thus the advantage of this technique proves to be more which makes this disadvantage to be eliminated.

The time chart figure 5.4 shows the time taken of one filtering time for one client to destination through three routers.

ISSN 1512-1232

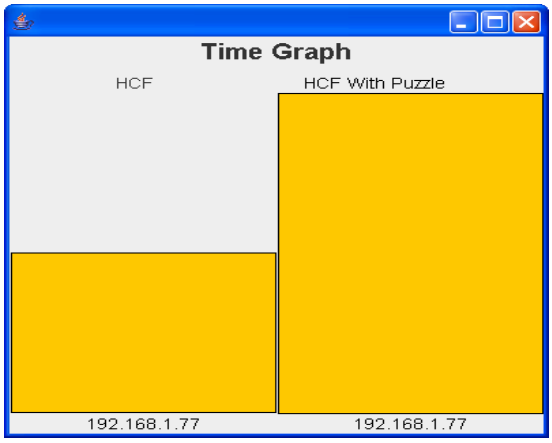

Fig 4. Time chart for one IP address

CONCLUSION

We have examined the problem of defending a server against Denial-of-Service attacks using a new technique based on client puzzles and hop count filtering. The system inspects the hop-count of incoming packets to validate their legitimacy. Using only a moderate amount of storage, HCF constructs an accurate IP2HC mapping table via IP address aggregation and hop-count clustering. A pollution-proof mechanism initializes and updates entries in the mapping table. By default, HCF stays in the *learning* state, monitoring abnormal IP2HC mapping behaviors without discarding any packet. Once spoofed DDoS traffic is detected, HCF switches to the *filtering* state and discards most of the spoofed packets. We observe that since puzzle distribution itself can be subject to attack, any viable system must have a robust puzzle distribution mechanism. We developed a new model for puzzle distribution using a robust service that we call a bastion. The bastion distributes puzzles.

ISSN 1512-1232

REFERENCES

1. TCP SYN flooding and IP spoofing. CERT Advisory CA-96.21, 2000 [Online]. Available: http://www.cert.org/advisories/CA-96-21.html
2. P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In Internet Measurement Workshop, 2002.
3. D. Dean, M. Franklin and A. Stubble.eld, "An algebraic approach to IP traceback," *Networks and Distributed System Security Symposium* (NDSS) 2001, pp. 3-12.
4. C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, CRYPTO '92, pages 139–147. Springer-Verlag, 1992.
5. S. Gibson, Distributed reflection denial of service Gibson Research Corp., Tech. Rep., Feb. 2002 [Online]. Available: http://grc.com/dos/drdos.htm
6. Hal Burch and Bill Cheswick, "Tracing anonymous packets to their approximate source," *14th Systems Administration Conference*, New Orleans, U.S.A., Dec. 2000, pp. 313-322.
7. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent and W. T. Strayer, "Hash-based IP traceback," *Proceedings of ACM SIGCOMM*, San Diego, California, U.S.A., Aug. 27-31, 2001, pp. 3-14.
8. Steven M. Bellovin, "ICMP Traceback messages," *Internet Draft*, March 2001.
9. H.Wang, C.Jin, and Kang G. Shin. "Defense against Spoofed IP Traffic Using Hop-Count Filtering" IEEE/ACM transactions on networking, vol. 15, no. 1, Feb. 2007
10. Waters, Ari Juels, J. Alex Halderman, and E. W. Felten, "New Client Puzzle Outsourcing Techniques for DoS Resistance" ACM *CCS'04.*
11. The Swiss Education and Research Network, Default TTL values in TCP/IP. 2002
12. [Online].Available:http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html.

Article received: 2010-12-21