International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 12 (2014), pp. 1167-1173
© International Research Publications House
http://www. irphouse.com

Vulnerability Analysis of Hash Tables to Sophisticated DDoS Attacks

D. Jessica Prathyusha

SREE VIDYANIKETHAN ENGINEERING COLLEGE TIRUPATHI

Abstract

In recent years, everybody experienced a flow of DDoS attacks threatening the welfare of the internet. These are launched by unauthorized users whose only motivation is to degrade the performance of other, innocent, users. The traditional systems turn out to be quite vulnerable to these attacks. The objective of this work is to aiming at laying a foundation that can be used in future computer/network designs taking into account the malicious users. This approach is based on proposing a metric that evaluates the vulnerability of a system.

Then use this vulnerability metric to evaluate a data structure which is commonly used in network mechanisms—the Hash table data structure and here presenting a new class of low-bandwidth denial of service attacks that exploit algorithmic deficiencies in many common application data structures. Then showing that Closed Hash is much more vulnerable to DDoS attacks than Open Hash. Due to this sophisticated DDOS attacks even after the attack has ended, the regular users still suffer from performance degradation or even a total denial of service.

Keywords Network Mechanism, Hash, metric, malicious, DDoS, vulnerability, low-bandwidth attack

1. Introduction

In recent years, the welfare of the Internet has been threatened by malicious Distributed Denial of Service (DDoS) attacks. A distributed denial-of-service (DDoS) attack is one in which a multitude of compromised systems attack a single target, 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 to the system to legitimate users. The basic form of a DDoS attack is the simple high bandwidth DDoS attack. That is, simple brute force flooding.

The first contribution in this work is a proposal of a new metric that evaluates the vulnerability of a system for any kind of sophisticated attacks. This approach is based on proposing a Vulnerability Metric[2] that accounts for the maximal performance degradation (damage) that sophisticated malicious users can inflict on the system using a specific amount of resources (cost) normalized by the performance degradation attributed to regular users, using the same resources.

The second contribution, presented in is an evaluation of the Hash data structure, commonly used in network mechanisms and presenting a new class of low-bandwidth denial of service attacks[3] that exploit algorithmic deficiencies in many common application data structures.

Finally shows that in the case of an attack either on an Open Hash system or on a Closed Hash system, the regular user still suffers from performance degradation or even a total denial of service, even after the attack has ended note that the degradation is not due to the extra load of the attack but due to its sophistication.

1. The Vulnerability Factor:

This work is based on the observation that the traditional performance analysis of computer systems and algorithms is not adequate to analyze the vulnerability of a system to sophisticated DDoS attack and a new metric is needed.

Such a vulnerability metric can help a system designer to

Choose between two equal systems:
Such universal vulnerability metric can be used to compare the vulnerability of two alternative systems.

Select the values of system operating variables:
The analysis (either analytically or empirically) of the vulnerability value can expose the contribution of the different system parameters to the system vulnerability and help the designer to balance tradeoffs between the system vulnerability and its efficiency.

Use vulnerable systems safely:
A vulnerability metric assessing the amount of damage that can be inflicted by malicious sophisticated users can help to use known vulnerable systems wisely. For example, consider the Bro intrusion detection systems that can drop up to 71 percent of the traffic during an attack.

This proposal for the definition of the Vulnerability Factor[1] of a system is to account for the maximal performance degradation (damage) that sophisticated malicious users can inflict on the system using a specific amount of resources (cost) normalized by the performance degradation attributed to regular users using the same amount of resources.

Formally defining of the Vulnerability Factor as follows:

Let the users Type parameter be equal to either regular users (R) or malicious attackers (Mst) with strategy st. using the plural terms since some of the attack types occur only in specific scenarios of multiple coordinated access of users to the system. Let the budget be the amount of resources that the users of users Type spend on producing the additional traffic to the system, i. e. , the cost of producing the attack is limited by the budget parameter.

The budget can be measured differently in the various models, e. g. , as the required bandwidth, or the number of required computers or as the required amount of CPU and so on. Let ΔPerf(users Type; budget) be the change in the performance of the system due to being subject to additional traffic added to the regular traffic of the system, where the traffic is generated by users from type users Type with resource budget. The performance can be measured by different means such as the CPU consumption, the delay experienced by a regular user, the number of users the system can handle, and so on.

We define the Effectiveness of a strategy st on the system as

$$E_{st}(budget = b) = \frac{\Delta Perf(M_{st}, b)}{\Delta Perf(R, b)}$$

and the Vulnerability Factor V of the system as

$$V(budget = b) = max_{st}\{E_{st}(b)\}$$

If the Vulnerability Factor of a system equals the effectiveness of a strategy st, then st is said to be an Optimal Malicious Users Strategy. Generally, there may be a system in which no strategy can be proved to be the optimal because the effectiveness of all other possible strategies cannot be evaluated or bounded. here, the most effective(known) malicious strategy is considered to be a lower bound on the vulnerability of the system.

2. Sophisticated Attacks On Hash Tables

Sophisticated low-bandwidth DDoS attacks use less traffic and increase their effectiveness by aiming at a weak point in the victim's system design, i. e. , the attacker sends traffic consisting of complicated requests to the system [1][2]. While it requires more sophistication and understanding of the attacked system, a low-bandwidth DDoS attack has three major advantages in comparison to a high-bandwidth attack:

1) Lower cost—since it uses less traffic
2) Smaller footprint—hence, it is harder to detect
3) Ability to hurt systems which are protected by flow control mechanisms.

2. 1 Algorithmic Complexity Attacks Against Hash Tables

In a hash data structure, an item is hashed through a hash function which produces a hash output. The output is then stored in the hash bucket, corresponding to the output modulo the number of buckets in the hash table. Items that hash into the same bucket form a linked list in that hash bucket[3][8]. In order to retrieve a stored item, the server first computes the hash function to find the correct bucket, and then traverses through the list in the corresponding hash bucket to locate the item.

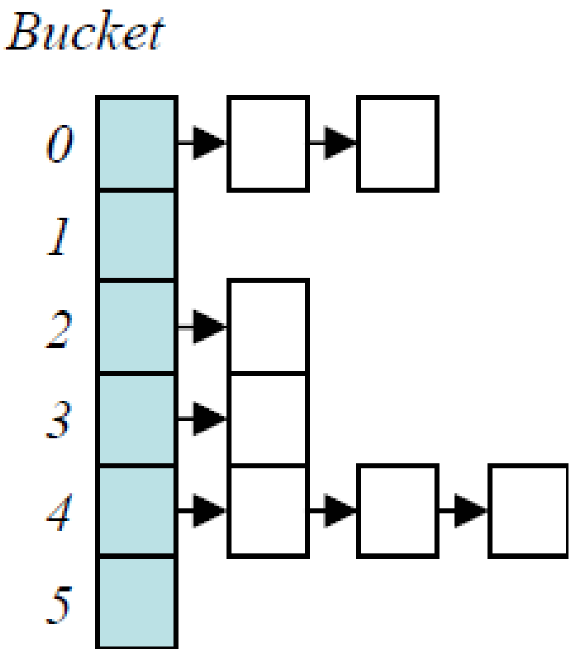


Figure 1: Normal operation of a hash table.

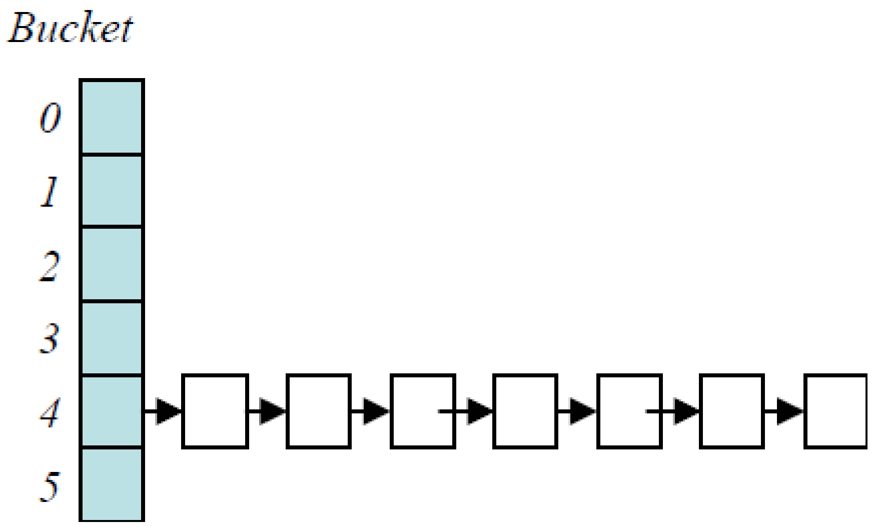


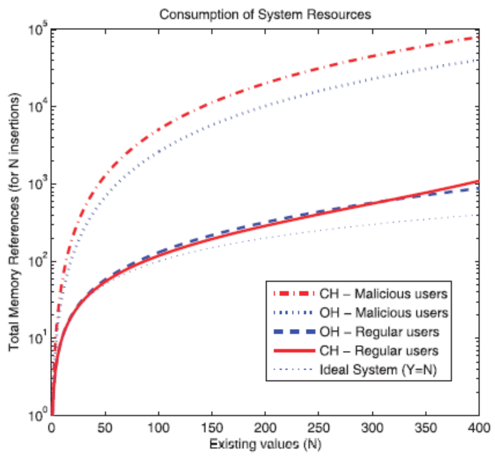
Figure 2: Worst-case hash table collisions.

A properly implemented hash function will distribute its inputs evenly throughout the array, creating very short lists in the buckets, so that retrieving a stored item will perform in an *O*(1) average lookup time. However, if an adversary knows the details of the hash function, can control its input, and if the number of buckets is small enough, then he/she can produce inputs that all collide into the same hash bucket. In the worst case scenario one will have to traverse a list of all the items stored, resulting in the same lookup time complexity as that of a regular linked list, *O*(*n*), assuming *n* elements were hashed.

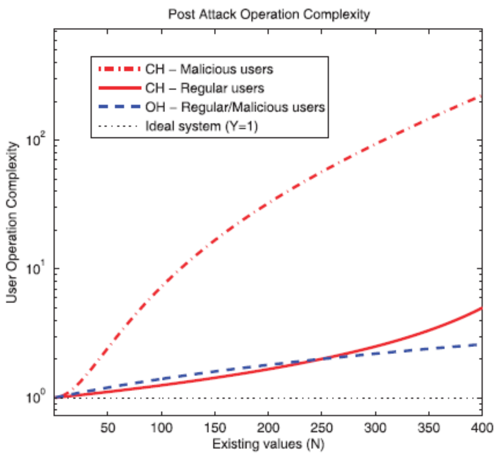
2. 2. Model

Consider a Hash system consisting of M buckets and N elements, where the Hash function is known to the attackers. In this model, the resource budget of the users is measured by the number of Hash operations (k) they can perform, where Hash operations can be either Insert, Delete, or Search. In our analysis, we will use three metrics:

1) **In-Attack Resource Consumption:** The number of memory references the k operations require during attack time.
2) **Post attack Operation Complexity:** The complexity of performing an arbitrary Hash operation after the attack has ended, i. e. , after these k Hash operations are completed.
3) **Post attack Waiting Time:** The waiting time of a regular user while performing an arbitrary Hash operation after the attack has ended, i. e. , after the k Hash operations are completed (Post attack Waiting Time).



(a)



(b)

Fig 3: The performance degradation due to k operations by regular versus malicious users, in Open and Closed Hash under the metrics: (a) In-Attack Resource Consumption (b) Postattack Operation Complexity. The curves are log scaled and as a function of the number of existing elementsN.

4. Application limits on hash tables

Many applications are sensitive about their overall memory usage, and thus have limits designed to control how large their hash tables might grow. If a hash table can never have enough elements in it for the worst-case $O(n2)$ behavior to dominate, then our attack will fail.

4. 1 Explicit limits

Some applications have explicit limits on their hash tables[2]. We first consider the Linux IP fragment reassembly code. In response to earlier attacks, Linux currently allows at most 256 kbytes of storage toward reassembling incomplete packets. If we wish to attack the hash table being used to store these packet fragments, the longest hash chain we can induce will still be under 256 kbytes in total.

4. 2 Implicit limits

The freedom of an attacker to construct arbitrary inputs may be limited. In the case of network packets intended to attack a network sniffer, the attacker is limited both by the packet fields being watched by the sniffer, and by the packet headers necessary to route the packet toward the targeted machine.

5. CONCLUSION

This report proposed such a Vulnerability Factor that measures the relative effect of a malicious user on the system. Using the Vulnerability measure, this report made interesting observations regarding the vulnerability of Hash systems and presented algorithmic complexity attacks, a new class of low-bandwidth denial of service attacks. Algorithmic complexity attacks against hash table, in particular, count on the attacker having sufficient freedom over the space of possible inputs to find a sufficient number of hash collisions to induce worst-case behavior in an application’s hash table. After an attack has ended, regular users still suffer from performance degradation. This performance degradation may reach the level of a total denial of service this report can calculate the exact magnitude of attack that can cause it.

6. REFERENCES

[1] U. Ben-Porat, A. Bremler-Barr, and H. Levy, “Evaluating the Vulnerability of Network Mechanisms to Sophisticated DDoS Attacks, ” Proc. IEEE INFOCOM, Apr. 2008.

[2] U. Ben-Porat, A. Bremler-Barr, H. Levy, "Vulnerability of Network Mechanisms to Sophisticated DDoS Attacks, " IEEE Transactions on Computers, vol. 62, no. 5, pp. 1031-1043, May 2013.

[3] S. A. Crosby and D. S. Wallach, “Denial of Service via Algorithmic Complexity Attacks, ” Proc. USENIX Security Symp. , Aug. 2003.

[4] M. Guirguis, A. Bestavros, I. Matta, and Y. Zhang, “Reduction of Quality (RoQ) Attacks on Dynamic Load Balancers: Vulnerability Assessment and

Design Tradeoffs, ” Proc. IEEE INFOCOM, May 2007.

[5] T. Moscibroda and O. Mutlu, “Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems, ” Proc. USENIX Security Symp. , June 2007.

[6] TCP SYN Flooding and IP Spoofing Attacks, CERT, http:// www. cert. org/advisories/CA-1996-21. html, Sept. 1996.

[7] J. L. Carter and M. N. Wegman, “Universal Classes of Hash Functions, ” J. Computer and System Sciences, vol. 18, pp. 143-154, 1979.

[8] N. Bar-Yosef and A. Wool, “Remote Algorithmic Complexity Attacks against Randomized Hash Tables, ” master’s thesis, Tel- Aviv Univ. , Tel-Aviv, Israel, 2006.