NOVEL SOFTWARE PUZZLE SCHEME TO DE-AMPLIFY RESOURCE INFALTED DDOS ATTACKS

Prachi R. Sorte[1], Prof. Shilpa Chougule[2], Dr. Chelpa Lingam[3]
Pillai HOC College of Engineering & Technology, Rasayani, India
Email:prsorte@mes.ac.in[1], schougule@mes.ac.in[1], chelpa.lingam@mes.ac.in[3]

Abstract

In the world of cyber-security, Denial of service (DOS) and Distributed DOS (DDOS) are popular resource depletion attacks. A client puzzle is a famous defensive measure against DOS and DDOS attacks. It forces a client to execute exclusive calculations before being allowed to access services from a server. But, an attacker may expand its ability of DOS/DDOS attacks using quick puzzle resolving software and incorporated graphics processing unit (GPU) hardware. This will notably fade the use of client puzzles. A novel software puzzle unified with a time threshold can improve the client puzzle. This method prevents DOS/DDOS invaders from expanding their puzzle-solving abilities. The present client puzzle schemes issue their puzzle algorithm a prior. In novel software puzzle scheme, a server randomly creates a puzzle algorithm provided a client request is acknowledged at the server side. In addition to this, a time threshold is enforced on the clients to submit the puzzle solution. The server maintains a detailed study of timing patterns of the clients which can be used to block the malicious nodes. The puzzle technique is deployed in a way that an attacker is incapable to formulate an implementation to crack the puzzle beforehand. It also requires significant effort in transforming CPU software puzzle to its corresponding GPU form. The transformation cannot be prepared in real-time.

Keywords: Software Puzzle, Distributed denial of service attacks(DDOS), Time threshold, CPU-GPU differentiation.

I. INTRODUCTION

The key principles of network security are authentication, integrity and confidentiality. Yet, all of them are dependent on the availability of the internet service. Generally, the aim of DOS attacks is to shut down an active server and destruct its service accessibility. An attacker achieves this by consuming the victim's bandwidth or resource. Such an attack is feasible because the attacker often pays very little for requesting a service, most of the time only the cost of sending a network packet. In DOS attack, an attacker basically employs use of a single computer and single Internet connection to flood targeted system or resource. But in case of DDOS attack, an attacker uses several computers and Internet connections to flood the targeted resource. For instance, when the attacker floods a victim server with several requests, the server consumes its resources and denies services to the genuine clients. On January 16th, 2016, the website of Torrents held down the whole day due to being hit by a massive DDOS attack. As per the statement of the website administrators, the attackers launched the DDOS attack on its DNS servers. Both the official site proxies and the main domain remained inaccessible by users on January 16th [1].

A classic example of an attack directed to exhaust the server's memory is TCP-SYN flooding attack. A three-way handshake protocol is followed by a TCP connection. Initially, a client wishing to access server resources sends a SYN message to the sever; secondly, upon receiving the SYN message, a server acknowledges the client with a SYN-

ACK message; and finally, the TCP connection is established when a client sends back an acknowledgement message. Data communication starts once the last message is received. As soon as the server sends its SYN-ACK message, it reserves a slot on the memory for the TCP connection requested by the client. An attacker only needs to sends several concurrent SYN messages to consume the server's resources. On the other hand, no final ACK for those SYN messages would be received by server. The server will grant certain memory resources for every single connection being requested. The attacker leaves the connections incomplete and thus exhausts the server's memory resources by keeping half-open connections [2].

DOS and DDOS are operative if the attackers invest comparatively few resources than the victim server or possess more power than normal users. The attacker requires minor efforts in generating a request, but the server has to invest significant computational efforts in establishing HTTPS handshake. The serious consequences of DOS/DDOS attacks have headed emergence of several defensive mechanisms against DOS attacks. In this paper a countermeasure to DOS/DDOS is proposed. Let α represents the ratio of resource depletion by a client to resource depletion by a server. Raising the ratio α can act as countermeasure to DOS and DDOS i.e., increase the computational cost of the client or decline that of the server [3]. An eminent technique to raise the cost of clients is a client puzzle. A client puzzle enforces the clients to perform computational calculations before being able to access services. Normally, a client puzzle scheme comprises of three steps: puzzle creation, puzzle cracking by the client and validation of puzzle solution by the server [3].

The prevailing client puzzle schemes works on the assumption that the malicious client cracks the puzzle using only the CPU resources. But, this hypothesis is not always correct. Currently, the multiple-core GPU (Graphic Processing Unit) section is typical configuration in almost all modern desktops, laptops and even smartphones. An attacker can thus, simply exploit the "free" GPUs or unified CPU-GPU to expand his processing capacity. Therefore the current client puzzle techniques become unsuccessful due to the considerable decrease in the computational cost ratio α. For instance, an attacker may offload one puzzle-cracking job to various GPU cores if the client puzzle is parallelizable. If the puzzle function is non-parallelizable then the attacker might send multiple appeals concurrently and enforce every single GPU core to solve one received puzzle challenge separately [4]. This parallelism strategy can efficiently decrease the total puzzle-solving time, and henceforth enhances the attack productivity.

This paper presents a novel puzzle, called software puzzle unified with a time threshold which de-amplifies the effects of resource inflated DOS attacks. This technique takes advantage of the CPU-GPU architectural difference. The current client puzzle schemes issue a puzzle function a prior. The novel software puzzle scheme unified with a time threshold dynamically creates the puzzle function in the form of a software core when it receives a client's request. The proposed scheme arbitrarily chooses a set of simple CPU functions and binds them altogether into the puzzle core C. It then creates a software puzzle C0x with the puzzle core C and a random challenge x. To defeat attackers who are able to reverse-engineer software, C0x is encrypted to generate a better software puzzle [3]. When a client receives a software puzzle from the server, he tries to solve it on the host CPU. The client then answers to the server with a puzzle solution. However, an attacker may try to transfer the puzzle-solving burden onto its GPU. To achieve this, an attacker needs to translate the CPU software puzzle into its functionally equal GPU version. But GPU and CPU have totally dissimilar instruction sets designed for diverse applications. Since the software puzzle is formed dynamically and arbitrarily, the transformation cannot be done in advance. Translating a software puzzle may require more time as compared to solving the puzzle on the host CPU directly [3]. Also the server uses a time threshold for the submission of puzzle solution. Thus the novel software puzzle prevents the GPU-inflated DOS attacks.

The reminder of this paper is organized as follows. Section II gives an overview of related work. Section III gives an introduction to GPU. Section IV introduces the proposed system and Section V evaluates the performance of novel software puzzle through obtained results. Section VI draws conclusion.

II. RELATED WORK

Maintaining information is very difficult in today's modern era. Deprived of security

measures and controls, the data might be exposed to an attack. DOS attack is the most prevalent attack on internet. DOS attacks and DDOS attack aims to exhaust online service's resources. Meanwhile when several prominent websites like eBay and Amazon became victims of DOS attacks; scientists and engineers started focusing on it. Over the past years, numerous methods have been suggested to avoid exhaustion attacks on systems.

Dwork and Naor offered the exclusive concept of client puzzle in junk mail defense [5]. Here the sender was supposed to pay a suitable cost for each message. This cost can be counted as a hardware resource invested for calculating a cryptographic puzzle. This cost is negligible for genuine clients, but it turns out costly for an attacker who sends junk mails. Their idea is creative and valuable and it put forward a defending concept of currency-based mechanism. It enlightens the fact that before a service provider allocates any of its resources to client's requests, clients should always consume their own resource for authentication. But Dwork and Naor limited this applicability of cryptographic puzzles to mail system only.

In 1999 Juels and Brainard employed use of client puzzle in SYN flooding attack [6]. They presented a simple client puzzle protocol. When the system is not suspected under an attack, a defending server responses the clients' requests as it does normally. If the server is doubted under a DOS attack, a small cryptographic puzzle is given out to each client who is requesting for a service. A client who success in solving the puzzle with a correct solution in specific time is allowed access the resources. But, Juels and Brainard did not take DOS attacks against authentication protocol into consideration. Moreover, the puzzle they proposed still has some redundancies and needs to be improved further.

Aura and Nikander proposed a one-way hash function as a form of client puzzle in 2000 [7], wherein a defending server sends the puzzle's parameters to a client. Based upon these parameters the client performs a brute-force search for some bits of the inverse of a hash function. The difficulty level of this puzzle can be attuned by changing the puzzle's parameter. Once the server verifies the puzzle solution; it commits its resource to the client. To verify the solution of the puzzle, the defending server has to perform similar cryptographic hash function as the clients do. It is disadvantage of this puzzle. An attacker can indulge the server consume significant resources for verification by sending numerous arbitrary values as his solutions. Hence, this puzzle mechanism redirects the process of puzzle verification to additional possible DOS target.

Waters [8] proposed a client puzzle method where puzzle building and delivery are positioned on to a secure unit called bastion. The bastion distributes puzzles periodically for a particular range of virtual channels. This remains effective for the duration of next time slot. Puzzle building is quite costly as it involves a modular exponentiation, but several servers can trust on puzzles provided by the similar bastion. A client resolves a puzzle by using brute force testing which is a parallelizable task. Verifying a puzzle at the server side includes a table scan and invests additional cost in modular exponentiation, which, can be achieved in advance during the earlier time slot.

III. GPU INTRODUCTION

Modern GPUs have various processing cores that can be used for general-purpose computing as well as graphics processing. GPU assists an attacker to launch GPU inflated DOS attacks. The architectural differentiation between CPU and GPU can be used to defend against the GPU-inflated DOS attack.

A. Difference between CPU and GPU

Modern CPUs are masters in the execution of single-thread programs. Current GPU executes extreme data-parallel programs. Initially GPUs were used for interpreting only graphics. But as advancement in technology came huge number of cores in GPUs relative to CPUs was misused. This made GPUs computationally capable for processing many parallel streams of data simultaneously. In other words, CPUs and GPUs vary in their architectural structures which make them applicable in different tasks. A GPU can process huge volumes of data in several streams, performing comparatively simpler operations on them, but is unsuitable for heavy or complex processing on a single or few streams of data.

As a consequence, GPUs are unsuited to handle tasks that are not parallelized, comprising many common consumer applications such as word processors. Moreover, GPUs uses a basically different architecture. A developer has

to program an application explicitly for a GPU and considerably different techniques are necessary to program GPUs. A CPU processor is comparatively slower than a GPU processor as a whole; on the other hand one CPU core is comparatively faster than one GPU core. All GPU cores utilize shared resources including the registers and caches whereas one CPU controls its resources such as memory and cache. If a GPU kernel uses its multiple shared resources, the number of cores used in the application would be much smaller than the available cores. Hence the power of GPU would not be fully utilized. In this situation, GPU may be slower than CPU. The proposed scheme exploits the above differentiation between CPU and GPU to thwart GPU from being used to accelerate the puzzle-solving process.

B. GPU inflated DOS attack

A client sends a request to the server whenever he/she wants to acquire a service. Upon receiving the client's request, the server responds with a puzzle challenge x. The legitimate client will calculate the puzzle solution y directly on the host CPU, and send the response (x, y) back to the server. However, as shown in Fig 1, a malicious user who controls the host will send the challenge x to GPU and exploit the GPU resource to accelerate the puzzle-solving process. This mechanism assists in accelerating calculation with GPU.

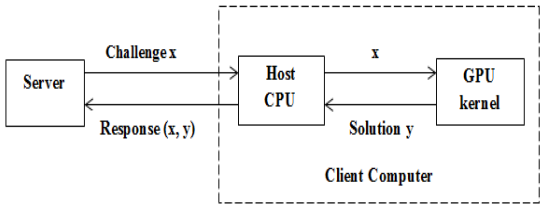

Fig.1: GPU-inflated DOS attack against data puzzle [3]

IV. PROPOSED SYSTEM

The prevailing client puzzle method assumes that the malicious client resolves the puzzle utilizing its CPU resource merely as shown in Fig. Though, this supposition is not always right. Currently, the many-core GPU (Graphic Processing Unit) module is a default configuration in modern desktop computers, laptop computers and even smartphones. Consequently, an attacker can simply make use of "free" GPUs or combined CPU-GPU to expand his computational capability. This makes the current client puzzle method ineffective due to the considerably decreased cost ratio γ which denotes the ratio of resource depletion by a client and a server.

A fresh effective puzzle scheme called software puzzle is introduced to avoid DOS/DDOS attackers from expanding their puzzle-resolving skills. This puzzle also includes a time threshold which gives a server the details of puzzle execution times by each client. When a client request is received at the server side, a software puzzle is constructed arbitrarily. The algorithm is created in such a fashion that an attacker is incapable to make an implementation to resolve the puzzle a prior. Thus an attacker requires major effort in transforming a CPU-based unit puzzle into its corresponding GPU form. This transformation cannot be completed in real time [3].

The novel scheme consists of four modules as shown in Fig. 2. 1) generating the puzzle C0x randomly and dynamically on client request. 2) binding the puzzle with CPU based instruction. 3) encrypting the puzzle C0x for high security to puzzle C1x. 4) employing a time threshold on the client for the submission of puzzle solution.

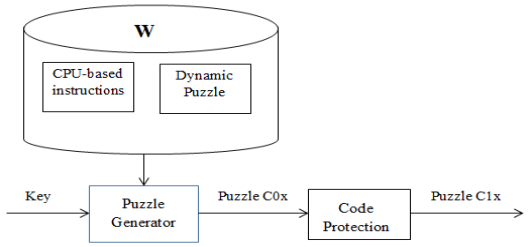

Fig. 2: Novel Software Puzzle Construction

As shown in Fig. 2 to defeat against GPU inflated DOS attacks, the proposed scheme use an improved version of client puzzle called as novel software puzzle. This scheme on receiving a client's request chooses CPU based instructions from a storage warehouse **W**. With this CPU-only instruction, it binds a dynamically and randomly created puzzle C0x. For security it further encrypts the puzzle C0x to produce C1x.This encrypted puzzle is then sent to the client. On receiving the encrypted puzzle, the genuine client attempts to resolve the puzzle on host CPU machine and submit the puzzle solution to the server. On deploying the puzzle C1x to the clients, server maintains a timer and

enforces a threshold for the submission of puzzle solution. The server uses this timer to analyze and further block or unblock the client nodes.

A. *CPU-specific instructions block*

Graphics Processing Unit (GPU) is intended to perform predictable graphic processing operations such as matrix operations, non-basic logic processing. Branching operations such as try-catch-finally, goto are non-predictable and are non-parallelable. Hence executing these operations on GPU is time consuming. This creates a disadvantage to an attacker as he/she cannot utilize the major importance of GPU. Second, GPU cannot execute human-interface and network-interface instructions such as reading hardware input and surfing network. Third, GPUs do not support dynamic thread generation. Fourth, GPU thread blocks shares the high-speed memory all together and hence the magnitude of fast accessible memory available to each thread is small. Therefore, the GPU parallelism potential will be restricted seriously if the puzzle demands large shared memory. All the GPU threads have to access the global memory at a considerable slower speed. Therefore, the difference between GPU and CPU instructions can be exploited to design the software puzzle components. Table I lists some of these instructions.

TABLE I

EXAMPLE CPU-ONLY INSTRUCTIONS

Instruction	Difference Exploited
1.Read Local cookie	GPU cannot directly read CPU storage
2.Allocate large memory	GPU has smaller memory than CPU
3.Try-catch	GPU does not support exception handling
4.Goto (address)	GPU does not support branch
5.Network interface	GPU cannot support networking function
6.Create new class	GPU does not support dynamic code
7.Create new thread	GPU does not support child thread

B. *Software Puzzle Construction*

The server requires three modules for the construction of software puzzle: puzzle core construction, puzzle challenge construction, software puzzle encrypting.

1) Puzzle Core Construction: The server initially selects CPU code blocks from the code warehouse. The selected code block is assembled together into a puzzle core known as C(•).

2) Puzzle Construction: The server creates a puzzle randomly and stores the client's public data such as IP addresses, hostname, timestamp etc. The puzzle is binded together with the puzzle core.

3) Code Security: It involves further encrypting the code and creating more confusion.

4) Time Threshold: A time threshold is used which allows a server to examine the nodes. This information can be further used by server to block the malicious nodes.

V. RESULTS

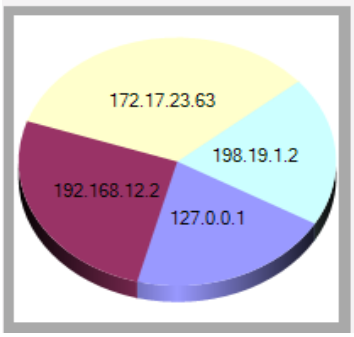

Fig. 3: Graph of IPaddress V/S Time

The pie diagram in Fig. 3 shows a graph of IPaddress versus time of various clients. This data is analyzed by a server to block or unblock nodes.

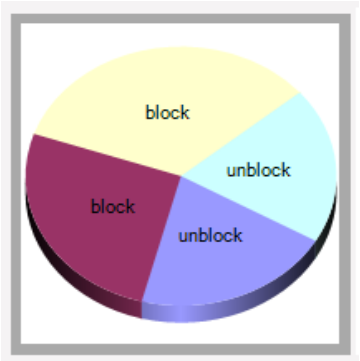

Fig. 4: Graph of Status V/S Time

The pie diagram in Fig. 4 shows a graph of Node status versus time of various clients. Using the time threshold, the server blocks and unblocks the nodes. Thus novel software puzzle scheme defeats the resource inflated DDOS attacks and increases access of resources to genuine clients.

VI. CONCLUSION

The proposed system is successfully designed to counteract resource inflated DDOS attacks. In this implemented system, novel software puzzle methodology setback GPU-inflated DOS attack to some extent. It agrees on a software protection technology to safeguard data secrecy and code security for an appropriate time span, e.g., 1-2 seconds. The software puzzle can be built upon a data puzzle. Thus any current server-side data puzzle system can be integrated with the software puzzle. Hence the novel software puzzle scheme can be easily used as the existing client puzzle schemes. This scheme uses the time threshold which allows the server to grant services only to the clients which submits the solution in the given timespan. Also the server maintains a databank of client's information and timespan required by each client to solve the puzzle respectively. Using this information the server blocks and unblocks the nodes so as to increase the access bandwidth for other genuine clients. Thus this proposed system presents a countermeasure to resource inflated DDOS attacks.

REFERENCES

[1] Uzair Amir. (Aug. 8, 2016). *Kickass Torrents The Latest Victims of DDOS attacks.* [Online]. Available:https://www.hackread.com/kickass-torrents-the-latest-victim-of-ddos-attacks/

[2] Vicky Laurens, Abdulmotaleb El Saddik and Amiya Naik, "Requirements for Client Puzzles to Defeat the Denial of Service and Distributed Denial of Service Attacks," University of Ottawa, Canada, International Arab Journal of Information Technology, vol. 3,no. 4, pp. 326-333, October 2006

[3] 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.

[4] R. Shankesi, O. Fatemieh, and C. A. Gunter, "Resource inflation threats to denial of service countermeasures," Dept. Comput. Sci., UIUC, Champaign, IL, USA, Tech. Rep., Oct. 2010. [Online]. Available: http://hdl.handle.net/2142/17372

[5] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. Advances in crptology - Proc. CRYPTO '98, volume 740 of LNCS,pages 139–147, Santa Barbara, CA

[6] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. Proc. 1999 Network and Distributed Systems Security Symposium(NDSS), pages 151–165, San Diego,CA, Februrary 1999. Internet Society.

[7] T. Aura. DOS-resistant authentication with client puzzles. Security Protocols, 8th International Workshop Cambridge, UK, April 2000, Revised Papers, volume 2133 of Lecture Notes in Computer Science, pages 178-181. Springer-Verlag, Berlin, Germany, 2001.

[8] B. Waters, A. Juels, J.A. Halderman, and E.W. Felten. New Client Puzzle Outsourcing Techniques for DoS Resistance. In Proceedings of CCS'04 ,August 27, 2004.