Safeguarding Against DoS Attacks: A Brief Study on Software Puzzle

Manaswini Kolluru,
Dept.of Computer Science
REVA University,Bangalore, INDIA
Email: manaswinikolluru@gmail.com

Bijay Kumar Jyotishi
Dept.of Computer Science
Associate Professor REVA University
Bangalore, INDIA
Email: bijayj@revainstitution.org

Abstract- Denial of Service (DoS)/Distributed Denial of service is a type of cyber offensive attack where resources like network bandwidth, memory and computational power are attacked by rogue elements. In order to safeguard the server against such attacks a new type approach was proposed by Zhao [1]. This approach is called software puzzle approach. The software puzzle approach basically runs on CPU. In this paper we studied and implemented this approach. This approach is found to be effective as it creates computationally difficult task for the malicious attackers so that they cannot overwhelm the server.

I. INTRODUCTION

A DoS attack targets a widely used server or website and prevents or degrades its access to the genuine user who is
not able to send or receive response from server. The attacker consumes most or all of the resources of the computer and forces the operating system of the host to fill the connection tables with illegitimate entries. The attackers aim to exhaust the resources of the system that includes CPU cycles, memory,disk space and network bandwidth. The attackers generate too many requests which is feasible since they pay very little or nothing to request a service. Often their cost is only of sending the request on the network. In DDoS attack the effect is same. Only the number of attackers are more than one. In above context the attacker is generally not a physical person,but a malicious program which can send multiple requests to overwhelm the server/ website. To counter the DoS attack a cryptographic puzzle scheme is used where, a client is required to solve a cryptographic puzzle and submit the puzzle solution as proof of work before the server commits substantial resources to its request. The malicious client has to spend time in finding the solution. Each puzzle requires a number of cryptographic operations, such as hashing, modular multiplication,or modular exponentiation, to compute the puzzle solution. Thus, the more an attacker wants to overwhelm the server, the more puzzles it has to compute, consequently the more computational resources are consumed. This way the attacker is restricted and cannot easily bring down the server/website. For example, an attacker could send a maliciously crafted HTTP request to a vulnerable web server and attempt to leverage errors or other unexpected application behavior. One of the significant scheme in Puzzle generation is Hash
Reversal scheme which it brings Client Puzzle into an action.Client Puzzle runs on a GPU (Graphical Processing Unit). GPUs enable the client puzzle to spend less time and in solving the puzzle. This way attacker can have lot of chances in attacking the server. This will ultimately make the loss in business for e-marketing. Considering an example an attacker may put one puzzle which takes Thirty GPU cores and client puzzle function is parallelisable, or attacker may send the continuously to the server and intimate the GPU core to solve the puzzle. The Parallelism method can narrow down the Puzzle solving time, and the number of attacks may become more. In order to eradicate this, clients IP address is tracked. In [2] the authors proposed a web-based client puzzle method that works for easily understanding and deploying.

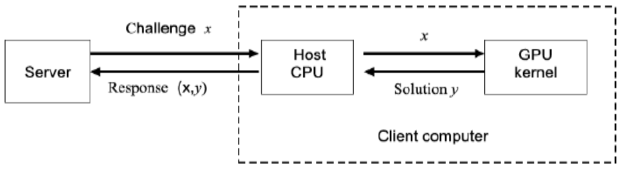

Fig. 1. GPU-inflated DoS attack against data puzzle.

II. GPU INTRODUCTION

A Graphics Processing Unit (GPU), is an electronic circuit which is specially designed to speed up the application particularly which has to perform high computation as a fast rate.These are being used by malicious programs to counter the puzzle approach to DDoS attacks.

GPU VS CPU PERFORMANCE: A CPU is having a single or small number of processors and generally used for serial processing of programs. Whereas, a GPU has a highly parallel architecture consisting of larger number of smaller processing units hence can perform the tasks quickly and is more suitable for parallel tasks

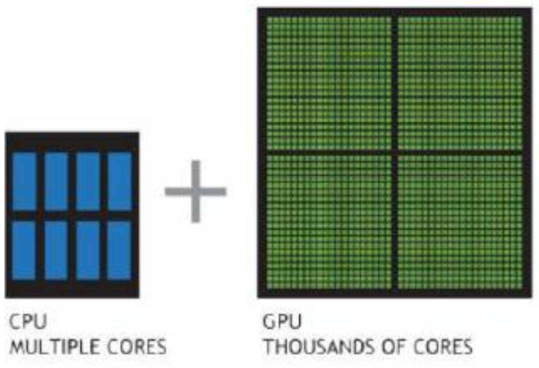

Fig. 2. Comparison of GPU and CPU

IV. Literature Survey

In [3], the authors, McNevin et.al. , give an approach for the novel client puzzle protocol that utilises a modification of the Extended Tiny Encryption Algorithm. An implementation of the client puzzle protocol was completed in the TCP stack of the Mandrake Linux 9.2 operating systems. They call this as modification to the TCP stack pTCP (for Puzzle TCP). This method is effective in the sense client puzzle protocol against various other resource depletion attacks within the transport layer, and helps in preventing the attacks in the application layer. In [4] the authors give a scheme to rank their vulnerability and reviewing a theoretical analysis of resource inflation attacks and carryout their application to a number of payment schemes. They find that the threat of Graphics Processing Units (GPUs)for inflation attacks is especially severe. They were able to demonstrate inflation of up to 630x with common
inexpensive GPUs. Later in 2011, in [5] a novel method was proposed to manage resources on a server using proof of work scheme on client puzzle, which provides a capacity to recover from DoS attack. Various defense mechanisms have been discussed to withstand these attacks. They have also demonstrated that, hash- reversal schemes which can be modified uniquely on server load and are very ineffective under attack by GPU utilizing adversaries; whereas, hash reversal schemes which adapt based on client behavior are effective even under GPU
based attacks. Subsequently, in [6] the authors proposed general mechanisms for safely and efficiently sandboxing software, such as dynamic language runtimes, that make use of advanced, low level techniques like runtime code modification. The language independent sandboxing builds on Software- based Fault Isolation (SFI), a traditionally static technique. They provide a more flexible form of SFI by adding new constraints and mechanisms that allow safety to be guaranteed despite runtime code modifications.

V. Design and Implementation

In this paper the following functionalities of the Software Puzzle has been implemented. The process begins with user sending a request to the server. The request is processed through the Middleware Host. The Function of a Middleware Host is to parse through the service request and divides the request into packets for easy assimilation by the server. The Middleware Host forwards the processed request to the server and creates log entry detailing the service request. Thereafter the server, before processing the request sends a puzzle which is deterministic or randomised. The puzzle after getting logged by the Middleware Host gets passed to the user/attacker.

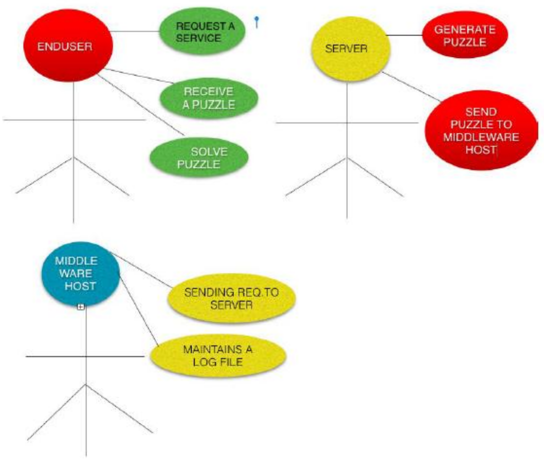

Fig. 3. UseCase Diagram

The attacker/user solves the puzzle and the solution gets recorded by the Middle ware host and passed on the server for the final redressal of the service request. In case the solution is correct the service is processed by the server. If not the Middleware Host tags the user as a Potential attacker. In this instance of a real DoS attack the user is most often a software generating multiple service requests in short span of time. The software also has to solve the puzzle in order to engage the server. The Goal of the attacker is to keep the server extremely busy so as to avoid the server from processing the honest user requests. In case the software knows the relationship between service request and the puzzle then the malicious software need only guess the puzzle solving function P. However the task of malicious software can be made more difficult and hence computational more intensive by randomising the relationship between the service request and puzzle. please note that this is not a one to one function, and hence there are multiple to a given value of N. We randomised the integer N to fit into the setting of the Software Puzzle.

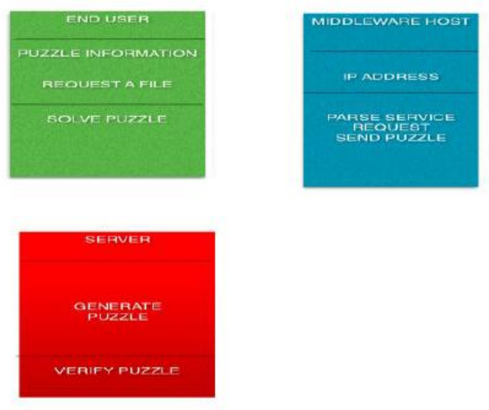

Fig. 4. Class Diagram

Puzzle Logic: Puzzle is formed and the output is sent to to end user and user is asked to provide the values of the puzzle coefficients. For example if puzzle is 1a+2b+3c and the answer is 14. The values a=1,b=2,c=3 can be guessed by the genuine user after few attempts, but it will be difficult for the malicious software. The class and sequence diagram explains the implementation diagrametically. Eclipse IDE has been used to program the Software Puzzle application using Java.The application has been tested in windows 7 version with 2GB RAM.

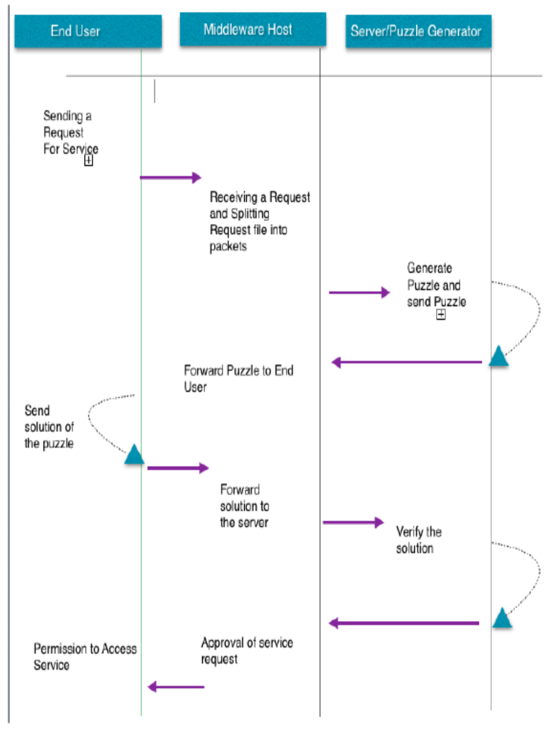
End User
Middleware Host
Server/Puzzle Generator
Sending a Request For Service
Receiving a Request and Splitting Request file into packets
Generate Puzzle and send Puzzle
Forward Puzzle to End User
Send solution of the puzzle
Forward solution to the server
Verify the solution
Permission to Access Service
Approval of service request

Fig. 5. Sequence Diagram

VI. EXPERIMENTAL RESULTS

A file service request is sent to the server through the user interface (Designed using Java Swings). The Server sends the puzzle and requests for the solution. A simple puzzle has been implemented which requests values of A, B, C (Fig 6). User provides the solution by giving the values of A, B, and C. For the Correct Solution the server intimates to the user of the success of the end user request through a user interface screen (Fig 9).

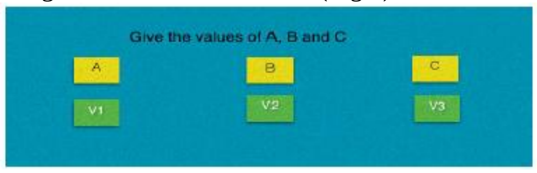

Fig. 6.

Fig. 7. Puzzle Solving Status

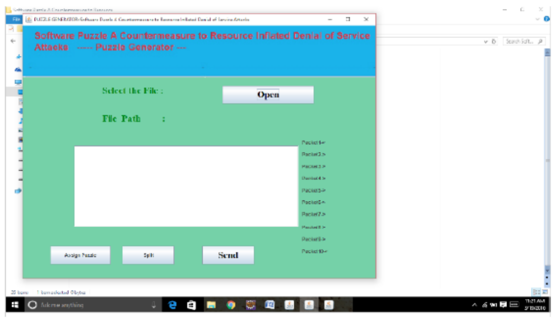
Software Puzzle A Countermeasure to Resource Inflated Denial of Service Attacks --- Puzzle Generator ---
Select the File :
Open
File Path :
Assign Puzzle
Split
Send

Fig. 8. Assigning puzzle

VII. CONCLUSION

DoS/ DDoS attack can put the server/website down causing business loss to the provider. Software Puzzle approach is an effective way to counter these attacks. In this paper we designed and tested Software puzzle approach to study its use and effectiveness. It is observed that before servicing the request the server necessitates the user to provide a solution to the software puzzle. The genuine user can guess the solution in few attempts, but it prevents a malicious user (generally program) to send large number of requests to server in short time and protects the server from malicious attacks. Further scope of work is to simulate DDoS attacks of large number of requests and using multiple nodes to send the genuine requests and measure the degradation in response time use the different puzzle schemes to find more effective puzzle schemes.

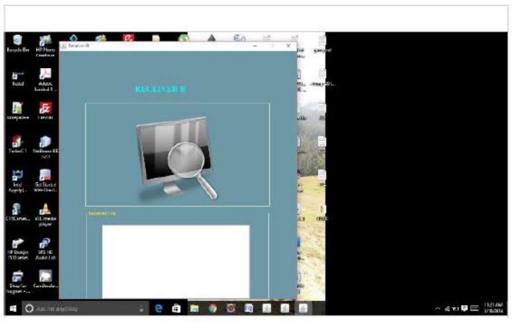

Fig. 9.

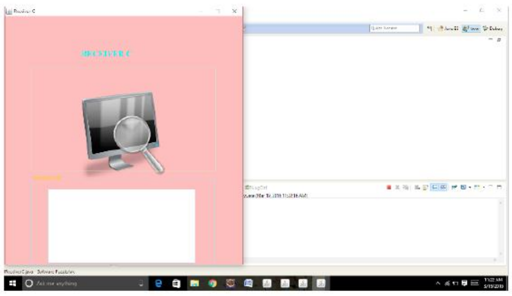

Fig. 10.

REFERENCES

[1] Yongdong Wu, Zhigang Zhao, Feng Bao and Robert H. Dang, “Software Puzzle: A countermeasure to resource inflated Denial of Service attack,” *IEEE Trans. Information Forensics and Security.*, Vol. 10, No.1, 2015

[2] E. Kaiser and W.-C. Feng, “Mitigating DoS with transparent proof of work: The case for public work*”.* In Proc. *IEEE Global Internet Symp.* pp. 43-48, 2007

[3] T. J. McNevin, J. M. Park and R. Marchany, “pTCP: A client puzzle protocol for defending against resource exhaustion denial of service attacks,” Technical Report, 2004.

[4] Ravinder Shankesi, Omed Fatemieh, Carl A. Gunter, “Resource inflation threats to denial of service counter measure,” Technical Report, 2010.

[5] Jeff Green, Joshua Juen, Omed Fatemieh, Ravinder Shankesi, Dong Jin and Carl A. Gunter, “Reconstructing hash reversal based proff of work schemes” In Proc. *4th Usenix Workshop on Large-Scale Exploit Emergent Threats*, 2011.

[6] Jason Ansel, Petr Marchenko, Ulfar Arlingsson, Elijah Taylor, Brad Chen, Derek L. Shuff, David Scher, Cliff L. Biffle and Bennet Yee, “Language independent sandboxing of just in time compilation and self modifying code,” In Proc. *ACM SIGPLAN Conf. Prog. Lang. Design Implement*, p. 355-366, 2011.