ISSN:0975-9646

An Efficient Framework to Handle DoS Attack by Using Software Puzzle and Decision Tree

Jyoti Shende[1], Asst.Prof. S. Todkari[2]

Computer Engineering Department,
JSCOE,
Pune, India.

Abstract— **Denial of Service (DoS) and Distributed Denial of Service (DoS) attacks are one of the major threats and among the hardest security problems in today's Internet. Which request a client to perform computationally very high operations before providing services from a server to client, is a well-known countermeasure to them. An attacker increase his puzzle solving capacity by using built-in graphics processing unit like (GPU) hardware to extremely weaken the effectiveness of client puzzles. In this paper, we study how to prohibit DoS/DDoS attackers from inflating their puzzle-solving capacities. To this end, we introduce a new client puzzle named as software puzzle. Unlike the existing client puzzle strategy, which publish their puzzle algorithms early and generate software puzzle for client request, a puzzle algorithm in the present software puzzle scheme is developed only after threshold value of client request exceed the CPU's request handling capacity by using technique like decision tree, fuzzy logic and generate algorithm such as: 1) an attacker is not able to prepare an implementation to solve the puzzle early and 2) the attacker needs extensive efforts required for translating a central processing unit puzzle software to its functionally identical GPU version.**

Keywords— **Software puzzles generation, information gain, GPU programming, distributed denial of service (DDoS).**

I. INTRODUCTION

Today Internet user's facing a very serious problem of Denial of Service (DoS) attacks and Distributed DoS (DDoS) attacks try to degrade the quality an online service's resources such as network bandwidth, memory, connectivity and computation power by outstanding the service with bogus requests. For example, a malicious client sends a large number of garbage requests to an HTTPS bank server. For making SSL handshake server has to spend lot of CPU time, it results insufficient resources remains to handle service requests from its customers, resulting in lost businesses reputation. These attacks achieve their aim by sending to a victim a stream of packets that swamps his network or processing scope denying access to his regular clients [2]. The client requires bulk of CPU time to make SSL handshake at the time of establishing connection with the server. It may result an insufficient resources are left to providing services. A DoS attack is recognized to take place only when access to a computer or network resource is intentionally blocked or reduced as a result of malicious action taken by another user. These attacks don't necessarily damage data directly or permanently, but they intentionally negotiation the accessibility of the resources.

In this paper, we are particularly interested in the overcome to DoS/DDoS attacks on server computation power. We are presenting the software puzzle which is overcome the limitations of existing systems like presenting the puzzle in advance. The overhead on the server is reduce by using middleware tool which generate and verify the puzzle, this reduce the server's CPU time and require to making SSL handshake. This framework overcomes the computational DoS attack by using decision tree. In this software puzzle is provide to the requested client on the basis of level identification of the client. On the basis of decision tree this method not only differentiates between attackers and legitimate clients but also types of attacks.

II. RELATED WORK

In this paper, introduce a new approach that client puzzle protocol, the goal is to fight against connection depletion attacks. When a server comes under attack, the server gives a unique client puzzle to each client wishing to make a connection with the server. The server resource allocated for a connection, the client must submit to itself for a connection, the client must give to the server an accurate solution to the puzzle it has been given by the server. This model is most robustness in stronger attack, able to handling attacks mounted at very high speed. Disadvantage is it requires special client side software and client early have a program capable of solving a client puzzle [1].

This paper, an idea of Proof of Work (PoW) scheme, clients prove that they have done work previously it prepare resources to their requests. Most PoW mechanisms are puzzle-based techniques in which clients solve processing thorough puzzles. This strategy can effectively restrict a resource scaling attacker's capabilities by adjusting puzzle difficulty based on past client behavior but as attacks use more resources, and the puzzle difficulties increases, weaker legitimate client may experience unnecessary requirements to obtain service[2].

This paper narrate the notion of timed release crypto where the aim is to encrypt a message so that it can't be decrypted by anyone, not even the sender, until a pre-arranged amount of time has passed. We study the problem of generating computational puzzles, called time-lock puzzles that require a predetermined amount of time to solve. Advantage is computational problems that can't be solved without running a computer constantly for at least a certain amount of time but demerit is the CPU time required to solve a problem can depend on the amount and nature of the hardware used and the parallelizability of the computational problem being solved [3].

This paper described an approach of mod_kaPoW system that has the human transparency and efficiency of proof-of-work strategy and also having the software backwards compatibility. There are disadvantages of using CAPTCHAs. A proof-of-work strategy modify the function

of a network protocol so that a client must rebound their challenge along with a right answer before being granted service. Merit of this paper system doesn't require special client software as well as in the system, a web server dynamically changes URLs. This technique has an overhead when processing files containing a variable number of URLs [4].

This paper introduces an idea of bread pudding protocol to be reworked by the verifier to achieve a separate, useful, and verifiable correct computation. In this paper, we depart from the standard cryptographic aim of proving knowledge of a secret, or the truth of a mathematical statement. This paper add bread pudding protocol to be a POW such that the computing effort provided in the proof may be harvested to accomplish a separate, useful and verifiably accurate computation. This techniques having advantage of achieve security goal and client pay for access to a resource by offering small amount of its computational power but demerit is highly computationally intensive operation of minting in the MicroMint strategy [5].

This paper gives idea of code obfuscation works by transforming an application so that the transformed program will be functionally identical to the original one but with much greater resistance to reverse engineering. A set of atomic operators for simple graph transformations that are guaranteed to preserve the functional behavior of the program and, hence, can be used as building blocks of a control-flow obfuscation algorithm. We propose a measure that we conjecture may be related to the robustness of the obfuscated program against reverse engineering [6].

III. PROPOSED METHODOLOGY

Here in this segment paper narrates about the methodologies that are incorporated in the experiment as depicted in the figure 1.

Step 1: Here in this step requests for a web transaction is received from all the clients by the web server along with the parameter like date, time and client IP to store in the database. Then all this data from the database will be retrieved in a vector for pre-processing, where selected data like IP is fetched in a single dimension vector for clustering process.

Step 2: Here Single dimension vector of IP addresses of the client that was fetched in the past step is been set to fuzzy-C Means clustering process. Fuzzy C means clustering (FCM) technique which eventually helps to analyse the patterns of the IP through interactive clustering. $\mu_{ij}=1/\sum_{k=1}^{c}(c_{ij}/c_{ik})^{(2/m-1)}$

where m is any real number whose value should be greater than 1, u_{ij} is the degree of membership of x_i in the cluster j, x_i is the i^{th} of d-dimensional measured data, c_j is the d-dimension centre of the cluster.

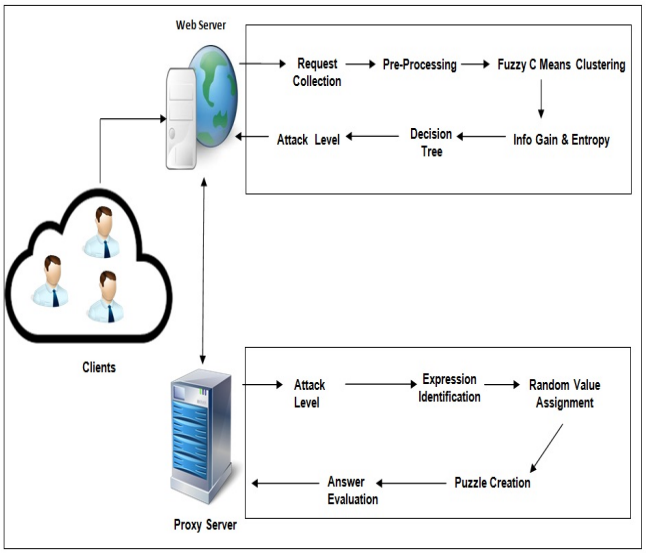


Fig. 1 Overview of System Architecture

Then the optimization of the clusters is carried out by the fact of fuzzy portioning which yields fine grained clusters which in turn indicates the abstract patterns of the input client IP.

ALGORITHM 1: FCM

Let $X = \{x_1, x_2, x_3 ..., x_n\}$ be the set of data points and $V = \{v_1, v_2, v_3 ..., v_c\}$ be the set of centers.
Step 0: Start
Step 1:Randomly select *'c'* cluster centers.
Step 3: Calculate the fuzzy membership $'\mu_{ij}'$ using:
$\mu_{ij}=1/\sum_{k=1}^{c}(c_{ij}/c_{ik})^{(2/m-1)}$
Step 4: Compute the fuzzy centers $'v_j'$ using:
$V_j = (\sum_{i=1}^{n}(\mu_{ij})^m X_i) / (\sum_{i=1}^{n}(\mu_{ij})^m)$,
for all j=1,2,…c
Step 5: Repeat step 2) and 3) until the minimum $'J'$ value is achieved or $||U^{(k+1)} - U^{(k)}|| < \beta$.
where, *'k'* is the iteration step. *'β'* is the termination criterion between [0, 1]. *'U = $(\mu_{ij})_{n*c}$'* is the fuzzy membership matrix. *'J'* is the objective function.

Step 6: Stop

Step 3: The clustered IP are then considered for their higher priority using the entropy distribution factor of Shannon information gain.
Here information gain is used to identify the most important and fluent IP address in the clusters which frequently affecting the web server for its performance. This can be given with the following equations 2.

$$IGR(C) = -\sum (|C_i| / |C|) \log (|C_i| / |C|) \quad ....(2)$$

Where C_i is the frequency of the IP address *add* in Cluster C.

Step 4: Decision trees are generally meant for the decision taking rules which indulge in putting conditions like if - else till to reach a decision. But here in our experiment of identifying DOS attack decision tree takes a two dimensional vector which is loaded with the attributes like IP address and their information gain values.

Here each of the indices of the vector is feed to the tree to form the nodes and at every levels of the tree with respect to the Shannon information gain values.

Then these values are keep accumulating the at the respective nodes to get the weighted decisions for judging attack level. Then this attack level is normalized in between the range 0 to 100 to get the desired level of software puzzle.

Step 5: This level of attack is been send to proxy server for puzzle generation process with a reference key generated through MD5 algorithm.

Once the proxy server receives the attack level it identifies the expression desired to tackle the attack in its raw form. Then the variables in the expression is set to change the variables by assigning random number from 1 to 9.

Once the expression is having the real numbers, then this is been evaluated using infix expression evaluation method as mentioned in the algorithm 2.

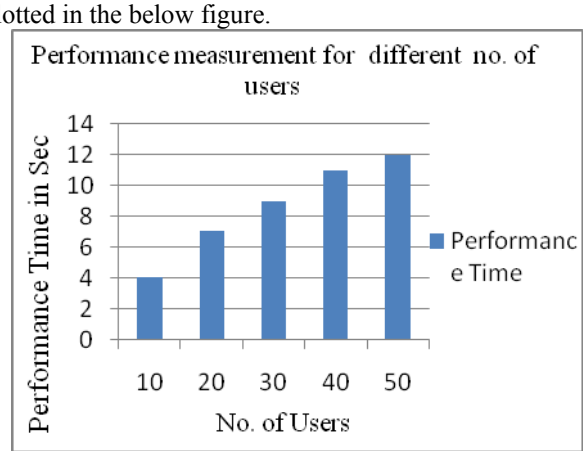
ALGORITHM 2: Infix expression evaluation algorithm

Step 0: Start
Step 1: If character exists to be read:
Step 2: if character is operand or (. push on the operand Stack
Step 3: else if character is operator
Step 3a :while the top of the operator Stack is not of smaller precedence than this character
Step 3b: pop op from operator Stack
Step 3c: pop two operands op1 and op2 from operand Stack
Step 3d:store op1 op op2 on the operand Stack back to 3a
Step 4:else if character is) [do the same as 3b – 3d till encounter (]
else // no more character left to read
Step 5: pop operators till operator stack is not empty
Step 6: pop top 2 operands and push op1 op op2 on the operand stack
Step 7: return the top value from operandStack
Step 8: Stop

IV. RESULT DISCUSSION

As most of the puzzle creation and puzzle generation systems are meant to be create the puzzle or solve the puzzle in independent manner. As the model does both the task of puzzle creation and evaluation on attack level identification for DOS attack in the network. For the experimental evaluation and some experiments are conducted on java based windows machine using Apache tomcat as the server.

To measure the performance of the system we set the bench mark on different number of users in the web application for railway ticket reservation system. And then we allow the number of users to seek the availability of the train between the source and destination places from the system. To evaluate the performance of the system WAPT 8.0 web load testing tool is used. And then experiment is plotted in the below figure.



The above plot expresses result of software puzzle creation time which is not directly proportional to number of users. So this indicates that the system over performs the software puzzling in matter of time.

And again when system is conducted for the performance time for puzzle solving task only and compared with other systems, obtained results are depicted in the below table 1.

Difficulty Level	Genetic Algorithm	Information Entropy Algorithm	Infix Evaulation Algorithm
Easy	3 Seconds	3 Milli seconds	3 Milli seconds
Medium	7 Seconds	4.6 Milli Seconds	4 Milli seconds
Hard	13 Seconds	4.66 Milli Seconds	4 Milli seconds
Evil	15 Seconds	5.99 Milli Seconds	5Milli seconds

Table 1: Comparison table with Different Algorithms

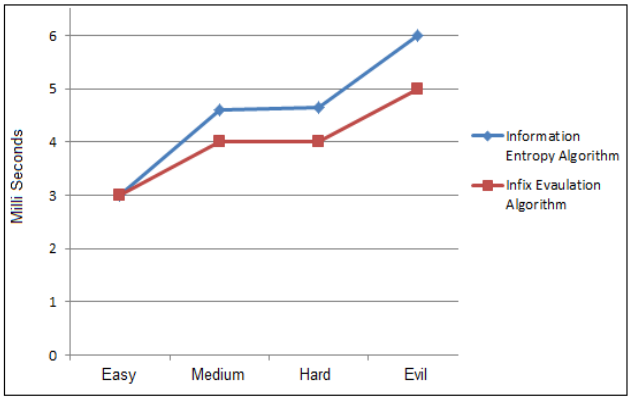


Figure 2: Comparison Plot with Information Entropy Algorithm

The above plot shows proposed system of Infix Evaluation algorithm over performs then of [11].

V. CONCLUSIONS AND FUTURE SCOPE

The proposed paper narrates about the incorporation of the DOS attack handling for any web applications through using of proxy server for puzzle generation instead of captcha or image number. Proposed system efficiently identifies the attack patterns using Fuzzy C means clustering and Shannon information gain for entropy estimation. Then system will powered with the decision trees to evaluate the attack levels to send to puzzle generator server. Puzzle generation will be done at the proxy server for the said level using infix expression evaluation technique for different levels from very easy to very hard ranges. This method of ours efficiently proves one of the best gateway for DOS attack counter measure technique using puzzle creation.

As the future scope of this paper system can be enhance to generate more complex puzzles for variable operands and operators.

ACKNOWLEDGMENT

The authors would like to thank the researchers as well as publishers for making their resources available and teachers for their guidance. We are thankful to the authorities of Savitribai Phule University of Pune . We also thank the college authorities for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

REFERENCES

[1] 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.

[2] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Comput. Netw.*, vol. 44, no.5, pp. 643–666, 2004.

[3] A. Juels and J. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in *Proc.* Netw. Distrib. Syst. Secur. Symp., 1999, pp. 151–165.

[4] R. L. Rivest, A. Shamir, and D. A. Wagner,"Time-lock puzzles and timed-release crypto," Dept. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. MIT/LCS/TR-684, Feb. 1996.

[5] Y. I. Jerschow and M. Mauve, "Non-parallelizable and non-interactive client puzzles from modular square roots," in *Proc. Int. Conf. Availability, Rel. Secur.*, Aug. 2011, pp. 135–142.

[6] J. Green, J. Juen, O. Fatemieh, R. Shankesi, "Reconstructing Hash Reversal based Proof of Work Schemes," D. Jin, and C. A. Gunter, in Proc. 4th USENIX Workshop Large-Scale Exploits Emergent Threats, 2011.

[7] E. Kaiser and W.-C. Feng, "mod_kaPoW: Mitigating DoS with transparent proof-of-work," in Proc. ACM CoNEXT Conf., 2007.

[8] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in Proc. IFIP TC6/TC11 Joint Working Conf. Secure Inf. Netw., Commun. Multimedia Secur., 1999.

[9] H.-Y. Tsai, Y.-L. Huang, and D. Wagner, "A graph approach to quantitative analysis of control-flow obfuscating transformations," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 2, pp. 257–267,Jun. 2009.

[10] 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.

[11] Gaoshou Zhai1 and Junhong Zhang ,] "Solving Sudoku Puzzles Based on Customized Information Entropy", International Journal of Hybrid Information Technology Vol. 6, No. 1, January, 2013

AUTHOR PROFILE

Jyoti B. Shende, is currently pursuing M.E (Computer) from Department of Computer Engineering, Jayawantrao Sawant College of Engineering, Pune, India. Savitribai Phule Pune University, Pune, Maharashtra, India - 411007. She received her B.E (Computer) Degree from S.B.Patil college of engineering, Indapur, Savitribai Phule Pune University, Pune, Maharashtra, India -411007. Her area of interest is network security.

Sachin V. Todkari, received his ME. (IT) Degree from MIT college of engineering Kothrud, Maharashtra, India -411007. He received his B.E (Computer Science) Degree from College of Engineering, Ambajogai, Maharashtra, India. He is currently working as HOD at Department of Information Technology Engineering, in Jayawantrao Sawant College of Engineering, Savitribai