International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 01 | Jan-2016 www.irjet.net p-ISSN: 2395-0072

Defend Software Puzzle against Denial of Services Attacks using a Countermeasure.

Saif A. Shaikh, Akash R. Deokar, Priti Joshi

[123] *Student, Computer Department, Trinity Academy of Engg. Pune, Maharashtra, India*

Abstract – ***Denial of services (DOS) and Distributed Denial of services (DDoS) are the major problem against network security and cyber security that allow a client to perform very expensive and vital operations, before the network services are provided to the respected client. However An attacker may be able to manipulate the DOS and DDOS or built in graphics processing Unit (GPU) and be able to destroy client puzzles. In this paper we study how to preserve DOS and DDOS attacker for being manipulating the puzzle-solving techniques. So now we introduce a new client puzzle referred to as Software Puzzle. It is unlike previous puzzle, which generate their puzzle algorithms in advance, a puzzle algorithm in the present software puzzle schemes is randomly generated only after a client request is received from the server side and the algorithm is generated such that:1)an attacker is unable to prepare an implementation to solve the puzzle in advance, 2)the attacker need very considerable effort such that he/she may be able to translate a central processing unit puzzle software to its functionally equivalent GPU version such that translation is not done in real time. How ever we show how to generate software puzzle in generic software –browser model.***

Key Words: *Software Puzzle, Denial of Service(DoS), Code Protection,GPU Programming, Distributed Denial Of Service (DDoS).*

1. INTRODUCTION

Denial of service (DoS) attacks are heard to be a very serious query In Internet whose virtue has well demonstrated in the computer network literature. The online services resources such as network bandwidth, memory and computation power are been overwhelmed by a bogus request as they are evacuated by Denial of Service (DoS) and Distributed Denial of Service attacks (DDoS). A denial of service attack is an attempt to make a machine or network resource unavailable to its intended users, such as to temporarily or suspend services of a host connected to the Internet DoS attack is been held in network resources as it conventionally blocks the network services and this attack is been created due to another user which creates malicious actions. DoS attacks target the network bandwidth or the connectivity. Bandwidth attacks flood by degrading the user requests i.e he/she is not able to send or receive requests. Connectivity is been attacked i.e all the available resources of operating system are consumed and the computer is not able respond the user request.

Distributed Denial of Service attacks (DDoS) is very simple but it is a very powerful for depleting internet resources. It make the DoS problem more difficult and hence it make it complex severe. DDoS attacks contain very powerful resources such as to execute very complex issues held by client and is very good at it. A distributed denial-of-service (DDoS) is where the attack source is more than one and often thousands–of unique IP addresses. DDoS attack does not destroy the victims computer but it is a attempt for any personal reason, either for a any information or for popularity. DDoS attacks is been distinguished from other such attacks hence it has the ability to deploy it own weapons in a distributed manner and create traffic over the connectivity or bandwidth of network resources.

1.1 EXISTING SYSTEM

The client solve the puzzle using the previous schems i.e the basic resources. Hence there are many modern computers, laptop which contain high configuration i.e they contain GPU component (for example ATI FIREPRO V3750 in Dell laptop which contain nVidia Quadro FX 880M). Therefore, an attacker can easily utilize the “free” GPUs or integrated CPU-GPU to inflate his computational capacity. This renders the existing client puzzle schemes ineffective due to the significantly decreased computing cost ratio γ. For eg. a hacker may amortize one puzzle-solving task to hundreds of GPU cores if the client puzzle function is parallelizable, or the hacker may simultaneously send to the server many requests and ask every GPU core to solve one received puzzle challenge independently if the puzzle function is non-parallelizable (e.g. modular square root puzzle and Time-lock puzzle [5]). To increase the attack efficiency the main parallelism strategy is used to reduce the total puzzle-solving time. Green et al. examined various GPU-inflated DoS attacks, and showed that attackers can use GPUs to inflate their ability to solve typical reversal based puzzles by a factor of

more than 600. To track client IP address it was proposed in order to defeat GPU-Inflated DOS attack to client puzzle. Nonetheless, if IP tracking is effective to thwart the GPU inflation, IP filtering can be used to defense against DoS attacks directly without utilizing client puzzles. Hence the defense against GPU-inflated DoS attack was not so much attractive in a proper manner.

The present search engines such as Microsoft Internet Explorer and Firefox do not explicitly support client puzzle schemes, many authors developed a web-based client puzzle scheme which focuses on transparency and backwards compatibility for incremental deployment. The scheme dynamically embeds client-specific challenges in webpages, transparently delivers server challenges and client responses. This scheme is vulnerable to DoS attackers who can implement the puzzle function in real-time. Technically, a hacker can rewrite the puzzle function P(·) with a native language such as C or C++ such that the cost of an hacker is much smaller than that the server expects(In our experiments, a native code is about 20 times faster than a Java byte code for the same function).However, a GPU-inflated DoS attacker can realize the fast software generation on the many-core GPU hardware and run the software in all the GPU cores simultaneously such that it is easy to defeat the web-based client puzzle scheme. If a puzzle is implement based on client's GPU capability, the GPU-inflation DoS does not work at all. Since, we do not recommend to do so because it is troublesome for high capacity deployment due to (1) not all the clients have GPU-enabled devices; and (2) an extra real-time environment shall be installed in order to run GPU kernel.

1.1.1. Notations

For ease of reference, important notations used throughout the paper are listed below.

x: A challenge chosen by server.

m: A message collected from environment.

y: A solution to the puzzle challenge x.

(˜ x, ˜ y): A puzzle response returned from client.

P(·): Puzzle algorithm such that x =P(y,m).

C: Puzzle core which is the software implementation of P(·).

C0x: Puzzle which embeds the information of x into C.

C1x: Obfuscated C0x.

2. Graphic Processing Unit (GPU)

Graphics Processing Units (GPUs) have become a popular choice for general-purpose high-performance computing. Encryption and decryption algorithms such as the Advanced Encryption Standard (AES) have been implemented on GPUs to gain good speedup. However, the security of the GPU architecture is not well studied, making it potentially risky to offload sensitive computation to GPUs. In this paper, we will introduce our ongoing work to improve GPU security against Denial of Service (DoS) attacks.

Modern GPUs have many processing cores that can be used for basic purpose computing as well as graphics processing. Hence the newly GPU vendors such as nVidia and AMD provide useful programming directories for massive computation application. Without loss of generality, nVidia GPU will be used to present our techniques in the following. For self-contained, this Section briefly introduces nVidia GPU [6], its application on the basic GPU-inflated DoS attacks, and its difference from CPU which will be exploited to defeat against the GPU-inflated DoS attack.

2.1 nVidia GPU

Nvidia designs graphics processing units (GPUs), as well as system on a chip units (SOCs) for the mobile computing market. Nvidia's primary GPU product line, labelled "GeForce", is in direct competition with Advanced Micro Devices' (AMD) "Radeon" products .In order to run high performance applications GPU has introduced parallel processing capabilities to scientist and researchers.

A GPU processor has fast but small shared memory. GPU contains man Streaming Multiprocessors (SMs) consisting of 'n' number of identical processing cores. For example, the nVidia GeForce GTX 680 consists of 1,536 cores. Besides, it has access to the host's global memory which is large but slow. CUDA, the major programming language for nVidia GPU, extends ANSI-standard C99 language by allowing a programmer to define C functions, or kernels. For instance, the client puzzle function P(·) can be generated as a GPU kernel. At any one time, a GPU device is dedicated to a single application which may include multiple kernels. When a kernel is loaded into GPU and invoked, it is executed by multiple identical threads in parallel for maximum efficiency.

2.2 Comparison Between CPU & GPU

CPU	GPU
The CPU or Central Processing Unit is where all the program instructions are executed in order to derive the necessary data.	GPUs where originally developed to render 2D graphics; specifically, to accelerate the drawing of windows in a GUI.

Advancement in modern day CPUs have allowed it to crunch more numbers than ever before, but the advancement in software technology meant that CPUs are still trying to catch up.	The need for 3D and faster graphics acceleration grew, the GPU became faster and more specialized in its task.
CPUs handle all of the computations and instructions in the whole computer, thus the use of the word 'central'.	A Graphics Processing Unit or GPU is meant to alleviate the load of the CPU by handling all the advanced computations necessary to project the final display on the monitor.
CPU are generally used for executing larger instruction in a respective manner.	GPU is designed for the predictable graphic processing such as matrix operations, not generic logic processing.

2.3 GPU Inflated DoS Attack

A client wants to obtain a service, he/she sends a request to the server. After receiving the client request, the server responds with a puzzle challenge x. If the client is genuine, he/she will find the puzzle solution y directly on the host CPU, and send the response (x, y) to the server. However, as shown in Fig. 1, by using the similar mechanism in accelerating calculation with GPU , 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.

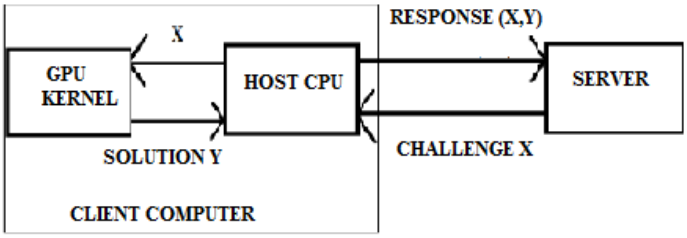

Fig -1: GPU-Inflated DoS attack against data puzzle.

3. Software Puzzle

If a puzzle function P, as all the existing client puzzle schemes ,[7] is fixed and disclosed in advance, the puzzle is called a data puzzle or it is referred to as a software puzzle. Data puzzle aims to enforce the client's computation delay of the inverse function P−1(x) for a random input x; while software puzzle aims to deter an adversary from understanding/translating the implementation of a random puzzle function P(·). That is to say, unlike a data puzzle challenge which includes a challenge data only, a software puzzle challenge includes a dynamically generated software C(·) which including a data puzzle function as a component.

3.1 System Architecture

The software scheme consist of a Warehouse which contains different kinds of block storing different information. Hence it contain two module 1.generate the puzzle C0x by randomly choosing code blocks extracted form Warehouse and obfuscating the puzzle C0x for higher security puzzle C1x.

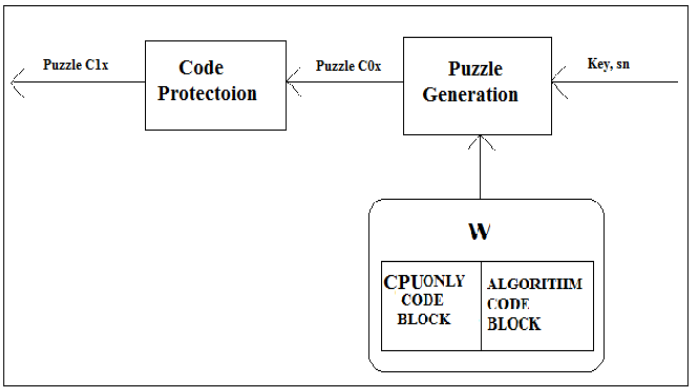

Fig-2 System Architecture

The compiled instruction code blocks(bi) are been stored in Warehouse(W) in the form of Java byte code or C binary code. In order to generate software puzzle , server chooses compiled codes rather than source code because if it chooses source code then server has to take extra time for puzzle generation. The warehouse stores both Java bytecode and the corresponding C binary code. . Because the former is applicable to different OS platforms but slow, it is suitable to deliver the software puzzle to the client in the format of Java bytecode. Code blocks can be classified into two categories: i)CPU-only instruction block ,ii)Data puzzle algorithm block.

I.CPU-ONLY INSTRUCTION BLOCK

Unlike CPU, GPU is designed for the predictable graphic processing such as matrix operations, not generic logic processing. As branching operations are inherently non-predictable and are non-parallelable, executing them in GPU is slow such that the major merit of GPU cannot be exploited by the attacker; Secondly, some hardware-related operations such as reading hardware input and surfing network, cannot be performed on GPU; Thirdly, the state-of-the-art GPUs do not support dynamic thread generation; Fourthly, the high-speed shared memory is shared by all the GPU thread blocks together such that the size of fast accessible memory available to each thread is small.

II.DATA PUZZLE ALGORITHM BLOCK

In this block mathematical operations are been done by data puzzle Algorithm. For example, in an AES round, ShiftRows code block outputs a transformed message matrix , which can be used as input of any other operation such as Mix Column code block without incurring parameter mismatch errors.

3.2 Software Puzzle Generation

In the software puzzle generation server has to execute three step: i) puzzle core generation, ii) puzzle challenge generation and iii) Code Obfuscation.

i) Puzzle Core Generation

The code block ware house chooses "n" code block based on hash function and secret key eg…the jth instruction block bij, where ij = H1(y, j), andy = H2(key,sn), with one-way functions H1(·) and H2(·), key is the server's secret, and sn is a nonce or timestamp. The chosen blocks are gathered into a puzzle core, denoted as C(·) = (bi1;bi2;⋯;bin).

ii) Puzzle Challenge Generation

The server calculates a message m from public data such as IP addresses, in-line constants, port numbers and cookies with the given auxiliary input messages and produces a challenge x =C(y,m), smiliar to encrypting plaintext m with key y to produce ciphertext x.

As the puzzle core C(.) function is not able to solve by the attacker and it can not force GPU to solve the problem in real time using basic GPU resource. It is possible for an hacker to generate the GPU kernel by mapping the CPU instructions in C0x to the GPU instructions one by one, i.e., to automatically translate the CPU software puzzle C0x into its functionally equivalent GPU version.

iii) Protecting the Code

A software puzzle consists of instructions, and each instruction contains of operands and opCode and it contains of operations such as additions,shift,jump , while the operands, varying with opCode, are the parameters to complete the operations. Operands and opcode are encrypted by code encryption technology and it behave software code as data string. The server produce an encrypted puzzle C1x = E(y,C0x), where E(·) is a cipher such as AES, and y is used as the encryption key. In practice, there are many commercial code obfuscation tool for C/C++ software such as VM protect which can be used to protect the software puzzle from hacking.

Encryption contains two layers i.e the inner layer and outer layer. In the encryption outer layer is used to encrypt the software puzzle C0x. In the encryption inner layer uses the puzzle software to encrypt the challenge as data puzzle does. Therefore, after receiving C1x, the client has to try ˜ y. If and only if ˜ y = y, the original software puzzle C0x can be recovered and further used to solve the challenges.

4. AES ALGORITHM

AES stand for Advanced Encryption Standard. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, whereas Rijndael can be specified with block and key sizes in any multiple of 32 bits, with a minimum of 128 bits. The blocksize has a maximum of 256 bits, but the key size has no theoretical maximum .AES operates on a 4×4 column-major order matrix of bytes .It is fast in both software and hardware.

The key size used for an AES cipher specifies the number of repetitions of transformation rounds. The numbers of cycles of repetition are as follows:

- 10 cycles of repetition for 128-bit keys
- 12 cycles of repetition for 192-bit keys
- 14 cycles of repetition for 256-bit keys

The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

Round are:

1. Initial Round
   i. AddRoundKey- Every byte of the state is combined with a block of the round key using bitwise XOR operation.

2. Rounds
   i. SubByte- A nonlinear substitution step where each byte is replaced with another according to a lookup table.
   ii. ShiftRows- A Transposition step where the last three rows of state are shifted cyclically a certain number of steps.
   iii. MixColumns- A missing operation which operates on the columns of the state, combining the four byte in each column.
   iv. AddRoundKey-

3. Final Rounds
   i. SubByte
   ii. ShiftRows
   iii. AddRoundKey
   iv.

5. CONCLUSIONS

Software puzzle scheme is prepaid for destroying GPU-inflated DoS attack. It hires software protection technologies to ensure challenge data confidentiality and code security for an respective time period, e.g., 1-2 seconds. It has different security requirement from the conventional cipher which demands long-term confidentiality only, and code protection which focuses on long-term robustness against reverse-engineering only. Since the software puzzle may be built upon a data puzzle, it can be integrated with any existing server-side data puzzle scheme, and easily deployed as the present client puzzle schemes do. Although this paper focuses on GPU-inflation attack, its idea can be extended to thwart DoS attackers which exploit other inflation resources such as Cloud Computing. For example, suppose the server inserts some anti-debugging codes for detecting Cloud platform into software puzzle, when the puzzle is running, the software puzzle will reject to carry on the puzzle-solving processing on Cloud environment such that the Cloud-inflated DoS attack fails. In the present software puzzle, the server has to spend time in constructing the puzzle. In other words, the present puzzle is generated at the server side. An open problem is how to construct the client-side software puzzle so as to save the server time for better defense performance. Another work is how to evaluate the effect of code de-obfuscation, which is related to the technology advance of code obfuscation.

ACKNOWLEDGEMENT

We express deepest gratitude to our project guide Prof. R.R.Ranawre, who modeled us both technically and morally for achieving greater success in life. As a mentor and torchbearer, he guided us to overcome the odds and evens faced during the project work. The supervision and support that he gave indeed paved the path for the smooth completion of the project. We are deeply indebted to our Head of the Department for their unwavering moral support and motivation during the entire course of the project. We also thank all the staff members of our college and technicians for their help in making this project a successful one. Finally, we take this opportunity to extend our deep appreciation to our Family and Friends for all that they meant to us during the crucial times of the completion of our project.

REFERENCES

[1] Yongdong Wu, Zhigang Zhao, Feng Bao, and Robert H. Deng. (Software Puzzle: A Countermeasure to Resource-Inflated Denial-of-Service Attacks)..

[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] X. Wang and M. K. Reiter, "Mitigating bandwidth-exhaustion attacks using congestion puzzles," in Proc. 11th ACM Conf. Comput. Commun. Secur., 2004, pp. 257–267.

[5] 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. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.110.5709.

[6] NVIDIA CUDA. (Apr. 4, 2012). NVIDIA CUDA C Programming Guide, Version 4.2. [Online]. Available: http://developer.download.nvidia.com/

[7] K. Iwai, N. Nishikawa, and T. Kurokawa, "Acceleration of AES encryption on CUDA GPU," Int. J. Netw. Comput., vol. 2, no. 1, pp. 131–145, 2012.