Using Software Puzzle for Reducing DDos/Dos Cost on SSL/TLS

Pankaj Kumar
Assistant Professor,
Dr. T. Thimmaiah
Institute of Technology,
Oorgaum, KGF,
Karnataka-563120

S. S. Ahluwalia, PhD
Dean-Academic, CSE
Dr. T. Thimmaiah
Institute of Technology,
Oorgaum, KGF,
Karnataka -563120

Tharun Kumar S. V.
Alex Maria Moses,
Bharath Kumar,
CSE,Student
Dr. TTIT

ABSTRACT
In cyber-security Denial-of-service and distributed Dos are the major threats, DOS and DDOS works by denying service users approved as genuine, traffic is jammed by the overwhelming illegal traffic frequencies. an attacker inflates its capability of attacks with fast puzzle solving software and graphics processing unit (GPU) hardware to significantly weaken the effectiveness of server. In this paper, we show to prevent DOS/DDOS attackers from inflating their challenge solving capabilities. To stop this, we introduce a client puzzle referred to as software puzzle.

In this paper the puzzle is generated randomly by selecting CPU only code, with time stamps .the generated puzzle cannot be easily solved through GPU with in real time

Keywords
Software Puzzle, GPU, Denial of Service, Distributed Denial of Service (DDoS), CPHS, MD5, DES

1. INTRODUCTION
Distributed DoS(DDoS) and Denial of Service (DoS) attacks deplete an internet service’s resources such as network, net bandwidth, computation power by affecting the service with bogus requests. For example, a illegal client sends a n number of garbage requests to server. The server has to spend a more time in completing TSL/SSL Handshake protocols, No sufficient resources left to handle client requests.

Company Server spent more resources and money in trying to responds to the requests generated from the illegal client. Whereas, the attacker client need not to spend as resources and money similar to server. This request degrades

the server resources capabilities in RSA encryption and decryption.

Company server uses TLS/SSL for the security of important resources ,TLS/SSL are very costly in handling each request handshakes ,so with multiple bogus requests from the attacker make the cost increase and reduce the CPU resources on the server side ,it impacts and make the server week and prone to more attacks.

In this paper, We particularly deal with DOS and DDOS counter measures in reducing Servers consumption of resources and money through software puzzle, Software puzzle is similar to existing data client puzzle, software puzzle is efficient in better ways.

i. Challenge is randomly generated

ii. Challenge should be solved in real time

iii. Client puzzle works in three steps

a. Puzzle core Generation

b. Puzzle challenge generation

c. Crypt Encryption /Decryption

The attacker impact on internet services is overcome by cryptographic mechanisms with physical layer attributes. These are Cryptographic Puzzles Hiding Schemes (CPHS), Strong Hiding Commitment Schemes (SHCS), and Cryptographic hash function using MD5 message Hash these are encrypted by DES crypt tools.

Client cost is increased by making use of an HASH reversal function which drives the resources of client in performing the process one way hash instance.as mentioned above that puzzle generation involves in 3 steps, first step results in an P puzzle function created by SHA-1 or AES block ,If we consider X as an challenge created by server for the client randomly, Challenge X is added to the puzzle functions as P(X) .P(X) is given to the client as an puzzle challenge ,the client need to provide as challenge solution to the server with a P(X,Y),where Y as an puzzle solution optioned, this validated on the server side for the checking whether the client is an authenticated or not, whereas the attacker cannot solve the puzzle in real time ,the time stamp acts major role in this puzzle software .this puzzle can improve in reduce the DoS and DDoS impact on the server side, as it increases the client work and significantly reduces the server spending resources on the bogus request.

The main advantages of the software puzzle is it reduces the major threat for the server from attacker is using of the GPU in solving the puzzle within the time ,the existing puzzle was designed to solved on the CPU whereas the increases in utilization of the GPU in every field lead to harden the failure of the existing data puzzle because of its vast multi cores present on the GPU, now days the phone also includes the GPU unit in it for graphical processing .If we consider the challenge given to one CPU core ,it solves with a given time, but in case of GPU challenge can be given to multicore ,the results are fast on the GPU because of it multicores ,in present days

Many core GPU is almost a standard configuration in modern computers eg are ATI Fire Pro V3750, NVidia Quadra FX 880M i, smartphones Power SGX540. Therefore, an attacker can easily utilize free GPUs inflate his

Computational capacity.

The proposed system make sure that need not to worry about the GPU utilization by the attacker, because the puzzle function written by the server is in CPU only code, which means attacker needs to convert the code blocks in GPU understandable blocks to solve it, which is time consuming.

The challenge can be broken down, but not in real time allocated to solve the puzzle, and the solution generated on GPU resources consumption is also no use, because the puzzle generates the challenge randomly each time.

This makes software puzzle to reduce the cost of the server by not letting the attacker to make TLS expensive handshakes.

2. INTRODUCTION TO CPU AND GPU

Earlier GPU has many cores of processing which can be used for General purpose computing as well as graphical processing, NVidia and AMD they are the major manufacturers, providing comfortable programming libraries to use GPU's for computational application. Without the loss of generality, NVidia GPU can be used to present this technique.

In this section we brief out the introduce NVidia GPU, its application on GPU-inflated DOS attack and difference between CPU to defeat against the GPU inflated DOS attack.

a. Overview of NVidia GPU

In NVidia Architecture, GPU has many multiprocessing, which contain many no of same processing cores. Ex: NVidia GTX 680 contains 1,536cores, GPU processor is very fast but uses a tiny shared memory, it access to global memory which is large but slow.

The programming language for NVidia GPU is ANSI-standard c99 language by allowing a developer to use C function (or) Kernel, the client Puzzle p (.) can be developed as a GPU kernel, at a single instance, GPU system is given to only one application which may include multiple kernel, when kernel is loaded to GPU it is executed by multiple threads in parallel for more efficiency.

b. Difference between CPU and GPU

Earlier CPU which is designed to optimize execution on the single thread program using a critical out of order execution, modern GPU executes multiple data in parallel in a different way, GPU cannot handle branching instructions.

Both GPU and CPU software can be developed using the same high level language such as C, but their low level instruction set are totally different, many instruction does not support in GPU software due to self-modification of code because codes are used for software protection it modifies the software itself.

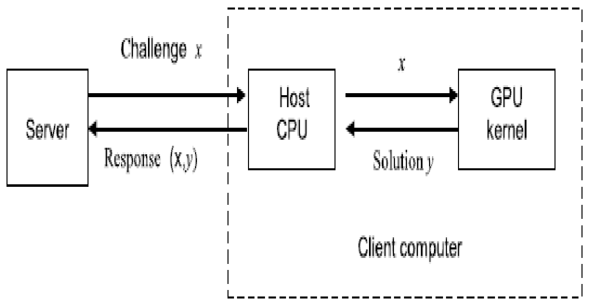

Fig. 1. GPU-inflated DoS attack against data puzzle.

CPU process is slower, when compared to GPU process but a single CPU core is much faster than one GPU core, CPU resources are reserved such as memory and cache, but in GPU cores are share the resources not only the memory and cache but also the register and cache. If the no of core in the application is ore then the no of cores in the GPU in this case the GPU core is slower than CPU, this paper exploit the difference between CPU and GPU and also tells you that how GPU is used to improve the puzzle solving process.

3. SOFTWARE PUZZLE

The client puzzle is broadly classified into two types the puzzle function P, the existing client puzzle is fixed and given to the client in advance, this is known as the data puzzle; otherwise it is referred as software puzzle. Data puzzle aims to force the clients computation delay of the inverse function p-1 (x) for the random input x, while software puzzle solving aims to determine the adversary for understanding /translating the development of the random puzzle function p(.), to brief out unlike the data puzzle challenge include data only, a software puzzle challenge which include a dynamically generated software C(.) which contain a data puzzle scheme which does not reveal the puzzle in advance, and it also make use of the Kirchhoff's principle because of an adversary knows the algorithm for constructing software puzzle and is able for reverse- engineering the software puzzle C1x to know puzzle from p(.) will be received after a several minutes of receiving the software puzzle.

A. Basic GPU inflated DOS attack

In order to brief out software puzzle, we recap GPU-inflated DOS attack in advance, whenever a client wants to get a service, he /she sends a request to the server, after receiving the client request, the server will respond to the client with a puzzle challenge x. if the requested client is a genuine user he/she will find the solution for the puzzle (y) directly on the host CPU, and response to the server as (x, y), by using this mechanism we calculate with GPU a malicious user will control the host and will send a challenge x to GPU and exploit the GPU resource to run the puzzle solving problem.

B. Framework of Software Puzzle

In order to overcome the GPU-inflated DOS attack we extend data puzzle to software puzzle as shown in Fig. 2. In server side, software puzzle scheme has a code block warehouse W for storing various software instruction blocks.

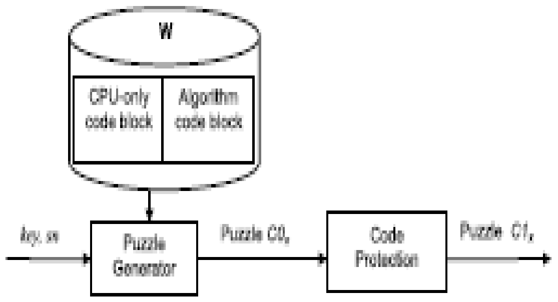

Fig. 2. Diagram of software puzzle generated with secret *key* and nonce *nn*.

The warehouse consists of two modules generating the puzzle C0x by assembling random code block taken from the warehouse; and the puzzle C0x for high security puzzle C1x.

C. Code block warehouse Construction

In code block warehouse W is used to store complied instruction blocks {bi}, in may be a java byte code, or c binary code. The use of storing the compiled code then source code is to save the server time, if not the server will take additional time to compile the source code in the process of software puzzle generation. The requirement for each blocks are

- To assemble the code blocks together each block consists of a set of defined input parameters and output parameters, it is followed in a sequential manner such that the output of one block is given as the input to the following next block.
- Size of the block is defined by the parameters of the security k. the size of the software puzzle is constant, if the block size is smaller, there may be more no of blocks, such that more no of puzzles can be generated/constructed.

Such that smaller block size has higher security level such that an attacker has to spend more effort to find the puzzle in the question. The short incoming of small block size is that the server has to give more time to extract the basic block and assembling the extracted block into a software puzzle. Perhaps the warehouse is used to store both the java byte code and the c binary code. Because of independent of different OS but slow, it is preferably uses to give the software puzzle to client in the form of java byte code.

Later it is fast and is used by the server to construct the stored pair (x, y). It also supports cross platform to be formed. The code block can be further divided into two categories: CPU only instruction block and data puzzle algorithm block.

1. **CPU-Only Instruction Block:** GPU is designed for the graphic processing like matrix operations, but not the generic logic processing. The branching operation like, try, catch, finally, goto can be inherited as non-predictable and non-parallel, executing these instruction in GPU is slow it is the main advantage of the GPU such that the it cannot be analyzed by the attacker, and some hardware related operated operation like reading hardware input and browsing through network cant not be done on the GPU. The GPU does not support the dynamic thread generation. It has a high speed shared memory such that the size of accessible is not so slow but the size is thread is very small, therefore if the puzzle kernel demands for large shared memory, the GPU parallelism restricts the access seriously, the thread has to access the global memory much slower.
2. **Data puzzle Algorithm Block:** this algorithm blocks perform the mathematical operation only likewise in AES round, shifts Rows code output is translated as a message to the matrix, which is taken as the input to the other operation like MixColumn code block without parameter mismatch error…

D. Software Puzzle Generation

To build a software puzzle, the server has to perform three modules:1) puzzle core generation,2)puzzle challenge generation, 3)software puzzle encrypting/obfuscating.

1. Puzzle Code Generation

The code block ware- house chooses" n" code blocks based on hash functions and a secret key, e.g., the jth instruction block bij, where ij = H1(y, j), andy = H2(key,sn), with one-way functions H1(·) and H2(·), key is the hidden information of server, and sn is a nonce or timeduration. The chosen blocks are gathered into a puzzle core, denoted as C(·) = (bi1;bi2;···;bin). As an illustrative example, Table III in the appendix shows an example puzzle core C generated from AES operation blocks stored in warehouse S. The puzzle core can be dynamically generated with hash function and a secret key for n code blocks.

2. Puzzle Challenge Generation

The server calculates a message m from public data such as their IP addresses, port numbers and cookies with the given some auxiliary input messages such 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(·) (or equivalently the puzzle function P(·)) function is not able to solve by the hacker in advance and it cannot force GPU to solve the puzzle problem C0x in real time using the 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.

3. Code Protecting

Code obscure is able to prevent the above translation threat to some extent. Though there are no generic obscure techniques which can prevent a patient and advanced attachker from understanding a program in theory [17], results in [18] show that obscure does increase the cost of reverse-engineering. Thus, although code obscure may be not satisfactory in long-term software defense against hacking, it is suitable for fortifying software puzzles which demand a protection period of several seconds only. A software puzzle consists of instructions, and each instruction contains of operands and opcode and it contains of operations such as addition, shift, jump, while the operands, varying the op Code, are the parameters to complete the operations.Operands and op code are encrypted by code encryption technology and it behave software 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. There are many commercial code obscure tool for C/C++ software such as VMprotect (http://vmpsoft.com/) which can be used to protect the soft- ware puzzle from hacking.

Encryption contains two-layers i.e. the inner layer and outer layer. In encryption the outer layer is used to encrypt the software puzzle C0x.In the encryption the 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. PUZZLE PACKAGING

The java class file C1x.class is created from the compiling C1x created at server site, it will be sent to client from a server through an internet source ,applet is delivery means .applet can run in any browser and many operating systems, but not all applicable to the mobile browsers .

Applet is embedded into an HTML page which is inurn embedded including the software puzzle C1x.class and a java class init.class for initializing the software classC1x.class

```
1.    < APPLET CODE=‘‘iniit.class’’
      ARCHIVE = ‘‘init.class, C1x.class’’
                 WIDTH=‘‘300’’ HEIGHT=‘‘50’’>
                 </APPLET>
```

Not all applets cannot run at client side with default access policy so that design for puzzle varies with browsers settings and configurations at client browser.so that we describe the packaging of the software puzzle based on the configuration at client end.

A. Reloading class into java sandbox
The execution of the C1x.class cannot be direct execute on JVM at client side ,because the software puzzle instructions has been encrypted ,has to be decrypting ,however ,a new instructions generation cannot be called by java class itself. Replacing the entire class by reloading a new class is illegal in JVM.

The init.class for reloading puzzle class on JVM. If a correct solution y is found, C0x.class shall be the same as the original puzzle C0x.class, where the challenge and solution is calculated in advanced and hard-coded into at the server side.

```
1. Read the C1x.class
2. Repeat
3. Randomly choose a small y
4. Decrypt C1x.class with key y into class Cox.class
5. Load class Cox.class to obtain m and further
6. X=Co(Y,M)
7. Until X=x
8. Output (X,Y)
```

B. Java native interface in dedicated sandbox
Java native interface provides java programs easy access to shared libraries with languages c/c ++, the software puzzle implemented with these codes can provide stronger performance, however JNI programming requires a dedicated execution platform

As dedicated sandbox the puzzle directly without reloading the class, the structure of init class is simplified as shown below

```
Read the Cx.class
Load class Cx.class
Repeat
Randomly choose a small Y
Decrypt Cx.class with key Y into class
Cyx.class
Invoke Cyx.class to obtain M and further X=Co(Y, M)
Until X=x
Output (X, Y)
```

5. SECURITY ANALYSIS

The aim of software puzzle is eliminate involvement of the GPU used for the solving puzzle process, by using different instruction set and real time between CPU and GPU, many attempts to deface software puzzle scheme on GPU cracking puzzle algorithm.

A. Host simulator on GPU

Attacker runs a CPU simulator over GPU

Environment, the software puzzle executed on GPU directly. However, simulator based is impractical in accelerating the puzzle-solving process because.

VM software emulate entire hardware environment, problems arise if the properties of hardware resources different in the host and the guest, No host simulator on GPU at present to develop a full-functional CPU simulator on GPU, because the CPU environment including Operating System, and imported Java libraries must be simulated. Only of simulator functions are implemented the GPU kernel as communicate with the host for the non-simulated functions. This leads to the GPU-inflation function is reduced significantly, because it cannot run in a parallel way and the GPU-CPU communication channel is slower;

A software running on a simulator is much slower than over its guest environment directly due to more processing steps to execute the software instructions.

B. Breaking Data Puzzle Algorithm

According to packaging of puzzle obtains the puzzle solution (X, Y) to the software puzzle C1x, such that x = X = _C0x (Y, M), where number x is hard coded in the software puzzle and M is derived on the fly. Since the software puzzle is encrypted on standard cipher, an attacker has to recover the puzzle software by brute force. For

the inner-layer encryption, as C(·) is an encryption function, theoretically, an attacker cannot find a valid solution (X,Y) than brute force given that y is over a small interval. Hence the attacker accelerates the brute force process by exploiting the parallel computation capability of GPU cores.

Even the code blocks combination may be not as secure as the original ones for the basic software puzzle.

in software puzzle, this problem can be easily overcome.

C. Data Puzzle Replaying

When a software puzzle is built on a data puzzle, the number puzzles is required to be very large such that an attacker is unable to construct the GPU-version software puzzles in advance and re-use them. Even though a service provider adds one AES round transformations between two AES transformations in the standard 10 rounds, the number of AES variants is up to 49×4+3 = 278.

Software can have many polymorphic codes such that the number of software puzzles is even larger. But, a smart adversary may collect all the code blocks in the warehouse **W**, and rebuild the GPU version code block warehouse **W,** *GPU* in advance. Once a new software puzzle is delivered , he reconstruct the GPU version puzzle by matching the puzzle code blocks against the software puzzle. In this case, it increases the attack performance. However, as the server encrypts the puzzle software *C*0*x* into *C*1*x*, the adversary has to recover *C*0*x* by brute force, and hence can not re-construct the GPU-version puzzle by matching code patterns.

D. Strong Software Code

To rewrite the GPU kernel, an attacker determine the instruction flow by debugging the

Software puzzle. Commonly dynamic translation can accelerate the attacking speed, but it is not very helpful to the GPU-inflated DoS attacker because

Dynamic translation is an human computer interactive process. If human interference is required, the DoS attack is very ineffective;

The attacker needs a simulation environment for “debugging” the software puzzle In order to use the dynamic translation . In the translation process, the decryption key Y has to be tested by brute force. Because it is impossible to decide whether a tested key is right based on the recovered opCode value.

Once the translated code has one error, the attacker fails to recover the software puzzle C0x,

Therefore, it is not easy for an attacker to develop a GPU kernel for solving the original software puzzle by analysing software puzzle.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we use the software puzzle scheme for reducing GPU-inflated DoS attack. Software puzzle ensure challenge data confidentiality and code security for an real time

Period. It has different security requirement from the conventional cipher which demands long-term confidentiality only, long-term robustness against reverse-engineering only.

Since the software puzzle built upon a data puzzle, so it can be integrated with any existing server-side data puzzle Scheme.

This makes the server to reduce cost on the TLS/SSL handshakes with the attackers; it does not only reduce the cost but also protects server resources to fail down

7. REFERENCES

[1] J. Larimer. (Oct. 28, 2014). Pushdo SSL DDoS Attacks. [Online].Available: http://www.iss.net/threats/pushdoSSLDDoS.html

[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] T. J. McNevin, J.-M. Park, and R. Marchany, “pTCP: A client puzzleprotocol for defending against resource exhaustion denial of serviceattacks,” Virginia Tech Univ., Dept. Elect. Comput. Eng., Blacksburg,VA, USA, Tech. Rep. TR-ECE-04-10, Oct. 2004.

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

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

[7] Y. I. Jerschow and M. Mauve, “Non-parallelizable and non-interactiveclient puzzles from modular square roots,” in Proc. Int. Conf. Availability,Rel. Secur., Aug. 2011, pp. 135–142.

[8] R. L. Rivest, A. Shamir, and D. A. Wagner, “Time-lockpuzzles 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=10.1.1.110.570