International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Software Puzzle: A Countermeasure to Resource-Inflated Denial-of-Service Attacks

Syeda Ghazala Nikhat[1], Zohara Begum[2], Dr. Mohammed Abdul Waheed[3]

[1]PG Scholar, Department of Computer Science and Engineering, VTU CPGS Kalburagi, Karnataka, India

[2]Assistant Professor, Department of E&CE, KBN College of Engineering, Kalaburagi, Karnataka, India

[3]Associate Professor, Department of Computer Science and Engineering, VTU CPGS Kalburagi, Karnataka, India

Abstract: ***Denial-of-service (DoS) and Distributed Denial of Service (DDoS) are dangerous to cyber-security, and client puzzle, 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. After all, an attacker can boost its capability of DoS/DDoS attacks with fast puzzle solving software and/or inbuilt graphics processing unit such like (GPU) hardware to extremely weaken the effectiveness of client puzzles. There are many system exist like Timelock puzzle, Client puzzle are used in this paper. 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 previously and generate software puzzle for each client request, a puzzle algorithm in the present software puzzle scheme is created only after threshold value of client request exceed which is accepted at the server side by using decision tree and the algorithm is generated such that: 1) an attacker is not able to prepare an implementation to solve the puzzle previously and 2) the attacker needs extensive effort in translating a central processing unit puzzle software to its functionally equal GPU version such that the transformation cannot be done in real time.***

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

1. Introduction

Denial of Service (DoS) attacks and Distributed DoS (DDoS) attacks try to damage an online service's resources such as network bandwidth, memory and computation power by outstanding the service with bogus requests. When client establishing connection with server needs a lot of CPU time to make SSL handshake. It may result an insufficient resource are left to providing services. In this case, conventional cryptographic tools do not enhance the availability of the services; in fact, they may reduce service quality due to expensive cryptographic operations. The seriousness of the DoS/DDoS problem and their increased frequency has led to the advent of numerous defense mechanisms [2]. In this paper, we are particularly excited in the countermeasures to DoS/DDoS attacks on server computation power. Client puzzle [1] is a well-known approach to increase the cost of clients as it pressures the clients to carry out heavy operations before being granted services. Generally, a client puzzle strategy consists of three steps: puzzle generation, puzzle solving by the client and puzzle verification by the server. Many of the system are existed which are using techniques like Timelock puzzle, client puzzle rather than this technology some other techniques also available like mod_kaPoW. So, this paper presents an idea of Software puzzle which takes input as request from client, and process the step using software puzzle. Therefore, in either case, a client puzzle can significantly reduce the impact of DoS attack because it permits a server to spend much less time in handling the bulk of malicious requests. Server gives threshold value of client requests, if requests exceeds the threshold value then software puzzle is given to client. Otherwise requested client is a legitimate client operate its task normally. This paper not only classify the attack is DoS/DDoS and but also request type. Optimizing the puzzle verification mechanism is very important and doing so will undoubtedly improve the server's performance.

2. Related Work

In [1] "Client puzzles: A cryptographic countermeasure against connection depletion attacks" this paper, introduces a new approach that we refer to as the client puzzle protocol, the aim of which is to fight against connection depletion attacks. The idea is quite simple, when there is no witness of attack, a server accepts connections request normally, that is aimlessly. When a server comes under attack, it accepts connections selectively. In particular, the server gives to each client wishing to make a connection a unique client puzzle. A client puzzle is a quickly computable cryptographic problem formulated using the time, a server secret, and additional client request information. The server resource allocated to it for a connection, the client must submit to itself for a connection, the client must submit to the server a accurate solution to the puzzle it has been given. Client puzzle are deployed in union with conventional time-outs on server resources. Thus, while genuine client will experience only a small degradation in connection time when a server comes under attack, an attacker must have access to large computational resource to create breach in service. Cryptographic puzzles have been used for several task, such as fighting against junk e-mail, creating digital time capsules, and metering Web site usage.

In [2] "Reconstructing Hash Reversal based Proof of Work Schemes" this paper, elaborated an idea of Proof of Work (PoW) mechanisms, in which a server request that clients prove they have done work previously it commits resources to their requests. Most PoW mechanisms are puzzle-based techniques in which clients solve processing thorough puzzles. For instance, Hash Cashes are puzzle-based

mechanisms that aim to prohibit an attacker from sending too much spam. As attacks use more resources, and therefore the puzzle difficulties increase, weaker legitimate clients may experience unacceptable requirements to obtain service. While computationally weaker clients would experience longer latencies during an attack, it would be extremely more functional than a protocol without the PoW based defense. Using Graphical Processing Units (GPUs) provides a powerful technique for launching resource inflation attacks. The attackers can use cheap and widely available GPUs to boost their ability to solve typical hash reversal based puzzles by a factor of more than 600. This paper is the calculation of Hash- Reversal PoW schemes in the presence of resource-inflated attackers. In this show that client-based adaptation is necessary for providing satisfactory service to genuine clients in this situation. Additionally, it show that an robust hash reversal PoW scheme based only on server load will fail to provide service, and can create a novel DoS attack against fair clients. Given these results, hash reversal PoW strategy proposed for DoS protection mechanisms should keep track of client behavior given the developing threat of GPGPU based attacks.

In [3] "Time-lock puzzles and timed-release crypto" this paper narrate the notion of timed-release crypto where the goal is to encrypt a message so that it can not be decrypted by anyone, not even the sender, until a prearranged amount of time has passed. The goal is to send information into the future. We study the problem of creating computational puzzles, called time-lock puzzles that require a precise amount of time to solve. The solution to the puzzle reveals a key that can be used to decrypt the encrypted information. This approach has the obvious problem of trying to make CPU time and real time agree as closely as possible but is nonetheless interesting. The more computational resources might be able to solve the time lock puzzle more quickly, by using large parallel computers. Another approach is the puzzle doesn't automatically become solvable at a given time; slightly, a computer needs work continuously on the puzzle until it is solved.

In [4] "mod_kaPoW: Mitigating DoS with transparent proofof-work" this paper described a approach of mod_kaPoW system that has the efficiency and human transparency of proof-of-work strategy and also having the software backwards compatibility. There are several disadvantages of using CAPTCHAs. One drawback is the user-interface problem they create; users with visual disabilities are unable to access content legitimately while natural users find it increasingly difficult to solve CAPTCHAs correctly as the images have become less readable in order to thwart sophisticated attacker that have developed automated solvers for simple CAPTCHAs. Another drawback is the static nature of the problems being given out. A proof-of-work scheme alters the operation of a network protocol so that a client must rebound their challenge along with a correct answer before being granted service. The challenge acts as a refine for clients based on their willingness to solve a computational task of varying difficulty. This paper describes the design, performance, and evaluation of a novel web based proof-ofwork system that provides the benefit of configurable PoW protocols in a portable manner. Unlike CAPTCHAs, the system is transparent to its users and supports backwards compatibility for traditional clients. The basic approach only requires changes to web servers and is similar to the URL rewriting approach employed by content-distribution networks such as Akamai. In the approach, the web server dynamically rewrites URL references by attaching a computational puzzle to them.

In [5] " Proofs of work and bread pudding protocols" this paper introduces an idea of bread pudding protocol. Bread pudding is a dish that originated with the purpose of reusing bread that has gone stale. In the same manner, a bread pudding protocol to be reused by the verifier to achieve a separate, useful, and verifiable correct computation. In this paper, we deviate from the standard cryptographic aim of proving knowledge of a secret, or the truth of a mathematical statement. POW is a protocol not defined or treated formally, POWs have been defined as a mechanism for a number of security goals, including server access metering, construction of digital time capsules, uncheatable benchmarks and denial of service. This paper contributes bread pudding protocol to be a POW such that the computing effort invested in the proof may be harvested to achieve a separate, useful and verifiably correct computation. These POWs can serve in their own right as mechanisms for security protocols as well as harvested in order to outsource the MicroMintminting operation to a large group of untrusted computational devices.

In [6] "Avoiding Permanent disabling of Wireless Sensor Network from the attack of malicious Vampire Nodes" this paper explores resource depletion attacks, which permanently disable networks by draining node's battery power. Here we propose a new mechanism to alleviate the attack from our network. The open nature of the wireless links attracts many security threats to the network. These attackers compromise them with the link and deplete the battery energy resource. One such attack is Vampire Attack.

3. Existing System

1) DoS and DDoS are effective if attackers spend much less resources than the victim server or are much more powerful than normal users. In the example above, the attacker spends negligible effort in producing a request, but the server has to spend much more computational effort in HTTPS handshake (e.g., for RSA decryption). In this case, conventional crypto-graphic tools do not enhance the availability of the services; in fact, they may degrade service quality due to expensive cryptographic operations.
2) The seriousness of the DoS/DDoS problem and their increased frequency has led to the advent of numerous defense mechanisms.
3) As the present browsers such as Microsoft Internet Explorer and Firefox do not explicitly support client puzzle schemes, Kaiser and Feng 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.

3.1 Disadvantages of Existing System

1) Puzzle is designed based on client's GPU capability, the GPU-inflation DoS does not work at all. However, we do not recommend to do so because it is troublesome for massive 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.
2) However, this scheme is vulnerable to DoS attackers who can implement the puzzle function in real-time.
3) Existing systems are not dynamic.

4. Proposed System

1) In this paper, software puzzle scheme is proposed for defeating GPU-inflated DoS attack. It adopts software protection technologies to ensure challenge data confidentiality and code security for an appropriate time period, e.g., 1-2 seconds. Hence, 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.
2) 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.
3) By exploiting the architectural difference between CPU and GPU, this paper presents a new type of client puzzle, called software puzzle, to defend against GPU-inflated DoS and DDoS attacks.

4.1 Advantages of Proposed System

1) SSL/TLS protocol is the most popular on-line transaction protocol, and an SSL/TLS server performs an expensive RSA decryption operation for each client connection request, thus it is vulnerable to DoS attack.
2) Our objective is to protect SSL/TLS server with software puzzle against computational DoS attacks, particularly GPU-inflated DoS attack. As a complete SSL/TLS protocol includes many rounds, we use RSA decryption step to evaluate the defense effectiveness in terms of the server's time cost for simplicity.
3) The software puzzle scheme dynamically generates the puzzle function.

5. System Architecture

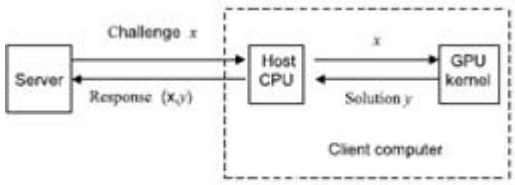

Fig. 1. GPU-inflated DoS attack against data puzzle.

Denial-of-service (DoS) and distributed DoS (DDoS) are among the major threats to cyber-security, and client puzzle, which demands a client to perform computationally expensive operations before being granted services from a server, is a well-known countermeasure to them. However, an attacker can inflate its capability of DoS/DDoS attacks with fast puzzlesolving software and/or built-in graphics processing unit (GPU) hardware to significantly weaken the effectiveness of client puzzles. In this paper, we study how to prevent DoS/DDoS attackers from inflating their puzzle-solving capabilities. To this end, we introduce a new client puzzle referred to as software puzzle. Unlike the existing client puzzle schemes, which publish their puzzle algorithms in advance, a puzzle algorithm in the present software puzzle scheme is randomly generated only after a client request is received at 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 and 2) the attacker needs considerable effort in translating a central processing unit puzzle software to its functionally equivalent GPU version such that the translation cannot be done in real time. Moreover, we show how to implement software puzzle in the generic server-browser model.

Block Diagram

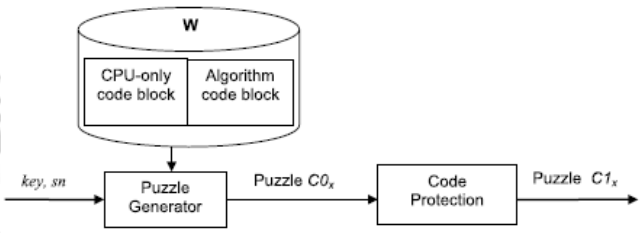

Fig. 2. Diagram of software puzzle generated with secret *key* and nonce *sn*.

A warehouse is constructed that contains CPU only code and algorithm code block. These code blocks are stored in compiled form so as to decrease server time and increase server performance. A Puzzle Generator generates a puzzle C0x using secret key and code protector will convert puzzle C0x to puzzle C1x for high security.

6. Conclusion

As this complete paper narrate different methodology on software puzzle, but none of the methodology are seems to be perfect. So, this paper as bit introduce an idea of software puzzle which is generated by using fuzzy logic and decision tree, server send query to those client reaching above the threshold value in the warehouse. In this paper also classify the type of request as well as types of attacks that is DoS/DDoS.

References

[1] 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.
[2] 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.
[3] 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.
[4] E. Kaiser and W.-C. Feng "mod_kaPoW: Mitigating DoS with transparent proof-of-work,", in Proc. ACM CoNEXT Conf., 2007.
[5] M. Jakobsson and A. Juels "Proofs of work and bread pudding protocols," in Proc. IFIP TC6/TC11 Joint Working Conf. Secure Inf. N.
[6] Rashmi Nayakawadi, Md Abdul Waheed, Rekha Patil "Avoiding Permanent disabling of Wireless Sensor Network from the attack of malicious Vampire Nodes", in International Journal of Advanced Scientific and Technical Research Issue 4 volume 3, May-June 2014 on http://www.rspublication.com/ijst/index.html.
[7] 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
[8] W.-C. Feng and E. Kaiser, "The case for public work," in Proc. IEEE Global Internet Symp., May 2007, pp. 43–48.
[9] D. Keppel, S. J. Eggers, and R. R. Henry, "A case for runtime code generation," Dept. Comput. Sci. Eng., Univ. Washington, Seattle, WA, USA, Tech. Rep. CSE-91-11-04, 1991.
[10] E. Kaiser and W.-C. Feng, "mod_kaPoW: Mitigating DoS with transparent proof-of-work," in Proc. ACM CoNEXT Conf., 2007, p. 74.
[11] NVIDIA CUDA. (Apr. 4, 2012). NVIDIA CUDA C Programming Guide, Version 4.2. [Online]. Available: http://developer.download.nvidia.com/
[12] 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.
[13] 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, pp. 258–272.
[14] D. Kahn, "The Codebreakers: The Story of Secret Writing", 2nd ed. New York, NY, USA: Scribners, 1996, p. 235.
[15] 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.
[16] B. Barak et al., "On the (Im)possibility of obfuscating programs," in Advances in Cryptology (Lecture Notes in Computer Science), vol. 2139. Berlin, Germany: Springer-Verlag, 2001, pp. 1–18.
[17] 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.
[18] S. Wang. (Sep. 18, 2011). How to Create an Applet & C++. [Online].Available: http://www.ehow.com/how_12074039_create-Applet-c.html#ixzz24Lsk0OJQ
[19] J. Bailey. (Oct. 28, 2014). How to Install Java on an iPhone, eHowContributor. http://www.ehow.com/how_5659673_install-java-iphone.html#ixzz24jIAyKiM
[20] J. Ansel et al., "Language-independent sandboxing of just-in-time compilation and self-modifying code," in Proc. ACM SIGPLAN Conf. Program. Lang. Design Implement., 2011, pp. 355–366.