iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** V **Month of publication:** May 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Defend Dos Attack with Geometric Hashing Function and Software Puzzle

Kiruthika Rengaraj[1], Matheshwaran Veerappan[2]
[1,2]*Master of Computer Application, Anna University*

Abstract- ***DOS attack can be exhaust the target server's resources have become major threat to today's internet. Even though client problem represent a promising way to defend against certain classes of DOS attack, several questions stand in the way of their use in practice. Existing software puzzle system issues their puzzle algorithm already. The attackers can easily solving the puzzle using puzzle solving software. We proposed a new client puzzle software algorithm that can choose puzzle algorithm randomly during the request form the client. The attacker cannot show progress since server chooses the puzzle algorithm randomly. We implement an efficient password authentication scheme based on a geometric hashing function in order to improve the confirmation. A single message prepared by the client and sent to the server, and a confirmation performs by the server. It efficiently defends beside DOS, verifier-stolen attack, replay attack, password guessing attack, min-in-the-middle attack.***
Keywords— hashing function, DOS attack, puzzle algorithm

I. INTRODUCTION

Denial of Service (DoS) attacks is definitely a very serious trouble in the Internet, whose crash has been fine recognized in the computer system literature. The most important plan of DoS is the disruption of services by effort to maximum access to a machine or service as an alternative of undermines the service itself. This kind of attack plan at illustration a network unable of provided that normal service by targets either the networks bandwidth or its connectivity. These attacks realize their aim by sending at a casualty a stream of packets that swamps his network or processing capability deny contact to his usual clients. In the not so remote past, there has been some large-scale attack targeting high profile Internet sites. Distributed Denial of Service (DDoS), is a moderately simple, yet very leading technique to attack Internet resources. DDoS attacks add the many-to-one dimension to the DoS difficulty creation the obstacle and alleviation of such attacks extra hard and the impact proportionally severe. DDoS exploits the essential fault of the Internet system architecture, its open source access model, which satirically, also happens to be its most benefit. DDoS attacks are including of packet streams from dissimilar sources. These attacks engage the power of a vast number of coordinated Internet hosts to consume some critical resource at the target and deny the service to legitimate clients. The traffic is usually so aggregated that it is difficult to distinguish legitimate packets from attack packets. More importantly, the attack volume can be larger than the system can handle. Unless special care is taken, a DDoS victim can suffer from damages ranging from system shutdown and file corruption, to total or partial loss of services. Background and Related Work In currency-based DoS defense mechanisms a server under attack demands some type of payment from all clients in order to raise the bar for provoking work by the server. In this section we explain some of the existing work on two classes of currency-based mechanisms: puzzle-based and bandwidth-based. We also explain resource fairness as a goal that currency mechanisms aim to achieve.

II. LITERATURE SURVEY

Ravinder Shankesi, Omid Fatemieh, and Carl A. Gunter, University of Illinois Urbana-Champaign [1]. Denial of Service (DoS) attacks on the Internet aim to prevent legitimate clients from accessing a service and are considered a serious threat to the availability and reliability of the Internet services. Numerous DoS defense mechanisms have been proposed in the literature. Among them, we particularly focus on currency-based mechanisms. The broad design objective of the currency-based defense mechanisms is to achieve resource fairness. Disparity is inevitable, but it has been argued that it is sufficiently modest for proposed resources that currency-based schemes can be effective if they adapt to attackers dynamically. In this paper we argue that such claims need to be analyzed in light of a threat that adversaries may have a way to achieve resource inflation, where they use resources or techniques that valid clients may not have implemented. We consider a range of resource inflation strategies and attempt to access their likely effectiveness. The paper makes three primary contributions. First, we introduce and analyze the concept of resource inflation as a 'thinking out of the box' approach to defeating DoS countermeasures. Such a puzzle scheme should also ensure that clients solving different puzzles of the same difficulty should require predictable and similar computational resources. Creating such a puzzle scheme, which is cheap for the server to generate and verify, is an open

problem.

Jeff green_, joshua juen, omid fatemieh, ravinder shankesi, dong jin, carl a. Gunter university of illinois at urbana-champaign[2]. A contribution of this paper is the evaluation of Hash- Reversal PoW schemes in the presence of resource-inflated attackers. Thus, we propose the use of protocols which adapt based upon client behavior. These are shown to be effective. Given these results, hash reversal PoW schemes proposed for DoS protection mechanisms should keep track of client behavior given the emerging threat of GPGPU based attacks. Modern GPUs are very efficient at processing large amounts of data in parallel. This is often referred to as Single Instruction Multiple Data (SIMD) programming. Recently, there has been significant interest in using GPUs for non graphical computation. This paradigm is known as General Purpose GPU (GPGPU) computing or Stream Computing. GPU-based architectures consist of a large number of SIMD engines. Tracking the behavior of each client provides a means of identifying those clients responsible for disproportionate load. Resource-scaling attackers are capable of solving stunning quantities of puzzles, and generating massive load. Tracking should correctly identify, and penalize, these clients. In conclusion, Hash-Reversal PoW schemes can effectively restrict a resource scaling adversary's capabilities by adjusting puzzle difficulty based on past client behavior.

Ronald L_ Rivest__ Adi Shamir___ and David A_ Wagner [3]. Our motivation is the notion of timed release crypto where the goal is to encrypt a message so that it cannot be decrypted by anyone_ not even the sender_ until a predetermined amount of time has passed. Use time_lock puzzles_computational problems that cannot be solved without running a computer continuously for at least a certain amount of time_ Use trusted agents who promise not to reveal certain information until a specified date. Using trusted agents has the obvious problem of ensuring that the agents are trustworthy secret sharing approaches can be used to alleviate this concern. Using time_lock puzzles has the problem that the CPU time required to solve a problem can depend on the amount and nature of the hardware used to solve the problem as well as the parallelizability of the computational problem being solved. We first explore an approach based on computational complexity. We study the problem of creating computational puzzles called time_lock puzzles that require a precise amount of time to solve. We propose an approach to building puzzles that appears to be intrinsically sequential in the desired manner Of course our approach yields puzzles with a solution time that is only approximately controllable since different computers work at different speeds. We first explore an approach based on computational complexity: we study the problem of creating computational puzzles called time lock puzzles that require a precise amount of time to solve. This approach has the obvious problem of trying to make CPU time and real time agree as closely as possible but is nonetheless interesting. Our goal is thus to design time_lock puzzles that_ to the greatest extent possible_ are intrinsically sequential in nature and cannot be solved substantially faster with large investments in hardware.

David Keppel, Susan J. Eggers and Robert R. Henry [4]. We define runtime code generation (RTCG) as dynamically adding code to the instruction stream of an executing program. An RTCG implementation dynamically compiles the expression into a special-case function and then calls the function once per datum. In some systems the view of RTCG is strongly in sequenced by the definition of code. Runtime code generation has been in use since the earliest programmable-store computers. In those days, memory was tight and clever ad-hoc self-modifying code sequences were often smaller and thus faster. Runtime code generation was used in early systems, but lost popularity as practices changed and as other concerns became more important. In older systems, memory space was always tight. RTCG was used in a variety of ways to build programs that it in small memories Portability became an issue with the proliferation of architectures and implementations, and the cost of writing and maintaining software for each platform. Many portability problems were solved by switching to high-level languages, but most high-level languages lack constructs for specifying RTCG. is profitable. In this paper we have analyzed RTCG as an optimization technique. A runtime compiler can generate better code because it has more specific information about the particular execution.

III.EXISTING SYSTEM

An efficient puzzle based on repeated-squaring and we demonstrated its usefulness in several security-critical applications. While we acknowledge that our puzzle, based on repeated-squaring, might not be ideal to compare the problem-solving performance of processors, we argue in this section that our proposal presents one of the very few alternatives to securely and efficiently assess the computing performance of devices. Few other directions seem interesting for Future work. First, performing wide experiments on other types of puzzle-based and bandwidth-based schemes would be of significant value. Such an analysis, for example, would result in exact numbers for inflation factors of attacks on memory-based puzzle schemes. Second, by collecting accurate data on the distribution of computing and bandwidth resources of a great body of hosts, one can build a mathematical model for danger analysis on various currency-based mechanisms. This division can be utilizing beside with the inflation factor for building a mathematical model for the threat study. This capacity occupy, for example, increasing a puzzle whose solution requires different computational operations based on the content of the puzzle itself. Such a puzzle scheme should also ensure that clients solving different puzzles of the same difficulty should require predictable and similar

computational resources. Creating such a puzzle scheme, which is cheap for the server to generate and verify, is an open problem.

IV. PROPOSED SYSTEM

Our puzzle is based on repeated squaring and uses the puzzle proposed in as its basic building block, but "outsources "most of verification of the puzzle's solution to the prover; this is achieved without compromising the application purpose by embedding a secret – only known to the verifier – within the trapdoor exhibited by the Euler function in modular square. We offer a security evidence for this structure. To confirm the client's key, the verifier only needs to execute a marginal number of modular multiplications. A well-known countermeasure against DoS attacks are client puzzles. The maltreated server difficulty from the clients to commit computing resources before it processes their requirements. To obtain service, a client must explain a cryptographic puzzle and submit the correct answer. They are either parallelizable, coarse-grained or can be utilize only interactively. In case of interactive client puzzles where the server poses the confront an attacker might mount a counterattack on the clients by injecting fake packets containing bogus puzzle parameters.

A. Advantages

A puzzle client further takes advantage of puzzle solution packets to solicit updated puzzle parameters.

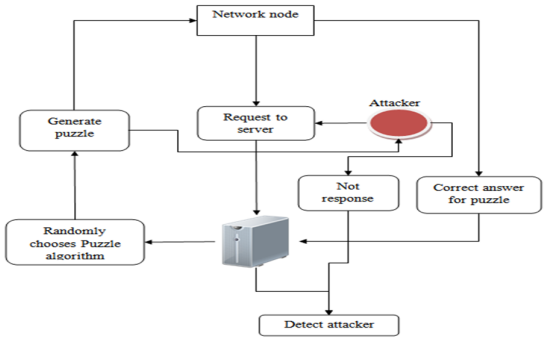


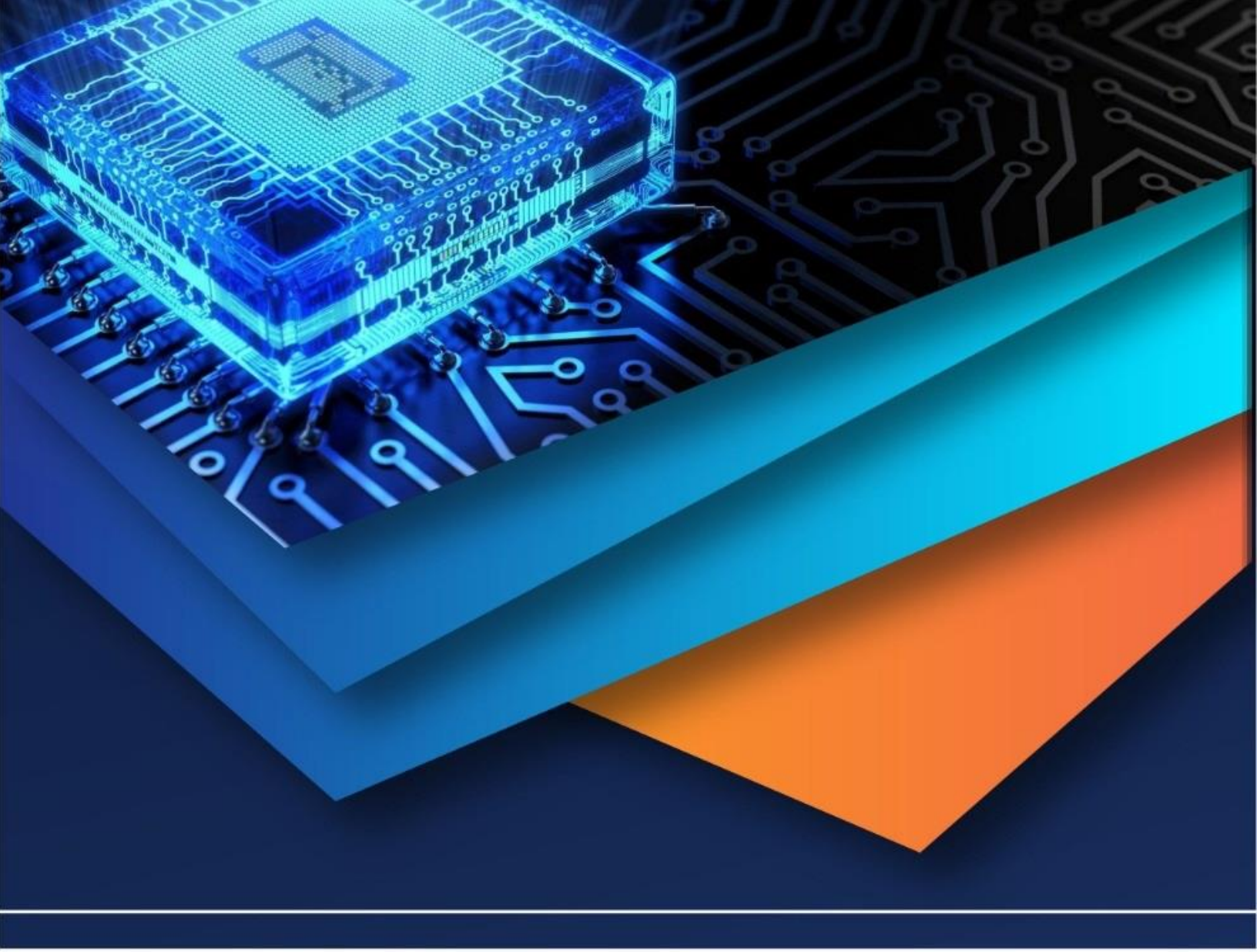
Figure 1: System Architecture

V. CONCLUSION

A new verification-efficient client puzzle based on frequent squaring. Our puzzle expands the time-lock puzzle permit a considerably more efficient verification of the puzzle solution that is reported by provers. More specifically, our scheme transfers the puzzle verification burden to the prover that executes the puzzle; we achieve this by embedding a secret – only known to the verifier – surrounded by the Euler trapdoor function that is used in repeated squaring puzzles. Given this, the development gain in the verification overhead of our puzzle when compared to the original repeated-squaring puzzle is almost 50 times for a 1024-bit modulus. We additional demonstrate how our puzzle can be integrated in a number of protocols, including those used for protection against DoS attacks and for the remote verification of the computing performance of devices

REFERENCES

[1] 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. [Online]. Available: http://hdl.handle.net/2142/17372
[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. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.110.5709
[4] W.-C. Feng and E. Kaiser, "The case for public work," in Proc. IEEE Global Internet Symp., May 2007, pp. 43–48.
[5] 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.
[6] E. Kaiser and W.-C. Feng, "mod_kaPoW: Mitigating DoS with transparent proof-of-work," in Proc. ACM CoNEXT Conf., 2007, p. 74.
[7] NVIDIA CUDA. (Apr. 4, 2012). NVIDIA CUDA C Programming Guide, Version 4.2. [Online]. Available: http://developer.download.nvidia.com/
[8] 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.
[9] 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.
[10] D. Kahn, The Codebreakers: The Story of Secret Writing, 2nd ed. New York, NY, USA: Scribners, 1996, p. 235.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY