Client Puzzle Approach for Improving Confidentiality

Ms. Sapna S. Khapre
IV SEM, M.TECH-CSE Department,
Smt. Bhagwati Chaturvedi College of Engg., Engg.,
Nagpur University

Prof. Shrikant Ardhapurkar
CSE Department,
Smt. Bhagwati Chaturvedi College of
Nagpur University

Abstract

In present scenario, security of data or information is main concerned. Different defense strategies are developed to avoid DoS-Denial of Service attack, but they don't provide optimal solution against DoS attack. Preventive mechanisms against flooding attacks can be effectively studied through game theory. This is mainly owing to the several trade-offs existing in a flooding attack defense scenario. For an attacker, there is a trade-off between the severity of his attack and the amount of resources he uses to do so; the more damage an attacker intends to cause, the more amounts of resources he should spend. For a defender, on the other hand, there is a trade-off between the effectiveness of his defense and the quality of service he provides for legitimate users; the more difficult it becomes to exhaust the defender's resources, the more workload, and hence, less quality of service is imposed on legitimate users. To improve the quality of service for legitimate user and also to improve confidentiality, puzzle based defense technique is used. With the help of puzzle based system we can avoid DoS attack.

1. Introduction

Denial of Service (DoS) vulnerabilities are one of the major concerns in today's internet. The Denial of Service attack makes a network service unavailable to its legitimate users. A denial of service attack may either be a brute force attack, where the attacker generates spurious network to exhaust server resources or a semantic attack, where the attacker exploits the vulnerabilities of the protocol used. Client-puzzles offer a mechanism for a server to counterbalance computational expenditure when subjected to a denial of service attack. On receiving a request, the server generates a puzzle of appropriate difficulty and sends it to the client. When a response is received, the server varies the solution and provides the requested service only if the solution is correct. Availability of services in a networked system is a security concern that has received enormous attention in recent years. Most researches in this area are on designing and verifying defense mechanisms against denial-of-service (DoS) attacks. A DoS attack is characterized by a malicious behavior, which prevents the legitimate users of a network service from using that service. There are two principal classes of these attacks: flooding attacks and logic attacks. A flooding attack such as SYN flood, Smurf, or TFN2K sends an overwhelming number of requests for a service offered by the victim. These requests deplete some key resources at the victim so that the legitimate users' requests for the same are denied. A resource may be the capacity of a buffer, CPU time to process requests, the available bandwidth of a communication channel, etc. The resources exhausted by a flooding attack revive when the attack flood stops. A logic attack such as Ping-of-Death or Teardrop forges a fatal message accepted and processed by the victim's vulnerable software and leads to resource exhaustion at the victim. Unlike flooding attacks, the effects of a logic attack remain after the attack until some appropriate remedial actions are adopted. A logic attack can be thwarted by examining the contents of messages received and discarding the unhealthy ones. This is due to the fact that an attack message differs from a legitimate one in contents. In flooding attacks, on the contrary, such a distinction is not possible. This causes defense against flooding attacks to be an arduous task. A large number of defenses have been devised against flooding attacks. A defense mechanism may be a reactive or preventive one. A reactive mechanism such as pushback, trace back, or filtering, endeavors to alleviate the impact of a flooding attack on the victim by detecting the attack and responding to it. A preventive mechanism, on the other hand, enables the victim to tolerate the attack without denying the service to legitimate users. This is usually done by enforcing restrictive policies for resource consumption. A method for limiting resource consumption is the use of client puzzles.

2. Weaknesses in existing definitions

A DoS countermeasure based on client puzzles should require appropriate work to be done for each client request: it should not be possible to solve many puzzles easily. While the existing models describe the difficulty of DoS countermeasures when faced with an adversary trying to solve one puzzle, these models do not adequately defend

against powerful adversaries who can expend more than the effort required to solve a single puzzle. In this section, we consider some puzzles where a single instance cannot be solved easily by an attacker, satisfying existing difficulty definitions, but where an attacker can solve n puzzles more efficiently than just n times the cost of solving a single puzzle.

2.1. MicroMint-Based Puzzle

The MicroMint micropayment scheme is effectively a client-puzzle-based micropayment scheme. A coin is a collision in a hash function: it is a pair of values x1, x2 such that H(x1) = H(x2) for a given hash function H. It is easy to verify the validity of a coin. Generating coins is harder. If H is a regular (or random) function with ℓ-bit outputs, then to find a collision one must rely on the "birthday paradox" : hash approximately $2^{\ell/2}$ distinct values and search for a collision. This puzzle can be shown to satisfy the puzzle difficulty definition of the Chen et al. model. However, many collisions can be found without too much more work: n collisions can be found with $\sqrt{n} \cdot 2^{\ell/2}$ hash function calls, much less than n times the $2^{\ell/2}$ cost of solving a single puzzle. We emphasize this is not an attack on the MicroMint scheme itself: MicroMint was in fact designed so that the amortized cost of generating multiple coins is smaller. While potentially a desirable 4 property in a micropayment scheme, this property is not desirable for client puzzles.

2.2. Generic Puzzle Construction of Chen et al.

Chen et al. proposed a generic client puzzle construction based on a pseudorandom function F and a one-way function ϕ. The challenger selects a secret $s \in K$ with $|K| = 2k$ and public parameters, denoted by $*$, to generate a puzzle. The challenger computes $x \leftarrow F(s, *)$, where $x \in X$ and $|X| \geq |K|$, and then sets $y \leftarrow \phi(x)$. The solver, given the challenge $(y, *)$, has to find a pre-image z such that $\phi(z) = y$. This generic construction satisfies the puzzle unforgeability and puzzle difficulty security properties provided certain bounds are met: namely, $|X| \geq |K|$ and $|\phi^{-1}(y)| \leq 1$ and $|X| = 2k$, for all y. Suppose we have that $|\phi^{-1}(y)| \leq 1$ and $|X| = 2k$. Then the bounds in the generic construction are satisfied and solving a single puzzle instance requires approximately 2k searches in X. But to solve n puzzles, the solver can find the value s with at most 2k searches and then obtain a solution with one application of F for each puzzle. That is, solving n puzzles would require 2k + n operations rather than the desired $n \cdot 2k$ computations.

2.3. Number-Theoretic Puzzles

Many client puzzles based on number-theoretic constructions have been presented, which uses modular exponentiation and argues for security in the Chen et al. model based on the intractability of the RSA problem. Given a puzzle consisting of an RSA modulus N, a challenge x, and a large integer R >> N, the solver must compute xR mod N. The security argument rests on the assumption that the best known algorithm for this computation requires O(log(R)) modular operations, assuming that factoring N requires more than O(log(R)) operations. But in fact a much smaller N would still suffice and would reduce the computational costs for the verifier, which is important when puzzles are used at extremely low levels in the network stack, such as TCP. Even with a smaller N, say 500 bits, the cost of solving a puzzle by computing xR mod N is still cheaper than factoring. However, if the adversary wants to solve 230 puzzles, the best technique is not to solve all these puzzles independently but to first factor N and then use this trapdoor to easily generate solutions.

3. Client puzzle approach

Currently intruders are beginning to more often use legitimate, or expected, protocols and services as the vehicle for packet streams. The resulting attacks are hard to defend against using standard techniques, as the malicious requests differ from the legitimate ones in intent but not in content. Filtering or rate limiting based on anomalous packets are not feasible at all. In fact, filtering or rate limiting an attack that is using a legitimate and expected type of traffic may in fact complete the intruder's task by causing legitimate services to be denied. The client puzzle approach provide solution to this problem. The client puzzle approach means that before engaging in any resource consuming operations, the server first generates a puzzle and sends its description to the client that is requesting service from the server. The client has to solve the puzzle and send the result back to the server. The server continues with processing the request of the client, only if the client's response to the puzzle is correct.

In client puzzle approach there are three components:

3.1. Sender
3.2. Server
3.3. Receiver

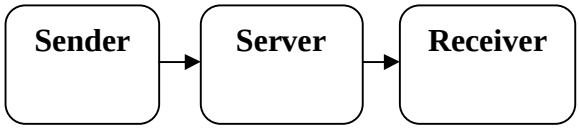

Figure 1. Client puzzle approach

3.1. Sender

Sender sends file to the intended user. In the user interface of sender, user has to first login, if user login is correct then hint to user is given to solve the puzzle. After solving puzzle correctly the user is allowed to send the file by using browse button to browse the file which sender wants to send to the user. The status of sending file is shown in status information text box of sender, then sender sends the file. The attack button is used to show DoS attack, if we press this button the selected file take more time for transmission compare to time taken for file transmission in normal environment.

3.2. Receiver

Receiver receives the required file from sender. The user interface of receiver shows the contents of receiving file and also display the message box which shows the time taken to transfer the file. It also shows the results of receiving file i.e. total size of packet, transmission rate, data loss size etc.

3.3. Server

Server gives the status of file transmission. In the user interface of server gives details of each packet present in file and number of packets in file. It also gives the status information about the delivery of the file to the receiver.

For transferring file from sender to receiver following steps take place:
Step 1: Sender enters login-id and password.
Step 2: Server generates puzzle & sends to sender.
Step 3: Sender solve the puzzle & sends back to the server.
Step 4: Server verifies the puzzle is correct or not.
Step 5: If puzzle is solved correctly and login-id and password is correct then sender can sends file to intended receiver. If puzzle is not solved correctly then service is not provided to the sender.

4. Database

Using Microsoft Access, you can manage all your information from a single database file. Within the file, divide your data into separate storage containers called tables; view, add, and update table data by using online forms; find and retrieve just the data you want by using queries; and analyze or print data in a specific layout by using reports. Allow users to view, update, or analyze the database's data from the Internet or an intranet by creating data access pages. To store your data, create one table for each type of information that you track. To bring the data from multiple tables together in a query, form, report, or data access page, define relationships between the tables. A common field relates two tables so that Microsoft Access can bring together the data from the two tables for viewing, editing, or printing. In table Design view, you can create an entire table from scratch, or add, delete, or customize the fields in an existing table.

In table Datasheet view, you can add, edit, view, or otherwise work with the data in a table. You can also display records from tables that are related to the current table by displaying subdatasheets within the main datasheet. With some restrictions, you can work with the data in subdatasheets in many of the same ways that you work with data in the main datasheet.

5. Conclusion

Game theory is used to propose a number of puzzle-based defenses against flooding attacks. Till now the DOS attack environment is implemented. It shows the interactions between an attacker, who launches a flooding attack and a defender who will counters the attack using a puzzle-based defense can be modeled as an infinitely repeated game of discounted payoffs. Database implementation and connectivity of database is also done.

References

[1] Raju Neyyan, Ancy Paul, Mayank Deshwal and Amit Deshmukh, “Game Theory based Defense Mechanism against Flooding Attack using Puzzle,” Emerging Trends in Computer Science and Information Technology, pg 5-10, no. 1, April 2012.

[2] Kumar Dayanand and S. Magesh, ”Defence Strategy against Flooding Attacks Using Nash Equilibrium Game Theory,” International Conference on Computing and Control Engineering (ICCCE 2012), April 2012.

[3] Tanmay Sanjay Khirwadkar, “Defense Against Network Attacks Using Game Theory,” University Of Illinois At Urbana-Champaign, May 2011.

[4] Mehran S. Fallah, ”A Puzzle-Based Defence Strategy Against Flooding Attacks Using Game Theory,” IEEE transactions on dependable and secure computing, vol. 7, no. 1, pg 5-19, 2010.

[5] D. Moore, C. Shannon, D.J. Brown, G.M. Voelker, and S. Savage,“Inferring Internet Denial-of-Service Activity,” ACM Trans.Computer Systems, vol. 24, no. 2, pp. 115-139, May 2006.

[6] Jelena Mirkovic, Janice Martin and Peter Reiher, “A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms,” ACM SIGCOMM Computer Communication, Vol. 34, no. 2, pp. 39 – 53, April 2004.

[7] E. Bursztein and J. Goubalt-Larrecq, “A logical framework for evaluating network resilience against faults and attacks,” Lecture Notes in Computer Science, Vol. 4846, 2007.

[8] T. Aura, P. Nikander, and J. Leiwo. “DoS-Resistant Authentication with Client Puzzles,” Lecture Notes in Computer Science, vol. 2133, 2001.

[9] Liqun Chen, Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi, "Security notions and generic constructions for client puzzles," In Mitsuru Matsui, editor, Advances in Cryptology,Proc. ASIACRYPT 2009, LNCS, volume 5912, pp. 505, 2009.

[10] Markus Jakobsson and Ari Juels, " Proofs of work and bread pudding protocols," In Bart Preneel, editor, Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security, IFIP Conference Proceedings, volume 152, pp. 258, 1999.

[11] Douglas Stebila and Berkant Ustaoglu, "Towards denial-of-service-resilient key agreement protocols," in Colin Boyd and Juan Nieto, editors, Proc. 14th Australasian Conference on Information Security and Privacy (ACISP) 2009, LNCS, volume 5594, pp. 389, 2009.

[12] Ronald L. Rivest and Adi Shamir, " Payword and micromint: Two simple micropayment schemes," in Security Protocols, LNCS, volume 1189, pp. 69, 1997.

[13] Timothy J. McNevin, Jung-Min Park, and Randolph Marchany ."pTCP: A client puzzle protocol for defending against resource exhaustion denial of service attacks," Technical Report TR-ECE-04-10, Department of Electrical and Computer Engineering, Virginia Tech, October 2004.