Playing Defense by Offense: Equilibrium in the DoS-attack Problem

A. Lukyanenko†, V. Mazalov‡, A. Gurtov†, I. Falko‡
† Helsinki Institute for Information Technology HIIT, Aalto University, Finland
‡ Institute of Applied Mathematical Research, Karelian Research Center RAS, Petrozavodsk, Russia

***Abstract*—We develop defenses from resource-exhausting Denial-of-Service attacks initiated by an attacker to a server. The attacker does not have a permanent identity but spoofs the IP addresses for other users. Generalizing the Defense-by-Offense approach we enable benign users to obtain low service time by re-submitting requests according to a game-theoretic strategy. The attacker that tries to overwhelm the server by a constant stream of requests cannot succeed as its requests are dropped by the server. We derive optimal strategies for the server, as well as the attacker. We show that in the equilibrium state, the server can successfully repel the attackers with selective processing of requests. Simulations using OMNeT++ support analytical results.**

I. Introduction

Internet every day introduces new threats which requires new defense mechanisms, while the study of the networking itself discovers new sides of the problems [7], [1], [4], [8].

In this work we present a two-person game related to protection of a network server against Denial-of-Service attacks (DoS-attacks) [6]. In the game, one player is a server and another player is an attacker. The attacker sends additional traffic to disturb the system. The server cannot identify if the visitor is benign or an attacker. The server replies to the demand following the protocol design. The standard protocol is FIFO (First In, First Out). In this case the attacker can visit the system many times with different names (spoofing the identity) and make the waiting time of benign users very large.

In this article, we introduce a novel protocol design MKFS (Most Knocking First Served). In the protocol we separate clients that already had entered the system (some initial handshake was done) and new clients that are waiting for a first server reply. This protocol deals with new clients only and aims at prevention of identity-spoofing DoS attacks. Identity spoofing means that an attacker can freely change its identity and present itself as a new client every time. In that case, the server suffers from bandwidth or resource consumption by a number of attacking packets, if it is not possible to distinguish them from benign clients' packets.

The idea behind the protocol is to serve clients based on the number of requests from the clients themselves. Hence, clients' packets are taken from a priority queue based on the number of times the client has sent the initial packet. In that case benign clients stay under same conditions as an attacker; if they want to enter the system then they need to send more initial packets. Thus, if an attacker spoofs an identity and sends only one packet, then all clients who sent at least two packets will be served before the attacker. On the other hand, if an attacker sends many packets with the same identity then it will spend some of its own capacity only for one reply. At the same time not knowing the number of packets required for one reply, an attacker can send more than required. It means that all packets that it sends after a reply do not interfere with the server's work.

In theoretical analysis, we consider a simplified model of such protocol, when an attacker uses only one identity at a time. The attacker decides only the interval during which the attacker wants to keep one identity before spoofing. Hence, that interval is identical to interarrival time of one attacking packet. In this case the strategy to visit the system with small time intervals is not a good strategy for the attacker. We consider attacker's optimal behavior and find the optimal behavior of the system for this DoS scenario. Simulation takes into account an attack with spoofing identities and supports our analytical results.

We proceed as follows. In Section II, we present the algorithm with notation used in the paper and present the network view of the algorithm implementation. In Section III, we obtain the distribution of waiting time for attackers and benign users. In Section IV, we derive optimal strategies for the players. Section V provides simulation results. Section VI concludes the paper.

II. Algorithm

The paper is devoted to a novel MKFS algorithm. An idea of "defense by offense" was suggested in [9]. However, MKFS is a completely different in the sense that it does not suggest to implement any thinner or to deal with all packets. MKFS mainly suggests a new novel queueing policy which gives the clients a right to fight to be served. This algorithm in addition uses the fact that the attacker "spoofs" its identity. Attacker's IP address cannot be distinguished and blacklisted, and the attacker cannot receive any reply on the spoofed identity. The attacker floods the system by initial packets appearing as a newcomer every time. From time to time an attacker can use real identities to send initial packets, and, hence, receive some feedback from the system, however the attacker cannot use the same identity for the same procedure again. We restrict users to enter the system only one time. Once a user enters the system (the handshake is done) the user cannot start this algorithm again and should be processed as a known user.

If creation of new identities for an attacker is not a cheap procedure then the attacker avoids revealing them.

The algorithm can be formulated as follows. For every new message a server checks if the id of a user, who sent the message, has already been processed then discard message. If the id is new, then add it to priority queue with priority 1. If the id is not new and has not been processed then it is already in the priority queue. We just increase its priority in the queue by 1 and update the queue. It was a part of processing new messages, another part of the algorithm is about replying to users, that is simply to reply to every message in the top of the priority queue, adding the id to processed list.

It is clear that MKFS should be working on the server side while clients have to implement the entering-message repeating mechanism. We will call the server-side part of MKFS scheme as *MKFS-server mechanism* and the client-side part of MKFS scheme as *MKFS-client mechanism*. For the server and the client these mechanisms can be implemented in a transparent way, if we place them outside the server/client in some daemon, router or overlay. The MKFS-client most of the time simply forwards all the messages from the client outside to the network. Whenever a client does not receive a reply message from some server (which "somehow" beforehand was added to server list supporting MKFS scheme) in the network the MKFS-client mechanism can be triggered. It starts to send continuous flow of entering messages to the MKFS server, while MKFS-server processes the incoming messages by MKFS Algorithm and forwards messages only from validated sources.

III. Analysis

Following classical works on queuing theory [5], [2] we analyze the model as follows. Consider two sequential requests from the users with times t_0 and t_1. Suppose $z = t_1 - t_0$ has the exponential distribution with parameter λ. Let them wait a service for the time γ_0 and γ_1. Denote CDF and PDF of γ_1 as $F(t), f(t)$.

Let $t_1 < t_0 + \gamma_0 + \tau$. The second request comes when the system is busy by servicing the previous request. The time until the end of service is $\gamma_0 + \tau - z$. If

$$\gamma_0 + \tau - z > \theta \tag{1}$$

then the second request is served otherwise the attacker's request is served. Thus, if $\gamma_0 + \tau - z > \theta$ then

$$\gamma_1 = \gamma_0 + \tau - z. \tag{2}$$

If $\gamma_0 + \tau - z \le \theta$ then

$$\gamma_1 = \gamma_0 + \tau - z + \tau = \gamma_0 + 2\tau - z. \tag{3}$$

Consider all events

$$\{0 \le z \le \gamma_0+\tau-\theta\} \cup_{i=1}^{\infty} \{\gamma_0+i\tau-\theta < z \le \gamma_0+(i+1)\tau-\theta\}.$$

In each set γ_1 has the form $\gamma_0 + (i+1)\tau - z$.

Now we can calculate the distribution of γ_1. Because $\gamma_1 \ge \theta$ then $F(t) = 0$ and $f(t) = 0$ for $0 \le t \le \theta$.

Note that γ_0 has the same distribution as γ_1, so

$$f(t) = \int_0^\infty \lambda e^{-\lambda u} du \Big[f(t+u-\tau) P\{z \le \gamma_0 + \tau - \theta\} + \sum_{i=1} f(t+u-(i+1)\tau) \times P\{\gamma_0 + i\tau - \theta < z \le \gamma_0 + (i+1)\tau - \theta\} \Big]. \tag{4}$$

The event $\{\gamma_0 + i\tau - \theta < z \le \gamma_0 + (i+1)\tau - \theta\}$ can be presented in the form $\{z - (i+1)\tau + \theta \le \gamma_0 < z - i\tau + \theta\}$.

Now (4) yields

$$f(t) = \int_0^\infty \lambda e^{-\lambda u} du \Big[f(t+u-\tau)(1 - F(u-\tau+\theta)) + \sum_{i=1} f(t+u-(i+1)\tau) \left(F(u+\theta-i\tau) - F(u+\theta-(i+1)\tau)\right) \Big].$$

or

$$f(t) = \int_0^\infty \lambda e^{-\lambda u} du \Big[f(t+u-\tau)(1 - F(u+\theta)) + \sum_{i=1} f(t+u-i\tau) \left(F(u+\theta-(i-1)\tau) - F(u+\theta-i\tau)\right) \Big].$$

Changing $v = t + u - i\tau$ in each term,

$$f(t) = \int_0^\infty \lambda e^{-\lambda u} du f(t+u-\tau)(1 - F(u+\theta)) + \sum_{i=1}^{\infty} e^{\lambda(t-i\tau)} \int_{t-i\tau}^{\infty} \lambda e^{-\lambda v} dv f(v) \left(F(v-t+\theta+\tau) - F(v-t+\theta)\right). \tag{5}$$

Denote

$$G(t) = \int_0^\infty \lambda e^{-\lambda v} dv f(v) \left(F(v-t+\theta+\tau) - F(v-t+\theta)\right).$$

then from (5) and because $F(t) = 0$ for $0 \le t \le \theta$ and $\forall i$ $v - t + \theta + \tau \le \theta \Rightarrow$

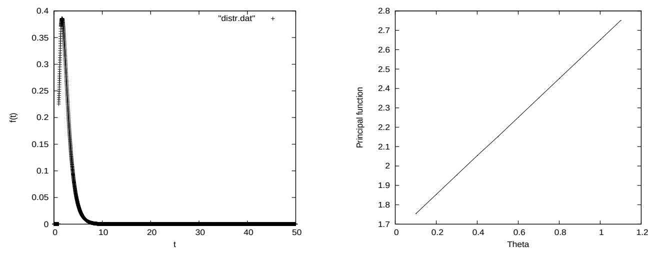
$$f(t) = \int_0^\infty \lambda e^{-\lambda u} du f(t+u-\tau)(1 - F(u+\theta)) + e^{\lambda t} \frac{G(t)}{e^{\lambda\tau} - 1}. \tag{6}$$

Here $G(t)$ is

$$G(t) = R(t-\tau) - R(t),$$

where

$$R(t) = \int_0^\infty \lambda e^{-\lambda v} f(v) F(v-t+\theta) dv.$$



(a) Distribution $f(t)$. (b) Principal function $E(1, \theta, 0.45)$.

Fig. 1. Numerical analysis.

Finally we can present (6) as

$$f(t) = e^{\lambda(t-\tau)} \left[\int_{(t-\tau)^+}^{\infty} \lambda e^{-\lambda v} f(v) dv - R(t-\tau) \right] + e^{\lambda t} \frac{R(t-\tau) - R(t)}{e^{\lambda\tau} - 1}. \quad (7)$$

We obtain the presentation $f(t)$ via integral operator R at the right side of (7). Now we can find $f(t)$ numerically for different parameters τ, θ and λ using the iterative procedure

$$f_n(t) = R f_{n-1}(t), n = 1, 2, ...$$

starting from some initial approximation $f_0(t)$. We use $f_0(t) = \exp(-(t-\theta))$. We can see the form of the function $f(t)$ in Fig. 1(a). It has a peak at point 1.69.

IV. Optimal strategies

Let us construct the optimal strategies in the game. The principal function here is $E = \int_1^\infty t f(t) dt$. It depends on λ, θ, τ. Without loss of generality let $\lambda = 1$, now we consider time not in seconds but in *units*. Fix the service time τ and let us change θ. We obtain from simulations that $E(\lambda, \theta, \tau)$ is an increasing function in θ. We can see the behavior of this function for $\tau = 0.45$ with respect to θ in Fig. 1(b). It is a linearly increasing function, hence, the optimal strategy for an attacker is to increase θ. At the same time the attacker is interested to have in the queue an infinitive number of the requests. Thus, it means that the best strategy for the attacker is $\theta = \frac{1}{\frac{1}{\tau} - \lambda}$. However, an attacker in fact does not know the value of τ, but the server knows it and the whole situation. The system strategy is τ. From one side the player presenting the system is interested to minimize the user's waiting time in queue $E(\lambda, \theta, \tau)$ and from other side the player is interested to minimize the service time τ as it maintains the service time on low level involving some costs (hardware, traffic, or computation power from third party, e.g. using cloud computing [3]). We suppose here that the improvement of the system (minimization of τ) has cost $\frac{const}{\tau}$. So, the payoff function for the system is $E(\lambda, \theta, \tau) + \frac{const}{\tau}$.

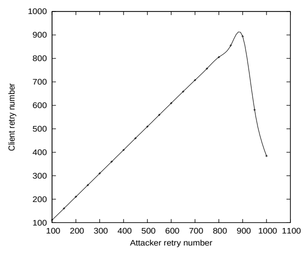
We numerically simulate this problem for $const = 1$. From the numerical analysis we found that the optimal strategy of the system is approximately at $\tau = 0.5$, i.e., a half of the intensity flow.

V. Implementation and simulation results

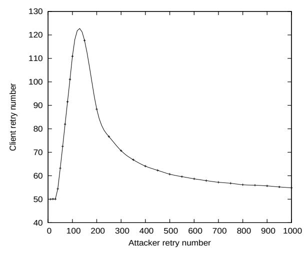
For simulation of MKFS algorithm in realistic environment we used OMNeT++ simulator. We created a simulated topology using BRITE network generator with modified source code in order to produce our own network format, which can be used to speed up the simulation process. We used 20 autonomous systems (AS) with 20 routers each, 400 nodes in total with 25% of zombie nodes (100 nodes). Every zombie node produces 100 messages per second or 10 messages per second depending on scenario. We also placed incoming clients in 5 random nodes in the network, each client is able to produce 100 messages per second. The server is able to handle only 10 messages per second (we made it quite slow); it is obvious that we have a DoS attack when a server cannot handle even one client in the system. It is made so, because we can receive data which represent real environment. If we use a server that is capable to handle 1000 messages per second, then it will have the same behavior with 10000 zombie nodes.

We implemented MKFS algorithm on server side using binary heap data structure for priority queue (every time we need only one element with the highest priority). Priority queue keeps only two copies of repeating messages received (which in our case is priority) and the id of element that sent it. All additional information, such as "enqueued", "position in queue", "is processed", "last update", is kept in additional structure — a map. Also we added two rules to the algorithm to keep the queue and the map size in reasonable size (too much memory consumption makes simulation quite slow). First of all, we added a rule that if a message from existing in map id came after 10 second delay, then we delete it and do not process the sender anymore (in reality it can be done by flushing sent data on the sender, hence, it will have to start from scratch). Secondly, we added a rule that all messages that are older than 30 seconds are purged from the queue and the map. This rule resembles the first one in a way, however, first one works when we still receive messages from the sender, and the second is global clean up rule, which we initiate only once in 30 seconds, for example.

In simulation we used a simple scenario for an attack. Globally an attacker sets the number of times before every zombie node spoofs its identity. This value, which we now call the *attacker retry number*, fully corresponds to the value of θ in the theoretical and numerical analysis in previous sections. Every zombie node generates an identity and sends it the defined retry number of times to the server, after that it generates a new identity and repeats the procedure. Every normal user, a client, enters one of 5 nodes for clients, placed randomly in the network, takes some new identity and starts to send the initial messages to the server until it receives a reply. After that reply the client is considered to have entered the system, and a new client comes to the system to repeat the entering procedure, etc. The number of the packets that a client sends to a server before receiving a reply we call the *client retry number*. It correspond to γ random variable in analysis. Here we study the average client retry number,

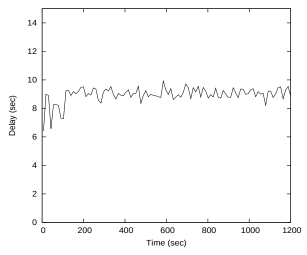


(a) Scenario A.

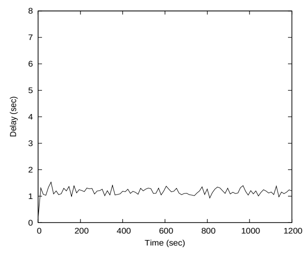


(b) Scenario B.

Fig. 2. Number of requests needed from benign client versus the number of requests by an attacker .



(a) Scenario A.



(b) Scenario B.

Fig. 3. Benign client waiting time when the attacker sends 900 requests (Scenario A) and 150 requests (Scenario B).

which corresponds to principal function of analysis above.

We divided simulation of the attacking model to two scenarios based on the speed of zombie nodes, one where the zombie nodes work with the same speed as client nodes. Hence, all of them are in the same conditions, except that zombie nodes do not know when their current identity was replied, if it was replied. We call this scenario Scenario A. The second scenario is, when zombie nodes are 10 times slower than client nodes. This is a reasonable assumption, as zombie nodes are in many cases hacked computers, with the owner unaware that the computer is attacking something. Hence, by aggressive performance a zombie node can reveal itself to owner by warnings from firewalls or antiviruses. Clients on another hand are not restricted by those conditions as they by themselves decide to pursue that algorithm. Hence, clients should have more opportunities to increase traffic size. This scenario we call Scenario B. In Scenario A clients and zombies send with speed of 100 messages per second. In Scenario B clients send with speed of 100 messages per second, while zombies with speed 10 messages per second.

Fig. 2(a) shows the dependency of client retry number on the attacker retry number (in Scenario A). As we can see, behavior corresponds to analytical results in Fig. 1(b), which was also linear for θ during fulfill of instability condition $\theta < \frac{1}{\frac{1}{\tau}-\lambda}$. In Fig. 2(a) linearity of the dependence stops almost at the point, when the condition of overloading of server stops holding. Fig. 2(b) shows the same behavior for Scenario B. As we can see in this case in the beginning we also have linear dependency, however, in Scenario B the load from zombie machines is 10 times less and the instability condition breaks earlier; as we can see in Fig. 2(b) it happens when the attacker retry number is about 150. But again the behavior fully corresponds to analytical results, while the condition holds.

We additionally studied the delay for a client to enter a system. The results are shown only from first node, where clients appear, as all of the nodes behave in resulting simulation identically. Fig. 3(a) shows the delay for the worst situation in Scenario A, when the zombie retry number is equal to 900, Fig. V shows the delay for the worst situation in Scenario B, when the zombie retry number is equal to 150. As we can see in spite of the fact that the server is responsible to answer a small fraction of requests, the server still replies in reasonable time to clients from different nodes. If the MKFS algorithm were not used and traditional ideal (infinite) queue were used, then in Scenario A, clients would have to wait more than $\frac{(V_I - V_O)t}{V_O} = 999t$ on the average, where t is the time when the client enters the system after a start of the DoS attack, V_I overall incoming number of messages to the system per second, V_O overall outgoing number of messages per second, and wait more than $99t$ in Scenario B.

VI. Conclusions

In the paper we introduced a novel MKFS algorithm. In spite of the algorithm is being easy-to-understand and easy-to-use, theoretical analysis of the algorithm is quite complicated. In the paper we showed some analysis for the algorithm, with numerical results where needed. We estimated the optimal behavior of the attacker and payoff representation of the server function. Our analysis is supported by simulation results.

Additionally, if the DDoS attack goes against the link capacity, the same MKFS algorithm can be easily distributed to a set of routers, that do not forward replies, but only the number of message replies.

References

[1] A. Akella, S. Seshan, R. Karp, S. Shenker, and C. Papadimitriou. Selfish behavior and stability of the internet: a game-theoretic analysis of TCP. *SIGCOMM Comput. Commun. Rev.*, 32(4):117–130, 2002.

[2] S. Asmussen. *Applied Probability and Queues*. John Wiley and Sons, New York, 1987.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616, 2009.

[4] R. Garg, A. Kamra, and V. Khurana. A game-theoretic approach towards congestion control in communication networks. *SIGCOMM Comput. Commun. Rev.*, 32(3):47–61, 2002.

[5] A. Y. Khinchin. *Mathematical methods of the theory of mass service*, volume 49 of *Trudy Mat. Inst. Steklov.* RAN USSR, Moscow, 1955.

[6] J. Mirkovic and P. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, 2004.

[7] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. *IEEE/ACM Trans. Netw.*, 14(4):725–738, 2006.

[8] S. J. Shenker. Making greed work in networks: a game-theoretic analysis of switch service disciplines. *IEEE/ACM Trans. Netw.*, 3(6):819–831, 1995.

[9] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker. DDoS defense by offense. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 303–314, New York, NY, USA, 2006. ACM.