Available online at www.sciencedirect.com

ScienceDirect

Computer Networks 51 (2007) 1013–1030

Computer Networks

www.elsevier.com/locate/comnet

Evaluation of a low-rate DoS attack against iterative servers

Gabriel Maciá-Fernández *, Jesús E. Díaz-Verdejo, Pedro García-Teodoro

Department of Signal Theory, Telematics and Communications E.T.S. Computer Science, University of Granada, c/Daniel Saucedo Aranda, s/n, 18071 Granada, Spain

Received 3 April 2006; received in revised form 21 June 2006; accepted 5 July 2006
Available online 2 August 2006

Responsible Editor: Christos Douligeris

Abstract

This paper presents a low-rate DoS attack that could be launched against iterative servers. Such an attack takes advantage of the vulnerability consisting in the possibility of forecasting the instant at which an iterative server will generate a response to a client request. This knowledge could allow a potential intruder to overflow application buffers with relatively low-rate traffic to the server, thus avoiding the usual DoS IDS detection techniques. Besides the fundamentals of the attack, the authors also introduce a mathematical model for evaluating the efficiency of this kind of attack. The evaluation is contrasted with both simulated and real implementations. Some variants of the attack are also studied. The overall results derived from this work show how the proposed low-rate DoS attack could cause an important negative impact on the performance of iterative servers.

Keywords: Denial of service; Low-rate attack; Network security; Iterative servers; Intrusion event

1. Introduction

Nowadays, denial of service (DoS) attacks are one of the most serious security problems in Internet, due to the fact that they threaten not only technical aspects of trade but also give rise to financial expenses. In recent years, many companies have been affected by this kind of attack, e.g. eBay, Amazon and Buy.com [1]. The increasing menace of these attacks parallels the difficulty in detecting, preventing and neutralising their effects. Moreover, modern DoS attacks use a distributed paradigm (DDoS) to achieve their purposes [2], which makes the process of detection and prevention even more difficult. Some examples of these attacks are presented in [3].

The aim of DoS attacks is to exhaust a resource in the target system, reducing or completely subverting the availability of the service provided. A common strategy used by an intruder to cause a DoS on a given target is to flood it with a continuous stream of packets that exhausts its connectivity. DoS attacks that use this kind of strategy are called brute-force attacks [2]. On the other hand, many

* Corresponding author. Tel.: +34 958 24 23 05; fax: +34 958 24 08 31.
E-mail addresses: gmacia@ugr.es (G. Maciá-Fernández), jedv@ugr.es (J.E. Díaz-Verdejo), pgteodor@ugr.es (P. García-Teodoro).

1389-1286/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2006.07.002

attacks rely on the exploitation of a specific vulnerability in such a way that it results in a denial of the service.

Many efforts have also been made, in parallel with the evolution of DoS attacks, in the field of prevention and detection in networking security. In terms of prevention, some of the approaches that have been proposed include egress [4] or ingress filtering [5], disabling unused services [6], and honeypots [7]. However, although prevention measures offer increased security levels, they cannot completely eliminate the risk of an attack. It is advisable to establish an intrusion detection system (IDS) [8] aimed at detecting attacks that seek to bypass prevention techniques. Many proposals have been made for IDSs designed to detect DoS attacks [9–11], most of these being based on the statistical detection of high traffic rates coming from the intruder or intruders.

As a main contribution of the present paper, a new application level DoS attack [12] is described. The attack is able to succeed in defeating an iterative server only by sending it low-rate traffic, in an intelligent way. Thus, it would be able to bypass detection mechanisms based on the monitoring of high-rate traffic. Recently, Kuzmanovic et al. presented in [13] a low-rate TCP attack that has similar characteristics to the one presented here; some methods for detecting it have since appeared [14–17]. Fundamentally, both types of attack (TCP and application attack) try to take advantage of the vulnerability caused by the possibility of an intruder being aware of a specific time value concerning a protocol or application. Thus, both attacks inflict an ON/OFF waveform attack that results in overall low-rate traffic but with high efficiency in service denial. However, despite certain similarities, there are key differences between them. In the first place, the attack presented in [13] is TCP-targeted, whilst ours works at the application level. Secondly, Kuzmanovic's attack attempts to trigger the TCP congestion control mechanism by creating outages in a link, while ours simply seeks to overflow a service running in a machine, not creating any network congestion at all. There are also differences in the vulnerability exploited in the two cases. In the TCP-targeted low-rate case, the knowledge of the RTO timer for congestion control implemented in TCP is exploited, whilst in our case, inter-output times are the key to building the attack, as shown below in Section 4. But the main difference between TCP and application attacks lies in the fact that although the former generates denial in some TCP flows, another flow with a special congestion control configuration could eventually bypass the attack and thus find the whole capacity of the link usable. In other words, the link capacity is free almost all the time. However, in our proposal the server is kept busy all the time, creating among legitimate users the perception that the server is not reachable. This latter feature is similar to the behaviour of the `Naptha` attack [18], although the main difference is that `Naptha` is carried out as a brute-force attack (high-rate traffic), while ours uses low-rate traffic to achieve its purpose. This discussion leads us to conclude that these attacks are not similar or comparable but complementary.

The contributions of this work can be summarized as follows: firstly, a new type of DoS attack, mainly characterized by low-rate traffic, and which works at the application level, is described. Secondly, a mathematical framework to model the behaviour of these kinds of attacks is proposed. Finally, the potential effects of the attack on both simulated and real applications are analyzed.

The present work is focused on iterative application servers. Whilst concurrent servers are able to process multiple service requests simultaneously in time, iterative servers are limited to handle just one at each instant. In other words, concurrency in services implies a "parallel operation", while iterativity is something like a "serial operation" in the server. Although the iterative server case is not as far reaching as the concurrent server case, the contributions of this study are intended to establish a basis for building future and more complex strategies for low-rate DoS attacks on concurrent servers.

The structure of this paper is as follows: the following section presents a brief study of the vulnerability exploited by the introduced attack. In Section 3, the fundamentals and technical details of the attack are described. Section 4 is focused on outlining a mathematical model to characterize and analyze the attack. In Section 5, all the results obtained from the experiments in simulated and real environments are compiled and discussed. Finally, some conclusions are drawn and suggestions for future work are made.

2. Scenario of analysis

Before addressing the description and analysis of the new low-rate DoS attack, let us briefly describe the scenario considered in this study. It consists of a

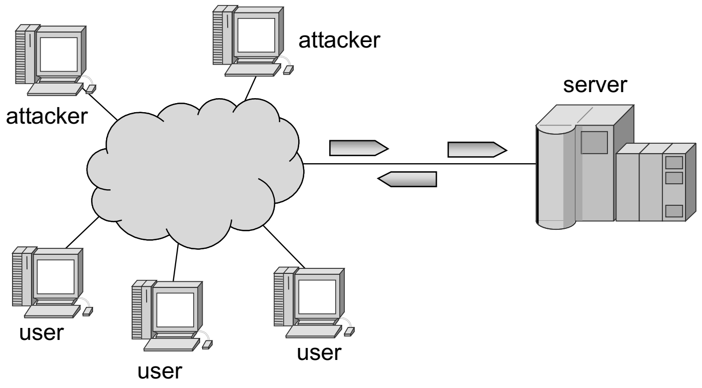


Fig. 1. Scenario of study.

generic client–server configuration in which an iterative server will receive aggregated traffic coming from both legitimate users and intruders. The incoming traffic to the server corresponds to requests for a specific service offered by the server (see Fig. 1). The sources of traffic reach the server through a generic network; the details concerning the network topology and structure are irrelevant for the purposes of the present study.

In this scenario, a preliminary simplification can be made related to the number of legitimate users accessing the server. As recommended in many teletraffic studies [19], the requests arrivals from a specific user are modelled using a Poisson distribution. Consequently, the distribution of user inter-arrival times, t_a, corresponds to an exponential probability (see [19]):

$$P(t_\mathrm{a} = t) = \lambda \cdot \mathrm{e}^{-\lambda t}, \tag{1}$$

where λ represents the mean arrival rate of packets from the user. Moreover, it has been shown in [20] that the aggregate traffic from n users whose inter-arrival times are exponentially distributed, with rates λ_i, results in a traffic pattern with inter-arrival times that also follow an exponential distribution, with the following inter-arrival time:

$$\lambda = \sum_{i=1}^{n} \lambda_i. \tag{2}$$

This means that, for the purposes of our study, it is possible to consider the aggregate traffic from n users as a single flow from one user that generates exponentially distributed traffic at a mean rate λ that complies with the above expression.

Concerning the number of attackers in the scenario, this is exclusively determined by the manner in which the intruder/s launch the attack. If a distributed DoS technique is used, this number will be high, otherwise just one attacker should be considered. As shown in Section 4, the conclusions drawn from this study are valid whatever attack technique is used. Therefore, for simplicity, we shall consider that the intruder launches the attack from a single point.

3. Vulnerability analysis in iterative servers

The server in the scenario shown in Fig. 1 is modelled as a black box where service requests are received from clients and specific responses are generated and sent to them (see Fig. 2). Thus, there exists a service queue where the clients' requests are sequentially stored while waiting for the server to process them. The waiting interval in the queue is called *queue time*, t_q. Since the service queue has a finite length, the server will raise an *overflow* message or event if there are no free positions in the queue when a new request arrives. The specific type of overflow notification is irrelevant for our study. Moreover, for the purposes of this work, the type of queue discipline used is not relevant either.

In the proposed model, a *service module* takes the requests, in order, from the service queue and processes each of them during a *service time* t_s.

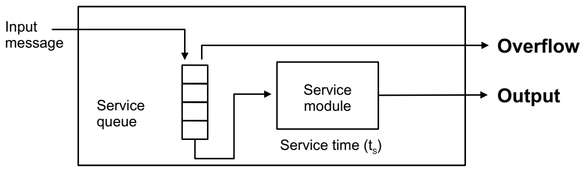


Fig. 2. Model for an iterative server.

The service module remains busy until a given input request is processed, while the other requests wait in the service queue, that is, an iterative operation is assumed. Once the request has been correctly processed, the server generates a message transmitting the appropriate answer to the corresponding source client. Henceforth, this message is termed an *output*.

In this context, a DoS attack could be aimed at filling the service queue with requests coming only from the intruder, preventing legitimate users from entering their requests in the queue. This aim is usually achieved by flooding the server with request packets, in the hope that most of the queued requests come from the attacker. This fact is observed by legitimate users as a denial of the offered service. The paradox is that, although the server is serving requests all the time, legal users cannot benefit from this.

Traditionally, the task of flooding the service queue has been carried out by means of a so-called *brute-force* attack, that is, the intruder (or intruders) sends requests to the server as fast as possible. Although the aim of the attack presented in this paper is the same, namely to "capture" the highest possible number of positions in the queue, the methodology used by the attacker is slightly different. In our case, the intruder predicts the instant when an output is raised in the server and, therefore, a free position in the service queue appears. This mechanism would allow the intruder to choose the key instants at which to send requests to the server, and thus to improve the efficiency of the attack, in comparison with the case in which no knowledge of the behaviour of the server is possessed. Thus, whenever an intruder gets information about the instant when a target server is going to raise an output, this knowledge allows an attack on the server in an intelligent way by the generation of low-rate traffic of request packets.

Therefore, the key to success in denying the service relies on the ability to forecast the optimum instant when requests should be sent in order to acquire newly freed positions. In this way, the attack would eventually capture all the positions in the queue, thus causing the desired effect of denial of service to legitimate users. Obviously, the first question we must ask ourselves is, is it possible to forecast the instant when a position in the service queue is going to be freed? Our hypothesis is that, under certain circumstances, and by a simple inspection of the outputs generated by the server, this is not only possible but also relatively easy. In fact, this is the vulnerability we point to.

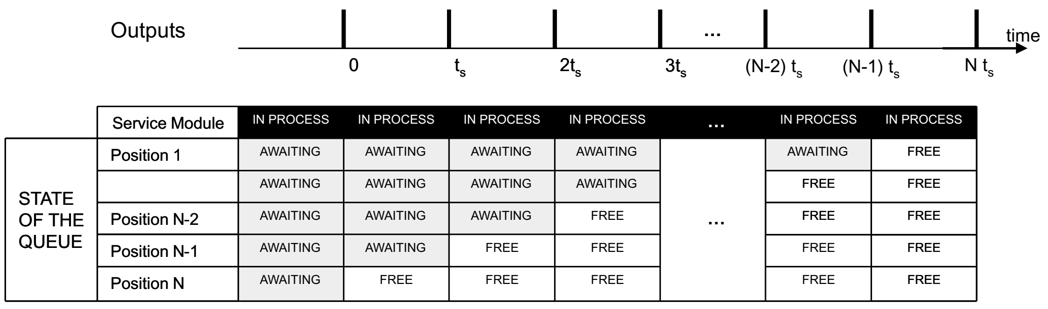
3.1. Inter-output time

Instead of addressing the task of forecasting the specific instant when an individual output will be generated by the server, we could tackle the problem by considering the time elapsed between two consecutive outputs from the server, that is, the *inter-output time*. In order to predict this time, it is necessary to analyze the complete process followed by a client request when it arrives at the server. This process is formalized as follows.

When a request enters the server, it is stored only if the service queue has free positions. Otherwise, an overflow message or event is raised. A queued request waits in the queue during a queue time t_q. After t_q, the request passes through to the service module, where it is parsed, processed and served during a service time t_s. Finally, the server sends the response to the client. After this, and if the service queue is not empty, the next request in the queue is fetched by the service module. At this point, a free position appears in the service queue. It is important to notice in the overall process that the instants when the free positions appear coincide with those at which the outputs are raised by the server. Because of this, the problem of forecasting the instants when a position is freed in the service queue is reduced to that of ascertaining the moments when an output is going to be raised.

In short, we can conclude that the knowledge of the inter-output time τ, defined as the time elapsed between two consecutive outputs in an (iterative) server, constitutes a vulnerability that could be exploited by an attacker. For clarity, let us examine a simple scenario to discover how the intruder could guess the inter-output time. In this scenario, a fixed service time is considered. Although, at first glance, this might seem a very restrictive case, it will be shown that the results are also valid for more complex cases in which this restriction is not made.

The scenario of interest consists in a server with N positions initially occupied in the service queue. We study the period of time for which all the requests are served, assuming that no new requests enter the system. It is clear that, during the observation period, the service module is always engaged in the processing of a request. Both the state of the service queue, with a per-position detail, and the timing of the outputs generated by the server are shown in Fig. 3. It can be observed that when a given request is in the "in process" state, the others are either waiting or free, due to the fact that the server is iterative.



	Service Module	IN PROCESS	IN PROCESS	IN PROCESS	IN PROCESS	...	IN PROCESS	IN PROCESS
STATE OF THE QUEUE	Position 1	AWAITING	AWAITING	AWAITING	AWAITING	...	AWAITING	FREE
		AWAITING	AWAITING	AWAITING	AWAITING		FREE	FREE
	Position N-2	AWAITING	AWAITING	AWAITING	FREE		FREE	FREE
	Position N-1	AWAITING	AWAITING	FREE	FREE		FREE	FREE
	Position N	AWAITING	FREE	FREE	FREE		FREE	FREE

Fig. 3. Time diagram of the state of the requests in the service queue (bottom) and the associated timing of the generated outputs (top). In black, service time.

Thus, a request enters the service module at $t = 0$, reaching the "in process" state. After t_s, the position is freed and the next request is fetched. Again, after t_s the other position is liberated and a new request comes into the service module. This mechanism continues until the service queue is empty. It is noticeable that in the complete process the inter-output time does indeed correspond to the service time, and is not influenced by the queue time.

The service time required by incoming requests normally varies depending on the specific resource or operation solicited. Moreover, due to the different service times required, the inter-output time also varies from one output to the next. However, if most of the incoming requests to the server come from an attacker, the duration of the inter-output time can be set to a fixed value by always asking the server for the same resource. In this latter scenario, the service time could be considered fixed and deterministic, as in the example shown in Fig. 3. However, even in this case, the main hypothesis in our study is that the service time behaves like a random process, T_s, due to the fact that the server processor is usually running other processes at the same time, which introduces variations into the value of the service time. Moreover, we hypothesize that the variations, in a stable load machine, will be low. Thus, we model the random process that represents the service time, T_s, by a normal distribution $\mathcal{N}(\overline{t_s}, var(t_s))$. The choice of the normal distribution is derived from the central limit theorem [21], because of the high number of random variables that contribute to T_s (e.g. number of processes or threads, memory occupation, disk utilization, etc.).

Under this hypothesis, the expression for the inter-output time τ, when all the requests are equal, corresponds to:

$$\tau = \mathcal{N}(\overline{t_s}, var(t_s)). \tag{3}$$

Once we have established the relation between the service time and the inter-output time, a mechanism could allow a potential intruder to acquire knowledge about the service time, based on the observation of the inter-output time. The estimation process can be performed by simply sending a set of probes, each one consisting of two consecutive requests, very close in time to ensure that they enter the target service queue consecutively, and, after that, observing the time elapsed between the reception of the corresponding outputs generated by the server. Note that the value of the inter-output time perceived by the intruder will differ from that directly observed in the server. This fact is due to the influence of the round trip time (RTT) experienced by the packets that traverse the portion of the network between the intruder and the server. Nevertheless, if all packets are assumed to be affected by this time, we can conclude that the round trip time will not modify the mean value of the observed inter-output time. However, this does introduce an additional variance into the general random process.

Some studies have shown that it is possible to represent the different values obtained for RTT as a random process modelled by a truncated normal variable [22]. This fact allows us to suppose that the estimation of the inter-output time made by a potential intruder located in the network, τ_{int}, also has a normal distribution, with the following characteristics:

$$\tau_{\text{int}} = \mathcal{N}(\overline{t_s}, var[t_s] + var[RTT]) \tag{4}$$

from which it can be stated that the round trip time does not affect the mean value of the observed inter-output time, but only its variance.

Finally, it is important to notice that the service module must always be engaged serving requests as a necessary condition to justify the above assumptions and conclusions. In other words, the service queue must always have at least one pending request.

3.2. Experimental validation

The scenario presented in Section 2 has been implemented in a simulated environment. Thus, Network Simulator (NS2) [23] has been used to check whether expression (4) is a good approximation for the observed inter-output time, τ_{int}. For this purpose, we have implemented a new class derived from the application class that behaves like a server (*server class*), which makes traces and logs to track all the events and statistics necessary for our study.

Several experiments have been carried out to confirm the expected values for the observed inter-output time, as presented above. The case where the service time and the round trip time are modelled with a normal distribution has been considered. As shown below, the results obtained are very promising, being very close to those predicted.

Fig. 4 shows some simulation results. In this case, the service module has been kept busy processing requests. To achieve this, a traffic with an inter-arrivals mean time of $\overline{t_a} = 1.0$ s has been offered to the server. Other values used in the simulation are $\overline{t_s} = 1.5$ s, $\text{var}(t_s) = 0.02$ s and $RTT = \mathcal{N}(0.6\text{ s}, 0.02\text{ s})$. Forty positions are considered for the service queue. It should be noted that $\overline{t_a}$ has been adjusted so that there is always at least one request in the service queue, that is, the arrival rate is greater than the service rate.

The simulation for a time period of 1000 s provides 694 outputs. The expected observed inter-output time distribution is $\mathcal{N}(1.5\text{ s}, 0.04)$, as derived from Eq. (4). The mean value obtained from the simulation for the inter-output time is $\overline{\tau_{\text{int}}} = 1.520$ s, with a variance of $var(\tau_{\text{int}}) = 0.041$ s, which accurately approximates $var[t_s] + var[RTT]$, as shown in expression (4).

We have also checked, through the Kolmogorov–Smirnov test of similarity [24], that the histogram obtained for the inter-output times approximates the normal probability function (see Fig. 4(b)). The value obtained for a significance level equal to 20% (very restrictive case) is 0.034, which indicates that a normal distribution is a good approximation for the histogram obtained.

We are also interested in the behaviour of the system when the service module is not always engaged in processing requests. This is illustrated in the following example. The parameters are the same as in the previous experiment, but the mean inter-arrival time $\overline{t_a}$ is changed to 1.5 s. This corresponds to the same value as the one for $\overline{t_s}$ in order to maintain requests in the service queue, although there may be instants at which the service queue becomes empty.

Fig. 5(a) represents the observed inter-output times obtained for this experiment versus the

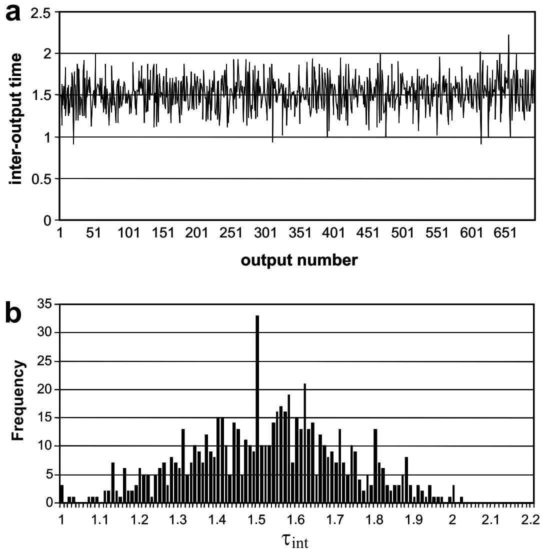


Fig. 4. Simulation of observed inter-output times in a flooded service queue scenario: (a) inter-output time values and (b) histogram of the samples.

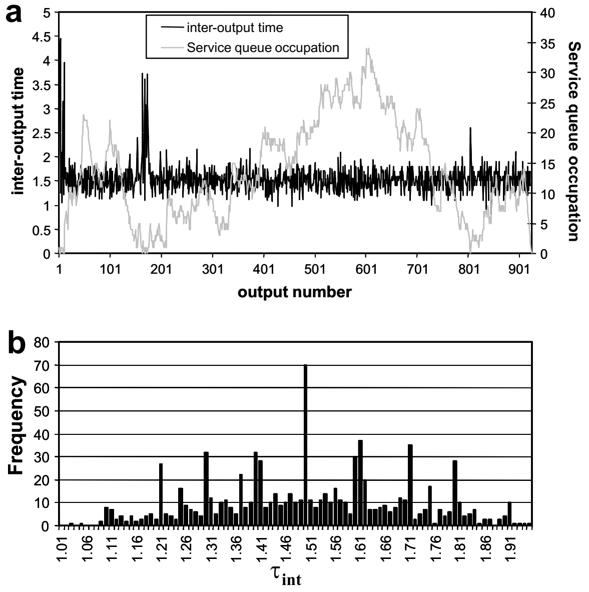


Fig. 5. Simulation of observed inter-output times in a non-flooded service queue scenario: (a) inter-output time values, and service queue occupation level and (b) histogram of the samples.

occupation of the buffer in the secondary axis (dashed lines). It can be observed that the periods with null occupation of the service queue result in sporadic peak values for the observed inter-output time.

The simulation for a period of 1400 s provides 923 outputs. The mean value obtained is $\overline{\tau_{\text{int}}} = 1.536$ s, and the variance $var(\tau_{\text{int}}) = 0.114$. The deviation is now greater than that obtained in congestion conditions, due to the peak values generated when the buffer is empty. As expected, the histogram obtained in this case (Fig. 5(b)) does not fit the Kolmogorov–Smirnov goodness of fit test, even for very low significance level values (0.5%).

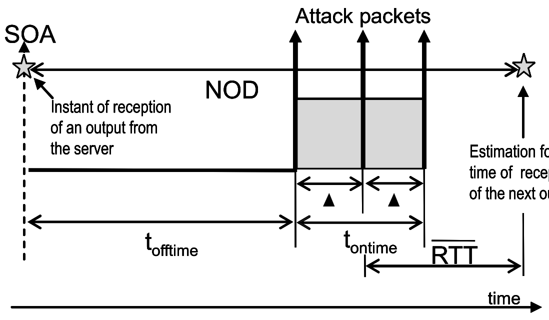
The results obtained from this set of experiments show that the assumptions made for the functioning when the service module is always engaged are good enough. They also show that, even in simulated scenarios in which the service time is variable as a normal distribution, the inter-output time may still be predictable for an observer. This makes it possible to build up an attack on the basis of this vulnerability.

4. Low-rate DoS attack specification

In this section, a proposal is made for building a low-rate DoS attack that takes advantage of the previously presented vulnerability. The first step in building the attack is to estimate the inter-output time, following the process described in the previous Section. The intruder uses this knowledge to send effective requests to the server. It is important to note that all the requests generated by the attacker should be of the same type so as to minimise variations in the inter-output time when the majority of positions in the service queue are occupied by these requests.

The low-rate denial of service attack is focused on maintaining the destination service queue occupied with malicious requests for as long a period as possible. Thus, requests coming from legitimate users will find no free positions in the service queue, and an overflow message or event will result. As explained above, the intruder seeks to forecast the instant when the next output will happen, and thus focuses the attack only during a short period around the predicted instant, in order to "capture" the newly freed position in the queue. Only in this way can low-rate traffic efficiently flood and, therefore, deny the service.

In consequence, the proposed attack is designed to follow an ON/OFF pattern, that is, it comprises



Attack packets
SOA
NOD
Instant of reception of an output from the server
Estimation for the time of reception of the next output
Δ
Δ
t_{offtime}
t_{ontime}
$\overline{\text{RTT}}$
time

Fig. 6. Attack specification: waveform and parameters.

a succession of consecutive periods composed of an interval of inactivity, called *offtime*, followed by an interval of activity, called *ontime*, as depicted in Fig. 6. The attack waveform is characterized by the following parameters:

- *Start of the attack period* (*SOA*): each attack period starts with the perception by the intruder of the occurrence of an output generated by the server. The reception of a response from the server delimits a new period of attack. However, it is important to remark that the beginning of the attack period is not always determined this way. In effect, the intruder will receive a packet containing an answer from the server only when this corresponds to a request previously sent by the intruder. Otherwise, the response would be sent to the corresponding origin, in such a way that the intruder would not notice this fact. In this case, the attack period will start at the estimated instant at which the output should arrive at the intruder.
- *Next output distance* (*NOD*): this is the time elapsed from the start of an attack period until the predicted instant value for the reception of the next output. Considering expression (4), this time will correspond to:

$$NOD = E[\tau_{\text{int}}] = \overline{t_s}. \tag{5}$$

- *Interval* (Δ): the period of time between the sending of two consecutive packets during *ontime*.
- *Ontime time* (t_{ontime}): the activity interval during which an attempt to seize a freed position in the service queue is made by emitting request packets at a rate given by $1/\Delta$. The value of t_{ontime} should be proportional to the variance of τ_{int}, the proportion factor being a design parameter of the attack.
- *Offtime time* (t_{offtime}): the inactivity interval before *ontime* in the period of attack, and during which there is no transmission of attack packets.

The intruder should adjust both t_{ontime} and t_{offtime} with the aim of synchronizing the middle of *ontime*, at the moment of arrival at the server, with the instant of the generation of the corresponding output. An additional difficulty in achieving this goal is that the intruder must consider the effects of the delay introduced by the round trip time to the server. To take this fact into account, the intruder schedules the middle of *ontime* $\overline{RTT}$ seconds before the estimated time for the reception of the next answer from the server, as shown in Fig. 6. Therefore, the value for t_{offtime} should be

$$t_{\text{offtime}} = NOD - \frac{t_{\text{ontime}}}{2} - \overline{RTT}. \tag{6}$$

Having described the philosophy of the attack, let us now describe in detail how it is effected over time. Starting from an initial prediction about the inter-output time of the target server, the attack takes place in two phases. Firstly, there is a transitory phase in which the intruder attempts to fill all the positions in the service queue. This process could be achieved through a brute-force strategy, although this is not the best solution if the intruder wants to bypass potential security detection mechanisms adopted at the destination. Instead, the intruder could use the attack dynamics proposed below, which are still effective, although requiring more time to finish the first phase.

After flooding the service queue, the second phase of the attack starts. The objective now is to maintain as many positions in the service queue as possible. This is carried out by sending a new request in such a way that it arrives at the server in the minimum possible time after a position is freed. To do so, the attack waveform presented in Fig. 6, synchronized with the outputs from the server, is used.

A detailed study of the dynamics for an attack period is made in the following. Fig. 7 shows a representation of the different events and parameters in an attack period over time, from the intruder's perspective as well as that of the server. Just after the reception of an output from the server (O1), an attack period starts (SOA_1) and the next output distance (NOD_1) is obtained, giving, along with the estimation of $\overline{RTT}$, the value of t^1_{offtime} for this period. At the end of this inactivity period, t^1_{ontime} starts by emitting an attack packet (A2). Within the *ontime* period, an attack packet is sent every *interval*, Δ (A3 and A4). When the timer associated to NOD_1 expires, a new attack period starts (SOA_2), again with a calculation for the corresponding parameters NOD_2 and t^2_{offtime}.

If no response is received from the server during this *offtime* time, the new attack period behaves in the same way as the previous one. This situation occurs whenever the response received from the server corresponds to a previous queued request coming from a legitimate user instead of from the intruder. However, when a response is received from the server, as illustrated in Fig. 7 (O2), the attack period restarts ($SOA_{2'}$) and new parameters $NOD_{2'}$ and $t^{2'}_{\text{offtime}}$ are calculated. This reset mechanism allows the attack to resynchronize each time it fails in the prediction of the timing of the outputs.

In accordance with the explained procedure, whenever an output is received by the attacker, the attack period is restarted, independently of the running interval (*offtime* or *ontime*) at the instant of reception. Moreover, upon this output arrival (O1 or O2), an attack packet is sent as a response

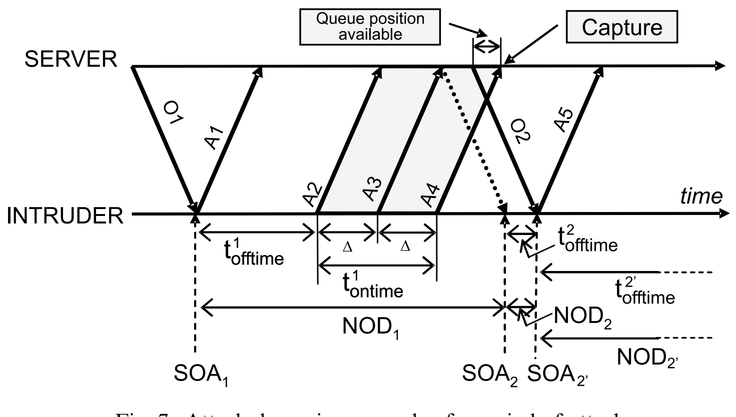


Fig. 7. Attack dynamics: example of a period of attack.

to it (A1 and A5). The reason for the sending of this packet is to minimize the time during which the newly freed position in the queue is available, in case the packets sent during the *ontime* period did not succeed in the seizure.

In summary, two possible events may trigger the start of a new attack period. First, the reception of a new output packet by the intruder. This fact starts a new period even if the current one has not finished. Second, the expiration of the *NOD* timer. In this latter case, the period is restarted but the intruder will not send an attack packet at the beginning of the attack period because it is not completely clear that a queue position has been freed. This behaviour reduces the traffic rate of the attack.

From the above procedure, the attack is expected to achieve good efficiency results, while the rate of the traffic involved is significantly reduced in comparison with a brute-force attack. Two key questions remain to be clarified: how efficient is the proposed strategy and what traffic rate is needed to achieve success? To answer these, a detailed evaluation is presented in the next section.

5. Evaluation of the attack

As previously indicated, the evaluation of the attack must respond to two main questions. On the one hand, what degree of denial of service could an intruder achieve? And on the other hand, what rate of traffic is necessary to succeed in this purpose? The answers to these questions are presented below.

5.1. Indicators for evaluating the attack

In order to measure the traffic rate needed to carry out the DoS attack, we define the *effort of the attack (E)*, as the ratio between the traffic rate generated by the intruder and the maximum traffic rate accepted by the server. It is important to note that although E measures the relative traffic rate involved in the attack, this parameter does not give an idea of the congestion level in the server, due to the fact that it depends not only on the traffic coming from the intruder but also on legitimate users' requests.

To measure the level of DoS suffered by legal users, let us examine the simplification made in Section 2, which allows us to consider a single user, who generates all the aggregated traffic coming from all the legitimate users. With this consideration in mind, we define the *user perceived performance* (*UPP*) as the ratio between the number of user requests served by the server, and the total number of requests sent by this user. This parameter gives us an idea about the service level experienced by legitimate users.

Although *UPP* is a good indicator of the DoS attack effectiveness, it depends on the traffic pattern of the legitimate users. Alternatively, we could consider a measure of the effectiveness of an attack independently of the user traffic pattern. This hypothetical indicator would enable a comparison between different design strategies and would make it easier to take a decision about the values to adopt for the attack parameters. With this fact in mind, we define, in a scenario free of user traffic, the *mean idle time* ($\overline{T_{\text{idle}}}$), as the average time during which a free position in the service queue is available within an attack period. $\overline{T_{\text{idle}}}$ is an objective measure about the efficiency of the attack, because by using it, the probability of a legitimate user seizing a position could be deduced.

The aim of the attack, in terms of the three indicators defined (see Table 1), should be to minimize the user perception on the availability of the service (*UPP*). This task can be achieved by minimizing the *mean idle time* in the server, which reduces the probability of a legitimate user seizing a position in the queue. Additionally, the attack should minimize its *effort (E)*, thus making the attack less detectable by intrusion detection systems. As an expense, it is expected that the reduction in the *mean idle time* will increase the *effort* of the attack, and vice versa.

5.2. Mathematical model for the evaluation of the attack

The choices of specific values for the different parameters that define an attack (see Section 4), its efficiency and traffic rate values are provided by the above specified indicators. The difficulty now is to find a relation between the specific values to choose for the parameters of the attack and their association with the indicators. For this purpose, it is necessary to develop a mathematical model that

Table 1
Defined indicators for efficiency and rate evaluation of a low-rate DoS attack

Rate evaluation	Efficiency evaluation
Effort (E)	User perceived performance (UPP)
	Mean idle time ($\overline{T_{\text{idle}}}$)

describes the behaviour of the indicators when the different attack parameters are varied.

Let us now examine a theoretical model that makes it possible to estimate, for a given server and attack characteristics, the effort and the efficiency, with respect to E and both the *mean idle time* and the *UPP*.

5.2.1. Mathematical model for the mean idle time

As remarked above, the intruder attempts to synchronize the attack periods with the instants at which the outputs are generated by the server. The time between these instants, τ (see expression (3)), is perceived by an observer as a random variable, τ_{int}, that is assumed to respond to a normal distribution with the characteristics given by expression (4).

If we represent the probability of occurrence of two consecutive outputs (a pair of gaussian curves), two possible scenarios appear. In the first case, the common area below the two curves is negligible. Thus, we say that there is no superposition between the two outputs. However, we could find a second scenario in which the two curves overlap, that is, the intersection area below the two curves is not negligible. In this case, there is said to be superposition. This scenario usually appears when the mean value of the inter-output time is low enough, or its variance is high enough, for a certain degree of overlap to exist between the probability functions corresponding to two consecutive outputs.

In the following development of the mathematical model, we will assume there is no superposition. Although this may not be realistic, it is still useful for the purpose of our work, because this allows us to evaluate the effectiveness of the attack in a broad number of scenarios, as shown below.

As a previous step to developing the model, let us establish some nomenclature. Fig. 8 represents the probability function (gaussian curve) associated with a server output over time. The ON/OFF pattern of the attack (offtime/ontime), as perceived by the server, is also depicted. Moreover, the arrival instants of the attack packets, represented by continuous vertical arrows, occur at time a_i ($i \geqslant 1$) within the *ontime* period. In this example, only three packets appear during the *ontime* period due to the chosen value for *interval*, Δ, although it can be easily extended to attacks with n packets. We will refer, henceforth, to the instants a_i when an attack packet arrives at the server as a *calculation point* in the model. Moreover, a special calculation point, a_0, which does not correspond to the arrival of an attack packet, is also defined. This point is situated, by definition, at a time $\overline{RTT}$ before the arrival of the first packet of *ontime*.

$$a_0 = a_1 - \overline{RTT}. \tag{7}$$

In summary, for a particular period of attack, a set of calculation points $\mathscr{A} = \{a_0, a_1, \ldots, a_n\}$ is defined, where $n = floor[t_{\text{ontime}}/\Delta]$. These calculation points are used by the model as references for the mathematical expressions.

The calculation points delimit a set of intervals at which we will calculate the instantaneous values of *idle time*, T_{idle}. In this way, if the output occurs in the interval $(-\infty, a_0)$, the value of T_{idle} will be $\overline{RTT}$, due to the fact that the intruder will respond to the output and this answer will arrive at the server before the arrival of the first attack packet corresponding to the *ontime* period. Therefore, the time involved is $\overline{RTT}$.

When the output is raised at an instant t situated within the interval (a_{i-1}, a_i) for all the possible values of i in $\mathscr{A}$, the *idle time* will take the value $(a_i - t)$. This is always true under the assumption that the duration of the *interval* parameter, $\Delta = a_i - a_{i-1}$, in the *ontime* period of the attack is lower than $\overline{RTT}$.

$$T_{\text{idle}_{(\Delta \leqslant \overline{RTT})}} = a_i - t. \tag{8}$$

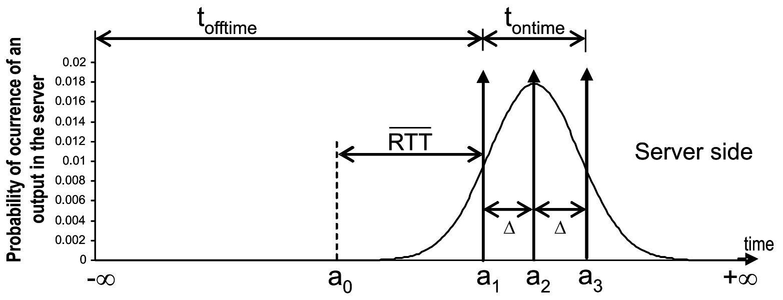


Fig. 8. Diagram of occurrence for an output: probability function and associated calculation points.

Otherwise, the interval is split into two parts in which the values for the instantaneous T_{idle} are different. For an instant t within the considered interval (a_{i-1}, a_i), the value for the instantaneous T_{idle} is

$$T_{\mathrm{idle}(\Delta > \overline{RTT})} = \begin{cases} a_i - t & \text{if } a_i - \overline{RTT} < t < a_i, \\ \overline{RTT} & \text{if } a_{i-1} < t < a_i - \overline{RTT}. \end{cases} \tag{9}$$

Finally, when the output occurs during the interval (a_n, ∞), we have the same case as in the first interval, and thus the value of the generated *idle time* is $\overline{RTT}$.

Thus, for this case in which $\Delta \leqslant \overline{RTT}$, the generated *mean idle time* in an output can be obtained from the instantaneous values previously deduced as:

$$\begin{aligned}\overline{T_{\mathrm{idle}(\Delta \leqslant \overline{RTT})}} = {} & \int_{-\infty}^{a_0} \overline{RTT} \cdot f(t)\,\mathrm{d}t + \int_{a_0}^{a_1} (a_1 - t) \cdot f(t)\,\mathrm{d}t \\ & + \int_{a_1}^{a_2} (a_2 - t) \cdot f(t)\,\mathrm{d}t + \cdots \\ & + \int_{a_{n-1}}^{a_n} (a_n - t) \cdot f(t)\,\mathrm{d}t \\ & + \int_{a_n}^{\infty} \overline{RTT} \cdot f(t)\,\mathrm{d}t, \end{aligned} \tag{10}$$

where $f(t)$ is the probability function for the occurrence of an output at the instant t. It is significant that this expression is independent of the distribution assumed for the probability of occurrence of an output. In our study, this probability function is taken from expression (4) in order to solve Eq. (10). Moreover, for simplicity, a temporal translation for the studied period is considered such that the mean value of the normal distribution is zeroed:

$$f(t) = \mathcal{N}(0, var[t_s] + var[RTT]) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{t^2}{2\sigma^2}}. \tag{11}$$

Assuming this distribution, the resolution of expression (10) leads us to the following:

$$\begin{aligned}\overline{T_{\mathrm{idle}(\Delta \leqslant \overline{RTT})}} = {} & \overline{RTT} \cdot (F(a_0) + 1 - F(a_n)) \\ & + \sum_{i=1}^{n} a_i \cdot (F(a_i) - F(a_{i-1})) \\ & + \frac{\sigma}{\sqrt{2\pi}} \cdot \left(\mathrm{e}^{-\frac{a_n^2}{2\sigma^2}} - \mathrm{e}^{-\frac{a_0^2}{2\sigma^2}}\right), \end{aligned} \tag{12}$$

where the operator $F(t)$ means the value of the distribution function associated with $f(t)$ at the instant t.

The very design of the attack means that the case $\Delta \leqslant \overline{RTT}$ is commonly encountered, since during t_{ontime} the traffic rate is not low, which results in a relatively low value for Δ, sufficient to meet the condition. However, for the case when $\Delta > \overline{RTT}$, it is also possible to find an expression for the generated *mean idle time*:

$$\begin{aligned}\overline{T_{\mathrm{idle}(\Delta > \overline{RTT})}} = {} & \int_{-\infty}^{a_0} \overline{RTT} \cdot f(t)\,\mathrm{d}t \\ & + \sum_{i=1}^{n} \left(\int_{a_{i-1}}^{a_i - \overline{RTT}} \overline{RTT} \cdot f(t)\,\mathrm{d}t \right. \\ & \left. + \int_{a_i - \overline{RTT}}^{a_i} (a_i - t) \cdot f(t)\,\mathrm{d}t \right) \\ & + \int_{a_n}^{\infty} \overline{RTT} \cdot f(t)\,\mathrm{d}t. \end{aligned} \tag{13}$$

Finally, it is important to highlight the relationship between the parameters of the attack, as defined in Section 4, and the different variables that appear in the model. Both the server behaviour and the particular attack dynamics are considered in the model as:

- *Server behaviour*: this is comprised within the $f(t)$ term, which is univocally defined by the mean value and the variance of the inter-output time observed by the intruder, as given by expression (4).
- *Attack dynamics*: the configuration of the attack is reflected in the calculation points of the expression. In effect, their position depends on the parameters of the attack, that is, t_{offtime}, t_{ontime}, and the *interval* Δ.

5.2.2. Mathematical model for the user perceived performance

In order to calculate the *user perceived performance* (*UPP*), the existence of legitimate users trying to access the server must be considered. As previously explained in Section 2, the packet arrivals are modelled by a Poisson distribution. Thus, the probability of packet reception from a user during a period of time T is given by the exponential distribution function:

$$F(T) = 1 - \mathrm{e}^{-\lambda T}, \tag{14}$$

where λ is the arrival rate of packets from the users.

In order to calculate *UPP*, it is necessary to previously assess the probability of a user capturing a position in the service queue during a period of the attack. This probability depends on the *mean idle time* generated during the attack periods. For the kth interval within an attack period, delimited by the corresponding calculation points in Fig. 8, this probability, denoted as P_u^k, is given by the expression:

$$P_u^k = 1 - \mathrm{e}^{-\lambda T_{\text{idle}}^k}, \tag{15}$$

where T_{idle}^k represents the generated *idle time* during the kth interval.

The probability of a user seizing a position in the service queue during a complete period of the attack is given by the sum of the corresponding terms from the different intervals:

$$P_u = \sum_{k=0}^{n+1} (1 - \mathrm{e}^{-\lambda T_{\text{idle}}^k}), \tag{16}$$

where n is the index of the last calculation point. The presence of $n+1$ calculation points delimits $n+2$ intervals to be considered in the sum.

Although the value of the *mean idle time* generated at each interval is given by the different terms in expression (12), this does not consider the presence of traffic from legitimate users. In effect, the *mean idle time* takes different values depending on whether the output corresponds to a user or to the intruder. If the answer raised in the server corresponds to a user, the intruder will not receive the output and, therefore, will not respond with a new attack packet. In consequence, the instantaneous *idle time* will take a greater value than when the intruder receives the output from the server. Therefore, it is necessary to incorporate this effect into the model.

In considering the above effect, and for the sake of simplicity, two approximations are made. First, the condition $\Delta \leqslant \overline{RTT}$ is retained, as discussed in the previous section, with the expression (10) being used to calculate the *mean idle time*. Second, the effect of the variation of the *mean idle time* is not considered when the packets coming from legitimate users arrive at the server in the intervals within a_0 and a_n. This is not an unreasonable approximation, due to the fact that the variation in the generated *idle time* for these intervals is up to Δ, if the intervals (a_1, a_n) are considered, and $\overline{RTT}$ for the interval (a_0, a_1). The experimental results shown in the next section confirm the goodness of these approximations.

The consequence of the second approximation is that only the first interval $(-\infty, a_0)$ and the last one (a_n, ∞) are affected by the fact that an output is sent to a legitimate user instead of to the intruder. For these two intervals, the value of the instantaneous *idle time* differs from $\overline{RTT}$ as proposed in expression (10). Within each of these two intervals, the instantaneous *idle time* will have to take the minimum value between the inverse of the mean arrival rate, $1/\lambda$, and the time between the last packet of one period of attack and the first attack packet of the next period, that is, $\overline{t_s} - t_{\text{ontime}}$. Moreover, this new value has to be considered only when the legitimate user captures one position in the queue, that is, with a probability given by P_u. Thus, the expressions for T_{idle}^k within all the intervals remain equal except for the calculation of the *mean idle time* during the first interval (T_{idle}^1) and the last one (T_{idle}^{n+1}), which are calculated as:

$$\begin{aligned} T_{\text{idle}}^0 &= \overline{RTT} \cdot F(a_0) \cdot (1 - P_u) \\ &\quad + \min\left[\frac{1}{\lambda}, \overline{t_s} - t_{\text{ontime}}\right] \cdot F(a_0) \cdot P_u, \qquad (17) \\ T_{\text{idle}}^{n+1} &= \overline{RTT} \cdot (1 - F(a_n)) \cdot (1 - P_u) \\ &\quad + \min\left[\frac{1}{\lambda}, \overline{t_s} - t_{\text{ontime}}\right] \cdot (1 - F(a_n)) \cdot P_u. \qquad (18) \end{aligned}$$

It is important to notice that the calculation of the expressions for T_{idle}^k and P_u should be made recursively, due to the fact that there is a crossed dependency between them. In all the experiments made, the value of P_u converges in a small number of iterations.

Finally, during an attack of duration T, with C seizures, the *user perceived performance* is given by:

$$UPP = \frac{P_u \cdot C}{T/\lambda}. \tag{19}$$

5.2.3. *Mathematical model for the effort of the attack*

The calculation of the *effort* generated by an attack is carried out taking into account the number of attack packets generated during a period of attack. Two factors contribute to the generation of attack packets: firstly, the activity period, *ontime*, during which these packets are generated at a rate of $1/\Delta$, and secondly, the packet sent as a response to the reception of an output by the intruder.

For the model, it is assumed that the intruder receives the output neither before nor within *ontime*. This is tantamount to assuming that the *ontime* period will always last t_{ontime} and thus, that the attack period will not be restarted during *ontime*. This fact allows us to calculate the number of packets generated during *ontime* as $(t_{\text{ontime}}/\Delta) + 1$.

Considering the packet generated as a response to an output received by the intruder, it should be borne in mind that not all the attack periods end with the reception of an output. As previously discussed, this is because the answer to a request from a legitimate user does not arrive at the intruder. Consequently, the percentage of attack periods during which an output is not received from the server is given by *UPP*. In these attack periods, no additional attack packet is generated as a response to the output.

The above considerations enable us to propose the expression that calculates the *effort* of the attack as:

$$E = \left(\frac{t_{\text{ontime}}}{\Delta} + 1\right) + (1 - UPP). \tag{20}$$

6. Experimental results

We now address the task of evaluating all the hypotheses presented, in both simulated and real scenarios. Firstly, we evaluate how efficient the attack would be in a simulated scenario. Then, an assessment of the proposed mathematical model is made, and finally, the attack is tested in a real environment.

6.1. Simulation of the attack

To evaluate the efficiency of the attack, its behaviour was tested by using Network Simulator (NS2) [23]. An implementation of the attack, as well as one from an iterative server, are described as derived application classes from NS2.

A set of attacks have been tested within this environment, and worrying results are derived, because of the high effectiveness demonstrated. For example, Fig. 9 shows the results obtained from an attack simulation comprised of 1352 outputs. The attack was launched against a server with $\overline{t_s} = 3.0$ s and $var(t_s) = 0.2$. The traffic generated by the user, with $\overline{t_a} = 4.0$ s, is close to keeping the server busy all the time by itself. The parameters of the attack for this example are: $t_{\text{ontime}} = 0.6$ s, $t_{\text{offtime}} = 2.7$ s, and $\Delta = 0.3$ s. On the other hand, $\overline{RTT}$ was set to $\mathcal{N}(0.6\,\text{s}, 0.2\,\text{s})$. As previously stated, a very high level of efficiency is obtained, $UPP = 4.59\%$, which means that only 4.59 percent of legitimate requests are attended to by the server, and the percentage of generated *mean idle time* during a period of attack is equal to 5.59%. On the other hand, an effort value of $E = 388\%$ was necessary to launch this attack, which indicates that the traffic offered to the server by the intruder approximates to four times the capacity of the server.

The results obtained from the experiments show that there are multiple attack configurations available for the intruder to be able to adapt the attack to the desired aim. Thus, Fig. 10 represents the *user perceived performance* versus the *effort* needed for a subset of the experiments carried out. As can be seen, it is possible to adjust the *effort* of the attack and, therefore, the ability to bypass an IDS system capable of detecting attacks at a given rate, by reducing the value of *ontime* time, t_{ontime}, and/or by increasing the *interval*, Δ, at the cost of a reduction in the effectiveness of the attack.

Although the above results show the potential risks of the kind of attacks reported in this work, it is still interesting to compare the proposed attack, which presents a certain degree of intelligence, to one that uses the same effort but without any intelligence, that is, sending the packets in instants taken from an aleatory distribution. The differences between these two strategies are revealed in the results shown in Fig. 11, where the values obtained for the efficiency and rate indicators, *UPP* and *effort*, corresponding to three different attacks are illustrated. The service time in the server, t_s, has

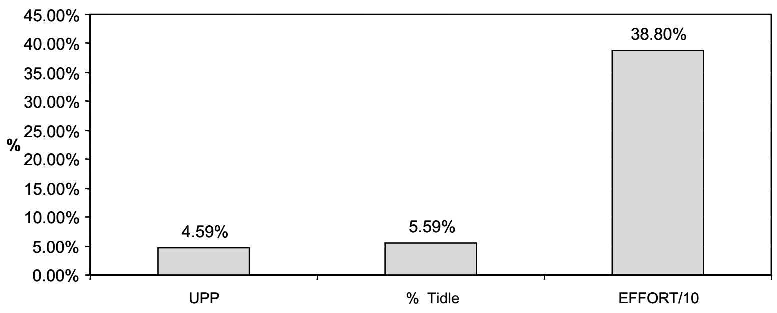


Fig. 9. Figures for indicators of an attack in a simulated environment.

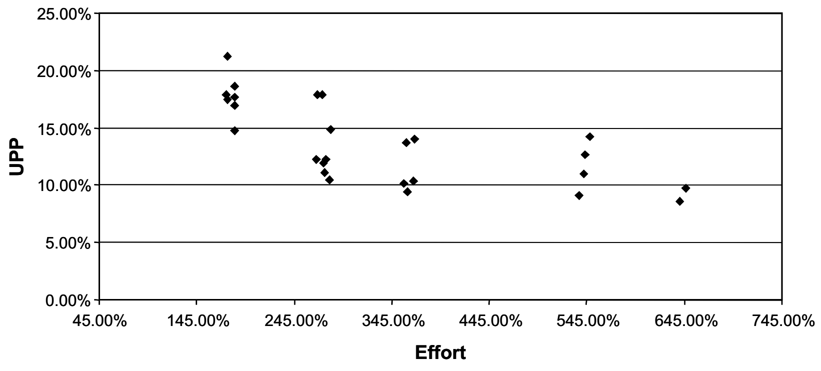


Fig. 10. User perceived performance *vs.* effort for 25 different configurations (t_{ontime} and Δ values) of the low-rate DoS attack.

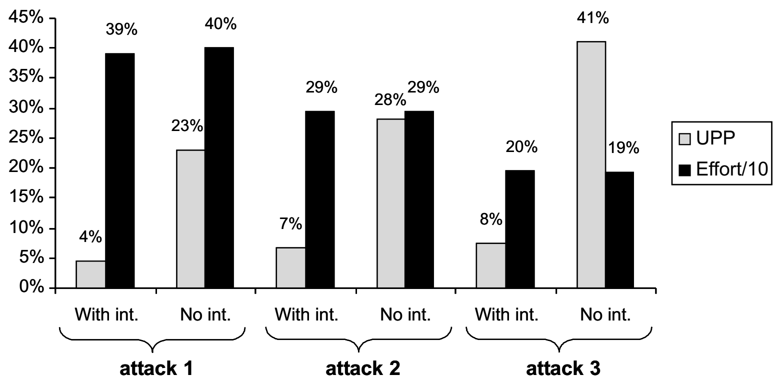


Fig. 11. Comparison between intelligent and non-intelligent strategies for three different attacks.

been adjusted to $\mathcal{N} = (5.0\ \text{s}, 0.2\ \text{s})$, and a traffic rate from legitimate users with $\lambda = 1/10$ s packets/second has been generated. For each of the three attacks, both intelligent ("With int." series) and non-intelligent ("No int." series) strategies have been used. As can be observed, the results show that, although the value of E is similar in both strategies, the derived values for *UPP* indicate that higher efficiency is achieved when running an intelligent strategy.

Finally, let us consider what would happen if the legitimate users increased their traffic rates so that the aggregate traffic was greater than that generated by the attack. We have made some experiments that demonstrate that even under these conditions, an intelligent attack attains a higher degree of DoS than does a non-intelligent strategy. Fig. 12 shows the results obtained for an attack carried out on a server with a value for t_s of $\mathcal{N} = (5.0\ \text{s}, 0.2\ \text{s})$, as in the previous experiments, but with legitimate users trying to

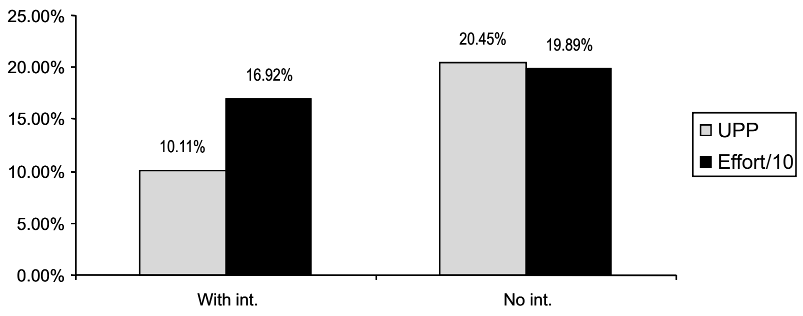


Fig. 12. Comparison between intelligent and non-intelligent strategies in a server flooded by legitimate users.

reach the server at a rate of $\lambda = 1/1.66$ packets/s. In the absence of attack, a $UPP = 33\%$ is expected with this type of traffic, due to the fact that the rate of the users is higher than the service rate of the server. As can be seen in Fig. 12, the intelligent attack ("With int." series) improves the value of *UPP* by up to 50% compared with that obtained in the non-intelligent case ("No int." series), which are worrying results, even under these conditions.

In summary, the proposed attack seems to be very effective in terms of the denial of service to legitimate users. Moreover, its design allows the attacker to tune the traffic rate sent to the server in order to select a threshold, usually determined by the sensibility of intrusion detection mechanisms, that allows them to be bypassed.

6.2. Evaluation of the mathematical model

At this point, it is interesting to determine the accuracy and precision of the formal expressions of *idle time* and *user perceived performance* given by the mathematical model proposed in Sections 5.2.1 and 5.2.2.

We performed a set of experiments based on the simulation, using NS2, of scenarios with different configurations for both the attack and server parameters. The results from these experiments have been compared with the values derived from the mathematical model, and a very good approximation between them was obtained. Fig. 13 shows the corresponding values of *idle time* for 13 simulations taken from this set of experiments. The maximum variation given by the model is 3.77%, with a mean value of 1.71%, which represents a very good approximation.

Fig. 14 shows a similar comparison for 14 different scenarios, where the *user perceived performance* is evaluated. The results are shown in absolute values. The values obtained from the model approximate the simulated ones very well, with a mean variation of 0.4% and a maximum variation of 1.46%.

Finally, Fig. 15 depicts the comparison of the effort between both the simulation and the model values, for 13 different simulations. It can be observed that the model approximates the simulated values very well, with a mean variation of 1.42% and a maximum variation of 4.02%.

In summary, the accuracy of the approximations made in the mathematical model confirms the validity of the model as a tool to evaluate the potential effect of an attack designed with certain parameters.

6.3. Experiments in a real environment

The proposed attack has also been tested in a controlled real environment to check its validity. The selected server is an Apache web server that observes the condition of serving requests in an iterative way (`ThreadsPerChild = 1`). Although this is not a typical example of an iterative server, it is still useful for our purposes. The reason for the adaptation of a web server to an iterative one is that we plan to extend this kind of attack to concurrent servers in a future project.

We assumed a client's petition to consist of a connection request to the server. The attack establishes connections and sends no messages on them, letting the web server close the connection after a timeout period specified by the Apache directive `Timeout`. This timeout is a deterministic and fixed

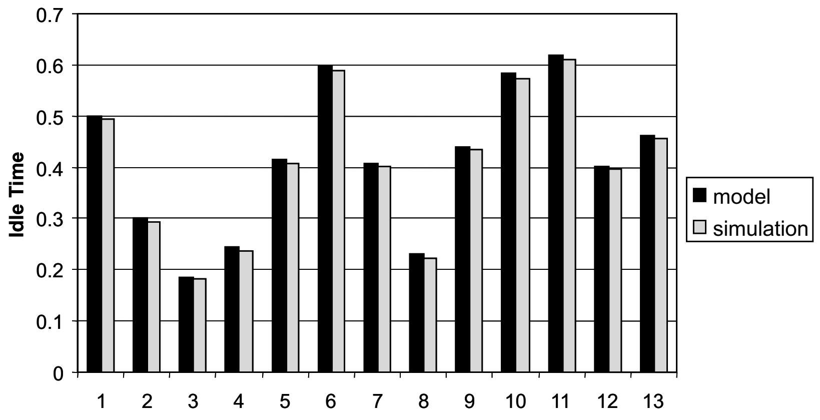


Fig. 13. Comparison between the obtained values of *idle time* from simulation and from the mathematical model, for 13 different scenarios.

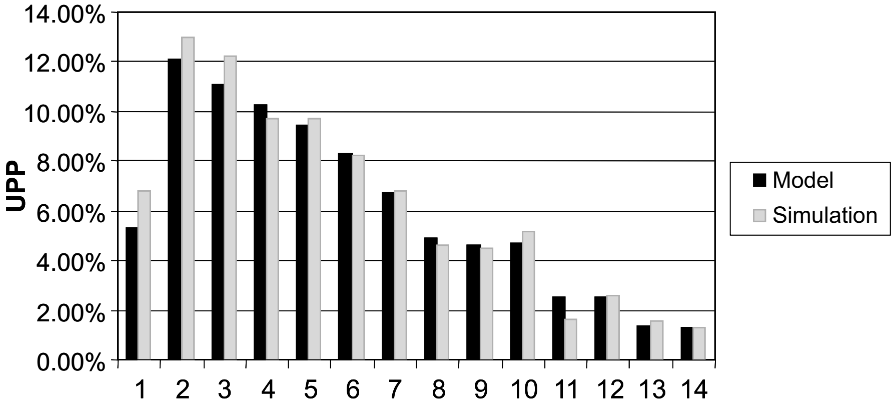


Fig. 14. Comparison between obtained values of *user perceived performance* from simulation and from the mathematical model, for 14 different scenarios.

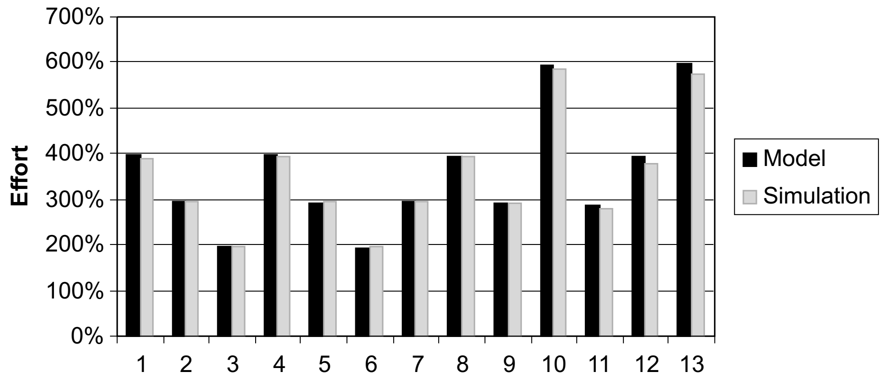


Fig. 15. Comparison between obtained values of *effort* from simulation and from the mathematical model, for 13 different scenarios.

value which corresponds, in our model, to the mean service time, $\overline{t_s}$. As discussed in previous sections, the variance in the inter-output time observed by the intruder, τ_{int} will be contributed to by variations in the service time caused by the operation of the server itself in a multiprocess environment within a non-real time operating system, and also by the variability of the round trip time.

The real scenario is analogous to the one considered for the theoretical analysis. The user traffic was generated following a Poisson process. Intruder software launches the attack from a single source. Both legitimate and intruder traffic traverse a WAN network to reach the server, with a round trip time modelled with the distribution $\mathcal{N}(17\,\text{ms}, 0.05\,\text{ms})$. Traces on the users' and the intruder's side were established to obtain the data necessary to calculate the attack indicators.

Table 2 shows some experimental results and a comparison between the real and predicted values for different mean service times ($\overline{t_s}$) and user traffic inter-arrival times ($\overline{t_a}$). These times were selected in such a way that there was no congestion on the

Table 2
Real and simulated attack performance

$\overline{t_s}$	$\overline{t_a}$		*UPP* (%)	*Effort* (%)
3	3.5	Simulated	10.4	260.8
		Real	9.8	239.7
5	6	Simulated	5.7	213.1
		Real	7.8	212.4
10	12	Simulated	3.0	198.3
		Real	6.4	201.6
15	17	Simulated	3.0	197.5
		Real	2.5	198.2
20	22	Simulated	3.0	197.5
		Real	4.3	196.2
25	28	Simulated	1.8	197.9
		Real	1.8	197.9

server when no attack was under way. The parameters of the attack were tuned to $t_{\text{ontime}} = 0.4$ s and $\Delta = 0.4$ s for all the experiments.

It can be seen that in some cases even better results in efficiency (lower *UPP*) can be obtained for the attack in a real environment. Moreover, differences can be appreciated between the simulated and the real values. Such variations are due to the difficulty of modelling the service time and round trip time distributions.

Two conclusions can be drawn from these results: (a) all the experiments made under simulation seem to provide results that are good approximations of the behaviour in real environments, and (b) the real impact of the attack is very high, showing that these vulnerabilities could be easily exploited in iterative servers.

7. Conclusions and future work

This paper presents an evaluation of the efficiency and the rate of a new kind of attack. It is based on the vulnerability present in iterative servers, which consists in the possibility of forecasting statistical metrics about the behaviour of the server. This vulnerability makes it possible to carry out a denial of service attack on the server.

Although this kind of attack could be achieved as a brute-force attack, its main characteristic is that it can make use of low-rate traffic to subvert the provided service. Thus, it is possible to tune the parameters of the attack in order to select appropriate values for the efficiency and the amount of load generated in the target server. This characteristic makes it possible, under certain circumstances, to bypass existing IDS systems intended to protect the attacked server, thus creating what is currently a non-detectable threat.

Although there are similarities with the kind of attacks presented in [13], there are also differences, namely that the attack presented here works at the application level, and takes advantage of knowledge of the inter-output times in the server to keep it constantly engaged serving requests from the intruder. On the other hand, the similarities imply the existence of a family of DoS attacks, characterized by a low-rate traffic that is shaped by the estimation of a specific timer or system behaviour on the server side.

The fundamentals and design of a possible attack to exploit the vulnerability described have been explained. We have confirmed that such an action could be very efficient and that it could be easily carried out. Moreover, its behaviour is neither detectable by parsing the packets that arrive at the server, due to the fact that they are similar to the users' requests, nor is it noticeable by high-rate analysis detection.

As future work, we plan to extend the study of these kinds of attacks to concurrent server environments. Methods for detection of and response to these attacks should be developed, as should methods to prevent their potential harmful effects.

Acknowledgement

This paper has been partially supported by the Spanish Government through MEC (Project TSI2005-08145-C02-02, FEDER funds 70%).

References

[1] M. Williams, Ebay, Amazon, Buy.com hit by attacks, 02/09/00. IDG News Service, 02/09/2000. Available from: <http://www.nwfusion.com/news/2000/0209attack.html>.

[2] J. Mirkovic, P. Reiher, A taxonomy of DDoS attack and DDoS defense mechanisms, ACM SIGCOMM Computer Communication Review 34 (2) (2004) 312–321.

[3] CERT coordination Center. Denial of Service Attacks. Available from: <http://www.cert.org/tech_tips/denial_of_service.html>.

[4] Global Incident Analysis Center – Special Notice – Egress filtering. Available from: <http://www.sans.org/y2k/egress.htm>.

[5] P. Ferguson, D. Senie, Network ingress filtering: defeating Denial of Service attacks which employ IP source address spoofing, RFC 2827, 2001.

[6] X. Geng, A.B. Whinston, Defeating Distributed Denial of Service attacks, IEEE IT Professional 2 (4) (2000) 36–42.

[7] N. Weiler, Honeypots for Distributed Denial of Service, in: Proceedings of the Eleventh IEEE International Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises 2002, Pitsburgh, PA, USA, June 2002, pp. 109–114.

[8] S. Axelsson, Intrusion detection systems: a survey and taxonomy, Department of Computer Engineering, Chalmers University, Goteborg, Sweden, Technical Report 99-15; March 2000.

[9] J.B.D. Cabrera, L. Lewis, X. Qin, W. Lee, R.K. Prasanth, B. Ravichandran, R.K. Mehra, Proactive detection of Distributed Denial of Service Attacks using MIB traffic variables – a feasibility study, in: Proceedings of the 7th IFIP/IEEE Internation Symposium on Integrated Network Management, Seattle, WA, May 14–18, 2001.

[10] J. Mirkovic, G. Prier, P. Reiher, Attacking DDoS at the source, in: Proceedings of ICNP 2002, Paris, France, 2002, pp. 312–321.

[11] R.R. Talpade, G. Kim, S. Khurana, NOMAD: Traffic-based network monitoring framework for anomaly detection, in: Proceedings of the Fourth IEEE Symposium on Computers and Communications, 1999, pp. 442–452.

[12] C. Douligeris, A. Mitrokotsa, DDoS attacks and defense mechanisms: classification and state-of-the-art, Computer Networks 44 (2004) 643–646.
[13] A. Kuzmanovic, E. Knightly, Low Rate TCP-targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants), in: Proceedings of ACM SIGCOMM 2003, August 2003, pp. 75–86.
[14] H. Sun, J.C.S. Lui, D.K.Y. Yau, Defending against low-rate TCP attacks: dynamic detection and protection, in: Proceedings of IEEE Conference on Network Protocols (ICNP2004), October 2004, pp. 196–205.
[15] G. Yang, M. Gerla, M.Y. Sanadidi, Randomization: defense against low-rate TCP-targeted Denial-of-Service attacks, in: Proceedings of the IEEE Symposium on Computers and Communications, July 2004, pp. 345–350.
[16] A. Shevtekar, K. Anantharam, N. Ansari, Low rate TCP Denial-of-Service attack detection at edge routers, IEEE Communications Letters 9 (4) (2005) 363–365.
[17] Haibin Sun, John C.S. Lui, David K.Y. Yau, Distributed mechanism in detecting and defending against the low-rate TCP attack, Computer Networks 50 (13) (2006) 2312–2330.
[18] SANS Institute. NAPTHA: a new type of Denial of Service Attack, December 2000. Available from: <http://rr.sans.org/threats/naptha2.php>.
[19] R.R. Martin, Basic Traffic Analysis, Prentice-Hall Inc, 1993, ISBN 0133354075.
[20] <http://mathworld.wolfram.com/ExponentialDistribution.html>.
[21] R.E. Walpole, R.H. Myers, S.L. Myers, Probability and Statistics for Engineers and Scientists, sixth ed., Prentice Hall College Div, 1997, ISBN 0138402086.
[22] T. Elteto, S. Molnar, On the distribution of round-trip delays in TCP/IP networks, in: Proceedings of the 24th Annual IEEE Conference on Local Computer Networks, October 17–20, 1999, p. 172.
[23] Network Simulator 2. Available from: <http://www.isi.edu/nsnam/ns/>.
[24] R. D'Agostino, M. Stephens, Goodness-of-Fit Techniques, Marcel Dekker, Inc., 1986.

Gabriel Maciá-Fernández is a Ph.D. candidate in the Department of Signal Theory, Telematics and Communications of the University of Granada (Spain). He received an MS in Telecommunications Engineering from the University of Seville, Spain. From 1999 till 2005 he has worked as a specialist consultant at 'Vodafone España', where he has been involved in several research projects. Since 2005 he is Assistant Professor in the Department of Signal Theory, Telematics and Communications of the University of Granada. His research was initially focused on multicasting technologies but he is currently working on computer and network security, especially in intrusion detection and response systems, and denial of service.

Jesús E. Díaz-Verdejo is Associate Professor in the Department of Signal Theory, Telematics and Communications of the University of Granada (Spain). He received his B.Sc. in Physics (Electronics speciality) from the University of Granada in 1989 and has held a Ph.D. grant from Spanish Government. Since 1990 he is Associate Professor at this University. In 1995 he obtained a Ph.D. degree in Physics. His initial research interest was related with speech technologies, especially automatic speech recognition. Currently he is working in computer networks, mainly in computer and network security, although he has developed some work in telematics applications and e-learning systems.

Pedro García-Teodoro received his B.Sc. in Physics (Electronics speciality) from the University of Granada, Spain, in 1989. This same year he was granted by "Fujitsu España", and during 1990 by "IBM España". Since 1989 he is Associate Professor in the Department of Signal Theory, Telematics and Communications of the University of Granada, and member of the "Research Group on Signal, Telematics and Communications" of this University. His initial research interest was concerned with speech technologies, especially automatic recognition, field in which he developed his Ph.D. Thesis in 1996. From then, his profile has derived to that of computer networks, and although he has done some works in telematics applications and e-learning systems, his main current research line is centred in computer and network security.