Assessment of a Vulnerability in Iterative Servers Enabling Low-Rate DoS Attacks

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

Dep. of Signal Theory, Telematics and Communications - University of Granada
c/ Daniel Saucedo Aranda, s/n - 18071 - Granada (Spain)
{gmacia, jedv, pgteodor}@ugr.es*

Abstract. In this work, a vulnerability in iterative servers is described and exploited. The vulnerability is related to the possibility of acquiring some statistics about the time between two consecutive service responses generated by the server under the condition that the server has always requests to serve. By exploiting this knowledge, an intruder is able to carry out a DoS attack characterized by a relatively low-rate traffic destined to the server. Besides the presentation of the vulnerability, an implementation of the attack has been simulated and tested in a real environment. The results obtained show an important impact in the performance of the service provided by the server to legitimate users (DoS attack) while a low effort, in terms of volume of generated traffic, is necessary for the attacker. Besides, this attack compares favourably with a naive (brute-force) attack with the same traffic rate. Therefore, the proposed attack would easily pass through most of current IDSs, designed to detect high volumes of traffic.

1 Introduction

The impact of Denial of Service (DoS) attacks in current networked systems is awesome, posing a very serious problem in many environments, both in economical and in performance sense [1]. The threat is specially overwhelming in Internet, with millions of interconnected systems and a practical lack of enforcement authorities. Furthermore, the possibility of performing the attack in a distributed way (DDoS, Distributed DoS) according to various methodologies [2], increases the risks and makes even more difficult the adoption of preventive and/or corrective measures. Recent incidents involving large-scale attacks that affected important Internet sites [3] [4] [5] demonstrate the vulnerability of the networks and services to this kind of attacks and its pernicious effects.

DoS attacks try to exhaust some resources in the target system with the aim of either reducing or subverting the availability of a service provided by the target. The intruders usually achieve their goal either by sending to the victim a stream of packets that exhausts its network bandwidth or connectivity, or exploiting a discovered vulnerability, causing an access denial to the regular clients [2].

* This work has been partially supported by Spanish government under project TSI2005-08145-C02-02 (70 % from FEDER funds).

D. Gollmann, J. Meier, and A. Sabelfeld (Eds.): ESORICS 2006, LNCS 4189, pp. 512–526, 2006.

Close to the evolution of the DoS attacks, many proposals have also appeared for preventing and detecting them. Many of the preventive measures are applicable to mitigate DoS attacks, like egress or ingress filtering [6] [7], disabling unused services [8], changing IP address, disabling IP broadcasts, load balancing, or honeypots [9]. However, although prevention approaches offer increased security, they can never completely remove the threat as the systems are always vulnerable to new attacks. On the other hand, it is advisable to establish an intrusion detection system (IDS) [10] capable of detecting the attacks. Although various approaches described in the bibliography [11] [12] [13] try to discover DoS attacks, most of them rely on the identification of the attack with techniques that are based on the hypothesis that a high rate flooding is going to be received from the intruder or intruders.

In this paper, a vulnerability in iterative servers is detected and exploited. This vulnerability allows an intruder to carry out an application level DoS attack [14] characterized by the use of a low-rate traffic against a server. Due to this fact, the attack would be capable of bypassing the detection mechanisms that rely on high-bandwidth traffic analysis. An attack with similar rate characteristics is described by Kuzmanovic et. al [15]. Although the attack presented here resembles in some aspects the previously cited one, there are key differences between them. First, both attacks take advantage of a vulnerability caused by the knowledge of a specific time value in the functioning of a protocol or application, allowing an ON/OFF attack that results in low-rate traffic but with high efficiency in service denial. However, the attack presented in [15] is TCP-targeted, while the proposed here threatens the application layer. Furthermore, Kuzmanovic's attack generates outages in a link, trying to trigger TCP's congestion control mechanism, while ours simply tries to overflow a single service running in a server. In other words, no noticeable effect on network traffic is expected. On the other hand, the proposed attack would not affect other services or users within the same network or host, as Kuzmanovic's does. There are also differences in the vulnerabilities exploited in both attacks. In the TCP-targeted low-rate case, the knowledge of the RTO timer for congestion control implemented in TCP is exploited, whilst in the iterative server case the inter-output times are the key to build the attack, as it will be presented in Section 3. But the main difference between both attacks lies in the fact that during the TCP-targeted attack, the link is only busy in the outages periods, while in the new proposal the server is always busy in processing service requests from the intruder, causing the legitimate users the perception that the server is not reachable. This last feature is similar to the behaviour of the *Naptha* attack [16], although the main difference is that *Naptha* is a brute-force attack executed with high rate traffic, while this attack uses low-rate traffic.

The rest of the article is structured as follows. Section 2 describes the scenario of the attack and the hypothesis about its behaviour enabling the vulnerability. Next, a validation of the hypothesis backing up the claimed vulnerability is presented. Section 3 describes the details and phases of the proposed attack, which is evaluated both by means of simulations and in a real environment in

Section 4. Besides, Section 4 shows the comparison of the proposed attack with a brute-force attack with the same rate of attack packets. Finally, some conclusions and further work to be carried out are compiled in Section 5.

2 Scenario and Vulnerability Analysis

The scenario to be considered for the analysis is a generic client-server configuration in which the server is going to receive aggregated traffic from legitimate users and from intruders. From our point of view, an iterative server is a black box system which acts in the usual way, that is, the server receives requests from the clients and responds them after some processing. Only after the processing of a request will the processor take a new petition (iterative operation). Thus, the existence of a finite length queue in which incoming requests are queued up while awaiting for the server to process them in a FIFO discipline is assumed. Therefore, the overall behaviour is that each request packet emitted by a client is first queued in the server and, after some time, processed and responded by the server. The possibility of rejecting an incoming request due to a queue overflow is also considered. Whether the rejection of a request packet is observable or not is irrelevant for our experiments.

The main objective of a DoS attack in this scenario is to keep the queue full of request from the attacker. This fact will avoid the acceptance of request packets from other (legitimate) users, thus causing a denial of service to these users. Usually, the DoS event is carried out by means of a so called brute-force attack, that is, the intruder or intruders send as many requests as they can with the aim of either obtaining the bulk of the queue positions or saturating the network. Our objective will be similar: to "capture" all the queue positions, but emitting a reduced number of requests, what is done by choosing the key instants at which they should be sent.

Therefore, the key to succeed in our approach of attacking the service is by forecasting the optimum instants at which a request should be made in order to acquire a just freed position in a previously full queue. This way, the attack would eventually capture all the positions and generate a denial of the service to legitimate users. Obviously, a main question remains unanswered: is it possible to forecast the instants at which the server is going to get a request from the input queue and, therefore, generate a free position? Our hypothesis is that, under certain circumstances, and by simple inspection of the outputs generated by the server, this is possible. In fact, this is the argued vulnerability.

2.1 Timing of the Outputs

In order to predict the instants at which a position is freed and, therefore, is available for the first request received, we need to review the operation of an iterative server in a more formal way, as follows.

A service request enters the system. If the service queue has free positions, the request is queued. Otherwise, an overflow event occurs and a denial of service is

perceived by the user. The request will stay in the service queue during a *queue time*, t_q, awaiting for its turn to be served. Afterwards, it will be processed by the service module during a *service time*, t_s. This time would have been employed in parsing the data, in a complex calculation or simply in building up the answer. Finally, once the processing is completed, the corresponding answer to the input request is generated and sent. Following, the next request in the queue is obtained to be processed. At this point, a free position in the queue is generated.

Our main hypothesis is that the service time is a random process, T_s that can be modeled by a distribution function. At this point, some studies suggest behaviours that depend upon the nature of the service [17]. Furthermore, various authors report different distributions even for the same service, depending on the network and other additional factors [18]. Nevertheless, it is not necessary to know the overall statistics of the service to carry out the proposed attack. The only needed knowledge concerns the statistics related to the processing and responses to the requests made by the intruder. In this context, if all the packets sent by the intruder contains the same request, it would be expectable to require the same service time. Obviously, many factors external to the service but related with the server itself will introduce some degree of variability. We lean on the central limit theorem [19] to characterize the model of this behaviour in a sufficiently complex system, e.g. a computer running a server, where a lot of variables are involved, as a normal distribution. Anyway, as the purpose of this paper is to show the existence of the vulnerability, which is solely based on the possibility of estimating the expected value for the service time, $E[t_s]$, and according to the central limit theorem, we use a normal distribution $\mathcal{N}(\overline{t_s}, var[t_s])$ to check our hypothesis. This approach allows us to consider the effects of the variance due to different CPU loads, occupation in the service network, etc. The behaviour of the queue time is irrelevant from our point of view, as will be argued hereafter. On the other hand, as it will be pointed later on, the attack procedure includes a mechanism to resynchronize its timing, which reduces the impact of the requests made by the legitimate users on the evolution of the attack and the relevance of the real distribution function of the service time. That is to say, we only need to model the distribution of the service time for the attack packets jointly with a mechanism to recover from mistakes.

The potential exploit that allows an intruder to carry out the attack is based on the knowledge about the statistics of the inter-output time of the server, τ, defined as the time elapsed between two consecutive outputs or answers provided by the server. With the knowledge of this time, the intruder can evaluate the timing at which free positions are generated.

For clarity, let us examine a case of study in which fixed service times are considered. Although, at first glance, this could seem a very restrictive case, it will be shown that the results are valid for more complex cases in which the service time is a random variable, as discussed above.

The scenario of interest is that in which the queue is always kept occupied with, at least, one pending request. Under this consideration, the behaviour of

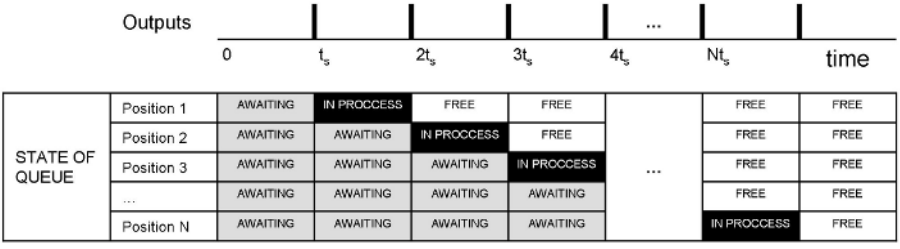


Fig. 1. Time diagram for the processing of the requests carried out by the server (bottom) and the associated timing of the generated outputs (top). In black, processing time; in gray, queue time.

the inter-output time, assuming fixed service times, can be described as follows (Fig. 1). Let us suppose a queue with N positions which is initially full of requests. At time $t = 0$, one of the pending requests is extracted from the queue and processed by the server. After t_s, an output is generated (vertical bar in the time diagram) and the next request is selected for its processing. Again, after t_s, a new answer is provided and the process is repeated while there are pending requests in the queue. Therefore, for this scenario, the time between two consecutive outputs from the server is equal to the service time, t_s. This rate of outputs will be maintained under the single condition that there always exists a pending request in the queue.

If, as previously hypothesized, the service time responds to a normal distribution, the inter-output time, τ, will behave as

$$\tau = \mathcal{N}(\overline{t_s}, var[t_s]) \tag{1}$$

As it can be seen, the queue time is not relevant, as it does not influence the timing of the outputs.

The distribution of the inter-output time could be estimated by a potential intruder by sending various requests close enough in time so as to occupy contiguous positions in the buffer. In a first approach, the time between two consecutive responses received by the attacker would provide the required information. By repeating this procedure, the potential attacker can collect enough information as to characterize the inter-output time distribution. Anyway, an effect that has not been previously considered appears in this mechanism. Both the requests and the answers must traverse the network to reach its destinations. Therefore, the round trip time (RTT) plays a role in the determination of the timings. Concretely, the variability of the RTT can produce deviations in the inter-output times. This way, the variance of the inter-output time perceived by a user, τ_{user}, will be affected by the variance of the RTT. Assuming that the service times and RTT are statistically independent variables, and that RTT can also be described by a normal distribution function, the perceived inter-output time will be:

$$\tau_{user} = \mathcal{N}\left(\bar{t_s}, var[t_s] + var[RTT]\right) \tag{2}$$

Experimental Validation by Simulation. In order to validate our hypothesis concerning inter-output time distribution, various simulation experiments have been carried out. For this purpose, Network Simulator (NS2) [20] have been used to check whether the assumption that inter-output time approximates to Eq. (2) is correct or not.

In a first set of experiments, an scenario with fixed service time has been considered. As expected, the distribution of the inter-output time, τ, behaves as predicted while there is at least one pending request in the queue.

In a second set of simulations, some scenarios in which the service time t_s and the round trip time RTT are modelled with normal distributions have been considered. The obtained results show low deviations in the behaviour of the system from that predicted theoretically.

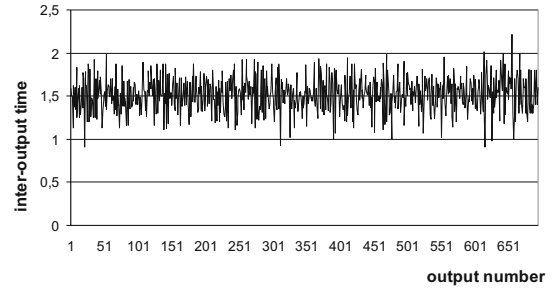
As an example, Fig. 2 shows the results of one of the simulations in which the queue is always full of requests, what is achieved by sending request packets at a higher rate than the service rate of the server. The service time and RTT are supposed to be $\mathcal{N}(1.5\ s,\ 0.02\ s)$ and $\mathcal{N}(0.6\,s, 0.02\ s)$, respectively, so that the inter-output time distribution is expected to be $\mathcal{N}(1.5\,s, 0.04)$ (see Fig. 2.a). The simulation results provide a mean value of 1.52 seconds and a variance of 0.041 seconds for the inter-output time, with a distribution that can be approximated by a normal distribution (Fig. 2.b). This fact have been tested through goodness of fit tests as the Kolmogorov-Smirnoff test [21].

On the other hand, Fig. 3 shows the results of another example in which the queue can become momentarily empty. The inter-output times and the occupation of the buffer are represented in the main axis and in the secondary axis (dashed lines), respectively, in Figure 3.a. For the same input parameters as in the previous example, the mean value obtained for the inter-output time is 1.536 s with variance 0.114. The deviation from theoretical results is greater than in the previous experiment due to the values generated when the buffer has no requests. In fact, the goodness of fit tests provide poor estimators for the obtained distribution when compared to a normal distribution. It is easily tested that the periods with no requests in the queue result in higher values for the inter-output times, greatly modifying the distribution of the samples. Therefore, the inter-output times becomes unpredictable if the queue becomes empty and the scenario of interest is that in which the queue has always at least one pending request, as previously stated.

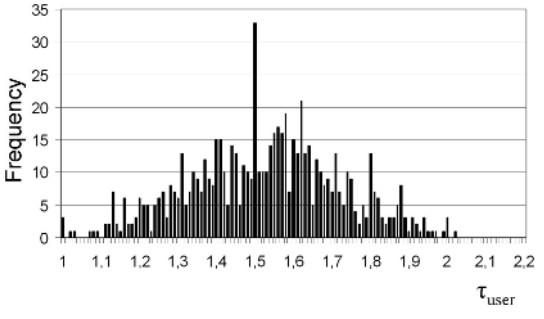
These experiments show that, even in simulated scenarios where the service time is variable, the inter-output time could still be predictable in some circumstances for a possible intruder to build up an attack based on this knowledge.

3 Low-Rate Attack Specification

As previously stated, the objective of the denial of service attack is to maintain the input queue of the target server full of requests coming from the attacker or attackers. This way, legitimate users are not going to be able to queue their requests, thus experimenting a denial of the service given by the server appli-



a)



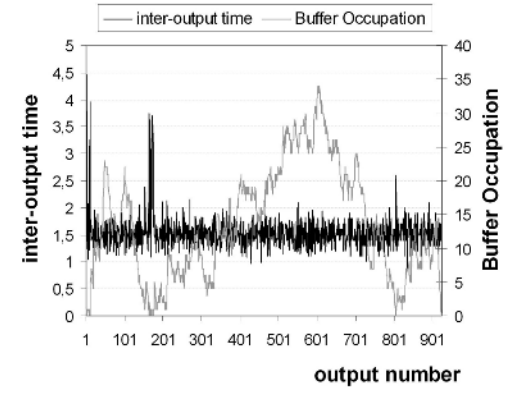
b)

Fig. 2. Simulation of inter-output time with flooded service buffer: a) inter-output time values, and b) histogram of the samples

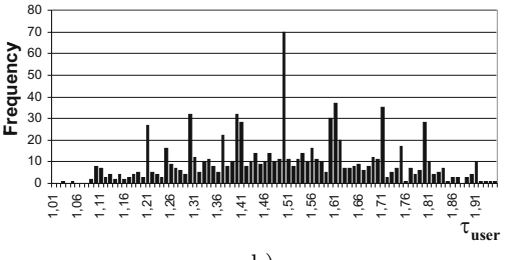
cation. This is achieved by making a prediction regarding the instants at which the server is going to answer the requests, that is, when an output is going to be generated. The intruder will flood the service queue only during a short period of time around the predicted output time, resulting in an overall low-rate flood experienced by the destination network. Therefore, the attack will follow an OFF/ON scheme, as described next.

The proposed attack strategy consists in the succession of consecutive periods composed by an interval of inactivity, *offtime*, followed by an interval of activity, *ontime*, as depicted in Fig. 4. The attack waveform is characterized by the following parameters:

- *Estimated mean inter-output time* ($E[\overline{\tau_{user}}]$): it is an estimation of τ_{user} made by the intruder.
- *Interval* (Δ): period of time elapsed between the sending of two consecutive attack packets during *ontime*.
- *Ontime time* (t_{ontime}): time during which an attempt to flood the service queue is made by emitting request packets, at a rate given by $1/\Delta$. The duration of ontime should be proportional to the variance of τ_{user}. It is centered around $E[\overline{\tau_{user}}]$.



a)



b)

Fig. 3. Simulation of inter-output time with the possibility of empty queue: a) Inter-output time values and buffer occupation level, b) histogram of the samples

– *Offtime time* ($t_{offtime}$): time during which there is no transmission of attack packets. Its duration should be

$$t_{offtime} = E[\overline{\tau_{user}}] - t_{ontime}/2 - \overline{RTT} \cdot \delta \tag{3}$$

where δ is equal to 0 if no response is received (a previous failure of the attacker in a seizure or loss of a packet) and 1 otherwise. This accounts for the delay among the emission of a packet and the reception of the response.

Both *offtime* and *ontime* are adjusted so that the generation of the output by the server and the reception of the requests from the intruder are synchronized. For this, the round trip time (RTT) has to be considered, as it represents the delay in the transmission from the server to the client and viceversa. Therefore, two points-of-view (server's and intruder's) should be considered in order to establish the timings and synchronization of the sendings. The descriptions made up to now have considered the server's point-of-view, i.e. all the events and sequences of events are timed according to the server's local clock. However, the communication among the server and the intruder will experience a delay due to the RTT which can make observation of the sequence of events slightly different. As an example, if an attack packet is due in the server at a given time,

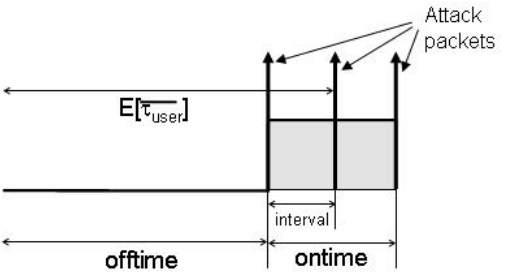


Fig. 4. Attack specification: Attack waveform and parameters

the intruder has to provide for it in advance (the packet should be sent RTT/2 time units before). Those effects will be considered in the following explanations concerning the attack execution.

The attack presents two phases. In the first phase, an attempt to capture all the positions in the queue is launched. After that, the attack tries to keep all the captured positions by sending a new request in such a way that it arrives at the server in the minimum time after the position becomes free, which is achieved by using the attack waveform in synchrony with the outputs from the server, as will be explained in the next paragraphs. For the first phase, a brute-force attack could be used although the same results can be achieved by using the attack mechanism proposed in the next paragraph. This alternative will achieve the aim in a longer time but reduces the chances of detection by an IDS rate-based mechanism.

The attack is composed by a continuous sequence of the attack waveform previously described, plus a mechanism to resynchronize the *ontime* periods or, equivalently, to restart the *offtime* period, just in case a response is received. Therefore, the execution of the attack can be explained as follows (Fig. 5). Just after the reception of an answer packet from the server (A1), the intruder sends a request packet (R1) and the *offtime* period starts. At the end of this period, the *ontime* period starts by emitting a new request packet (R2). While in the *ontime* period, another request packet is sent every *interval* (R3 and R4). At the end of the *ontime* period a new *offtime* starts inmediately, considering $\delta = 0$ in Eq. 3. On the reception of an answer packet during *offtime* (packet A2), a request packet is sent (R5) and the *offtime* period is restarted with the value given by using $\delta = 1$ in Eq. 3. If an answer packet had been received while in the *ontime* period, an additional request packet would have been sent, the *ontime* period would have been finished and the *offtime* period would have been started with $\delta = 1$ (not depicted in Fig. 5). This way, a request packet is sent whenever an answer is received. This is done to reduce the probability of losing the free position due to its capture by a legitimate user.

Finally, two comments about the behaviour of the attack should be pointed out. First, according to the described attack procedure, the intruder will flood the service queue only during a short period of time around the predicted output time, resulting in an overall low-rate flood experienced by the destination network. On the other hand, the behaviour during the flooding (*ontime* period) is

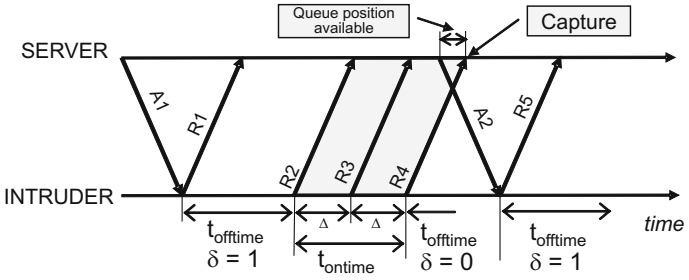


Fig. 5. Attack dynamics

designed to send packets that must arrive at the server at the precise time. Obviously, this can be made from a single attacker or in a distributed way, becoming a DDoS in the last case.

4 Experimental Results

In this section, the attack behaviour is evaluated and its impact analyzed. The attack has been tested in a simulated scenario, by using Network Simulator 2, as well as in a real environment.

Prior to the evaluation of the attack, some indicators are going to be defined in order to measure the attack performance.

4.1 Performance Indicators

The parameters of interest are:

- *Percentage of seizures* (S): percentage of the seizures in the server that corresponds to the attacker.
- *Effort of the attack* (E): percentage of request packets emitted by the intruder, related to the total number of packets that can be accepted by the server.
- *User success percentage* (U): percentage of seizures made by the legitimate clients related to the total number of requests generated by them.
- *Overflow percentage* (O): percentage of unattended requests due to full queue condition related to the total number of received requests.

It should be noticed that not all the parameters are observable by the agents in the scenario. For example, only the effort of the attack is observable by the intruder during the attack due to the fact that only the server knows the total number of packets and seizures generated in the observation period.

The aim of the attack should be to minimize the user perception of the availability of the server (U). This can be achieved by maximizing S, due to the fact that if the server is engaged more time with intruder requests, the user success percentage (U) will be lower. Besides, in order not to be detected by any active

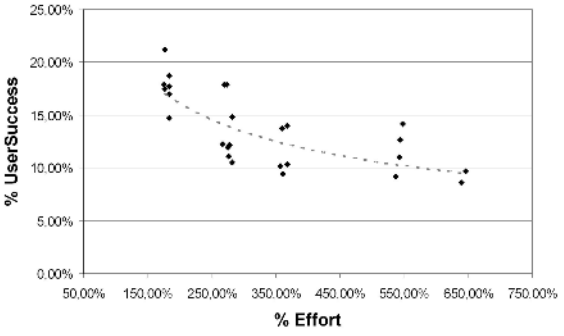


Fig. 6. User success and effective effort for 25 different configurations (t_{ontime} and Δ values) of the low-rate attack

intrusion detection system, the attack should also minimize its effort E. Minimizing E will contribute to a lower overflow O in the server, thus making the attack more undetectable.

4.2 Simulated Scenario

We are interested in discovering how effective the low-rate DoS attack can become. For that, a set of attacks has been analyzed in the simulator. The obtained results are really worrying due to the high effectiveness demonstrated.

As an example, in what follows the results obtained from one attack simulation composed by 1332 outputs are discussed. The attack has been launched against a server with $\overline{t_s} = 3.0$ seconds and $var(t_s) = 0.2$ seconds. The traffic generated by the user is enough by itself to keep the server busy all the time. The parameters of the attack for this example are: $t_{ontime} = 0.6$ s, $t_{offtime} = 2.7$ s, and $\Delta = 0.3$ s. Round trip time has been set to $\mathcal{N}(0.6\,s, 0.2)$. As expected, a very high efficiency is obtained: $S = 94\%$ and $U = 9\%$, which implies that only a 9.04 percent of the user requests are attended by the server. On the other hand, the overflow percentage $O = 77\%$ indicates that the traffic offered to the server by both the legitimate users and the intruders is about four times its capacity. In other words, only 22.9% of the requests are attended.

It is possible to adjust the effort of the attack (E) and, therefore, the ability to bypass an IDS system able to detect attacks on a given rate, by reducing *ontime* time and/or increasing *interval* at the expense of decreasing the effectiveness of the attack. In this context, Fig. 6 shows the variety of obtained vaules for the user success percentage and the effort for 25 possible settings for the attack to the previously defined server.

4.3 Comparison with a Naive Attack

The proposed attack strategy has to be measured not only in terms of effectiveness, according to the proposed indicators, but also in comparison with a *naive*

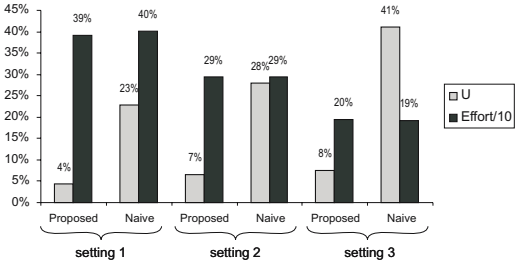


Fig. 7. Naive attack vs. proposed attack user success percentage values for three different efforts (packets rates)

attack. By *naive attack* we mean an attack with the same objective (to seize all the positions in the buffer) but carried out by simply sending packets in a fixed rate or randomly. Therefore, a naive attack is, in some sense, a brute-force attack, as it lacks any intelligence about the server's behaviour and is based in the exhaustion of the server's capabilities. However, we prefer the term "naive" as opposed to "brute-force" due to the fact that we are considering relatively low-rates for the attack.

The proposed attack would become useless if its figures does not improve those from the performance of a *naive attack* with a similar rate. The experimental results (Fig. 7) suggest an important impact in the expected behaviour when using the knowledge of the estimated inter-output time. As shown, the user success percentage of the proposed attack is about 20% lower, in absolute terms, when compared with the naive attack for the same effort of the attack. That's to say, the same rate of attack packets provides better results in denying requests from legitimate users when using the proposed dynamics for the attack. Furthermore, the difference for U between the naive and the proposed strategies increases as the attack rate (effort) decreases. This is an expected result; when using high rates both kinds of attacks should get the same performance due to the fact that the rate of arrivals at the server will be enough to saturate the capacity by itself (becoming, in this case, a brute-force attack).

4.4 Real Scenario

The proposed attack has been also tested in a controlled real environment to check its validity. The selected server is an Apache web server that keeps the condition of serving requests in an iterative way (`"ThreadsPerChild`= 1"). Although we positively know this is not a realistic scenario, as most of web servers are concurrent instead of being iterative, there exist some reasons for considering it. First, the argued vulnerability is present in every iterative server under the single condition of a predictable time distribution of the inter-output time or, equivalently, of the service time. In our opinion, this makes the iterative web server valid to test the behaviour of the proposed attack. Second, our interest is to extend this kind of study to concurrent servers, mainly to web servers, which

Table 1. Real and simulated attack performance

$\overline{t_s}$	$\overline{t_a}$		U	O	S	E
3	3.5	Simulated	10.4	71.4	90.5	260.8
		Real	9.8	69.4	91.4	239.7
5	6	Simulated	5.7	67.7	94.2	213.1
		Real	7.8	67.6	92.5	212.4
10	12	Simulated	3.0	64.4	97.2	198.3
		Real	6.4	65.5	94.3	201.6
15	17	Simulated	3.0	66.6	96.8	197.5
		Real	2.5	65.6	97.7	198.2
20	22	Simulated	3.0	65.0	97.2	197.5
		Real	4.3	65.1	96.0	196.2
25	28	Simulated	1.8	64.6	98.0	197.9
		Real	1.8	65.4	98.3	197.9

makes the iterative web server an interesting starting point. Furthermore, the next steps in our research, still in preliminary stages, confirm the existence of the vulnerability in concurrent servers.

We have considered that a client's petition consists in a connection request. The attack establishes connections and sends no messages on them, letting the web server to close the connection after a timeout period specified by the Apache directive `"Timeout"`, which corresponds to the service time, $\overline{t_s}$, in our model. Although it could be argued that there is no variance in the service time, it is not true due to two main reasons: there still exists some variability due to the processing of the connection request and, mainly, due to the variability in the RTT.

The real scenario is analogous to that one considered for the theoretical analysis. The user traffic has been generated following a Poisson process. A piece of software launches the attack from a single source. Both legitimate user and intruder traffic flows traverse a WAN network to reach the server, with a round trip time $\mathcal{N}(17\ ms, 0.05\ ms)$. Traces on the users and the intruder side have been issued for collecting the necessary data to calculate the attack indicators.

Table 1 shows some experimental results with a comparison among real and predicted values for different service times ($\overline{t_s}$) and user traffic arrival rates ($\overline{t_a}$). These rates have been selected in such a way that there is no congestion on the server if the attack is not carried out. The parameters of the attack have been tuned to $t_{ontime} = 0.4$ s and $\Delta = 0.4$ s for all the experiments.

We can even obtain better results in efficiency (lower U and higher S, with lower O) for the attack in a real environment in some cases. Two conclusions can be derived from these results: a) All the experiments we have 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 can be very high, showing that these vulnerabilities could be easily exploited in iterative servers.

5 Conclusions

In this work, a vulnerability present in iterative servers is described. It consists in the possibility that a potential attacker becomes aware about the statistics of the inter-output time of a given server. This vulnerability allows an intruder to perform a denial of service attack against an iterative server. The attack could be designed to nearly or completely saturate the capacity of the target system but, as a difference from the generalized brute force DoS attacks, it uses a relatively low-rate traffic to achieve its goals. Moreover, it is possible to tune the attack parameters in order to select the appropriate values for efficiency and load generated in the server. This distinctive characteristic could allow the attack to bypass, in many cases, existent IDS systems based on rate thresholds, becoming a non-detectable threat.

As a difference from other existent low-rate DoS attacks, this one threatens the application level, maintains the server engaged serving intruder requests and gets advantage of the knowledge of the inter-output time of the target server. This is opposed to the TCP-targeted low-rate attack defined in [15], that relies on selectively saturating the link in order to trigger TCP's congestion control mechanism. However, it has some common features with [15], what points out the existence of a new family of DoS attacks, characterized by the fact that they rely on vulnerabilities that consist in the a-priori knowledge of one timer of a protocol or end-system behaviour, and that allow the intruder to carry out the DoS attack with a low-rate traffic.

The fundamentals and details of the design of a possible exploit have been explained. We have demonstrated that this attack can be easily carried out and that it can obtain very efficient results. The potential risk presented by the attack is really worrying, due to the fact that it could behave very similar to legacy users, bypassing IDS systems and possibly affecting many services in a server.

The extension of this kind of attacks to concurrent servers is being researched jointly with a mathematical framework able to model the described behaviour. The preliminary experimental results obtained up to now show evidences of the existence of a analogous vulnerability in concurrent servers. On the other hand, the mathematical model under study points out some possible improvements in the proposed attack, which makes the associated threat more awesome.

References

1. Computer Security Institute and Federal Bureau of Investigation: CSI/FBI Computer crime and security survey 2001, CSI, March 2001. Available from <http://www.gocsi.com>.
2. Jelena Mirkovic and Peter Reiher: A taxonomy of DDoS attack and DDoS defense mechanisms. SIGCOMM Comput. Commun. Rev., 34(2):39–53, 2004.
3. M. Williams. Ebay, amazon, buy.com hit by attacks, 02/09/00. IDG News Service, 02/09/00, http://www.nwfusion.com/news/2000/0209attack.html - visited 18.10.2000.
4. CERT Coordination Center. *Denial of Service attacks*. Available at http://www.cert.org/tech_tips/denial_of_service.

5. D. Moore, G. Voelker, S. Savage: Inferring Internet Denial of Service activity, Proceedings of the USENIX Security Symposium, Washington, DC, USA, 2001, pp. 9-22.
6. P. Ferguson, D. Senie: Network ingress filtering: defeating Denial of Service attacks which employ IP source address spoofing, in: RFC 2827, 2001.
7. Global Incident analysis Center: Special Notice - Egress filtering. Available from <http://www.sans.org/y2k/egress.htm>.
8. X.Geng, A.B.Whinston: Defeating Distributed Denial of Service attacks, IEEE IT Professional 2(4)(2000) 36-42.
9. N. Weiler: Honeypots for Distributed Denial of Service, Proceedings of the Eleventh IEEE International Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises 2002, Pitsburgh, PA, USA, June 2002, pp. 109-114.
10. S. Axelsson: Intrusion detection systems: A survey and taxonomy. Technical Report 99-15, Department of Computer Engineering, Chalmers Univ., Mar. 2000.
11. R. R. Talpade, G. Kim, and S. Khurana: NOMAD: Traffic-based network monitoring framework for anomaly detection. Proc. of IEEE Symposium on Computers and Communications, pages 442–451, 1999.
12. J. Cabrera et al.: Proactive detection of distributed denial of service attacks using MIB traffic variables - a feasibility study. Proc. of the IFIP/IEEE International Symposium on Integrated Network Management, 2001.
13. J. Mirkovic, G. Prier, and P. Reiher: Attacking DDoS at the source. Proc.of ICNP 2002, pages 312–321, 2002.
14. C. Douligeris and A. Mitrokotsa: DDoS attacks and defense mechanisms: classification and state-of-the-art. Comput. Networks, 44(5):643–666, 2004.
15. A. Kuzmanovic and E. Knightly: Low rate TCP-targeted denial of service attacks (The shrew vs. the mice and elephants). Proc. ACM SIGCOMM'03, pages 75–86, Aug. 2003.
16. SANS Institute: NAPTHA: A new type of Denial of Service Attack. Available at http://rr.sans.org/threats/naptha2.php.
17. A. Adas, Traffic models in broadband networks, IEEE commun. Mag. 35 (7) (1997) 82-89.
18. M. Izquierdo, D. Reeves, A survey of statistical source models for variable-bit-rate compressed video, in Multimedia systems, pp. 199-213, Springer Verlag, Berlin, 1999.
19. R.E. Walpole, R.H. Myers, and S. L. Myers, Probability and Statistics for Engineers and Scientists, Sixth Edition, Prentice Hall College Div, 1997. ISBN: 0138402086
20. K. Fall and K. Varadhan: The ns manual. Available at http://www.isi.edu/nsnam/ns/.
21. R. D'Agostino, M. Stephens, Goodness-of-Fit Techniques. Marcel Dekker, Inc. (1986).