Computer Networks 150 (2019) 234–249

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Introducing the SlowDrop Attack

Enrico Cambiaso[a,*], Giovanni Chiola[b], Maurizio Aiello[a]

[a] IEIIT-CNR, National Research Council via De Marini, 6 Genoa, 16149 Italy
[b] DIBRIS Department, Università degli Studi di Genova via Dodecaneso, 35 Genoa, 16145 Italy

A R T I C L E I N F O

Article history:
Received 24 April 2018
Revised 21 December 2018
Accepted 7 January 2019
Available online 12 January 2019

A B S T R A C T

In network security, Denial of Service (DoS) attacks target network systems with the aim of making them unreachable. Last generation threats are particularly dangerous because they can be carried out with very low resource consumption by the attacker. In this paper we propose SlowDrop, an attack characterized by a legitimate-like behavior and able to target different protocols and server systems. The proposed attack is the first slow DoS threat targeting Microsoft IIS, until now unexploited from other similar attacks. We properly describe the attack, analyzing its ability to target arbitrary systems on different scenarios, by including both wired and wireless connections, and comparing the proposed attack to similar threats. The obtained results show that by executing targeted attacks, SlowDrop is successful both against conventional servers and Microsoft IIS, which is closed source and required us the execution of so called "network level reverse engineering" activities. Due to its ability to successfully target different servers on different scenarios, the attack should be considered an important achievement in the slow DoS field.

1. Introduction

The Internet is today used in almost every corner of the world, providing important benefits to its users and enhancing their lives reducing distances between individuals through communication. Due to a large scale adoption, its infrastructure and systems are often target of cyber-criminals. Internet users are indeed exposed to several security and privacy issues [20]. This exposure also has an enormous impact on critical infrastructures, national strategic assets whose incapacity or destruction would compromise public safety and security, leading to possible loss of humans life or social unrest. Such systems are expected to become increasingly dependent on Internet services [3] and exposures to emergent cyber-threats should not be underestimated. In particular, the network security field is nowadays threatened by several types of intrusive mechanisms [1], designed for different aims and targeting users or the network infrastructure itself.

In the cyber-security context, Denial of Service (DoS) attacks represent one of the most important intrusive technique to online systems [59]. These attacks severely compromise the availability of a network node, like a single host, a server, a network router, or even an entire network. First generation attacks were designed to either exploit particular vulnerabilities or flood the targeted system with huge amount of useless packets [52]. Under a DoS condition, that may last from a few minutes to even several days, users trying to communicate with the affected systems are not able to properly interact with them. In virtue of this, serious damages are caused not only to targeted services and organizations, but also to users themselves.

Although both exploit-based and flood-based threats are nowadays considered dangerous, various protection systems have been provided during the years [2,23,40,42,48]. Nevertheless, last generation threats, known as Slow DoS Attacks (SDA) [8], low-rate DoS (LBR DoS, or LDoS) [26], or application DoS attacks [57], are more difficult to counter [30,37,60], since they represent a mixed category of threats inheriting characteristics of both exploit-based and flood-based approaches. Although they make use of tiny amount of network bandwidth, SDA effects are similar to other attacks (i.e. flood-based), since they are able to successfully obtain a DoS state on the targeted systems. This is often possible due to the direct communication with the listening application daemon, instead of working only at the transport or network layer [8]. By targeting at a higher layer of the ISO/OSI model, resources needed to carry out an attack are reduced, due to the reduction of the number of simultaneous connections a daemon is able to handle [8]. In virtue of this, an attack can be carried out even by non powerful devices, such as small sized computers like Raspberry PI, network routers, or even mobile devices like smartphones or tablets [5,10].

* Corresponding author.
E-mail address: enrico.cambiaso@ieiit.cnr.it (E. Cambiaso).

https://doi.org/10.1016/j.comnet.2019.01.007

With the aim of protecting network systems from denial of service threats, Intrusion Detection Systems (IDSs) represent today an essential element of the network security field, designed and configured to efficiently detect attacks. Exploit-based threats are usually mitigated through signature-based IDS approaches [2], extrapolating prepared patterns of known menaces to efficiently detect and block them. Instead, flood-based attacks protection systems are typically based on anomaly detection models [47], analyzing the behavior of measurable characteristics and comparing it to a normal behavior. If a significant deviation from normal behavior is detected, an alarm is raised. For instance, a common feature adopted for detecting flood-based attacks exploits the changes statistically introduced by the attack in the network traffic flow [47]. These changes may affect several different parameters, such as the type/size of packets, the number of connections over the time, or the rate of packets associated with a particular communication protocol. Instead, concerning Slow DoS Attacks, although promising IDSs have been designed [30], an efficient detection system is currently missing, due to not only the novelty of such menaces, but also to the their behavior which is similar to the one of legitimate situations [8]. Therefore, slow DoS threats are today considered extremely dangerous and not mitigated in practice.

In this paper, we introduce an innovative Slow DoS Attack called SlowDrop. While most of the known SDA exploit specific server side timeouts [8], SlowDrop simulates a legitimate situation involving several nodes communicating with the server through an unreliable network connection. This behavior makes the proposed menace more difficult to counter.

Although the proposal of a novel threat may be unusual, we believe that knowing in advance offensive tools is fundamental in order to properly design efficient detection and mitigation systems. The proposed work should therefore not be considered as the release of a tool for cyber-criminals, but instead an essential resource for the network security world, providing researchers an important element to properly investigate the denial of service phenomenon.

The rest of the paper is structured as follows: Section 2 reports the related work on the topic. The proposed SlowDrop attack is described in Section 3, analyzing in detail how the attack works. Appropriate tests of the offensive tool are reported in Section 4. Section 5 introduces instead protection systems and methodologies for SlowDrop. Finally, Section 6 concludes the paper and proposes future work on the topic.

2. Related work

Since their appearance, Slow DoS Attacks have attracted researchers around the globe for different purposes. Although many works are focused on proposing mathematical models to represent SDAs [15,56], categorize them [8,9], analyze [21,41] or model [52] their behavior, other studies are focused on developing novel threats, such as the Shrew [26], LoRDAS [16] or Slow Next [10], or to design appropriate protection mechanisms, based on signature extraction [28], statistical analysis of network traffic [31], clustering [38] or machine learning approaches [30,53].

Historically, low-rate DoS attacks were first proposed as the Shrew attack, designed to exploit the TCP retransmission timeout mechanism, throttling TCP flows to a small fraction of their ideal rate to cause intermittent packet losses [26,33,51]. Zhang et al. [58] and Schuchard et al. [39] demonstrate how an attacker can launch a Slow DoS Attack on BGP routing sessions for compromising stability and network reachability of the Internet backbone. More recently, researchers examined the exposure to low-rate attacks from other applications such as Internet services [34,35], load balancers [36], wireless networks [50], or peer-to-peer networks [55]. Guirguis et al. prove that a low-rate attack can force a feedback control system to oscillate between the desired state and another state. The authors analyzed the effect of such attack on a web server [34,35].

Concerning slow DoS threats, web servers represent in general a particularly exploited typology of services exploiting specific timeout implementations of network servers. Macià-Fernàndez et al. proposed a threat called Low Rate DoS attack against Application Servers (LoRDAS), in which the attacker tries to estimate/foresee the instants when resources are going to be freed by the server [17], in order to seize them before any legitimate client can. Such preliminary estimations are exploited to concentrate short burst of new connection requests to specific periods, with the aim of seizing available connections on the server as soon as they are released [14–16].

Considering other threats exploiting specific servers timeout, Slowloris is considered the most known menace [27], most probably as a consequence of its adoption by the Anonymous group of hacktivists executing cyber-attacks in opposition of the 2009 Iranian presidential elections [54]. The attack works by sending incomplete requests to the server. Similar attacks are Slow HTTP POST [18], also known as RUDY, varying sent payload to hinder detection systems, and SlowReq [29], proposed by our research group, an attack also implemented for mobile devices under the name of SlowDroid [11], reducing required bandwidth at minimum. Unlike these threats, working at the application layer, the proposed SlowDrop targets the listening application daemon by working at lower layers.

Another well-known threat is Slow Read, designed to force the server to slowing down the responses [22]. The attack exploits the Δ_{resp} parameter [8], related to the duration of the responses of a request over TCP. In this case, the aim of the attacker is to seize connections as long as possible, by inducing the server to take long time to send the entire response to the client. The Δ_{resp} parameter identifies in fact the time passing between the start of a response and the end of the same response. Under a Slow Read attack, such parameter assumes long values. The approach exploited by the proposed attack is similar to Slow Read one: indeed, as Slow Read, SlowDrop exploits server response. Nevertheless, the technique adopted by SlowDrop is based on dropping selected packets, instead of simulating a low reception buffer [8,22]. Another attack similar to the one we propose is Sockstress, designed to reduce TCP window size to slow down (even indefinitely) the communication [43]. Unlike Sockstress, the proposed threat is not bounded to specific flags or data of the TCP protocol. In addition, since an extremely low (or null) window size may be easily flagged as an anomaly, thus making the attack mitigated in practice, we believe that SlowDrop's approach is more difficult to detect as malicious, because it simulates potentially legitimate situations. This characteristic, followed by SlowDrop, is also shared by Slow Next, a low-rate attack we have proposed in a previous work [9]. Like Slowloris or Slow Read, Slow Next exploits a specific server timeout, known in this case as Δ_{next} [8] and identifying the time passing between the end of a response and the start of the next request on the same connection stream. Being this exploitation accepted by many (TCP based) protocols supporting persistent connections (for instance, HTTP 1.1, SMTP, SSH), Slow Next presents a behavior particularly similar to a legitimate one, since it is expected that a legitimate user attends some seconds, or even minutes (for instance, in case of SSH), before sending an additional request/message to the server. Indeed, unlike other slow DoS threats, by analyzing Slow Next payload directed to the application daemon and packets sending times during the attack, the behavior is compliant to the protocol and it is not trivial to distinguish a malicious behavior from a legitimate one [9]. SlowDrop presents some similarities with Slow Next, since our aim here is to mimic the behavior of a legitimate client connected through a poor network link, hence, a particular but legitimate behavior is reproduced by the attack.

In fact, although the behavior during a SlowDrop attack may be associated to a common client, potentially, it is particularly difficult for the application to identify an anomaly. Such identification would indeed lead to a potential drop of lawful connections. In addition, to the best of our knowledge, the proposed Slow DoS Attack represents one of the first application layer threat able to target Microsoft IIS web servers. Being the proposed attack a slow DoS attack affecting Microsoft IIS, this ability makes SlowDrop an innovative cyber-threat, although a wide variety of well-known attacks against Microsoft IIS are available. Such threats exploit different approaches and vulnerabilities of IIS, such as buffer overflow [6], worm spreading [7], or exploit based DoS attacks like the recent vulnerability in IIS's HTTP protocol stack (HTTP.sys) with codename MS15-034 [19].

Attacks like Slowloris are almost ineffective against Microsoft web servers, since such threats are designed to seize all available connections the server is able to simultaneously serve. Since IIS is able to potentially manage thousands of connections simultaneously (against a few hundreds of Apache, for instance), a DoS state can't be reached through this approach. Although SlowDrop 's approach is unchanged when targeting, e.g. , an Apache web server, when attacking a different web server (like Microsoft IIS), instead, the attack behavior leads to an overload of the buffer resources of the server, thus potentially leading to the DoS. For instance, it may be needed to establish less than 100 connections with the server to exploit its weaknesses. We believe that these different effects caused by SlowDrop make the proposed threat unique in its kind. In addition, the proposed threat leads to the possibility to define a novel subcategory of Slow DoS Attacks specifically designed to simulate particular legitimate behaviors, including existent threats such as Slow Read, due to its simulation of a client with reduced buffers [8,22], and Slow Next, in virtue of its behavior, accepted by a protocol supporting persistent connections [9]. Due to its nature, the SlowDrop attack we introduce in this paper represents therefore an innovative Slow DoS Attack and it should be considered an important advance in the cyber-security field.

3. The SlowDrop attack

SlowDrop is focused on the exploitation of TCP based protocols and the attack emulates a client connected through an unreliable connection channel, such as a weak wireless connection. Since this exploitation simulates a legitimate user, the concept behind SlowDrop is similar to the one of Slow Next [9], a Slow DoS Attack simulating a legitimate user, by sending a legitimate request to the targeted server, hence receiving the related response, and finally exploiting the Δ_{next} parameter [8] before sending an additional request on the same connection stream. If we analyse a persistent TCP connection, the client is allowed to use the same communication channel for multiple subsequent requests. In this case, as previously anticipated, the Δ_{next} parameter identifies the time passed between the end of a response and the beginning of the next request, on the same connection stream. During such time, known as Wait Timeout [8], the attacker does not send any data to the server (on the same connection stream). By adopting such behavior, if communication is analyzed, the Δ_{next} parameter itself assumes values higher than usual [9]. When repeated on many connections, this behavior may lead to a denial of service on the victim, since all connections manageable by the listening daemon process are seized by the attacker and additional clients are not able to properly communicate with the server. An exhaustive description of the Slow Next behavior can be found in [9].

By analyzing Slow Next, it is characterized by legitimate traffic, both at the network and application layer, exchanged between the attacker and the victim. In addition, it may not be easy to flag its timeout exploitation as suspicious, since legitimate clients may behave similarly, e.g. similar Δ_{next} values may be related to a fair client parsing a web page before requesting an internal resource. Therefore, the attack simulates particular scenarios a legitimate client could experience.

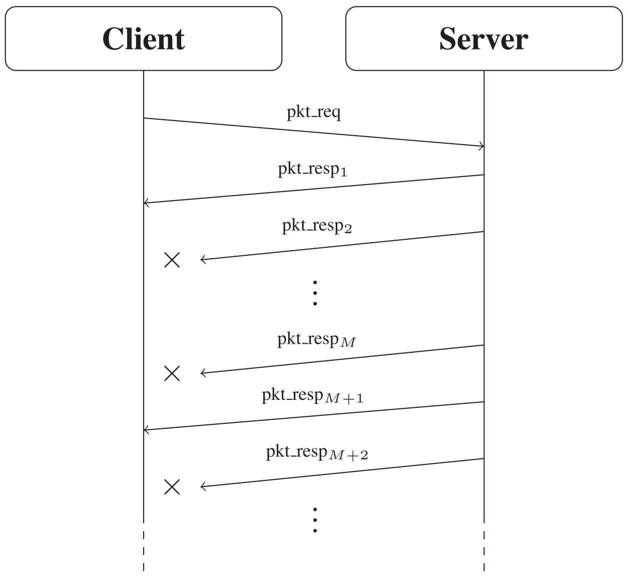

Fig. 1. Request-Response Packets Flow During a SlowDrop Attack.

On the same concept, the SlowDrop attack simulates a set of clients associated to unreliable connections. In this case, the Δ_{resp} parameter is exploited, related to the time required to send the response to the client [8]. In particular, as depicted in Fig. 1, the idea behind SlowDrop is to repeatedly request a specific (possible large, due to fragmentation needs) resource to the server, hence dropping received packets. This packet discarding action may occur on several legitimate cases like on a weak wireless connection.

In particular, we define R the packets dropping ratio, $0 \leq R < 1$, where $R = 0$ implies a legitimate situation (no packets are discarded), while $R = 1$, not considered, implies that all packets are dropped. Let $M - 1$ be the number of discarded packets before accepting a single packet, Fig. 1 reports $R = 1 - \frac{1}{M}$. In general, mathematically speaking, $M > 0$ must be satisfied. Therefore, accepted packets are indexed (by arrival order) $kM + 1$, with $k \geq 0$.

Considering the taxonomy reported in [8,9], SlowDrop may be considered a threat exploiting the Δ_{resp} parameter. As many others Slow DoS Attacks, SlowDrop approach is to directly target the listening application daemon. In particular, packets dropping is applied only on packets including application layer data.

It should be noted that a "smart" attack may accept packets by considering sequence numbers ordering. In this case, a packet may be accepted, for instance, by considering its sequence number, to be subsequent to the sequence number of the previously accepted packet. Nevertheless, since SlowDrop simulates an unreliable connection link, this "smart" acceptance is not implemented. Indeed, if we consider for instance a poor wireless connection, received packets order may not be driven by their sequence numbers, but instead by network connection characteristics[1]

3.1. Effects of the attack

If we analyze a single connection, in general, a connection is typically seized for the entire resource transfer operation. Considering the SlowDrop attack, although it may be thought that a

[1] This is not true for 3G and 4G connections, since the radio link protocol makes sure that packets arrive at the other end in the same order that they are received.

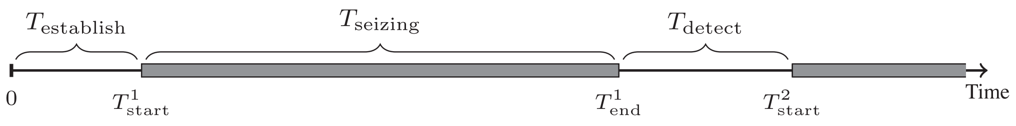

Fig. 2. Attack Behavior in Case of Server Side Connection Closure, Analyzed on a Single Connection.

packet discard leads to high retransmission rates, in practice the server would interpret the packet loss as a communication problem, hence extending connection seizing times. Therefore, induced packets retransmission and sending slow-down [32] may lead a vulnerable server to long connection resources occupations. This characteristic is shared with many other Slow DoS Attacks [8]. Although the packet discarding activity may not be necessary, in this case it is accomplished not only to enlarge resources occupation timings, but also to reduce the network bandwidth required for the transfer. This reduction leads the server to maintain the connections alive, until the transfer is completed. For R values tending to 1, SlowDrop may theoretically postpone the completion to extremely long times. In practice, an appropriate trade-off is needed to avoid server side connection closures. This R trade-off is needed in order to efficiently exploit servers resources. Although R values tending to 1 theoretically lead to extremely long seizings, in practice the server may interrupt the connection. In Fig. 2, it is reported the behavior of a generic slow DoS attacker[2] targeting a network service, from a single connection point of view.

According to the figure, we assume that the attack begins at the instant 0 . After some seconds, reported in Fig. 2 as $T_{\text{establish}} = T^1_{\text{start}}$, a connection is established with the server. The aim of the attacker is usually to reduce to a minimum this time [12], hence connections are often established at the beginning of the malicious activity. This is not always true: for instance, in case of a server applying limits to clients connection rate. Data transfer is then accomplished, if needed, according to the adopted attack strategy. Assuming the victim identifies some sort of anomalous behavior, or in general a connection closure activity is triggered, the server interrupts/closes the established connection. This closure is often applied at server-side. Nevertheless, in some cases a client-side closure may be preferable [14]. Assuming a server-side closure, according to the figure, it is accomplished at time T^1_{end}. Hence, connection established duration is $T_{\text{seizing}} = T^1_{\text{end}} - T^1_{\text{start}}$.

Identifying a connection closure takes some time to the attacker, ranging from a few seconds, even to some minutes in the worst cases, depending on the implementation of the threat. Once the closure is detected, the attacker usually instantiates a new connection with the victim, with the aim of seizing the resource again before any legitimate client does. According to Fig. 2, attacker side connection closure detection time is $T_{\text{detect}} = T^2_{\text{start}} - T^1_{\text{end}}$. Although not always true [16], it is in general important for the attacker to reduce T_{detect} values at a minimum, in order to quickly seize resources again once a closure is identified. Duration T_{seizing} of all the established connections typically assumes similar values, since it is often related to a specific server side timeout [8]. In the next section we will try to estimate the T_{seizing} value.

3.2. Connection seizing duration

We will now estimate the T_{seizing} duration of a single connection, during a SlowDrop attack. Let's assume a resource r sized $S'(r)$ bytes is requested by the attacker. Fragmentation of the resource operated at levels lower than the application one leads the server to send $P(r)$ packets, sized in average $S(r)$ bytes. Note that due to additional lower layers payloads (i.e. TCP or IP), $S'(r) < P(r)S(r)$, although in the same order of magnitude.

We define B the attacker reception bandwidth, in terms of bytes per second. We assume a packet discard ratio equal to R, according to the definition above.

We define $P_{\text{effective}}(r)$ the number of packets the server effectively sends during a SlowDrop attack to completely transfer the requested resource r to the client, according to Eq. (1).

$$P_{\text{effective}}(r) = \frac{P(r)}{1 - R} \tag{1}$$

According to Eq. (2), we also define Q the number of packets per second the communication channel can tolerate for the transfer of r, in proportion to the bandwidth capability.

$$Q(r) = \frac{B}{S(r)} \tag{2}$$

Hence, an estimation of the T_{seizing} seizing time in seconds of SlowDrop for a single connection (regarding the transfer of the r resource) is reported in Eq. (3).

$$T_{\text{seizing}}(r) \simeq \frac{P_{\text{effective}}(r)}{Q(r)} = \frac{P(r)S(r)}{B(1-R)} \tag{3}$$

For simplicity, since we focus on the application layer, we do not consider some additional packets, such as those of the request, the 3-way-handshake, or exchanged ACK messages. Also, we do not model congestion and flow control algorithms. Nevertheless, in case of high R values, the connection will be closed by the server, hence requiring the attacker to establish a new connection, hence, to accomplish a new 3-way-handshake. Instead, concerning ACK messages, since SlowDrop focuses on packet discarding, some packets will not be received during an attack, hence, ACK messages received by the server will contain different (usually, lower) acknowledge number values than in case of a normal connection. This would require from one side a client retransmission of unreceived packets, but it will also keep the connection alive for longer periods, also considering flow control algorithms, that in case of a SlowDrop attack would lead to a bandwidth reduction.

In addition, our estimation assumes all accepted packets are consumed by the client: for instance, instead, the client may receive and drop a packet with a wrong sequence number. In this case, T_{seizing} value may increase.

3.3. The packets dropping approach

As already mentioned, SlowDrop simulates a situation involving unreliable clients. In order to properly simulate such environment, the client host should not receive/interpret dropped packets, since packets reception may induce him to send related ACK packets to the victim, (wrongly) specifying a correct data reception. Our aim is instead to avoid such behavior, making the client application believe that packets have not arrived at all. Although it could be possible to selectively drop ACK packets directed to the victim, hence making the victim erroneously believe that sent packets have not reached their destination, this behavior would not correctly simulate an unreliable connection. A more accurate approach is therefore needed. The adopted method involves instead the drop of packets *before* they are parsed/interpreted by the client, thus maintaining an affinity with the behavior associated to an unreliable connection.

[2] Actually, if we think of a legitimate client, the scenario is pretty much similar.

In general, in order to drop packets before they are interpreted by the client application, two possible approaches are possible: (i) to drop packets *before they reach the destination*, for instance through a apposite network devices (like a firewall or a network tap), or (ii) to drop packets on the destination host *before they are interpreted by the application*. Of course, in case of network propagation of the packets, the overall network will be affected from the injected packets. Nevertheless, in the context of a SlowDrop attack, both the approaches lead to similar results, since the aim is to prevent interpretation from the server application.

3.4. Limits of SlowDrop

The SlowDrop attack presents two important limits. As specified above, a weak aspect of the proposed threat is represented by *the resource size*. Since transport layer fragmentation is exploited by SlowDrop, the attack is unsuccessful in case of low-sized responses. In particular, in order to exploit fragmentation, the size of the requested resource should be higher than the data carried out by a packet sized equal to the maximum segment size [32]. In this context, in order to bypass this limit of SlowDrop, further work on the topic may be directed to deeply investigate the behavior of the server when low-sized resources are requested by the attacker.

Another limit is represented by *the transport protocol*: the proposed threat is designed to only work over specific transport layer protocols (such as TCP) supporting packets retransmission. Considering for instance unreliable protocols, such as UDP, a packet loss does not induce a server side packet retransmission. Although this characteristic limits the range of exploitable services, many slow DoS threats are bounded to TCP-like protocols as well [8].

4. Executed tests and obtained results

In this section we focus on the tests we have executed to validate the effectiveness of SlowDrop. Our tests have been executed with the following aims: (i) to identify proper attack parameters, (ii) to analyze the effects of SlowDrop on a targeted server, (iii) to analyze SlowDrop performance on different network scenarios, (iv) to analyze the ability of SlowDrop to target different server applications, and (v) to compare SlowDrop to other similar threats.

With reference to the packets dropping approaches reported in Section 3.3, although effects of both the approaches are equivalent, we directly work on the attacking host, filtering and dropping packets before they are interpreted. This choice allows us to make use of a single malicious machine to execute the attack. In particular, on Linux based systems, it's possible to intercept network packets through *iptables* [45], a software allowing system administrators to directly configure the tables provided by the Linux kernel firewall. The attacking host has been configured to forward selected packets to the user-space, for processing. This is possible by using the *NFQUEUE* target [24] of Linux iptables packet filter. Packets processing is possible by interfacing to the packets' queue through apposite programming libraries, available in different languages. Adopted programming language used for the development of the filter is Python and related library is *python-nfqueue*.

4.1. Identifying proper R *values and resource sizes*

Standing to our statement reported in Section 3.4, the attack success depends on the size of the requested resource. Considering that, a first test set has been focused on identifying an appropriate size for the requested resource. Although a resource bigger than the maximum segment size [32] may be sufficient, by fixing $T = 600$ seconds the duration of each considered test, we focus on identifying the minimum size of the resource able to maintain connections alive (hence preventing server side closures) for at least T seconds. Our aim is therefore to adopt $T_{\text{seizing}} \geq T$.

As described in Section 3.2, T_{seizing} value directly depends on resource size and R. By fixing the resource size to a large sized file (adopted file size is about 700MB), we have to find a proper value of R. This value is found by iterating over possible values of R and opening a single connection with the server, with the aim of identifying $R_{\max}$ the maximum value of R before a server side connection closure is detected within T seconds. We focus on the maximum value, instead of the minimum, since reducing the number of packets accepted by the server, bandwidth is reduced too. Therefore, we expect in general to save more bandwidth using higher R values. Nevertheless, as previously explained, extremely high values may lead to server side connections closures. Although a server side closure may induce the attacker to establish newer connections with the server, for our tests it is not important to quickly identify the closure, since our goal here is to retrieve appropriate R values.

During our tests we have involved a single attacking host targeting a Linux based web server running Apache2 web server. Both the machines are connected through a high speed LAN connection.

Concerning the identification of $R_{\max}$, Fig. 3 reports results obtained considering single connections with the victim by varying R between 0 and 1 (due to the considerations reported in Section 3, $R = 1$ has been excluded), adopting an increasing step equal to 0.02, hence considering 50 different scenarios. The adopted step is not able to provide the real $R_{\max}$ value of our selected scenario, but it allows us to identify a neighborhood of that value. Once a closure is detected, no new connections are established with the server. Shown results are related to bandwidth requirements, analyzed on the attacking host. Fig. 3 also highlights connections closed by the server within T seconds.

It's possible to notice how attacker bandwidth usage directly depends on the R value: by increasing the dropping ratio, the overall attack bandwidth decreases. A first result is related to resource size: in the most bandwidth expensive case, with $R = 0.0$, overall bandwidth usage is about $2.193 \cdot 10^6$ bits per second, hence equal to a total of 157 MB exchanged from the client and the server, during $T = 600$ seconds. Therefore, since the requested resource is bigger than effectively exchanged bandwidth, it should be considered in this case a good resource.

In addition, tests show how bandwidth is significantly reduced by passing from $R = 0.24$ to $R = 0.30$. Therefore, a first good $R_{\max}$ selection may be to adopt $R_{\max} \geq 0.30$. In order to identify the best value, according to our scenario, we have to analyze performance for greater R values. In particular, for $R > 0.90$, since server side packets reception is hindered, as a consequence of the client side data dropping, connections are closed by the server, hence, we obtain $T_{\text{seizing}} < T$. Nevertheless, as bandwidth directly depends on the R value, in general decreasing with the increase of R, our final selection is $R_{\max} = 0.90$.

By adopting $R = 0.90$ and fixing $T = 600$ seconds, our aim is now to identify a proper resource size, in order of avoiding transfer completion within T seconds. We define $r_{\min}$ the minimum amounts of bytes the requested resources has to be composed of, in order to seize a connection for at least T seconds. In particular, according to Fig. 3, measured overall bandwidth rate for $R_{\max} = 0.90$ is about 2.5kbps for a duration of T seconds, corresponding to an object sized 192 KB. Therefore, we identify an average $r_{\min}$ value as $\bar{r}_{\min} = 192\text{KB}$. This value is not accurate, since it also includes the request, the response headers, and non application layer packets and payloads. Nevertheless, effective resource size $r_{\min}$ satisfies the equation $r_{\min} < \bar{r}_{\min}$. Therefore, any size equal or higher to $\bar{r}_{\min}$ is good for our purpose, including the adopted large size of about 700 MB. Indeed, is important to notice that in this case resource size does not affect attack performance: resource size di-

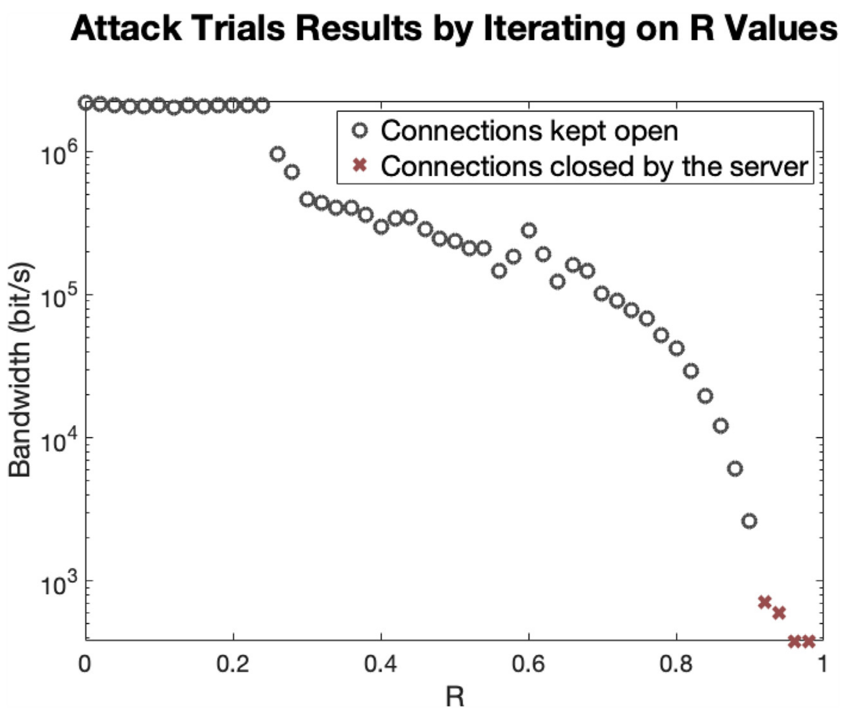

Fig. 3. Obtained Results by Varying R for Different Attack Executions of Duration $T = 600$ Seconds Each.

rectly depends on the adopted T and R values, but once an $r_{\min}$ size has been found, each value greater or equal to $r_{\min}$ is good. In particular, according to our definition in Section 3, by choosing $r \geq r_{\min}$, for $R = R_{\max}$ we always expect to obtain $T_{\text{seizing}} \geq T$.

4.2. SlowDrop Effects on the targeted server

Adopting the R and resource size values identified in the previous section, we have targeted an Apache2 web server with SlowDrop. The victim server has been configured to serve at most $N_{\max} = 256$ simultaneous connections. This value represents the maximum value for Apache2 web server [49], while default value is equal to 150. Higher values are possible only by adopting and configuring appropriate additional software modules.

During our test, we have configured SlowDrop to establish $N_{\max}$ simultaneous connections with the victim. Similarly to other Slow DoS Attacks, such as Slowloris [27], SlowDroid [11], or Slow Next [9], this behavior leads the attacker to seize all the available connections on the server, thus causing a denial of service. According to previous findings, we have adopted $R = 0.90$ and a request size able to allow us to maintain the connections seized for the entire attack duration, equal to $T = 600$ s.

Fig. 4 reports obtained results, analyzing on the targeted server the amount of established connections over the time.

In this case, the total amount of bandwidth required to the attack is equal to 652 Kbps. Our results show that the attack is successful, because all available connections are seized by the attacker, thus causing a DoS on the targeted server. In particular, according to the parameters introduced in Section 3.1, we have measured a maximum $T_{\text{establish}}$ value equal to 26, as the DoS condition occurs after 26 seconds from the instant the attack starts. Nevertheless, by deeply analyzing the number of connections, a single connection is closed by the server after about 593 seconds since the beginning of the capture, hence leading to a T_{seizing} time lower than the overall attack duration T. Under these conditions, the victim is able to serve a legitimate client (a single connection, in particular) communicating with the server. This situation has been confirmed through additional tests, executed for different $R > 0$ values and times, in which we have also observed that considering longer times, the number of connections involved in the closure slowly increases.

This behavior derives in particular from the *TimeOut* directive[3] of Apache2. This directive defines the amount of time the server will wait for certain events before failing a request. In particular, when writing data to the client, the TimeOut directive defines the length of time to wait for an acknowledgment of a packet when the send buffer is full. During our tests we have adopted a timeout equal to $TO = 300$ seconds, which represents the default value. Due to the randomness of packets receiving (hence dropping), the timeout may be triggered at any time, after 300 seconds from the beginning of each connection. In addition, it may refer to each connection.

Although this represents a limit of SlowDrop, it is simple for the attacker to identify a connection closure and re-instantiate the connection, hence maintaining the DoS over the time. Moreover, a different attack may send acknowledgments after about TO seconds, or, similarly, apply an acceptance delay of about TO seconds.

4.3. SlowDrop Performance on different network scenarios

Tests considered in the previous section refer to an attacker belonging to the same (LAN) network of the victim: the local network of our institute. In order to analyze attack behavior on real conditions, we have to consider different network types. In this context, we have considered two test cases, respectively related to wired and wireless networks. Due to the results obtained above, for this test set we have adopted $R = 0.90$. Instead, in this case the victim server has been configured to serve at most $N_{\max} = 150$ simultaneous connections. As mentioned, this value represents the default value of Apache2 web server, preconfigured after the installation of the daemon software.

[3] Regarding the TimeOut directive of Apache2, more information are available at the following address: http://httpd.apache.org/docs/2.2/mod/core.html#timeout.

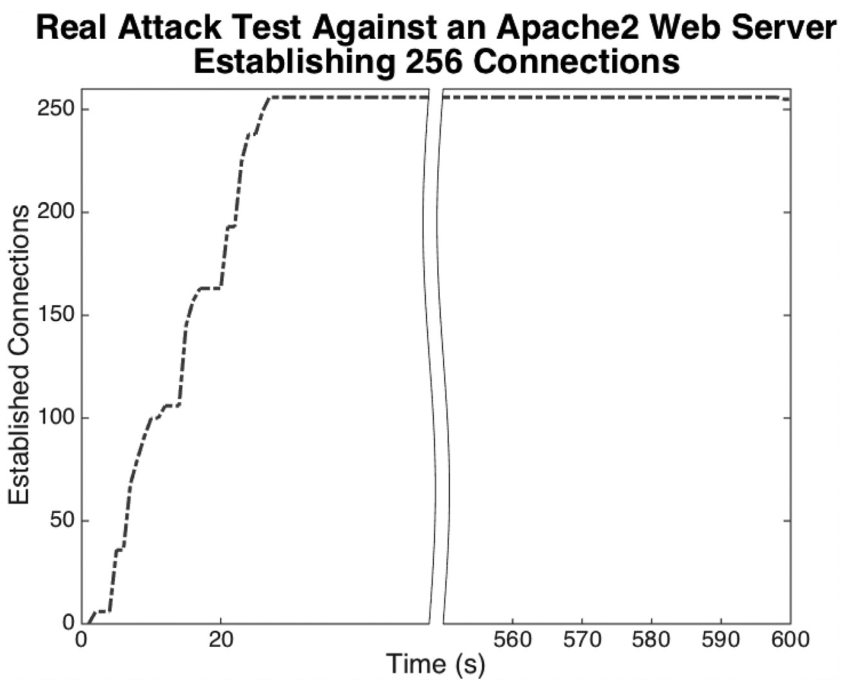

Fig. 4. Attack Effects on an Apache2 Web Server by Establishing 256 Connections and Adopting $R = 0.90$.

4.3.1. *Wired network tests*

Concerning wired networks, we have considered three situations:

- *LAN network*, placing the attacker on the same network of the victim (as previously considered);
- *WAN network*, placing the attacker on the same wide area network of the victim, on the network of our university. In particular, the adopted network is commonly known as *GARR network* and it represents the Italian Research & Education Network[4];
- *Internet network*, placing the attacker on a global Internet host, connected through an ADSL connection.

In all the considered situations, the victim server has been placed on the network of our research institute. Connection/link speeds measured during the tests are the following: 925 Mbps in the LAN scenario, 93.8 Mbps for the WAN one, and 321 Kbps for the Internet test case.

Obtained results are shown in Fig. 5, showing that for all the considered scenarios the DoS state is reached after a few seconds since the beginning of the attack, as all the $N_{\max} = 150$ connections are seized by the attacker.

Although results for the LAN scenario are similar to the previous ones, in this case the DoS is maintained for the entire attack duration. This result derives from the randomness of the network communications: for instance, the TCP protocol implementation may close connections related to a missing reception of proper sequence numbers. We therefore expect that for longer execution times, connections are “slowly” closed by the server. On the other cases, instead, a partial number of connections is closed after about 300 seconds since the beginning of the attack. While for the WAN scenario, the closure only affects 110 connections, concerning the Internet tests, all the connections are closed.

[4] Further information concerning the GARR network are available at the following address: http://www.garr.it/eng.

4.3.2. *Wireless network tests*

In this case, the communication with the server is established through a wireless network. We have considered in particular three network types:

- *Wi-Fi network*, placing the attacker on the same local network of the victim and connecting the attacker to the network through a wireless access point;
- *4G/LTE network*, connecting the attacker to the Internet through a 4G/LTE mobile network.
- *3G/HSPA network*, connecting the attacker to the Internet through a 3G based High Speed Packet Access (HSPA) mobile network;

Connection/link speeds measured during the tests are the following: 60.4Mbps in the Wi-Fi scenario, 23.3Mbps for the 4G one, and 1.19Mbps for the 3G test case.

Obtained results are shown in Fig. 6, showing that the attack is always successful: all the $N_{\max} = 150$ connections have been established by the attacker and are kept active during the time, thus leading to a DoS on the targeted server.

Nevertheless, as experienced in the results reported on Section 4.2, some of the established connections are closed. Although this closure does not affects the LAN case, characterized by T_{seizing} values greater than the overall attack duration T, and it marginally affects the Wi-Fi case, where attacker and victim belong are part of the same LAN network, in case of a mobile connection, after about 300 seconds, all connections are closed.

4.3.3. *Network type results discussion*

Analyzing the results we have obtained, it should be clear that attack performance differ by varying the typology of network adopted. The difference we have observed is in particular associated to the quality of the network connection: since SlowDrop may be seen as an attack that simulates a poor wireless connection link, associated to high packet losses, depending on the nature of the network, packet losses may be less frequent on a LAN wired connection, instead of on other scenarios that may be characterized by packet losses (e.g. due to limited radio coverage on a wireless scenario). Therefore, in order to execute an efficient attack, it is

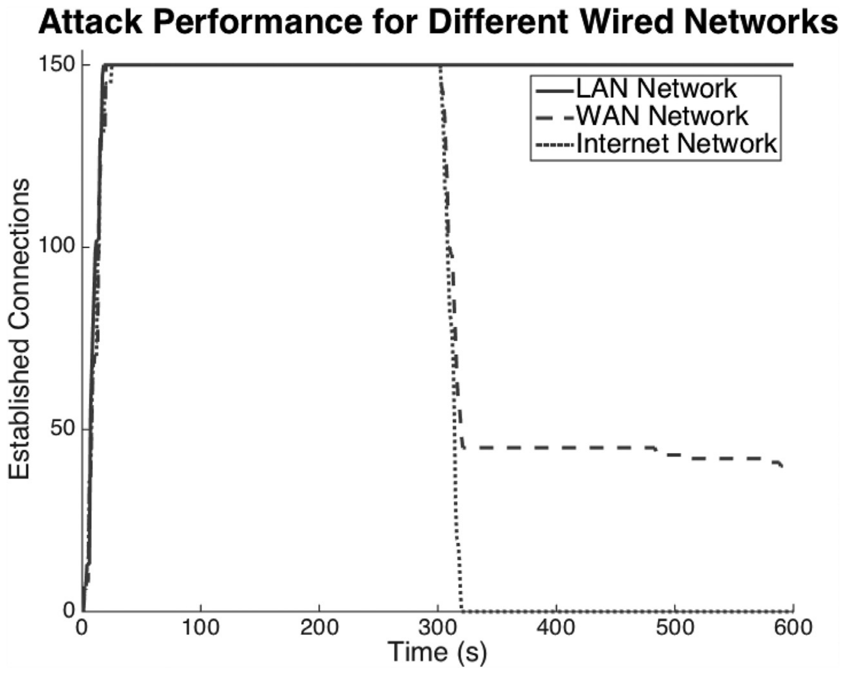

Fig. 5. Attack Effects on an Apache2 Web Server by Communicating through a Wired Network and Adopting $R = 0.90$.

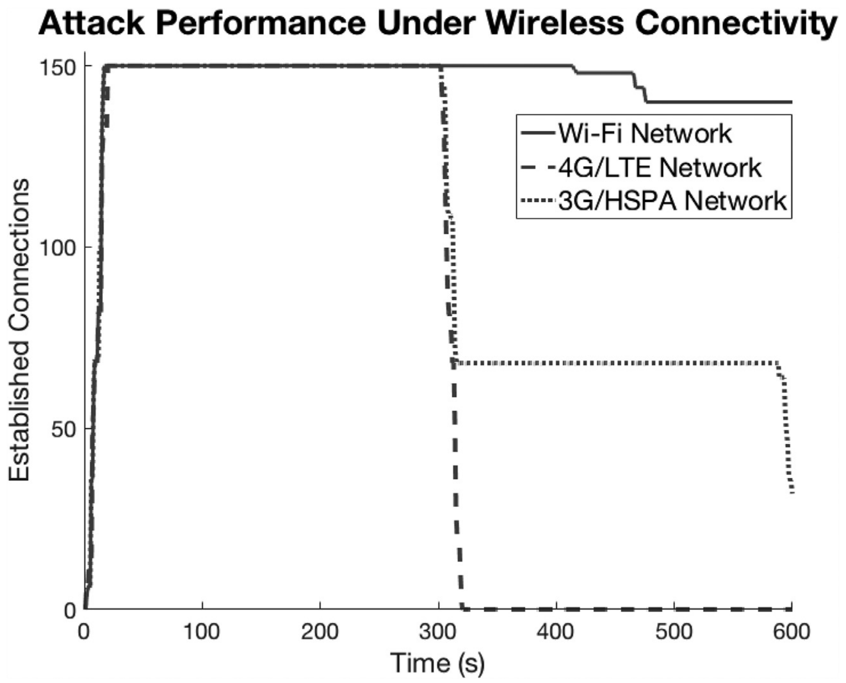

Fig. 6. Attack Effects on an Apache2 Web Server by Communicating through a Wireless Network and Adopting $R = 0.90$.

fundamental to adapt the R value according to the network characteristics.

Regarding wireless tests, instead, it is also important to consider the quality of the connection link (signal strength, collisions, etc.). Since the access point used for Wi-Fi (over LAN) tests was in proximity of the attacking network, packet losses are reduced. On the contrary, connection quality in case of mobile network tests was limited, especially for LTE connectivity, most probably because associated to low signal strength during the tests. Analogously to the wired situation, by reducing wireless connection quality, effective R value is implicitly increased, thus increasing packet closure possibility. Therefore, also in this case it is important to choose a proper R value. Nevertheless, since wireless connection characteristics depend on many factors (proximity to the cell, network load, interferences, etc.), it may be extremely difficult to identify an optimal R value.

Although connection closures have been experienced during the tests, we have preferred to maintain a common R value, in order to analyze different results for different networks, executing the attack with common parameters. We have obtained that the R packet dropping ratio value depends on the characteristics of the adopted network. In particular, by decreasing the performance of the network link, it is fundamental to reduce the R value accordingly. Nevertheless, it should be noted that the attack is always successful and independent from the network bandwidth available to the attacker. Moreover, an optimized attack may adopt (or set up) SlowDrop's approach to reduce needed bandwidth, hence re-establish connections once a server-side closure is detected. In this case the DoS state would be indefinitely maintained. In virtue of these considerations, SlowDrop should be considered an extremely dangerous threat, not only for its deployment over different communication systems, but also for its adaptation to the network.

Considering a better R value choice, we have for instance identified that by choosing $R = 0.80$, hence a lower R value, for the WAN scenario, performances are similar to the LAN case: no connections are closed in this case within $T = 600$ seconds. Therefore, choosing a good R value is crucial not only to reduce network bandwidth requirements, but also to enhance the performance of the attack. Further work on the topic may be focused on an accurate dynamic identification of R values, independently from the characteristics of the adopted communication medium.

4.4. IIS Servers affection

Slow DoS Attacks are able to successfully cause a DoS on a server through low bandwidth. Nevertheless, if we consider other timeout based threats [9] like Slowloris, SlowReq, Slow Read, or Slow Next, the range of potential victims does not include widely adopted server systems. Indeed, existent timeout based threats are often designed to target open source software, such as Apache, Squid, or common FTP servers. Among all the servers such threats can't exploit, Microsoft systems could represent the most important unexploited servers. This limit comes from the (mentioned above) approach of other timeout based DoS attacks, seizing all possible simultaneous connections on the victim, thus depriving legitimate clients to access the server. This approach leads to successful results on many typologies of servers. Moreover, the "open implementation" limit is reasonable, since open source software is easier to understand and, as a consequence, server's behavior is simpler to analyze and foresee. Nevertheless, when possible, a threat able to target a server independently from the implementation should be preferable.

In this context, during our study on SlowDrop we have found that its ability to target a different typology of server: Microsoft Internet Information Server (IIS) in particular, the proprietary web server implemented by Microsoft [25]. Being both Microsoft IIS and Microsoft Windows (the operating system hosting IIS software) closed source [44], it is more difficult to accurately analyze the server behavior. As previously mentioned, in virtue of this characteristic and their wider adoption, open source systems are often favorite by researchers and ethical hackers implementing cyberattacks. Nevertheless, when possible, it is important to consider also closed source systems, especially when, as in the case of Microsoft IIS (IIS in the following), their market penetration is considerable.

Analyzing the possibility to perpetrate a SlowDrop attack against IIS, we have found that the approach of seizing all available connections of the server can not be applied here. IIS implementation differ in fact from other software like Apache, since it is potentially able to serve thousands of connections simultaneously. Indeed, preliminary tests against IIS, executed by normally establishing thousands of connections with the server and requesting the default home page of IIS, showed us that the server can successfully handle all the connections without losing in performance. From these initial results, IIS appeared to be a particularly resilient server. Moreover, we can state that the connections management approach of the server differs from, i.e., Apache2, and, consequently, the attack approach should vary accordingly.

We will now describe the approach we have followed to target Microsoft IIS web server and the results we have obtained during our tests.

Considered duration of each test is equal to $T = 600$ seconds. During our tests against Microsoft web server, we have configured an IIS 8.5 server on a Microsoft Windows Server 2012 machine. Server and attacker are connected through a LAN network. Regarding the attack parameters and server configuration, we have maintained the same configuration which successfully leaded Apache2 to a denial of service. In particular, we have adopted $R = 0.90$ and during the attack each request to the server is related to a resource sized about 700MB. Due to the preliminary tests results (mentioned above), a smaller resource size would make SlowDrop ineffective.

4.4.1. Same configuration considered against apache2

Analyzing IIS configuration, it is possible to customize the maximum number of concurrent connections. By retrieving the default amount, it could be possible for the attacker to establish such amount of connections, thus leading to a DoS on the server, through the same approach adopted against Apache2 (see Section 4.2). Nevertheless, by default, such value assumes an extremely high value[5]: 4294967295, equal to $2^{32} - 1$. A different attack approach is therefore needed, since a "slow DoS attacker" can't afford to establish such amount of connections. Concerning SlowDrop, the attack works by slowing down reception through packet discarding. As reported in Section 3.4, this activity requires the attacker to request large resources. Therefore, even with low amounts of connections, the attack may be successful. For instance, in case of a not optimized implementation, a vulnerable server may replicate in memory the requested resource, in order to serve it. Such approach would lead to a DoS on the server, since by repeatedly requesting large resources, the server would easily consume its entire memory.

Since we couldn't measure a proper number of connections to establish with the server, a first test set has been accomplished by establishing the same (higher) amount of connections adopted for Apache2. In particular, we have established 256 connections with the server. As for previous tests, once a connection closure occurs, no additional connections are established with the server. Such approach is not optimal, from the attacker point of view, but it allows us to better understand the server behavior. According to previous tests, we have analyzed reachability capabilities at server side. The attack is not successful in this case, since the DoS state is not reached on the server.

We have therefore decided to insert an additional HTTP compression header to the requests sent to the server: `Accept-Encoding: gzip,deflate`. Through this header, the client specifies the server the supported compression schemes. Particularly, data are compressed in GNU zip format, by using the deflate algorithm. In this case, in particular, a gzip-encoded resource may be returned as a response. Such behavior would induce the server to compress the (large-sized) requested resource, hence leading to greater consumption of its computational resources. Obtained results are shown in Fig. 7. As shown, the DoS state is in this case reached and the server is not able to serve any client. Nevertheless, we have found that the DoS state is maintained for 187 s, instead of the entire duration of the attack. This makes such configuration not optimal in this context. Nevertheless, we analyzed that required overall attack bandwidth is in this case equal to 35.4Kbps. while previous tests related to Apache2 (see Section 4.2) reported bandwidth requirements of

[5] More information about Microsoft IIS connections limits are available at the following address: https://www.iis.net/configreference/system.applicationhost/sites/site/limits.

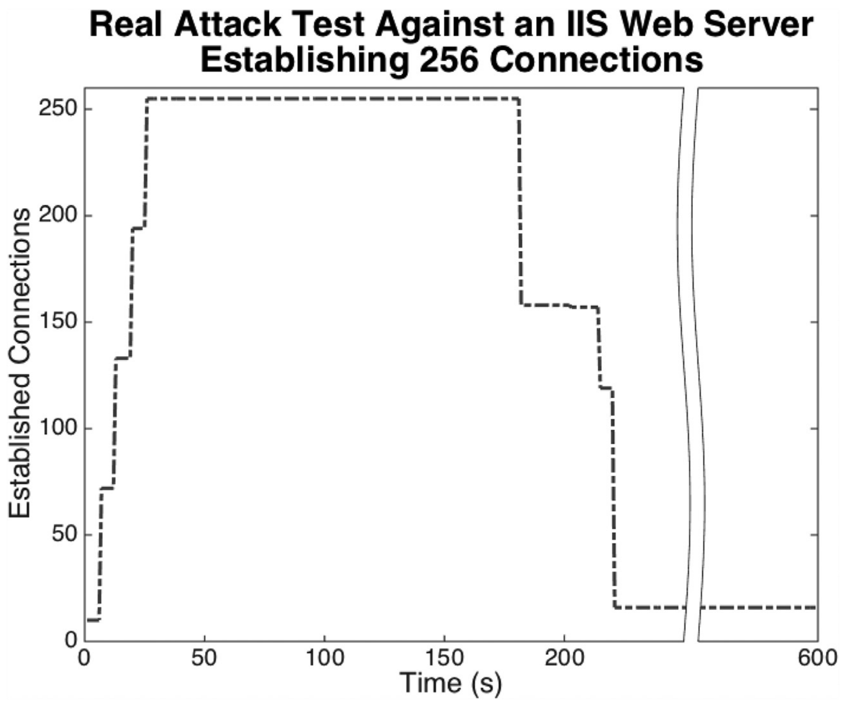

Fig. 7. Results Obtained by Targeting Microsoft IIS Server with SlowDrop Establishing 256 Connections and Adopting $R = 0.90$.

652Kbps. Although this is also related to a less effective threat, bandwidth requirements are in this case extremely reduced. This is an important characteristic of a Slow DoS Attack.

4.4.2. Attack variations

As a consequence of the results obtained previously, we have executed additional tests, with minor attack configuration changes. Variations are related to the size of the requested resource and the adopted R value. These changes derive from the following facts:

- according to the statements reported in Section 3.1, the connection time may depend on the resource size, since a completed transfer would normally induce the server to interrupt the communication. Therefore, as a consequence of the connection closures identified during the tests reported in the previous section, we have analyzed attack success in case a bigger resource is requested. We expect that for a bigger resource, seizing times may be longer. Since our aim is to maintain the DoS state for the entire attack duration, we do not consider smaller resources;
- by adopting $R = 0.90$, we have found that the attack is (even if not indefinitely) successful. We have decided to use different R values and analyze if the attack is still successful or not;
- since the number of connections we have adopted is related to the Apache2 configuration, providing the server the ability to serve at most 256 simultaneous connections, we have varied this number to analyze the success of the attack for different connection sizes.

Different size of the requested resource

We have executed tests by requesting a resource with different size. File size is about 4GB, hence nearly six times bigger than previously. Obtained results are similar to the case reported in previous section, related to a 700MB resource. In this case, the duration of the DoS state is 188 seconds and attack bandwidth is equal to 38.02 Kbps. Therefore, we can state that for high resource sizes, the effects of the attack are almost unchanged. Regarding the tests reported in the following, in order to maintain analogy with the tests considered so far, we have decided to adopt a 700 MB resource. Nevertheless, analyzing attack bandwidth, we can state that each file bigger than about 2.72MB is good in this context (assuming $R = 0.90$) and leads to similar results.

Different R *value*

Since the retrieved $R = 0.90$ value has been obtained analyzing the behavior of an Apache2 web server, a different value may be preferable when targeting IIS. Therefore, we have executed two additional tests by adopting respectively $R = 0.10$ and $R = 0.99$.

Concerning both the tests, we have found that results are similar to the tests related to $R = 0.90$, although in this case bandwidth requirements vary according to the adopted packet discarding ratio. In particular, regarding $R = 0.10$ tests, the DoS duration period is equal to 226 seconds, while attack bandwidth is equal to 7.08 Mbps (506 MB in total). Instead, concerning $R = 0.99$ tests, the DoS is maintained for 204 seconds and measured attack bandwidth is equal to 30.76 Kbps (2.2 MB in total). Bandwidth results are expected, since overall attack bandwidth depends on the R value. Since results are in line with previous findings, we can state that by varying the R value, performance of the attack are unchanged. This characteristic is particularly important, since results related to Apache2 attacks provided an asymmetry dependent on the network typology. We can therefore state that, unlike for Apache2, when targeting an IIS web server, network typology should not be relevant for a proper identification of the R value. As a consequence of the considerations made in Section 4.3.3, the "R is network independent" characteristic represents a meaningful result and an important feature of SlowDrop. In general, for the tests reported in the following, we have adopted $R = 0.90$.

Different number of connections established during the attack

We have executed tests by varying the number of established connections. In this case also, connections are not established after closure. We have executed two different tests by establishing respectively 50 and 1000 connections with the server. Obtained results show that, regarding the 50 connections test, attack results are not better than the 256 connections case. In particular, the

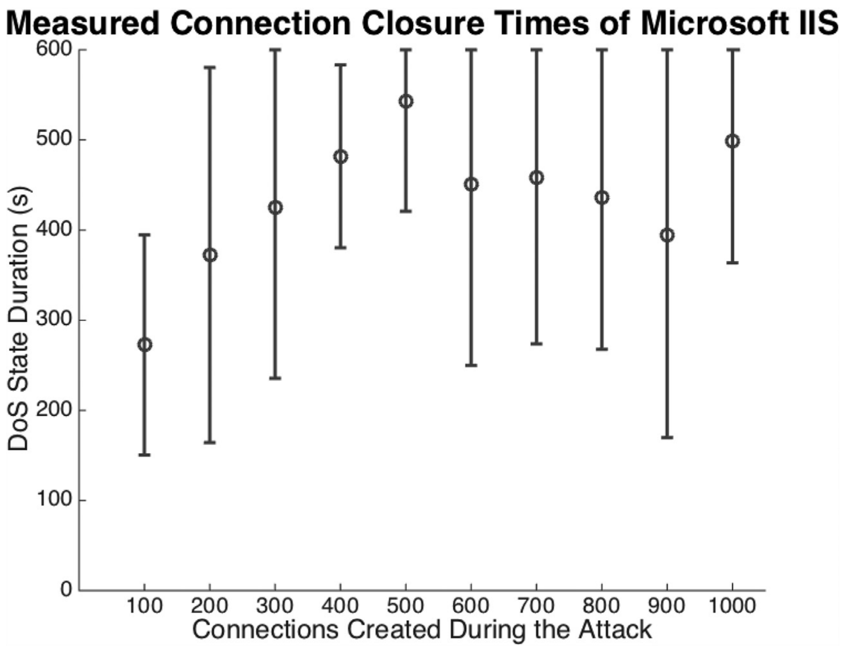

Fig. 8. Obtained Results for Different Attack Scenarios by Analyzing Service Reachability When Targeting Microsoft IIS.

DoS state is maintained for just 32 seconds. Concerning the tests on 1000 connections, instead, results reported that the test is successful and the server experiences a denial of service that is maintained for the entire duration of the attack. In addition, in this case connection speed is about 22.5 Kbps and about 1.6 MB of attack bandwidth are consumed: such tiny amount of bandwidth makes the attack extremely dangerous. Although the reduced amount of bandwidth may not be expected, it derives from the lack of connection closures, hence from a reduced number of packets.

Considerations

From the tests reported in this section we have concluded that adopted resource size, equal to 700MB, is acceptable for our aim. Moreover, the success of the attack does not depend on the adopted R packet discarding ratio value. Finally, the number of connections established during the attack assumes an important role for the success of the attack. In the next section we report the tests executed to identify server side connection closures in function of the adopted number of connections established by the attacker.

4.4.3. Monitoring DoS duration

Tests reported previously returned interesting results by varying the number of connections established by the attacker. We have therefore decided to analyze server status by varying connection closures. Since, as mentioned, IIS is a proprietary software, we can't know how it works and we can't understand in detail its behavior. Nevertheless, we can execute some sort of *network level reverse engineering*. We have decided to execute repeated tests by varying the number of established connections, within a range of 100 to 1000. Each test has been repeated for 12 times, in order to statistically analyze obtained data. For each test, we have measured the duration of the DoS state, in seconds, over a total of $T = 600$ seconds, and the attack bandwidth.

Fig. 8 reports DoS duration mean and standard deviation retrieved after the executed tests. The obtained results show that there is not a linear relationship between the DoS duration and the number of established connections. Moreover, variance of the results is particularly high (in some cases, the DoS duration exceeded the measured time T), hence, the duration of the DoS is in general dependent on many additional factors. Being IIS a proprietary software, properly identifying and monitoring connection closure triggers is not an easy task, and may be considered an extended work on the topic.

Regarding the obtained results, we analyzed that in a few cases (about 6.67%) the attack is not successful and DoS duration is equal to 0 seconds. Nevertheless, in general, Fig. 8 shows that the attack is in average successful and that mean DoS duration is equal to about 3 minutes, with peaks of about 9 minutes. Although retrieved 0 DoS durations represents ineffective executions of the attack, it is easier for the attacker to detect its failure, hence take appropriate actions. Similarly, it is possible to identify connection closures and establish additional connections with the aim of reach the DoS again.

The obtained results concerning the attack bandwidth are shown in Fig. 9, in terms of mean and standard deviation. As it may be expected, results show that attack bandwidth grows almost linearly by increasing the number of considered connections, with peaks of about 20Kbps.

Analyzing the obtained results, Fig. 8 shows that the most successful cases are (in average) related to 500 connections. As a consequence, due to the considerations made in Section 4.4.2, Fig. 9 shows that in case of attack success we have a reduced bandwidth, deriving from a lack of connection closing packets. Therefore, we can state that, in virtue of the obtained results, better results are achieved by establishing 500 connections with the server. Nevertheless, since the significance of such obtained value is not clear, further work on the topic may focus in this direction, by executing additional experiments in order to confirm or reject our hypothesis.

4.5. Comparison with other slow DoS attacks

We have executed tests comparing the proposed menace to other timeout based DoS attacks [9], on a real test environment in our institute LAN. We have targeted a common Apache2 web server with the proposed SlowDrop, in conjunction with Slowloris, SlowReq, Slow Read, and Slow Next attacks described in Section 2. Since attacks like Slowloris and Slow Read are designed to target only the HTTP protocol [29], we have focused on Apache2, which

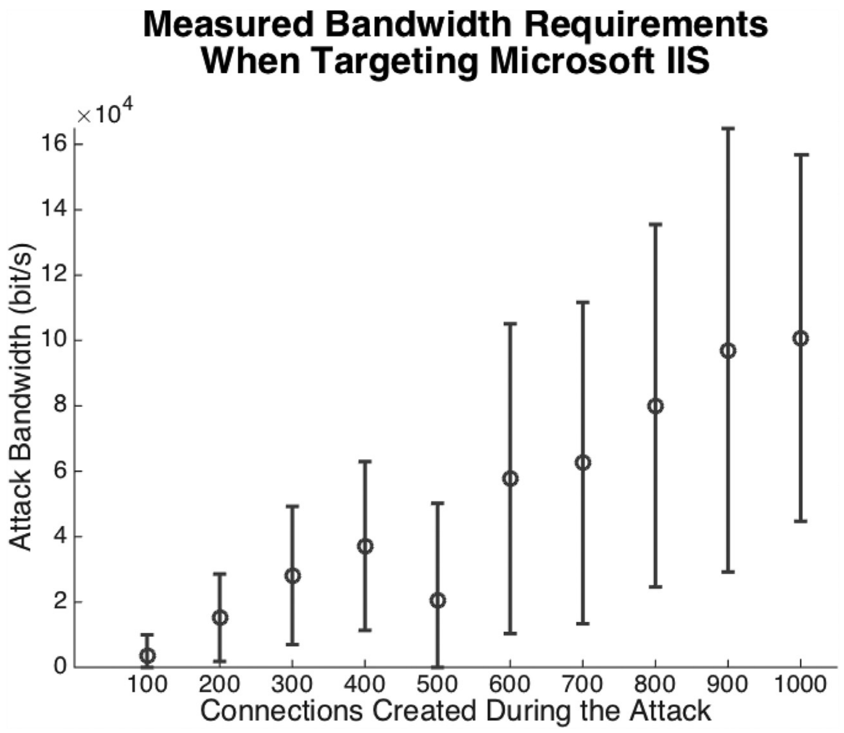

Fig. 9. Obtained Results for Different Attack Scenarios by Analyzing Attack Bandwidth When Targeting Microsoft IIS.

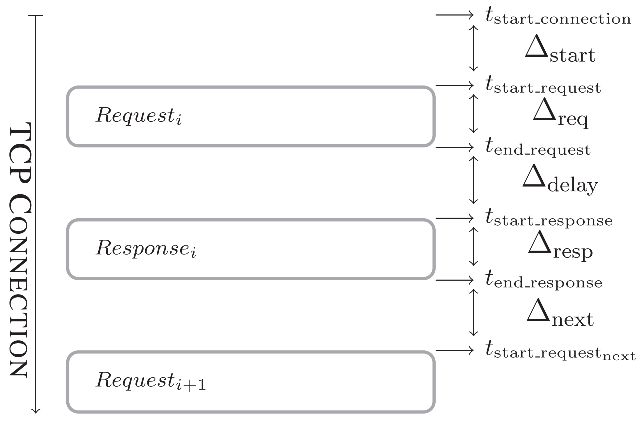

Fig. 10. TCP Connection Stream for an HTTP Connection.

represents one of the most important services in this context. As described in Section 2, since Microsoft IIS systems are not vulnerable to existent low-rate attacks (except the proposed one), we have excluded this service as well.

4.5.1. Timeout exploiting DoS attacks

As previously mentioned, existent slow DoS threats, known as Timeout Exploiting DoS attacks [9], work by exploiting a specific server side timeout, as reported in Fig. 10. Such an exploitation often gives the attacker the ability to reduce the network bandwidth. As reported in Fig. 10, depicting a TCP connection stream, various Δ parameters may be exploited (more information are available in [9]). For instance, since Slowloris threat exploits the Δ_{req} parameter, the connections it generates are characterized by extremely long requests. Instead, the proposed SlowDrop attack exploits the Δ_{resp} parameter by inducing the server to take long times to send a response resource to the client. Existent threats considered during the tests do not consider the exploitation of Δ_{start} and Δ_{delay} parameters. Regarding the Δ_{start} parameter, as described in [9], it has been excluded since its exploitation is similar to Δ_{next} one. Instead, concerning the Δ_{delay} parameter, as reported in [8], existent threats are nowadays mitigated and no more effective.

4.5.2. Testbed

During our tests, we have generated captured network traffic on the targeted server. Each considered trial refers to a capture of $T = 600$ seconds of live network traffic. Since our aim here is to compare the ability to carry out an attack and analyze required bandwidth for the attacker, we have considered a SlowDrop execution through a (LAN) network link associated to a limited connection speed. We have decided to limit the connection speed related to SlowDrop since our aim here is to reduce bandwidth requirements for the attacker, although maintaining the ability of SlowDrop to cause a DoS on the server, thus executing a Slow DoS Attack [8]. Through traffic shaping techniques, we have limited both download and upload bandwidth to 10kbps. We used a discard ratio $R = 0.90$. These values represents the best options obtained during preliminary tests. The bandwidth reduction choice derives from the results obtained in Section 4.3.3, stating that SlowDrop is able to adapt resources usage in function of the available network. We believe that this is an important characteristic of SlowDrop, since it gives the client the ability to carry out an attack independently from the connection network.

As previously mentioned in Section 3, timeout based threats make use of a Wait Timeout to alternate activity periods to inactivity ones, for data sending operations, with the aim of reducing attack bandwidth [8]. Instead, the proposed SlowDrop does not make use of such timeout, in favor of a packets dropping approach. Adopted a Wait Timeout for Slowloris, SlowReq and Slow Read is equal to 60 seconds, which allows us to maintain the connections alive on most situations, since default Apache2 timeout is equal to 300 seconds. Regarding the Slow Next attack, a different timeout is needed [9], due to different directive of the server, equal by default to 5 seconds. Therefore, concerning Slow Next only, in order to avoid a server side connection closure, we adopted a Wait

Table 1
Obtained Results Analyzing Different Situations on an Apache 2.2.22 Web Server for $T = 600$ seconds.

Attack		Slowloris	SlowReq	Slow Read	SlowDrop	Slow Next
Expl. Timeout		Δ_{req}	Δ_{req}	Δ_{resp}	Δ_{resp}	Δ_{next}
N_{max}		150	150	150	150	100
DoS		✓	✓	✓	✓	×
$C \to S$	pkts	1376	1915	4989	1129	27424
	B	138518	129132	469992	142187	2632774
	bps	1846.91	1721.76	6266.56	1895.83	35103.65
$S \to C$	pkts	1184	1701	4667	2348	13976
	B	79360	113474	3373556	2022042	5132705
	bps	1058.13	1512.99	44980.75	26960.56	68436.07
Total	pkts	2560	3616	9656	3477	41400
	B	217878	242606	3843548	2164229	7765479
	bps	2905.04	3234.75	51247.31	28856.39	103539.72

Timeout equal to 4 seconds. Attacks were configured to send default payload messages.

The server has been configured to serve at most (the default value of) $N_{server} = 150$ simultaneous connections. During our tests, we have configured the attack tools to establish and maintain alive such amount of connections with the victim, which represents the minimum amount of connections needed to potentially cause a denial of service on the targeted server. Indeed, under these conditions, an additional connection would not reach the listening daemon until an already established connection is closed. Nevertheless, by default, the maximum number of persistent simultaneous connections is equal to $N_{persistent} = 100$. Therefore, concerning Slow Next only, without varying default configuration of the host, we have configured the attack to establish in this case $N_{persistent}$ simultaneous connections with the server. In virtue of this, since the default Apache2 configuration is adopted, the Slow Next attack is not able to cause a denial of service on the server. Nevertheless, in our scenario, the attack is able to seize a considerable amount of resources of the victim, hence reducing its ability to manage legitimate connections. Because of this, and since it is considered an important slow DoS threat [9], we have included it into the tests as well.

4.5.3. Obtained results

We define N_{max} the maximum number of simultaneous connections established by the attacks. For each scenario, we considered client-to-server ($C \to S$), server-to-client ($S \to C$), and aggregate/total communications. The results we obtained during the tests are shown in Table 1.

The obtained results show that all the attacks successfully establish the predefined amount of connections with the server. Therefore, as expected, except in case of Slow Next, a DoS is always reached on the server.

Comparing the results of SlowDrop to Slow Next, it is clear that bandwidth requirements are extremely lower for the proposed threat. In addition, as previously mentioned, SlowDrop is able to successfully carry out a denial of service on the victim, while Slow Next leads the server to a “partial DoS”.

Considering instead SlowDrop versus Slow Read, although the same timeout is exploited by both the threat, in this case also, SlowDrop required bandwidth is considerably lower. This result should be interpreted as an important achievement, since in the context of attacks exploiting the Δ_{resp} parameter, the proposed threat represents an innovative attack making use of lower amount of bandwidth than other similar threats.

Regarding Slowloris and SlowReq, their exploitation of the Δ_{req} parameter makes such threats require particularly low amounts of network bandwidth. This is a recognized characteristic of these offensive tools [8,27]. These results were therefore expected. Nevertheless, results show that analyzing the number of exchanged packets, the results we obtained are similar for all the threats. Therefore, anomaly detection systems such as [30], identifying anomalies on exchanged packets may not reveal a running SlowDrop attack, especially considering that among the considered attacks, SlowDrop is associated to the lowest amount of client-to-server packets.

Moreover, if we analyze the behavior of the threats, it is well-known that the Wait Timeout parameter makes existent timeout exploiting DoS attacks repeatedly execute short attack bursts, followed by inactivity periods [8]. This characteristic is not associated to the proposed SlowDrop, which is instead related to a continuous traffic, more similar to a legitimate one. Fig. 11 reports the network traffic flow in terms of exchanged packets, comparing SlowDrop with Slowloris and SlowReq.

As shown in Fig. 11, Slowloris and SlowReq threats are characterized by short attack bursts/peaks, corresponding to the Wait Timeout expirations. Instead, SlowDrop traffic is distributed in a more uniform way during the time. Comparison with Slow Read and Slow Next (not reported in Fig. 11, for readability reasons) evidenced instead a network traffic flow similar to SlowDrop. The network traffic flow characteristics, not characterized by peaks, in conjunction to the ability of SlowDrop to successfully carry out an attack, make the proposed threat more dangerous and more difficult to counter.

Finally, concerning SlowDrop only, Fig. 11 shows that the packets flow is slowed down during the time, as a consequence of the dropping activities. In addition, results reported in Table 1 prove that an important characteristic of SlowDrop is related to the asymmetry between incoming ($C \to S$) and outgoing ($S \to C$) bandwidth, from the server point of view. This characteristic should not be underestimated, since protection systems often assume as legitimate the outgoing traffic flow of the server. In virtue of the retrieved results, the proposed SlowDrop threat should be considered extremely dangerous. In addition, although it is associated to higher bandwidth consumption, its ability to adapt on the network of the attacker (*network independence*) makes such threat able to successfully perpetrate an attack even on extremely slow connections.

5. Attack detection and mitigation

Concerning denial of service threats detection and mitigation, it is important to distinguish between DoS attacks and Distributed DoS (DDoS) attacks. While in the former case a single attacking host is involved in the malicious activity, in the latter case more nodes are (willingly or not) involved in the attack. It should be noted that the execution of a non distributed SlowDrop attack would be associated to a single IP address, the one of the attacking node: since SlowDrop exploits the TCP protocol to reach the listening daemon on the victim host, connections have to be actually established with the server, and, unlike flooding attacks, IP spoofing activities have to be excluded here, hence exposing the attacking host on the network. Therefore, considering standalone DoS attacks, it's possible to efficiently detect and mitigate a running SlowDrop attack by analyzing clients IP addresses of the received packets. For instance, it's possible to limit the number of simultaneous connections associated to a common IP address, in order to maintain reachability on the server. This approach is particularly effective and it could involve network firewall devices, or the server itself, through appropriate software or modules [29]. Although this node may be directly used by the real perpetrator, an astute attacker may expose a third party (infected) node instead of his own machine. This characteristic is shared with many other Slow DoS Attacks, such as timeout based threats like Slowloris, Slow Read, or Slow Next [8].

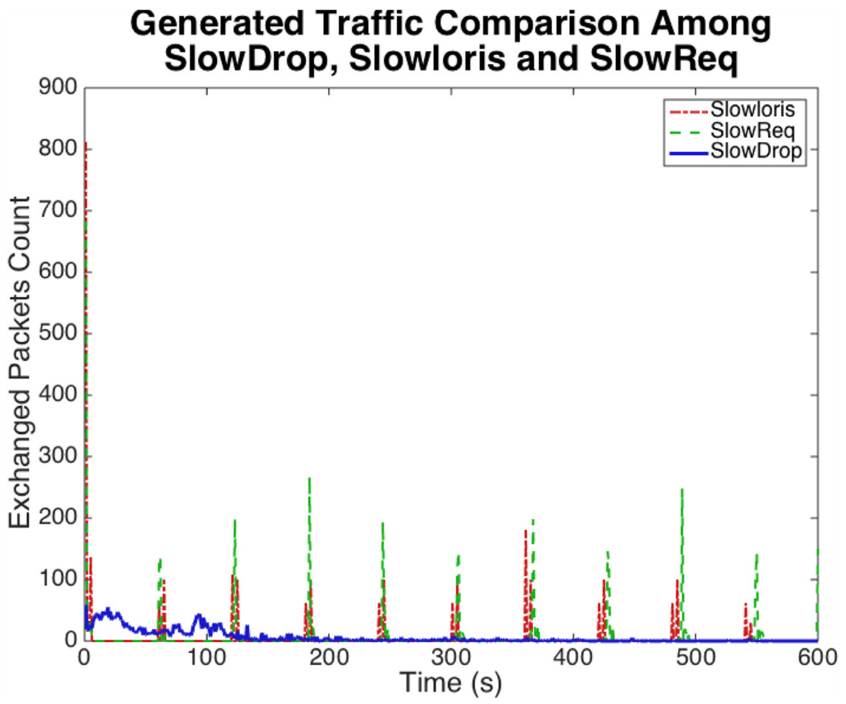

Fig. 11. Obtained Results by Analyzing Exchanged Packets Count Comparing SlowDrop With Similar Threats.

Although during our tests we have considered a DoS attack executed from a single attacking host associated to a single IP address, it should be simple to distribute the attack on many machines, executing a coordinated malicious activity against a common server system. In this case, attack detection and mitigation is more difficult, as it's particularly difficult to distinguish between a legitimate situation or a malevolent one [37,60]. Slow DoS Attacks are indeed especially difficult to counter, due to the reduced attack bandwidth and behavior, similar to a fair one, analyzing packets payload or the shape of the generated traffic.

Regarding SlowDrop, we have the following facts that should be considered for an efficient attack detection and mitigation:

- sent and received payload is legitimate, since it's composed of legitimate requests and responses. Attack detection/mitigation should therefore not be based on analyzing the payload directed to the application daemon;
- as previously described, the same packet dropping characteristics may be related to legitimate clients connected through a weak link. This fact makes detection particularly challenging, since an appropriate detection system should be able to distinguish a SlowDrop attack from those legitimate clients;
- the induced anomaly is related to communications coming from the server, while, typically, attacks tend to generate anomalies on data sent by the client/attacker. In virtue of this, a detection/mitigation system should appropriately consider server-to-client communications.

Although, to the best of our knowledge, an efficient detection and mitigation system against a distributed execution of SlowDrop is still missing, promising algorithms in this context (such as the ones proposed in [13,30,46]), involving research areas such as statistics, machine learning, neural networks, and spectral analysis, may be adapted to efficiently protect a network system from the proposed threat.

6. Conclusions

In this paper we proposed an innovative slow DoS threat attacking application servers called SlowDrop. The possible range of targets of SlowDrop is very wide, ranging from simple personal servers to widely used services on the Internet, or even critical infrastructure system services. Targeted services are based on the TCP protocol. Samples of such services are HTTP, SMTP, or SSH. We described in detail how SlowDrop works, proposing different possible implementations, the limits of the attack, and illustrating the effects of SlowDrop on the victim host. Since the attack may work at the operating system level (or at the network level, if deployed on a network tap), it should be considered particularly dangerous: for instance, a malicious kernel software upgrade may be released, thus recruiting thousands (or even millions) of attacking nodes unwillingly becoming part of a botnet. In this case, the eyes of the recruited user, the attack effects may be visible only as an, even extreme, reduction of the network bandwidth.

In order to execute accurate tests, we identified appropriate attack parameters, by analyzing how the success of SlowDrop varies by changing those parameters. We then targeted a common web server vulnerable to similar slow DoS threats and we observed that the proposed attack can successfully lead to a DoS on the server, hence presenting results similar to other available Slow DoS Attacks. We also analyzed the success of the attack when it is executed from different networks, by considering both wired and wireless (Wi-Fi, 3G/HSPA, 4G/LTE) networks, obtaining that by changing the network, attack parameters may change accordingly. Extension to the work may be focused on an accurate identification and categorization of packet discarding ratio values, in function of the characteristic of the adopted communication medium. Additional work may also be focused on deeply analyzing and modeling the effects of the attack on the server, in case of wireless network connections, by properly considering wireless properties such as signal strength, interferences, or wireless network load.

We also executed tests against Microsoft IIS, a proprietary web server not vulnerable to the majority of slow DoS attacks, due to its requests handling procedures. We obtained that SlowDrop is almost always successful, but its success depends on various factors, internally implemented on the implementation by Microsoft. Indeed, being the software proprietary, it is more difficult to identify the source of the triggers that makes the server close malicious connections. Our approach is based on some sort of *network level reverse engineering*, where we analyzed the behavior of the server during the attack, in order to deduce the internal working of the system. Further work could therefore concern a deeper analysis of the server's ability to respond to a running attack, by also defining in detail the adopted approach, and to execute additional experiments in order to validate our hypothesis.

Additional tests we have executed were focused on a comparison between SlowDrop and other similar slow DoS threats, in terms of success of the attack, required bandwidth, and generated amounts of packets. The obtained results show that, although required bandwidth is higher than other threats such as Slowloris, the attack is successful and its behavior is not characterized by traffic peaks like for other Slow DoS Attacks, hence resulting more similar to a legitimate traffic. Additional tests in this topic may concern the comparison with other unconsidered threats, or, similarly to the work reported in [29], on analyzing the success of SlowDrop when targeting a protected server. Similarly, further work may be directed to adopt the concepts behind SlowDrop to exploit systems serving resources of small size, like common web pages or simple texts.

In addition, further work may be directed to the execution of SlowDrop on transmission protocols different from TCP. For instance, it may be interesting to analyse how SlowDrop behaves on the Quick UDP Internet Connections (QUIC) protocol [4], an UDP based protocol proposed by Google to reduce retrieval time of web resources. Finally, additional work on the topic may be focused on a proper design and proposal of an innovative detection and mitigation system. Being the attack an innovative threat able to target previously rarely considered systems, it is fundamental to define appropriate protection systems able to counter SlowDrop.

Acknowledgement

This work has been supported by the following research projects:

ANASTACIA, funded by the European Commission (Horizon 2020, call DS-01-2016). Grant Agreement Number 731558

MHMD, funded by the European Commission (Horizon 2020, call ICT-18-2016) Grant Agreement Number 732907.

References

[1] L. Molina, A. Furfaro, G. Malena, A. Parise, A simulation model for the analysis of ddos amplification attacks, Modelling and Simulation (UKSim), 2015 17th UKSim-AMSS International Conference on (2015) 267–272.
[2] A.A. Alfantookh, Dos attacks intelligent detection using neural networks, J. King Saud Univ.-Comput.Inf. Sci. 18 (2006) 31–51.
[3] F. Cao, S. Malik, Vulnerability analysis and best practices for adopting ip telephony in critical infrastructure sectors, Commun. Magaz., IEEE 44 (4) (2006) 138–145.
[4] G. Carlucci, L. De Cicco, S. Mascolo, Http over udp: an experimental investigation of quic, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM, 2015, pp. 609–614.
[5] L. Caviglione, A. Merlo, The energy impact of security mechanisms in modern mobile devices, Netw. Security 2012 (2) (2012) 11–14.
[6] E. Chien, P. Ször, Blended attacks exploits, vulnerabilities and buffer-overflow techniques in computer viruses, Virus 1 (2002).
[7] J.R. Crandall, F.T. Chong, Minos: control data attack prevention orthogonal to memory model, Proc. 37th Annual IEEE/ACM Int. Sympos. Microarchitecture (2004) 221–232.
[8] G. Chiola, E. Cambiaso, G. Papaleo, M. Aiello, Slow dos attacks: definition and categorisation, Int. J. Trust Manag.Comput. Commun. - In Press Article (2013).
[9] G. Chiola, E. Cambiaso, G. Papaleo, M. Aiello, Designing and modeling the slow next dos attack, Int. Joint Conf. (2015) 249–259.
[10] G. Chiola, E. Cambiaso, G. Papaleo, M. Aiello, Mobile executions of slow dos attacks, 2015b.
[11] G. Papaleo, E. Cambiaso, M. Aiello, Slowdroid: turning a smartphone into a mobile attack vector, 2014 2nd International Conference on Future Internet of Things and Cloud (FiCloud) (2014) 405–410.
[12] G. Papaleo, E. Cambiaso, M. Aiello, Slowcomm: design, development and performance evaluation of a new slow dos attack, J. Inf. Secur. Appl. 35 (2017) 23–31.
[13] M. Ficco, G. D'angelo, F. Palmieri, S. Rampone, An uncertainty-managing batch relevance-based approach to network anomaly detection, Appl. Soft. Comput. 36 (2015) 408–418.
[14] J.u.E. Díaz-Verdejo, G. Maciá-Fernández, P. García-Teodoro, Evaluation of a low-rate dos attack against iterative servers, Comput. Netw. 51 (4) (2007) 1013–1030.
[15] J.u.E. Díaz-Verdejo, G. Maciá-Fernández, P. García-Teodoro, Mathematical model for low-rate dos attacks against application servers, Information Forensics and Security, IEEE Transactions on 4 (3) (2009) 519–529.
[16] P. García-Teodoro, G. Maciá-Fernández, J.u.E. Díaz-Verdejo, F. De Toro-Negro, Lordas: a low-rate dos attack against application servers, in: Critical Information Infrastructures Security, Springer, 2008, pp. 197–209.
[17] R.A. Rodríguez-Gómez, G. Maciá-Fernández, J.u.E. Díaz-Verdejo, Defense techniques for low-rate dos attacks against application servers, Comput. Netw. 54 (15) (2010) 2711–2727.
[18] N. Stakhanova, H. Gonzalez, M.A. Gosselin-Lavigne, A.A. Ghorbani, The impact of application-layer denial-of-service attacks, Case Stud. Secure Comput. (2014) 261.
[19] Z.-L. Pan, H. Guo, Y.-Y. Wang, S.-W. Liu, Research on detecting windows vulnerabilities based on security patch comparison, Instrumentation & Measurement, Computer, Communication and Control (IMCCC), 2016 Sixth International Conference on (2016) 366–369.
[20] M. Jiang-B, W. Zhou, Gao-T. Pan, H. Zhou, C. Wu, M. Huang, Evolving defense mechanism for future network security, Commun. Mag., IEEE 53 (4) (2015) 45–51.
[21] S.M. Helalat, An investigation of the impact of the slow http dos and ddos attacks on the cloud environment, 2017.
[22] H. Tanaka, J. Park, K. Iwai, T. Kurokawa, Analysis of slow read dos attack and countermeasures, Proc. Int. Conf. on Cyber-Crime Investigation and Cyber Security (2014) 37–49.
[23] S. Jamali, G. Shaker, Pso-sfdd: defense against syn flooding dos attacks by employing pso algorithm, Comput. Math. Appl. 63 (1) (2012) 214–221.
[24] M. Ahmadi, K. Karimi, A. Ahmadi, B. Bahrambeigy, Acceleration of iptables linux packet filtering using gpgpu, Symp. Comput. Sci. Softw.Eng.(CSSE) 2013 (2013).
[25] S. Köszegi, G. Kersten, On-line/off-line: joint negotiation teaching in montreal and vienna, Group Decis. Negotiation 12 (4) (2003) 337–345.
[26] A. Kuzmanovic, E.W. Knightly, Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants, Proc. 2003 Conf. Appl.Technol., Arch., Protocol.Comput.Commun. (2003) 75–86.
[27] I.M. De Diego, L.C. Giralte, C. Conde, E. Cabello, Detecting denial of service by modelling web-server behaviour, Comput. Electr. Eng. 39 (7) (2013) 2252–2262.
[28] Z. Liu, L. Guan, Attack simulation and signature extraction of low-rate dos, Intell. Inf. Technol. Secur.Inf. (IITSI), 2010 Third Int. Symp. on (2010) 544–548.
[29] G. Papaleo, M. Aiello, E. Cambiaso, Slowreq: a weapon for cyberwarfare operations. characteristics, limits, performance, remediations, Int. Joint Conf. SOCO'13-CISIS'13-ICEUTE'13 (2014) 537–546.
[30] M. Mongelli, M. Aiello, E. Cambiaso, G. Papaleo, An on-line intrusion detection approach to identify low-rate dos attacks, Secur. Technol. (ICCST), 2014 Int. Carnahan Conf. on (2014) 1–6.
[31] S. Scaglione, M. Aiello, E. Cambiaso, G. Papaleo, A similarity based approach for application dos attacks detection, Comput. Commun. (ISCC), 2013 IEEE Symp. on (2013) 000430–000435.
[32] V. Paxson, M. Allman, E. Blanton, TCP congestion control - RFC 5681, Technical Report, IETF, 2009.
[33] A. Bestavros, M. Guirguis, I. Matta, Exploiting the transients of adaptation for roq attacks on internet resources, Netw. Protocols, 2004. ICNP 2004. Proc. 12th IEEE Int. Conf. on (2004) 184–195.
[34] I. Matta, M. Guirguis, A. Bestavros, Y. Zhang, Reduction of quality (roq) attacks on internet end-systems, INFOCOM 2005. 24th Ann. Joint Conf.IEEE Comput. Commun. Soc. Proc. IEEE 2 (2005) 1362–1372.
[35] I. Matta, M. Guirguis, A. Bestavros, Y. Zhang, Adversarial exploits of end-systems adaptation dynamics, J. Parallel Distrib. Comput. 67 (3) (2007) 318–335.
[36] I. Matta, M. Guirguis, A. Bestavros, Y. Zhang, Reduction of quality (roq) attacks on dynamic load balancers: vulnerability assessment and design tradeoffs, INFOCOM 2007. 26th IEEE Int. Conf.Comput. Commun. IEEE (2007) 857–865.
[37] D. Bhattacharyya, M.H. Bhuyan, J. Kalita, An empirical evaluation of information metrics for low-rate and high-rate ddos attack detection, Pattern Recognit. Lett. 51 (2015) 1–7.
[38] R. Kabra, M. Salunke, A. Kumar, Layered architecture for dos attack detection system by combined approach of naive bayes and improved k-means clustering algorithm, Int. Res. J. Eng. Technol. 2 (3) (2015) 372–377.
[39] D. Foo Kune, N. Hopper, Y. Kim, M. Schuchard, A. Mohaisen, E.Y. Vasserman, Losing control of the internet: using the data plane to attack the control plane, Proc. 17th ACM Conf. Comput.Commun.Secur. (2010) 726–728.
[40] J. Mirkovic, P. Reiher, A taxonomy of ddos attack and ddos defense mechanisms, ACM SIGCOMM Comput. Commun. Rev. 34 (2) (2004) 39–53.

[41] N. Muraleedharan, B. Janet, Behaviour analysis of http based slow denial of service attack, Wireless Commun., Signal Proc.Netwo. (WiSPNET), 2017 Int. Conf. on (2017) 1851–1856.
[42] R.C. Baishya, D. Bhattacharyya, N. Hoque, M.H. Bhuyan, J.K. Kalita, Network attacks: taxonomy, tools and systems, J. Netw. Comput. Appl. 40 (2014) 307–324.
[43] C. Eze Elias, O. Elechi Onyekachi, C. Eze Joy, Use of information centric network (icn) as a viable alternative to traditional ip network in forwarding mechanism: a practical approach to preventing dos using bloom filter packet forwarding. iiste, Comput. Eng. Intell. Syst. 4 (5) (2013).
[44] N. Petreley, Security report: windows vs linux, Register 22 (2004) 1–24.
[45] G.N. Purdy, Linux Iptables Pocket Reference, O'Reilly Media, Inc., 2004.
[46] T.G. Sekhar, S.B. Bhushan, D. Shamki, A. Al-Arussi, S. Tamrakar, M. Aloney, M. Kandpal, R.P. Kumar, J. Babu, D. Sah, Mitigating application ddos attacks using random port hopping technique (2014).
[47] V.A. Siris, F. Papagalou, Application of anomaly detection algorithms for detecting syn flooding attacks, Comput. Commun. 29 (9) (2006) 1433–1442.
[48] C. Leckie, T. Peng, K. Ramamohanarao, Survey of network-based defense mechanisms countering the dos and ddos problems, ACM Comput. Surv. (CSUR) 39 (1) (2007) 3.
[49] S. Temme, Measuring and enhancing apache performance, 2001.
[50] Y. Zhang, W. Chen, Y. Wei, The feasibility of launching reduction of quality (roq) attacks in 802.11 wireless networks, Parallel Distrib. Syst., 2008. ICPADS'08. 14th IEEE Int. Conf. on (2008) 517–524.
[51] R.K. Chang, X. Luo, E.W. Chan, Performance analysis of tcp/aqm under denial-of-service attacks, Model., Anal. Simulat. Comput.Telecommun. Syst., 2005. 13th IEEE Int. Symp. on (2005) 97–104.
[52] L. Qi, X.-m. LIU, G. CHENG, M. ZHANG, A comparative study on flood dos and low-rate dos attacks, J. China Univ. Post.Telecommun. 19 (2012) 116–121.
[53] J. Chen, X. Zhang, Z. Wu, M. Yue, An adaptive kpca approach for detecting ldos attack, Int. J. Commun. Syst. (2015).
[54] V. Nigam, Y.G. Dantas, I.E. Fonseca, A selective defense for application layer ddos attacks, Intell. Secur. Inf. Conf.(JISIC), 2014 IEEE Joint (2014) 75–82.
[55] Y. Han, L. Wu, Y. He, Q. Cao, T. Liu, Reduction of quality (roq) attacks on structured peer-to-peer networks, Parallel Distrib. Process., 2009. IPDPS 2009. IEEE Int. Symp. on (2009) 1–9.
[56] Q. Hui, Y. Tang, X. Luo, R.K. Chang, Modeling the vulnerability of feedback-control based internet services to low-rate dos attacks, Inf. Forensics Secur., IEEE Trans. on 9 (3) (2014) 339–353.
[57] M.T. Thai, Y. Xuan, I. Shin, T. Znati, Detecting application denial-of-service attacks: a group-testing-based approach, Parallel Distrib. Syst., IEEE Trans. on 21 (8) (2010) 1203–1216.
[58] Z.M. Mao, Y. Zhang, J. Wang, Low-rate tcp-targeted dos attack disrupts internet routing, NDSS (2007).
[59] X. He, P. Nanda, A. Jamdagni, R.P. Liu, A system for denial-of-service attack detection based on multivariate correlation analysis, Parallel Distrib. Syst., IEEE Trans.on 25 (2) (2014) 447–456.
[60] D. Li, Z. Wu, M. Yue, K. Xie, Sedpbased detection of lowrate dos attacks, Int. J. Commun. Syst. (2014).

Enrico Cambiaso graduated in Computer Science at the University of Genoa, Italy, in 2012, with a thesis titled "Analysis of Slow DoS Attacks". He continued the studies on Slow DoS Attacks and achieved in 2016 a Ph.D in computer science at the University of Genoa, in collaboration with the National Research Council of Italy, producing a thesis titled "Development and Detection of Slow DoS Attacks". His scientific interests are related to computer and network security, intrusion detection systems, covert channels and cloud computing.

Giovanni Chiola graduated in Electronics Engineering at the Polytechnic of Turin, Italy in 1983. He was an Assistant Professor and subsequently an Associate Professor of Computer Science from 1985 to 1993 at the University of Turin, Italy. Since 1994, he is a Full Professor of Computer Science at the University of Genoa, Italy. In the past, his main scientific contributions have been in the fields of distributed simulation, performance modelling and evaluation of distributed systems, and stochastic Petri nets. His current research and teaching activities are focused on operating systems, distributed systems, peer-to-peer and computer and network security.

Maurizio Aiello graduated in 1994, worked as a freelance consultant both for universities and research centre and for private industries. From August 2001, he is responsible of CNR network infrastructure. He is a Teacher at the University of Genoa and University College of Dublin; students Coordinator, fellowships and EU projects in the computer security field. His research are on network security and protocols.