Volume 1, No. 2, July-August 2010

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Improving Distributed Denial of Service (DDOS) Attack Detection Performance using Priority Queuing

Y. Rebahi, J. J. Pallares, D. Vingarzan, F. Gouveia,T. Magedanz, Andreea Ancuta Onofrei

Fraunhofer Fokus Institute, Kaiserin Augusta Allee 31, 10589 Berlin, Germany

{yacine.rebahi, jordi.jaen.pallares, dragos.vingarzan, fabricio.gouveia, thomas.magedanz, Andreea.ancuta.onofrei }@fokus.fraunhofer.de

Abstract: Distributed Denial of Service (DDoS) attacks are a major threat for the deployment of SIP-based VoIP networks. Various solutions have been suggested to mitigate the impact of these attacks, each bringing its advantages and disadvantages. Classifying the SIP-based VoIP traffic into different queues according to the suspiciousness of the traffic, can be an efficient solution if it is correctly implemented. Unfortunately, adding queues to the SIP servers affects severely its performance. The current paper suggests a mechanism called "virtual queuing" that implements a priority queue on a SIP server while keeping its performance at its best.

Keywords: DDoS attacks, Prioritization, Queuing, Scheduling Algorithm.

I. INTRODUCTION

In Voice over IP (VoIP) environments, DoS attacks target the VoIP components, for instance, servers, registrars, and clients. If no appropriate detection and prevention mechanisms are in place, these attacks can lead to the instability of the VoIP network and the disruption of the related services. DoS attacks can be carried out under different forms and launched either by one source or different sources. In the second case, they are called Distributed DoS (DDoS) attacks.

In the literature, different countermeasures were suggested, each with its pros and cons, and will be discussed later on in this paper. However, there is a solution that looks promising and in which, traffic is classified into two or more categories, one for legitimate traffic, and another one for suspicious traffic. This classification is based on appropriate filters. The "good" traffic has always priority and is processed before the suspicious one. This concept has been discussed in fighting flooding attacks in IP networks, however, it was not explicitly investigated in the case of VoIP networks.

Prioritization is usually achieved through some physical queues in which, messages are classified and stored according to their importance. Unfortunately, adding such queues to the VoIP components will increase their load as more processing is needed. This will certainly yield to a degradation of the performance of these components and to a potential requests dropping, which is not acceptable in the case of emergency services if we take the latter as an example. In addition, an overload situation will worsen if retransmission messages to non-received in time requests are added to the overall load of the VoIP server.

The focus of this paper is not on how the filters are built as this has already been discussed in the literature. Please refer to the "related work" section for more details. However, we will stress the discussion on how to implement the queuing mechanism in order to keep the performance of the VoIP components at an acceptable level while preventing congestion and avoiding non-useful retransmission.

The rest of the paper is organized as follows: Section 2 presents some background information about the Session Initiation Protocol (SIP), its retransmission mechanisms, and the related DoS attack mitigation mechanisms. Section 3 motivates the proposed solution and section 4 describes in details its implementation. Section 5 presents the experiments and testing scenarios carried out, and finally section 5 concludes the paper.

II. BACKGROUND

A. The Session Initiation Protocol (SIP)

SIP is an application-layer control protocol that allows users to create, modify, and terminate sessions with one or more participants. It can be used to create two-party, multiparty, or multicast sessions that include Internet telephone calls, multimedia distribution, and multimedia conferences. The Session Initiation Protocol (SIP) protocol was published by the IETF in 1996, but the first recognized standard was disclosed later in 1999. SIP was revised over the years and republished in 2002 as RFC 3261, which is the currently recognized standard for SIP.

The SIP standard can run over the reliable transport protocol (TCP) or the unreliable transport protocol (UDP). When UDP is used, the messages INVITE, 200OK and BYE are retransmitted if no appropriate responses are received in a predefined time interval. The SIP protocol considers two types of retransmission, which different from each other in the way the requests are resent and confirmed. The first type is related to the INVITE transaction, and the second one deals with the other non-INVITE transaction types.

SIP (RFC 3261) defines the timer T1 (default value for T1 is 500ms) for the retransmission, in which the SIP client retransmits an INVITE request at a time interval that starts at T1 seconds. This interval is doubled after each retransmis-

sion. The retransmission stops if a provisional response is received or the time from the first transmission is greater than 64* T1. This means, if no response is received after 32 seconds, the SIP client stops the retransmission. As for the retransmission of non-INVITE transactions (200OK, BYE), RFC 3261 defines another timer T2. The retransmission, in this case, works as follows: A second packet is sent T1 seconds after the first transmission, the next is sent 2*T1 seconds after the second, the subsequent packet 4*T1 seconds after, and so on until the time interval matches T2. Then, the subsequent retransmissions are carried out every T2 seconds. If after 32 seconds, no answer is received, the SIP client stops the retransmission.

B. DoS attacks in SIP

DoS attacks are a common security threat in the Internet trying to utilise the target's available resources with the aim of rendering the offered service unavailable [1]. These resources might be bandwidth, CPU or memory. Such threats can also occur in SIP environments; however, some appropriate application-layer attacks will be used.

In previous papers, the vulnerabilities that allow dedicated SIP attacks to occur have been listed [1], [2], [3]. Basically these are missing or wrongly applied sender authentication for packets, software errors in SIP implementations or poorly designed implementations that allow resource depletion to occur.

When talking about a DoS attack, one generally means flooding attacks that overwhelm the victim's resources. In our case, flooding can be achieved with different SIP messages (INVITE, REGISTER, etc.) and the attack can be launched from a single source or from multiple sources. In the latter, the attacker employs a large number of (usually unaware) computers with different IP addresses to generate a higher-bandwidth stream of messages than would be possible from one single machine.

C. Related work

DoS handling strategies have been discussed in the literature in various forms. As there are both multiple and different types of DoS attacks, there is no unique solution that is able to cover all types of attacks. Different approaches have therefore been proposed. Initial approaches for DoS protection have been simple rate-limiting algorithms that allow a limited number of requests per time interval from each sending IP address [4]. These mechanisms are effective for single source DoS attacks but fail for highly distributed DDoS attacks. Other researchers have proposed mechanisms to detect Denial-of-Service attacks using state-machine specifications [5],[6]. These mechanisms are helpful against single-source DoS attacks and can also detect DDoS attacks, but they lack the possibility of mitigating DDoS attacks. Other researchers have developed lightweight statistical algorithms to detect DDoS attacks, e.g. by using the Hellinger Distance calculation [7] or calculating cumulative sums [8]. These algorithms can successfully detect DoS and DDoS attacks on SIP proxies, but do not allow any prevention mechanisms.

To our knowledge, using priority queuing in the context of fighting VoIP DDoS attacks has not been investigated yet. However, in the context of IP networks, some work has already been achieved in [9], [10], and [11]. In these papers, classifiers, queuing algorithms and intrusion filters are combined in order to deal with the DDoS attacks.

The usage of priority queuing in the context of VoIP DoS attack detection is feasible, however, it is worth to mention that queuing also adds an overhead to the SIP servers as a two-phase processing is needed. As a consequence, it is very important to find a trade-off between implementing priority queues and keeping the servers performance at the desirable level. This is, in fact, the focus of this paper.

III. THE VIRTUAL QUEUING MODEL

Summing up, the main objectives of the virtual queuing mechanism are:

- Prioritize legitimate calls upon suspicious calls;
- Consider virtual queues, where not the entire SIP messages are stored in these queues according to their priorities, but only some information related to the SIP messages and their eventual retransmission is considered. Such information involves: the headers fields "From", "To", estimated duration of the request, and the time when the SIP server is available to process the SIP message under consideration;
- Control the load in the system avoiding unnecessary retransmissions.

When a SIP request hits the SIP server, the latter will first parse the message's SIP header and matches the corresponding information against some appropriate filters to determine whether the request is legitimate or suspicious. Once this is done, the next step is to schedule the incoming call in the queue to determine when this call will be served. The algorithm to place the calls in the queue is depicted in the figure 1 below:

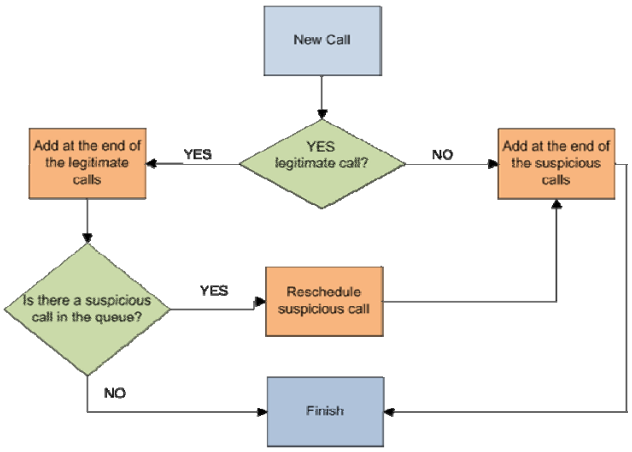

Figure. 1. Algorithm to place incoming calls in the virtual queue

Now, we still have to deal with avoiding unnecessary retransmissions to reduce the SIP traffic. This corresponds in the above algorithm to the "if necessary notify caller" box.

According to the normal flow in a SIP call, the proxy receiving the INVITE request will respond with a provisional 1xx response, usually a 100 (Trying), to prevent the caller to resend the SIP INVITE over the time. Upon reception of the provisional response, the caller will set up a timer that will trigger an error when it expires, as mentioned in section A. After this time, the caller takes the session as terminated. On the other hand, if there is no provisional response, the caller will resend the call issuing unnecessary traffic.

According to this, the value of the Timer C represents the maximum re-scheduling time inside the virtual queue. In order not to have the caller dismiss the current session there are two main alternatives:

- Use of the Retry-After header together with responses such as 480 (Temporarily Unavailable), 486 (Busy here), 503 (Server Busy) or 600 (Busy everywhere) providing the estimated number of seconds where it is foreseen that the call will be able to be processed by the queue;
- Issuing a 182 (queued) response. This response may include the estimated number of seconds that will be spent in the queue. After this time, the call will be processed. The server may issue more than one 182 responses to update the caller about the status in the queue.

The differences between both approaches is that 182 queuing is stateful (the call information will remain in memory) and will issue less traffic from caller to proxy, since the caller will be updated with the current status of the call periodically, whereas the Retry-After queuing is stateless (the call is dismissed) and will issue more signaling between caller and proxy: at least one more INVITE and a provisional response.

If after a caller dismisses one call, tries to re-send it later in a different session, then the call will have lost the grace priority of having already been waiting in the queue once: the call will have a new SIP Call-ID and will be treated as new. For this reason an internal call identifier needs to be defined to be able to map between past calls that have been queued and new calls from the same user. The proposed queue ID is based on a hash function as shown below:

queueID = md5sum { RESPONSE/REQUEST : SIP method : From : To }

This will ensure that the re-issue of the call after a Retry-After will be treated with priority compared to other calls that have never been in the queue.

Another issue is how to estimate the waiting time inside the queue. This is done by keeping statistics of the real duration of the calls inside the queue. Each call is time stamped when it arrives and when it leaves and the duration is used to estimate the duration of the next call. This involves having a long term estimation that it is used as a reference each time the system is started and a short term estimation (last 100 calls or last minute: depending on the server load this might be configurable) which is used to weight the current conditions of the system provided that in a congestion or overload situation, call processing will take more time than in a normal situation and the system needs to adapt the estimation to the current conditions.

A. Implementation

Currently, the virtual queue has been implemented as a standalone test module and will be later on integrated with a SIP proxy such as the SIP Express Router (SER) [12].

Our virtual queuing mechanism implements the following functionalities,

a) *Queue Update:*Each incoming call triggers the queue update procedure. It first checks whether there are calls in the queue whose scheduled time is less than the current time. These are what in figure 2 is shown as “Abandoned or processed calls”, left from the “now” reference point. All processed calls - these are the calls where the out time value (see Table 1) has been set - are used to update the estimation of the next call delay time duration statistic and deleted from the queue.

On the other hand, abandoned calls with scheduled time less than T_{min} are deleted from the queue. These are mostly calls that received a Retry-after response and did not call back in time or that haven't been processed due to other reasons such as internal call routing problems or misconfiguration. LC simply refers to Legitimate Calls, and SC to Suspicious Calls.

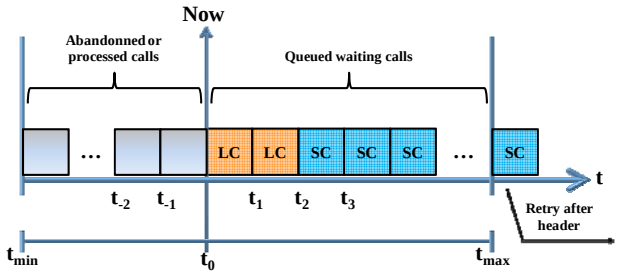

Figure. 2. Virtual Queue.

The update procedure guarantees that the queue will not contain outdated information and will not grow indefinitely. On the other hand, the time information for the processed finished calls is used to update the statistics to estimate the next call delay.

b) Placement: After this first step, the call is processed and inserted in the queue. The queue itself has been implemented as a double linked list and the information about each call that is kept in the queue is summarized in the following table:

Table I. Call information stored in the virtual queue.

Field	Rationale
Queue identifier	Unique identifier for a call. See details in Section 3.
Type of call	Stores the type of SIP method.
Priority	Indicates the type of priority.
Rescheduled	Indicates whether the call has been already rescheduled or not.
Arrival time	Stores the time when the call arrived to the queue.
Scheduled time	Stores the estimated scheduling time for the call.
Out time	Stores the time when the call did leave the queue.

Let us look the proposed placement algorithm at figure 1 in more detail: a new legitimate call that arrives to the queue will get a queue identifier that will be used to reassess that the call is not in the queue and hence it is not a retransmission. Then according to the type of call and SIP method, the incoming call will be assigned a time duration estimation t_{legit} based in the current time statistics. As shown in figure 3, suspicious calls (SC) will always be added in the rear of the queue and legitimate calls (LC) will always be placed at the end of the “legitimate calls subqueue”. This algorithm can be easily formulated in a more general way and extended to more than 2 types of priority. In the placement algorithm, there is the special case where we want to add a legitimate call and there are some suspicious calls already in the queue each of them with an associated scheduled time t_i.

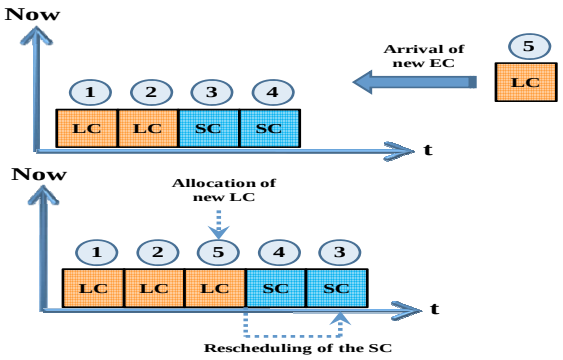

Figure. 3. Virtual queue scheduling algorithm.

Some of these suspicious calls will need to be rescheduled since their place in the queue will now be occupied by the new legitimate call. This rescheduling is also shown in figure 3. In this case, and based on the time estimation for the new incoming legitimate call, suspicious calls are moved to the end of the queue as necessary to leave enough place in the queue for the new prioritized call.

c) Queue Update: After a call is placed in the queue, the scheduled time t_i is calculated as an addition of the scheduled time of the previous call in the queue and the duration of the processing time t_{call}. The processing time initial value is set depending on the SIP method and the call nature. For instance, for an INVITE message, the processing time initial value is set to v_{init} = 15 ms if the call is legitimate and to v_{init} = 22 ms if the call is not legitimate. The processing time is not a constant value: once a call has been successfully processed, the processing time value estimated is adjusted accordingly using some statistics as discussed in section 0 . As we are limited in space, the statistics algorithm will be provided in an upcoming paper.

If the scheduled time is greater than T_{max}, in order to avoid retransmissions the call information is left in the queue and the SIP client is informed with a 503 or a 182 response as depicted in Figure 4. As mentioned in section III, in this case the queuing algorithm issues a 503 "Retry after" or a 182 "Queued" response message with the estimated delay value in seconds.

d) Call Delay Estimation: To calculate the statistics for the call delay estimation, we use the time information described in the table 1. When the call enters the queue, both fields arrival time and scheduled time are filled according to the current time and the current estimation of the call duration from the statistics for each type of SIP method and legitimate or suspicious call. After the call has been successfully processed, the out time field of the call information is filled in.

The call statistics are updated with each queue update: the call time is then *arrival time* minus *out time* and the prediction error is for each call the difference between *output time of the previous call* and the *scheduled time* of the current call.

There are also different types of statistics to be gathered: first, a long term prediction using historical values such as weekdays, weekend, day or night that may be used at initialization time. On the other hand, statistics of the calls in the last 3 minutes or even the last 5 seconds will be more helpful to predict call delay in server overload situations.

IV. PERFORMANCE

Again the main focus of this paper is to develop an efficient queuing mechanism that can allow the classification of the VoIP traffic into legitimate and suspicious, and give priority to the legitimate traffic in terms of processing while keeping the performance of the SIP server at an acceptable level especially in case of congestion. The implementation of the filters is out of the scope of this paper, and to distinguish between legitimate calls and malicious calls in our simulation, we wrote a test program that generates random calls at a certain rate in order to have them processed by the virtual queue.

To test the performance of the queuing algorithm, the latter was stressed with various amounts of requests with different legitimacy probabilities and a fixed processing time for each type of call between 10 ms and 75 ms. According to our experiment, the processing time does not really vary if there is no severe congestion, that is why we assigned to it a fixed value. In fact, we started with an amount of 1000 requests, and then increased it gradually until we reached the amount of 10000 requests. Going beyond 10000 requests is possible but does not really change anything in the behavior of the queuing algorithm. For this reason, we have just considered results related to the cases of 1000 and 10000 requests.

The results are show in the figures 4, 5, 6 and 7 below. The horizontal axis represents the time in seconds and the vertical axis the number of calls generated:

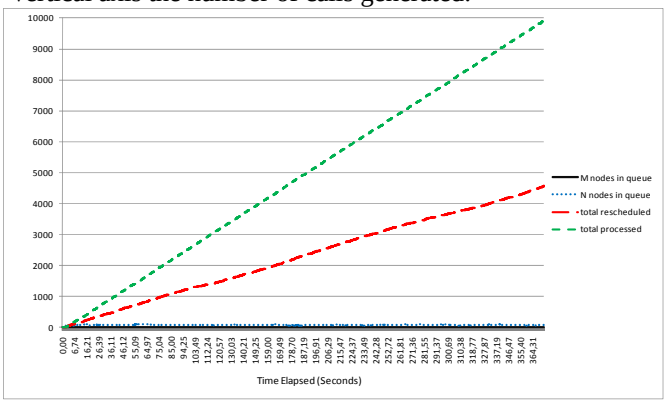

Figure. 4: Results for 10000 calls at 25% legitimate rate.

The call per second rate used in the testing is around 25 call/s, which makes 1500 calls per minute. This amount is likely to be well within the capacity of most SIP servers. Using a value of 25% legitimate rate (so 75% is malicious traffic) for the calls is enough to simulate a severe DDoS attack situation.

As we see in all the graphics, the number of legitimate calls stays always close to zero whereas the number of suspicious calls in the queue oscillates around 100 calls. As the time of testing advances, the number of rescheduled calls increases as a product of the queuing algorithm.

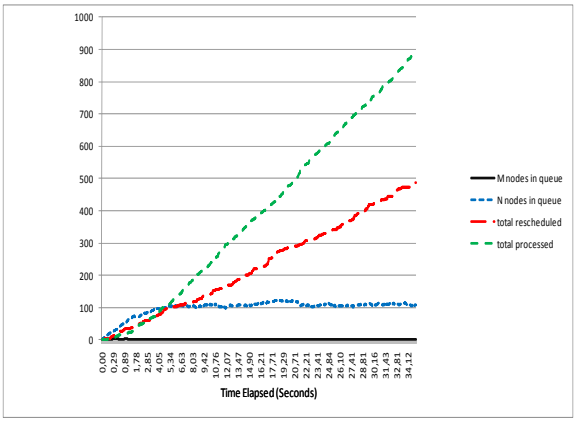

Figure. 5: Results for 1000 calls at 25% legitimate rate.

Figure 6 depicts the testing results for the scenario where 1000 call requests with only 5% legitimate probability are used:

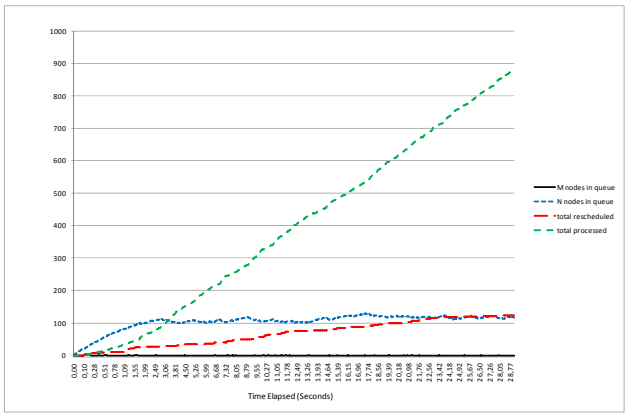

Figure.6: Results for 1000 calls at 5% legitimate rate.

Comparing Figure 6 with the previous ones, we see that provided we have less legitimate calls, the rescheduled nodes are not growing as fast as before, and the queue is getting filled with suspicious calls. The number of legitimate calls in the queue stays close to zero, and the total number of entries in the queue stabilizes around 100 calls as in the 25% legitimate calls case. The queue throughput (processed calls) is similar in both cases.

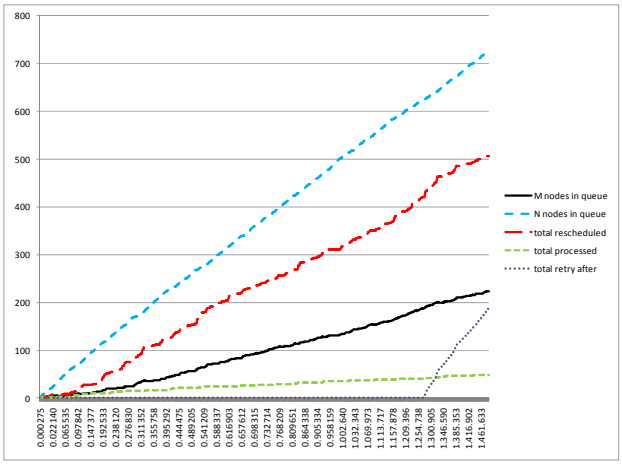

Figure. 7: Results for 1000 calls at 25% emergency rate.

In the realization shown in Figure 7, 1000 calls are sent at a very high rate in order to emulate a flood of SIP messages: the legitimate rate is 25% and the average call rate during the test is 672 calls per second which represents a severe overload for the SIP server. As we see in the graphic, the queue starts filling very fast. After the second 1.27, the scheduled time of the incoming or rescheduled calls exceeds T_{max}, so the queue starts answering the requests with 503 Retry-after responses. At this time, we have in the queue a total of 815 calls (189 legitimate and 626 malicious calls). The number of calls in the queue continues increasing, but this is only the reference to the calls that are kept in the queue (see Table 1). The rest of the server resources are still available for the calls that will be processed in time. 503 Retry-after responses close the transaction at the server side and at the same time avoid retransmissions from the client side. The reference to the call that it is kept in the queue makes it possible for the retransmitted calls to get a slot for processing after the transmission time specified in the 503 response.

In this case, only legitimate calls are served, since the call arriving rate is too high. If we take an average processing time for one call equal to 35 ms, during the time one call is being processed, already around 24 new calls will be entering the queue. From these 24 calls, we will have 6 legitimate calls and 18 suspicious calls (at a 25% legitimate rate). This implies that under these circumstances, the SIP server will only be processing legitimate calls, since they are prioritized and inserted at the beginning of the queue whereas the suspicious calls in the queue will just keep being added to the end and/or being rescheduled.

V. CONCLUSION

In this paper, we have described the concept of virtual queuing that gives priority to legitimate calls over suspicious ones, minimizes the time for legitimate calls establishment, and regulates overload situations while it avoids the dropping of legitimate calls.

VI. REFERENCES

[1] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher. Internet Denial of Service: Attack and Defense Mechanisms. Prentice Hall, 2005.

[2] S. Vuong and Y. Bai. A Survey of VoIP Intrusions and Intrusion Detection Systems. In 6th International Conference on Advanced Communication Technology (ICACT 2004), Phoenix Park, South Korea, February 2004.

[3] D. Sisalem, J. Kuthan, and S. Ehlert. Denial of Service Attacks Targeting a SIP VoIP Infrastructure - Attack Scenarios and Prevention Mechanisms. IEEE Network – Special

[4] B. Iancu. SER PIKE Excessive Traffic Monitoring Module, 2003.

[5] E. Y. Chen. Detecting DoS Attacks on SIP Systems. In 1st IEEE Workshop on VoIP Man-agement and Security, Vancouver, Canada, April 2006.

[6] S. Ehlert, G. Zhang, D. Geneiatakis,G. Kambourakis, T. Dagiuklas, J. Markl, and D. Sisalem. Two Layer Denial of Service Prevention on SIP VoIP Infrastructures. Computer Communications, 31(10):2443-2456, June 2008.

[7] H. Sengar, H. Wang, D. Wijesekera, and S. Jajodia. Detecting VoIP Floods using the Hellinger Distance. IEEE Transactions on Parallel and Distributed Systems, 19(6):794-805, June 2008.

[8] Y. Rebahi. Change-Point Detection for Voice over IP Denial of Service Attacks. In 15. ITG/GI - Fachtagung Kommunikation in Verteilten Systemen (KiVS 2007), Bern, Switzer-land,February 2007.

[9] Q. Huang, et Al, "Analysis of a New Form of Distributed Denial of Service Attack" , In the Proc of the 2003 Conference on Information Science and Systems, the JohnsHopkins University, March 12-14, 2003

[10] W. Lai, et Al, "Using Adaptive Bandwidth Allocation Approach to Defend DDoS Attacks", In the International Journal of Software Engineering and its Applications, vol 2, No 4, October 2008

[11] Lack of Priority, white paper, link: http://delivery.acm.org/10.1145/1040000/1036502/p64-gill.pdf?key1=1036502&key2=2899399621&coll=GUIDE&dl=GUIDE&CFID=84064419&CFTOKEN=89282354

[12] SIP Express Router (SER), link: http://www.iptel.org/ser/