A SIP delayed based mechanism for detecting VOIP flooding attacks

Khaled Dassouki[1], Haidar Safa[2], Abbas Hijazi[1], and Wassim El-Hajj[2]
[1]Lebanese University, Lebanon
[2]American University of Beirut
{kdassouki@ubilitynet.com, hs33@aub.edu.lb, abhijaz@ul.edu.lb, we07@aub.edu.lb}

***Abstract*— SIP is amongst the most popular Voice over IP signaling protocols. Its deployment in live scenarios showed its vulnerability to flooding attacks. In this paper, we present a SIP flooding attack detection mechanism that dynamically detects SIP flooding attacks and correlates in real time the temporal characteristics of SIP reliable mechanism and the number of received INVITE requests. Experimental results show that the proposed mechanism is able to detect SIP flooding rapidly and does not suffer from false alarms. When compared to other similar approaches in literature, the proposed approach outperformed the other approaches in terms of detections speed and accuracy.**

Keywords—SIP; DDOS; INVITE flooding; Ack delays.

I. INTRODUCTION

Voice over IP (VoIP) is being widely adopted to provide affordable, effective and dynamic voice solutions. The Session Initiation Protocol (SIP) [1] is the most popular signaling protocol used to deliver a variety of VoIP services like voice conferences, call transfers, and instant messaging. It is also used by IP Multimedia Subsystems (IMS) [2] to establish and manage sessions. Since SIP is used over the Internet, it inherits most of the underlying IP vulnerabilities; especially those related to Distributed Denial of Service (DDoS) where an attacker can send a high rate of SIP requests to a server in order to disconnect it. Many studies showed that hundreds of calls per second are able to bring down a SIP server [16].

Several flooding attacks detection and protection mechanisms were proposed in literature [3-14, 21-24]. These mechanisms can be classified into two categories: anomaly-based [4-8] and specification-base [10-14]. An anomaly-based detection system models the normal behavior of traffic using statistical interpretations during a training phase. Different statistical methods were implemented in SIP anomaly detection, including but not limited to: Hellinger distance [5][8], CUSUM [6], bloom filters [7] and stateful rule tree [4]. Any deviation from the normal behavior that exceeds a certain threshold is considered an attack. Because traffic is dynamic and unpredictable in nature, it is difficult to find data sets simulating the Internet's normal behavior. This renders these mechanisms prune to high false negative and false positive alarms. Although, some of these anomaly detection systems auto adapt to the dynamicity of the Internet [5-6], they fail in detecting stealthy DOS attacks [9].

The specification-based intrusion detection system (IDSs) takes the correct behavior of the system, and then manually abstracts the security specifications of the system's critical objects. In actual deployment, communications that change the behavior of the critical objects are flagged as intrusions. These approaches can detect unseen attacks with low false alarm rate [17] and are well suited to protect SIP systems as it is easy to model its protocol using state machines.

Unlike our work in [21], which focused on detecting congestions by monitoring passively an aggregation link and deducing parameters from public internet traces using statistical approaches and TCP delays, in this paper we propose a dynamic INVITE flood detection mechanism that correlates in real time the temporal characteristics of SIP reliable mechanism and the number of received INVITE requests. The detection threshold is proportional to the attack speed in the sense that the more intense is the attack, the quicker is the detection. The paper is organized as follows. In Section 2, we present SIP background and related work. In Section 3, we introduce our proposed approach. In Section 4, we describe the implementation environment and present some numerical results. We finally conclude in section 5.

II. BACKGROUND AND RELATED WORK

A. *SIP protocol overview*

SIP is a text based application layer protocol that is used to establish, control and terminate multimedia and voice sessions. Fig. 1 illustrates an example of a successful VOIP session using SIP protocol. In this example, UA calls UB by sending it an INVITE message. Once UB had answered the call, it replies with an OK response to UA. Upon receiving the OK response, UA sends an Ack to UB to confirm the session establishment. The party that wishes to terminate the call must issue a BYE request.

SIP is a reliable protocol. It is designed to operate using either UDP or TCP as underlying transport protocol. To ensure reliability, SIP uses timers and counters. For example, after replying by an OK response, the callee expects an Ack from the caller within T1 seconds (T1 is usually set to 500ms [1]). If no Ack is received within T1, the callee retransmits the OK response and waits for 2xT1. For each retransmission, T1 is doubled until it reaches T2 seconds (T2 is usually set to 64xT1). If T2 is reached without receiving an Ack response the session is terminated.

B. *SIP Flooding attacks*

As mentioned above, SIP has to keep track of connection states in order to ensure reliability. A malicious user can thus trigger an INIVITE flooding attack that exhausts the memory of a SIP server, which is storing the state information of all attempted calls. The attack is realized by sending a fast stream of INVITE messages with different session identifiers such as To, From or Call-Id [20]. In the case of a nonexistent recipient (invalid URI in the generated SIP message), the proxy should forward the message and keep the state information for at least 30 secs [1]. In other cases, the server might fork a request to different destinations, and maintain a copy of the incoming request as well as a copy of all the forked requests. Other SIP messages may also be used in launching flooding attacks [15]. Examples of such attacks are out of state Bye, Ack and OK flooding. In this case, the attacker sends a large amount of messages that do not belong to any session rendering the server busy replying to these messages.

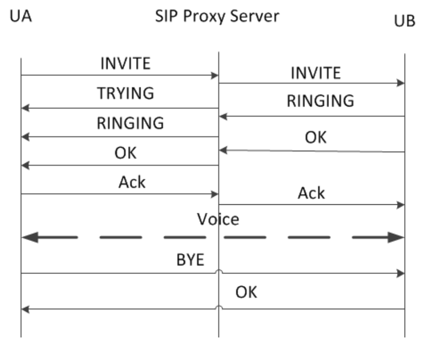

Figure 1. Example of a successful SIP session

C. *State of the art of SIP flood detection*

The first SIP-VOIP specification-based intrusion detection was proposed in [10]. The system is made of a state machine that models SIP, Real-time Transport Protocol (RTP) and the interaction between them. Also, the system encloses a state machine that models the INVITE flooding attack. Indeed, for every destination IP address, the state machine counts the number of received INVITE requests. When the number of these requests reaches a threshold N, the IDS fires an alarm. N is defined as the maximum number of requests that a proxy can handle within a certain period of time. Compared to SIP specifications, this state machine is limited and immature because it only models the establishment phase of a VOIP session and does not address the error states that a SIP session can experience, which makes the system unable of detecting unknown attacks.

A more detailed and advance specification-based IDS was proposed in [11]. In this approach, SIP session is made of one Dialog between two end points. A Dialog encloses many transactions; each of which is made of a request and the responses triggered by this request. An example of a SIP transaction is the list of messages exchanged during the establishment phase: INVITE, Trying, Ringing, OK and Ack. There are two types of transactions: a client transaction, created when a request is sent, and a server transaction, created when a request is received. Any message received by one of the state machines that does not correspond to their design is considered abnormal and make the state machine transit to the Error state. Moreover, every state has a counter assigned to it. Every time the counter exceeds a predefined threshold, the state machine classifies that as an abnormal event and transits to the Error state. Attack detection is based on four thresholds; the upper bound of the number of allowed transaction errors per second, the upper bound on the number of allowed SIP application errors per second, the upper bound on the number of allowed transactions per node and the upper bound on the number of allowed packet rate (in terms of packets per second) per transaction. One major limitation of [11] is that it did not specify values for the above parameters and did not implement the IDS.

[12] implemented and enhanced the specification of the detection system proposed in [11] by adding two new models; malformed SIP detection module and session management module. The malformed message attack module checks if the SIP message is conforming to the rules described in SIP specifications. If a message passes the malformed SIP messages module, it is passed to the Session management module, which creates a new session after receiving or sending INVITE request, and destroys the session after receiving or sending BYE request. The session management detects anomalies by comparing current sequence number and Call-ID of each session with previous ones. After passing both modules' tests, the message is validated by the state machine module. The INVITE flood detection method checks both thresholds: the upper bound of the number of allowed transactions per node and the upper bound of the number of allowed packet rate (in terms of pps) per transaction.

Similar to [11], the state machines presented in [13] model the transaction layer of a SIP session with one major difference between the two: [11] models only the error states while [13] models events that are not allowed within any state. Moreover, [13] relies only on proxy state machines and does not models the client state machine because it consider that the server state machine is enough to protect servers from flooding attacks. [13] also differs from the previously presented specification mechanisms in the way it monitors SIP servers. All the previous specification detection mechanisms evaluated an operation based on its conformity with respect to a specification (i.e. state machine). [13] gets more insight from the state machine by gathering and correlating statistical measurements such as active transactions, active Replies, new transactions, transaction termination and intrastate event. The retransmission messages within each state are also counted. Thresholds are specified for every parameter. To detect INVITE flooding attacks, the system monitors whether the number of replies generated by the server exceeds a certain threshold. If it does, an INVITE flooding is detected.

In [14] one state machine is used to model both the client and the server. Arrival packets are used to determine corresponding events in the machine. Almost all states except close are connected to the Abnormal state. There are two transitions into Close state; one is from connection close implying that the connection has been closed normally; the other is from Abnormal state implying that the connection has been closed abnormally. Each state of the machine has a counter that counts the number of transitions to it. The packet stream is considered abnormal if this counter exceeds a certain value m. Moreover, a timer is provided with each state as a time limit to wait for corresponding traffic that is supposed to arrive. When the timer expires, the previous message is abnormal. Every state has its own timer value. The threshold used in this work to detect INVITE flooding attacks is the number of INVITE received by a certain node during a specified period of time.

III. Proposed Mechganism

The above works have two major limitations:

- Detection time is an important metric in detecting DDOS attacks. The sooner the attack is detected the easier it is to deploy the countermeasures and the less is the effect of the attack on the protected system. All the aforementioned detection mechanisms [10-14] use fix and static thresholds that are not dynamically tuned with respect to the attack behavior. These types of thresholds are efficient in the case of sudden increase of SIP requests and responses. However, DDOS attacks that are increased gradually are not detected directly by these systems. The system waits for the attack to reach a certain threshold to detect it.
- Ability to differentiate between flash crowds and DDOS. Flash crowds are sudden increase in received legitimate traffic by a SIP server. The increase in traffic is caused by simultaneous normal calls. It is difficult to differentiate between flash crowds and DDOS attacks. [10] and [14] count the number of received INVITE by a server during a specified period of time. Similarly [11] and [12] count the number of transactions which may be initiated by a sudden increase in the number of new INVITE requests. [13] counts the number of replies generated by a certain node. Whether it is the number of INVITE or the number of replies, both metrics are symptoms of flash crowd and INVITE flood. This makes the proposed mechanisms unable of differentiating between flash crowd and INVITE flood.

In order to overcome the aforementioned limitations, we propose a detection mechanism that protects SIP servers form flooding attacks and more precisely INVITE flooding. This is because, INVITE flooding are the most harmful and wide spread SIP flooding attacks [16]. Following is a detailed description of our mechanism.

A. *Notations, definitions and parameters*

Before describing our flood attack detection algorithm, we first define the following notations and parameters:

1) *OK delays:* after answering a call, the callee's end point sends an OK response to the caller endpoint and waits for an Ack. If the Ack is not received in T1 ms, the OK is retransmitted. The delays in receiving an Ack response are an important indication to the presence of an INVITE flood attack. As shown in Fig. 2, we propose to classify Ack delays into three different phases

a) Normal delay: It is a delay during which most of the Ack responses are received by the callee.

b) Probably abnormal delay: When the delay is not normal anymore and before getting to the abnormal phase, the Ack delay is considered probably abnormal. During this phase we are not sure whether the SIP session will be resumed or it enters the abnormal phase.

c) Abnormal delay: It is a delay after which most of the delayed Ack are not received. When reaching this delay, the callee resends many times the OK response without receiving any response from the caller.

2) *ThPrAb:* It is the probably abnormal delay threshold after which the session is considered as probably abnormal.

3) *ThAb:* It is the abnormal delay threshold after which the session is considered as abnormal.

4) *Δ:* It is the difference in seconds between *ThAb* and *ThPrAb*.

5) *t:* It is the time accumulated by the session delay after reaching the probably abnormal threshold *ThPrAb*.

6) *x:* It is the number of sessions reaching the probably abnormal phase during a period *P*.

7) *Max:* It is the maximum acceptable number of sessions reaching the abnormal phase during a specified period of time. If the amount of abnormal sessions exceeds *Max*, INVITE flood is detected.

8) *y:* It is the number of sessions that reach the abnormal phase at a specified time. INVITE flood is detected when *y* becomes greater than *Max*.

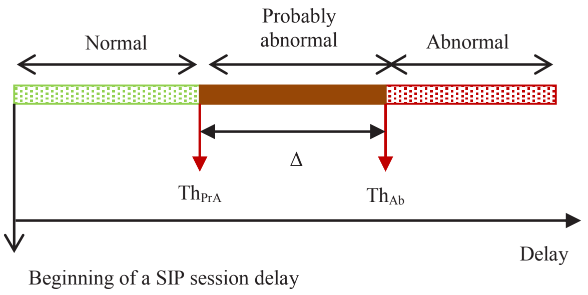

Figure 2. SIP session delays

According to our delay classification, a session is considered abnormal when its delay reaches *ThAb*, the abnormal threshold. Thus, to minimize the detection time, we propose that a session should transit from the probably abnormal to the abnormal phase, when *t* reaches *E* given as:

$$E = \begin{cases} \Delta - \Delta/\text{Max} & \text{when } x = 1 \\ \Delta - \Delta{*}x/\text{Max} & \text{when } 1 < x < \text{Max} \\ 0 & \text{when } x = \text{Max} \end{cases} \qquad (1)$$

This equation is made of two constants *Δ* and *Max* and one variable *x*. When *x* (the number of sessions that had reached the probably abnormal phase) increases, *Δ-Δ*x/Max* decreases which consequently reduces the time needed for a session to reach the abnormal phase and accordingly the time needed to reach *Max*.

B. *Basic concepts*

Every new SIP session is handled by a state machine that is similar to the one suggested in [13]. Based on the fact that our detection mechanism is mainly designed to detect INVITE flood attack, our detection mechanism monitors at the callee state machine the number of received INVITEs and the Ack delays.

Every monitored session may lead to an increase or decrease in the value of x and y. Fig. 3 illustrates the proposed detection algorithm. The algorithm has two levels of analysis; a session level analysis and an aggregation level analysis. A session level analyzer is created for every active SIP session as shown in Fig. 3(a). There is only one aggregation level analyzer which is influenced by the different session analyzers as shown in Fig. 3(b). When the Ack delay experienced by an active SIP session becomes greater than *ThPrAb*, the session level analyzer created for this session transits from the normal state to the probably abnormal state. At this stage, the aggregation level analyzer is informed by the session level analyzer to increase the value of x by 1. If the SIP session receives a packet, the session analyzer goes back to the normal state and the value of x is decreased by 1. If the session is still in the probably abnormal state and the delay persists until the value of t becomes greater than $\Delta - \Delta * x/Max$, the session is classified as Abnormal and the counter y of the aggregation level analyzer is increased by 1. When y becomes greater than *Max*, INVITE flood is detected.

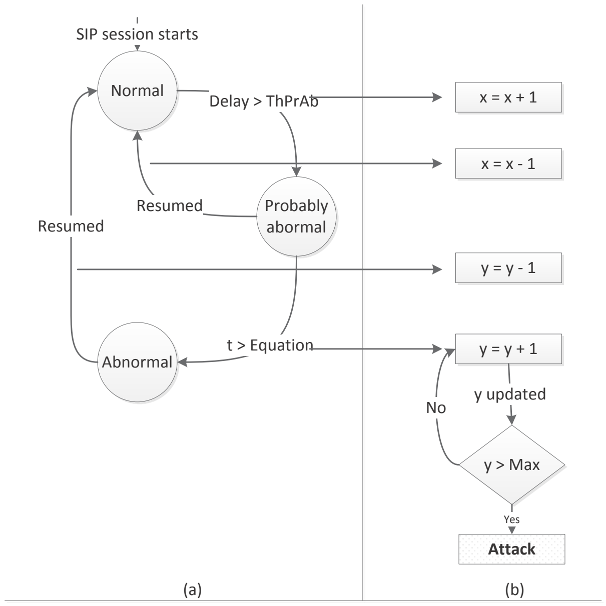

Figure 3. Flow of the proposed algorithm

IV. Implementation and Result Analysis

In this section, we describe our implementation environment, experiments and the obtained results. We also specifiy how *ThPrAb*, *ThAb* and *Max* are specified.

A. Implementation test bed

As previously pointed by most of the SIP related work, no public SIP traces are available on the internet. To evaluate our proposed mechanism, we implemented a test bed that simulates a real world SIP scenario. The architecture of this test bed is shown in Fig. 4. We connected two sites through an internet connection provided by the same ISP. The first site to the left simulates the SIP clients. It is made of a Linux Ubuntu machine that runs five SIPP [18] agents. The second site to the right simulates a SIP server that responds to the SIP machines. The server is also made of a SIPP server that is run over a Linux Ubuntu machine. Every Linux operating system is installed on a machine made of 4 GB RAM and two 2.76 GHz processors. Both sites are connected to the ISP through a dedicated 1 Gb/sec links. Moreover, the end to end connection between both sites is not congested.

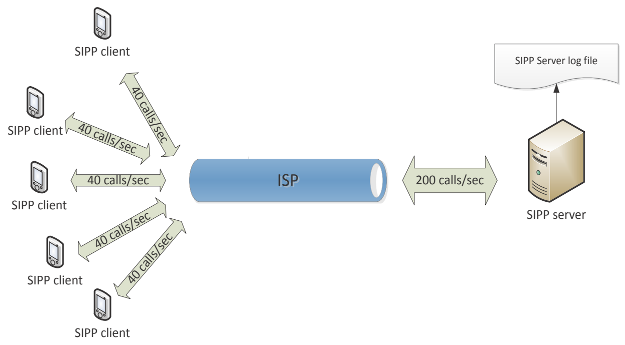

Figure 4. Implemented test bed

B. ThPab, ThAb, and Max specification

In order to validate our detection mechanism, we should specify three thresholds; *ThPab*, *ThAb* and *Max*. In the case of an INVITE flooding attack, the malicious user keeps on sending INVITE messages without acknowledging the OK responses sent by the victim. The server resends the unacknowledged OK until a certain time threshold. Passing this threshold, the server considers the session as abnormal.

In order to specify the aforementioned three thresholds, we setup every SIPP client to initiate 40 calls/sec with the SIPP server, for a duration of 10 minutes. Every call is made up of the SIP messages exchanged in Fig. 1. As a result, the server handles 200 calls/secs coming from the five SIPP clients for a duration of 10 minutes. The call rate specification was chosen, after testing the capabilities of the server and clients, where every client was able to handle 40 calls/secs and the server was able to handle 200 calls/secs without being overloaded. We repeated the 10 minutes run, five times during different periods of the day. No delay was programed. Every SIPP entity responds directly upon message reception. For every run, the SIPP server logged the arrival time of all the SIP messages.

For every run, we deduced for every call, the time difference between the server that had sent an OK and the reception of the ACK response. Then we applied equation (2) to calculate the percentage of delayed sessions for duration less than T with respect to the total number of sessions exchanged during one run. Equation (2) is given as:

$$\frac{\sum_0^d DS_{Td}}{\sum_0^n S_n} \times 100 \qquad (2)$$

Where DS_{Td} is the d_{th} session that is experiencing a delay less than T, d is the total number of DS_{Td} sessions during one run of 10 minutes, S_n is the n^{th} session of a certain run, and n is the total number of sessions exchanged during one run. We computed the percentages for T equals 10, 500, 1000 … 6000 msecs. Fig 5 shows the results for every run while Fig. 6 shows the average of the five runs.

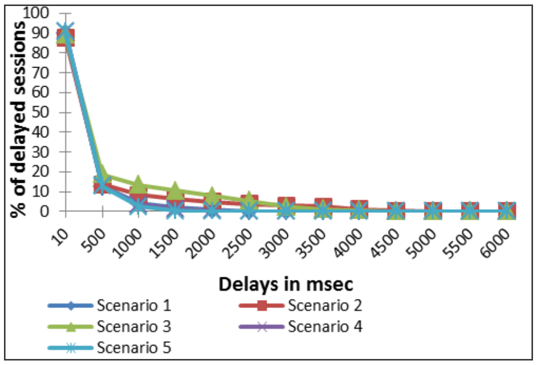

Figure 5. Results for every run

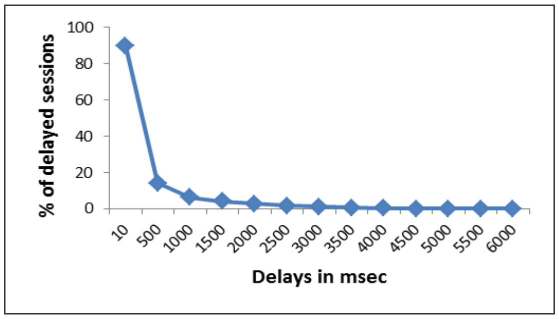

Figure 6. Average of the five runs

As we can deduce from the results that around 95% of the sessions are resumed before a delay of 1 second. Based on this finding, we specify *ThPab* to be 1 second. Due to the small difference between the delay percentages, the graph above is not able to show the small delay percentages on the very right of Figures 5 and 6. Table I and Fig. 7 show the detailed values of the deduced average delays for a period greater than 3000 msecs. They show that after 5.5 seconds, the delayed sessions are not resumed. Based on this result we specify that *ThAb* is 5.5 seconds. Only 0.029% of the sessions did not respond to the OK messages. Based on this, *Max* was set to 0.029%. However, by testing our detection algorithm using the deduced *Max* value, we found that our algorithm is vulnerable to a high rate of false positive. After performing many tests, and following the same approach proposed in [13], we found that choosing *Max* to be 1% can minimize the false positive rates.

TABLE I. DETAILED VALUES

	Delay in ms						
	3000	***3500***	***4000***	***4500***	***5000***	***5500***	***6000***
% of delayed sessions	1.104	0.732	0.347	0.111	0.033	0.029	0.029

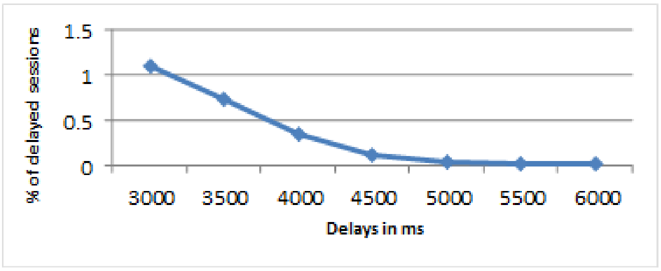

Figure 7. Delays greater than 3 seconds

C. *Attack detection*

In this section, we compare our detection mechanism with previous work [12, 13]. The approach proposed in [12] counts the number of received INVITE messages in order to detect flooding INVITE attacks. The approach proposed in [13] considers that retransmissions ratio is an indication of an INVITE flooding attack. To compare the proposed approach with the approaches presented in [12] and [13], we implemented two scenarios: a sudden increase in normal traffic and a slow rate INVITE flood attack.

We have used Bro IDS [19] to implement our detection mechanism on the same machine on which the SIPP server is deployed in Fig. 4. Algorithm 1 presents the pseudo code of our policy script, which is described in high level functions. The script monitors and analyzes the active sessions for every received packet. During this period, the script computes the number of sessions that had experienced a delay greater than *ThPrAb* seconds (i.e. *x*). Then the script compares the time elapsed since the session had reached the *ThPrAb* seconds delay with the Δ - Δ *x/Max*. If *t* is greater than Δ - Δ *x/Max*, *y* (i.e., the number of sessions that had reached *ThAb*) is incremented. Finally the script compares *y* with *Max*. If *y* is greater than *Max*, flooding attack is detected.

In order to check the sensitivity of our detection mechanism with respect to false negative detection, we implemented a scenario of unexpected increase of normal traffic using the same test bed presented in Fig. 4. The five SIPP agents in the scenario were generating an equal number of calls/sec. All the calls generated in this scenario are similar to the one presented in Fig. 1. After 35 seconds of receiving 100 calls/sec, the server receives a sudden increase of traffic that reaches 300 calls/sec for duration of 70 seconds. Our proposed mechanism and [13] did not signal an attack. This is because both mechanisms monitors SIP anomalies (i.e. abnormal delays and retransmissions). However, [12] signaled an attack at the beginning of the sudden increase of calls and persists on signaling an attack until the end of the sudden increase of calls (i.e. 105 seconds). This is because following the implementation guidelines presented in [12], we set the attack threshold to 200 calls/seconds. [12] was not able to differentiate between normal and abnormal sessions.

```
For every 100 ms do
    Compute the number of SIP sessions that were delayed
    for more that ThPrAb and save it in x:
        For every session where delay is > ThPrAb do
            E = Delta – Delta*x/Max;
            t = time elapsed since reaching ThPrAb;
                If t > E, then
                    y = y + 1;
                Endif
                If y > Max, then
                    Alarm attack;
                Endif
        Endfor
Endfor
```

Algorithm I

In order to check the performance of the detection time in our approach, we implemented a slow rate flooding attack using the same test bed showed in Fig. 4. Four SIPP agents were generating normal traffic whereas the fifth agent was set to generate abnormal attack traffic. A normal call follows the same messaging sequence presented in Fig. 1, whereas the abnormal traffic sent by the fifth phone is made only of INVITE requests. The normal flows received by the server are made of 100 calls/sec and the attack traffic started at the 15th second. The attack started by an attack rate of 1 call/sec that increases by 1 call every 1 second. Our detection mechanism was able to detect the attack in 2.6 seconds, whereas the detection mechanism proposed in [13] took 9.2 seconds to detect the same attack. The method in [13] took around four times the time needed by our proposed mechanism to detect flooding attacks. In real case scenarios, this difference in detection time may have an important impact on the performance of the system. The approach in [12] took 110 seconds to detect the attack which is far from any satisfactory result. A major advantage of our approach is the fact that we start our mitigation process much earlier than other approaches.

V. Conclusion

In this paper, we proposed a novel specification-based detection mechanism that protects SIP from flooding attacks. The proposed mechanism detects dynamically the INVITE flood and correlates in real time the temporal characteristics of SIP reliable mechanism and the number of received INVITE requests. We showed that the proposed mechanism is capable of reducing the detection time and not sensitive to false alarms. We aim in future work to cover all kinds of SIP flooding attacks.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Spark, M. Handley, and E. Schooler, Session initiation protocol. RFC 3261, June 2002.

[2] Z. Chen, W. Wen, D. Yu "Detecting SIP flooding attacks on IP Multimedia Subsystem (IMS)," International Conference on Computing, Networking and Communications (ICNC 2012), February 2012.

[3] AD. Keromytis "A survey of Voice over IP security research."*Information Systems Security. Springer Berlin Heidelberg*, Vol. 5905, 2009, pp 1-17.

[4] D. Seo, H. Lee, and E. Nuwere. "SIPAD: SIP–VoIP Anomaly Detection using a Stateful Rule Tree." *Computer Communications*, Vol. 36, no. 5, March 2013, pp. 562–574.

[5] J. Tang and Y. Cheng "SIP Flooding Attack Detection." *Intrusion Detection for IP-Based Multimedia Communications over Wireless Networks. Springer New York*, Sep. 2013. pp. 53-70.

[6] Y. Rebahi, M. Sher, and T. Magedanz "Detecting flooding attack against IP multimedia subsystem (IMS) networks". In proceeding of the International Conference on Computer Systems and Applications (AICCSA 2008), April 2008.

[7] D. Geneiatakis, N. Vrakas, and C. Lambrinoudakis "Utilizing bloom filters for detecting flooding attacks against SIP based services."*computers & security* Vol. 28, no. 7, 2009, pp. 578-591.

[8] H. Sengar, H. Wang, D. Wijesekera, and S. Jajodia "Detecting VoIP floods using the Hellinger distance." *IEEE Transactions on Parallel and Distributed Systems* Vol. 19, no. 6, 2008, pp. 794-805.

[9] J. Tang and Y. Cheng "Quick detection of stealthy SIP flooding attacks in VoIP network". in proceeding of the IEEE Internation Conference on Communications (ICC 2011), June 2011.

[10] H. Sengar, D. Wijesekera, H. Wang and S. Jajodia "VoIP intrusion detection through interacting protocol state machines." In proc. of the Int. Conf. on Dependable Systems and Networks (DSN 2006), June 2006.

[11] EY Chen "Detecting DoS attacks on SIP systems." In proceeding of the 1st IEEE Workshop on VoIP Management and Security, April 2006.

[12] D. Seo, H. Lee, and E. Nuwere "Detecting more SIP attacks on VoIP services by combining rule matching and state transition models." In proceedings of The Ifip Tc 11 23rd International Information Security Conference. Springer US, September 2008.

[13] S. Ehlert, C. Wang, T. Magedanz, and D. Sisalem "Specification-based denial-of-service detection for sip voice-over-ip networks." In proc. of the IEEE 3rd Int. conf. Internet Monitoring and Protection, June 2008.

[14] PH. Thyda and AB. Koki "A Protocol Specification-Based Intrusion Detection System for VoIP and Its Evaluation." *IEICE transactions on communications*, Vol. 91, no. 12, 2008, pp. 3956-3965.

[15] EY. Chen and M. Itoh. "A whitelist approach to protect SIP servers from flooding attacks." in the proceeding of the IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR 2010), June 2010.

[16] B. Roh, JW. Kim, KY. Ryu, JT. Ryu "A whitelist-based countermeasure scheme using a Bloom filter against SIP flooding attacks." *Computers & Security,* Vol. 37, 2013, pp. 46-61.

[17] G. Vigna and R. Kemmerer. NetSTAT: A Network based Intrusion Detection Approach. in Proceedings of the 14th Annual Computer Security Application Conference (ACSAC 1998), December 1998.

[18] http://sipp.sourceforge.net/

[19] http://www.bro.org/

[20] E. Alomari, S. Manickam, BB. Gupta, S. Karuppayah, and R. Alfaris. "Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art." arXiv preprint arXiv:1208.0403 (2012).

[21] K. Dassouki, H. Debar, H. Safa, and A. Hijazi "A TCP delay-based mechanism for detecting congestion in the Internet," in the *3rd int. conf. on Communications and Information Technology (ICCIT), 2013*, pp.141-145, 19-21 June 2012.

[22] H. Mukhtar, K. Salah and Y. Iraqi "Mitigation of DHCP Starvation Attack," International Journal of Computers and Electrical Engineering, Elsevier Science, Vol. 38, No. 5, September 2012, pp. 1115-1128.

[23] K. Salah, J. Alcaraz Calero, S. Zeadally, S. Almulla, and M. Alzaabi "Using Cloud Computing to Implement a Security Overlay Network," IEEE Security and Privacy, Vol. 11, No. 1, January 2013, pp. 44-53.

[24] K. Dassouki, H. Safa, and A. Hijazi End to End Mechanism to Protect Sip from Signaling Attacks. In proc. Of the 6th IEEE int. conf. on New Technologies, Mobility and Security (NTMS 2014), March 2014 .