A Selective Defense for Mitigating Coordinated Call Attacks

Marcilio O. O. Lemos[1], **Yuri Gil Dantas**[2], **Iguatemi E. Fonseca**[1], **Vivek Nigam**[1] **and Gustavo Sampaio**[1]

[1]Federal University of Paraíba, Brazil

[2]Technische Universität Darmstadt, Germany

{marcilio.cc.lemos,gbritosampaio}@gmail.com,

{iguatemi,vivek}@ci.ufpb.br,

dantas@mais.informatik.tu-darmstadt.de

Abstract. *Telephony Denial of Service (TDoS) attack is a form of Denial of Service (DoS) attack that targets telephone services, such as Voice over IP (VoIP), not allowing legitimate users to make calls. This paper proposes a new selective defense for mitigating a type of TDoS called Coordinated Call Attack where attackers simply call to each other exhausting the target VoIP server's resources and denying service to legitimate users. Our defense builds on the defense SeVen for mitigating Application Layer DDoS attacks. We implemented and integrated SeVen in usual VoIP systems using the SIP protocol and carried out a number of experiments: without SeVen, less than 15% of users can access the target VoIP service, whereas with SeVen, around 90% of users can access the same service.*

1. Introduction

Distributed Denial of Service (DDoS) attacks have been a great concern to network administrators since the origins of the Internet targeting all sorts of services. Voice over IP has been widely used for audio and video communications due to its low costs and acceptable quality. As such, it has been target of DDoS attacks such as VoIP amplification attacks [30, 22] and the SIP flooding attack [30]. Telephony Denial of Service (TDoS) attacks are DoS attacks that target telephone services, such as VoIP, not allowing legitimate users to make calls. TDoS attacks have been reported targeting hospital systems [2, 7] and systems for emergency lines (like the American 911 system) [6]. Moreover, according the FBI, 200 TDoS attacks were identified only in 2013 [7].

Available defenses for DDoS targeting VoIP services, such as [27, 17, 29, 31], are constructed to mitigate attacks by analysing traffic flows and whenever there is an unusual increase of traffic, suitable mechanisms are placed, such as blocking IPs[1]. A recent class of DDoS attacks, called Application-Layer DDoS Attacks (ADDoS), are able to bypass such defense mechanisms because they are carried out by generating traffic that is similar to usual client traffic. Examples of ADDoS attacks include the Slowloris [23], POST [20] and Slowread [24] attacks exploiting the HTTP protocol, and the Coordinated Call Attack [11] exploiting the SIP protocol used by VoIP applications.

The Coordinated Call Attack [11] exploits the fact that pairs of attackers, Alice and Bob, can collude to exhaust the resources of the VoIP server. Assume that Alice and

[1]see also http://itsecurity.telelink.com/tdos-attacks/ accessed on March 2016

Bob are valid registered users. This can be easily done for many VoIP services. The attack goes by Alice simply calling Bob and trying to stay in the call as long as she can. Since the server allocates resources for each call, by using enough pairs of attackers, they can exhaust the resources of the server and deny service to legitimate clients. This is a simple, but ingenious attack, as only a small number of attackers is needed generating a small network traffic (when compared to SIP flooding attack for example), being, thus, hard for network administrators to detect and counter-measure such attack.

Our recent work [16] proposed the use of selective strategies for mitigating (HTTP) Application-Layer DDoS attacks in the form of the tool called SeVen. Our defense mechanism is governed by probability functions that specify the chances of a request to be dropped whenever the service is overloaded. We showed in our previous work that by using simple uniform distributions, SeVen can be used against a number of attacks exploiting the HTTP protocols (Slowloris and POST). However, that work only considered ADDoS attacks exploiting the HTTP protocol against web-servers.

This paper investigates the use of selective strategies for mitigating TDoS attacks, in particular, the Coordinated Call Attack. Our contributions are three-fold:

1. **A New Selective Strategy for VoIP Services:** We propose a new selective strategy suitable for VoIP services. While in our previous work [16], we used simple uniform probabilities, this paper uses more sophisticated definitions for deciding which calls that are going to be selected to continue and which should be dropped;
2. **Integration of SeVen with VoIP Servers:** We investigated how SeVen can be used in conjunction with typical VoIP servers, in particular, Asterisk [1]. We implemented our solution using SeVen as a proxy for VoIP server;
3. **Experimental Results:** Finally, we carried out a number of experiments showing that our solution can mitigate Coordinated Call Attacks. When under attack and without using our defense, we observed that only 15% of users could access the VoIP service. All of these clients were able to complete their calls. When using SeVen, we observed that around 90% of clients could access the VoIP service. Of these, 63% were able to complete their calls. The remaining 27% were interrupted by SeVen being able to stay using the VoIP service in average 60% of the intended call duration.

We start in Section 2 explaining the SIP protocol used by VoIP services and the Coordinated Call Attack. Section 3 introduces the new selective strategy used to mitigate the Coordinated Call Attack. In Section 4, we describe the experimental set-up used to validate our defense mechanism as well as the quality measures used. We also show our main findings which validate the efficiency of our defense mechanism. Finally in Section 5, we discuss related work and conclude pointing to future work.

2. VoIP and the Coordinated Call Attack

Nowadays the most widely used signaling protocol in VoIP communication is the Session Initiation Protocol (SIP) [25]. Figure 1 depicts the messages that are exchanged during a call. In the initiation phase, Alice, the caller, sends an INVITE message to the SIP Proxy, such as Asterisk [1] server, with the details of Bob, the party Alice wants to talk with. The proxy searches for Bob's details (such as his IP) and sends an INVITE message to

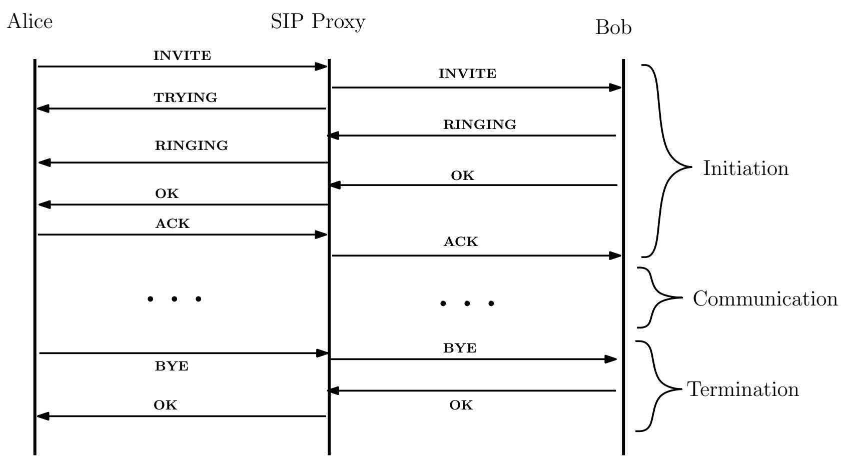


Figure 1. Session Initiation Protocol (SIP)

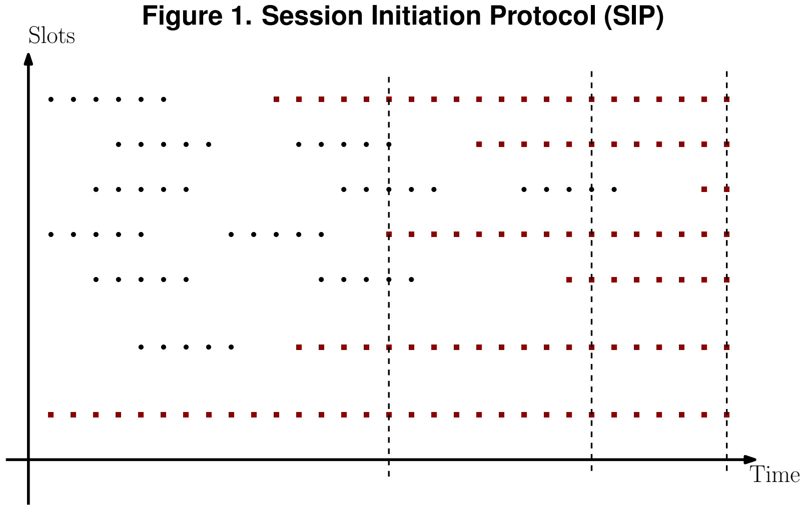


Figure 2. Illustration of the Coordinated VoIP Attack. (Black) Circles represent legitimate calls, while (red) squares represents an attacker call.

Bob with Alice's identification.[2] At the same time, the SIP Proxy sends Alice a TRYING message indicating that it is checking whether Bob is available. If he is available, Alice and Bob exchange through the proxy a number of messages (RINGING, OK, ACK) to establish the connection. At this point, the communication phase starts when Alice and Bob can talk or exchange data. When the call is terminated by one of the parties (in Figure 1, Alice), then a BYE message is sent to the proxy which forwards it to Bob who responds with an OK message acknowledging the end of the call.

2.1. Coordinated Call Attack

A pair of colluding attackers, A_1 and A_2, that are registered in the VoIP service,[3] call each other and stay in the call for as much time as they can. The attackers exchange with the SIP server the same initiation messages as in a normal SIP session as shown in Figure 1. Once the call is established, the attackers stay in the call for indefinite time. They may be disconnected by some Timeout mechanism establishing (very large) time bounds on call duration. During the time that A_1 and A_2 are communicating, they are using resources of

[2]In practice, Asterisk reformulates the data send by Alice according to its own user database adding further information such as its own identification tags.

[3]Or alternatively two honest users that have been infected to be zombies by some attacker.

the server. As usual, VoIP servers have an upper-bound on the number of simultaneous calls they can handle. By using enough colluding attackers, the target VoIP server will reach its limit denying new (legitimate) calls to be established.

Figure 2 illustrates why this attack is so effective. The attacker can slowly occupy the call slots available in the SIP Server. Since they stay indefinite time, once the slot is occupied it is never made free (without a defense mechanism) and therefore eventually the attacker will be using all slots not allowing legitimate clients to use the VoIP service.

An important difference of this attack to other DDoS on VoIP services, such as SIP-flooding, is the fact that the attacker's traffic is not significantly different to usual legitimate traffic. Firstly, the attackers follow correctly the protocol sending the expected messages as specified by the SIP protocol. Secondly, the attack does not have to generate a large number of calls at a particular instance. It can slowly populate the VoIP resources with traffic load, *i.e.*, call rate, similar to client call rate. The overall traffic would correspond to expected traffic load and therefore not activating any defense mechanism.

3. SeVen

Our defense, SeVen, uses a selective strategy [16]. In a nut-shell, *whenever the application is overloaded*, that is, its capacity is full, and a new request R arrives, SeVen acts as follows:

1. SeVen decides (using some probability distribution P_1) whether the application should process R or not;
2. If SeVen decides not to process R, then it simply returns a message to the requesting user that the service is not available;
3. If SeVen decides to process R, then as the application is overloaded, it should decide (using some probability distribution P_2) which request currently being processed should be dropped. This decision is governed by P_2, a distribution probability *which may depend on the state of the existing request.*

Intuitively, this strategy works because when the application is overloaded, it is very likely that it is suffering a DDoS attack. Therefore, whenever a new request arrives and SeVen decides to process it, the probability of dropping an attacker is much higher. Figure 2 illustrates this situation: the vertical dashed lines indicate some moments when the application is overloaded and new requests arrive. The number of square dots, representing attackers, is much greater than the number of circle dots, representing legitimate clients, when the attack is being carried out. While without SeVen the application would simply deny service to all requesting clients, SeVen allows new clients to be served.

The first main contribution of this paper is on the definition of the probability distributions P_1 and P_2, in particular, the probability P_2. While in our previous work [16] P_2 was uniform which was suitable for Web-Servers, we need a more elaborate distribution. P_2 will depend on (1) the status of the request and (2) on the duration of a call.

We consider two types of status for requests:

- WAITING: A request is WAITING if it did not yet completed the initiation phase of the SIP protocol (see Figure 1). That is, it is still waiting for the responder to join the call and start to communicate;
- INCALL: A request is INCALL if the initiation phase of the SIP protocol have been completed and the initiator and the responder are already communicating (or simply

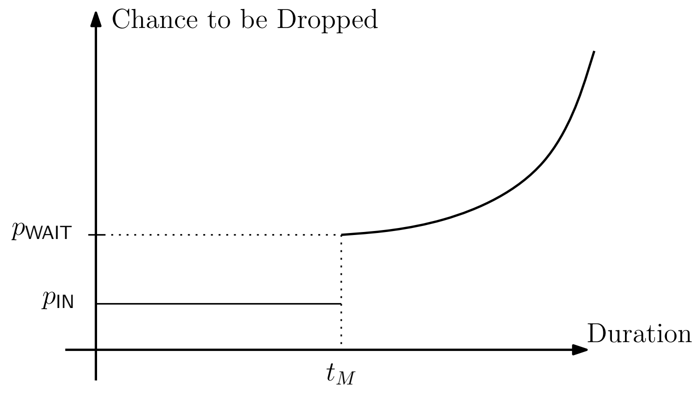


Figure 3. Graph (not in scale) illustrating the behavior of SeVen according to the status of a request and the duration of INCALL request. p_{WAIT} is the factor for dropping a WAITING request, while p_{IN} the factor for dropping a INCALL request.

in a call).

Thus, any incoming INVITE request assumes the status of WAITING, and it can change its status to INCALL once the initiation part of SIP is completed.

We assume here that it is preferable to a VoIP server, when overloaded, to drop WAITING requests than INCALL requests that are communicating not for a *very long duration*. In many cases, it is true that interrupting an existing call is considered to be more damaging to server's reputation than not allowing a user to start a new call. This could also be modeled by configuring the probability distributions of SeVen accordingly. Moreover, to determine whether a call is taking too long, we assume that the server knows what is the average duration, t_M, of calls.[4]

The chance of an INCALL request to be dropped increases exponentially. This function was based on using a Poisson distribution[5] once this has a duration of more than t_M. Figure 3 depicts roughly how the chance of dropping a call increases with the call duration. The actual equation is of the form, where t is the call duration:

$$d(t) = \begin{cases} p_{\mathsf{WAIT}} & \text{if } t = 0 \\ p_{\mathsf{IN}} & \text{if } 0 \leq t \leq t_M \\ p_{\mathsf{WAIT}} + e^{\alpha t / t_M} & \text{if } t > t_M \end{cases} \tag{1}$$

We apply this function to the roulette-wheel selection method [19] in order to produce the probability distribution P_2 that decides which request currently being processed should be dropped. This function is used because we think it is reasonable for many applications. Of course, there are many decision options for these probabilities which will depend on the intended application. For instance, one could consider that the Poisson distribution should begin only a period after t_M, or that it should be another distribution, etc. It will depend on the specific requirements of the defense.

Finally, we point out that we also formally specified this strategy in the computational tool Maude [13] and used advanced model checking techniques, namely Statistical Model-Checking [21], to have a first impression of whether this defense would work or

[4]The value of t_M can be obtained by the history of a VoIP provider's usage.

[5]We used a Poisson distribution because such distributions are normally used for modeling telephone calls arrival [12]

not. As our model checking results (which are detailed in another work [15]) were satisfactory, we implemented the defense and carried out experiments, which will be detailed in Section 4.

3.1. SeVen by Example

The formal specification of SeVen is detailed in our previous work [16]. We illustrate how it works with an example referring the details of the mechanism to [16].

Assume for this example that the VoIP server can only handle 3 simultaneous calls. Moreover that the average time of calls is $t_M = 5$ (just for illustration) and that currently, the server is processing two call requests identified as id_1 and id_2 as follows:

$$\mathcal{P}_0 = [\langle id_1, \mathsf{INCALL}, 8\rangle, \langle id_2, \mathsf{WAITING}, 0\rangle]$$

where $\langle id, st, dur\rangle$ specifies that the call id has status st and dur is the duration of the call, being 0 whenever $st = \mathsf{WAITING}$. Thus, id_1 is calling for 8 time units, while id_2 is still waiting for the responder.

Assume that a new request arrives, identified as id_3. Since the server's capacity is not reached, it accepts this request updating the status of the server to the following reaching its maximum capacity:

$$\mathcal{P}_1 = [\langle id_1, \mathsf{INCALL}, 8\rangle, \langle id_2, \mathsf{WAITING}, 0\rangle, \langle id_3, \mathsf{WAITING}, 0\rangle]$$

Assume that one time unit passes and that the id_2 completes the initiation phase. The status of the server is updated to:

$$\mathcal{P}_2 = [\langle id_1, \mathsf{INCALL}, 9\rangle, \langle id_2, \mathsf{INCALL}, 1\rangle, \langle id_3, \mathsf{WAITING}, 0\rangle]$$

Now consider that a new request id_4 arrives. Since the server is in its maximum capacity, SeVen should decide whether it processes id_4 or not. SeVen throws a coin (according to the probability distribution P_1 which is not important for this paper. You can safely assume a fair coin). If it returns 0, then it does not process id_4; otherwise it does process id_4.

Assume that SeVen decides to process id_4. SeVen now has to decide which one of the requests currently being processed, id_1, id_2 or id_3, should be dropped. It chooses one using the probability distribution generated from function depicted in Figure 3. Thus, id_1 has a greater chance of being chosen than id_2 and id_3 because its call has a duration greater than t_M $(9 > 5)$. Moreover, id_3 has a greater chance of being chosen than id_2 as $p_{\mathsf{IN}} < p_{\mathsf{WAITING}}$.

Assume that SeVen decides to drop id_1. The new status of the server will be:

$$\mathcal{P}_3 = [\langle id_2, \mathsf{INCALL}, 1\rangle, \langle id_3, \mathsf{WAITING}, 0\rangle, \langle id_4, \mathsf{WAITING}, 0\rangle]$$

Finally, say that id_2 finishes its call. The resulting server status will be:

$$\mathcal{P}_4 = [\langle id_3, \mathsf{WAITING}, 0\rangle, \langle id_4, \mathsf{WAITING}, 0\rangle]$$

where the server is no longer overloaded.

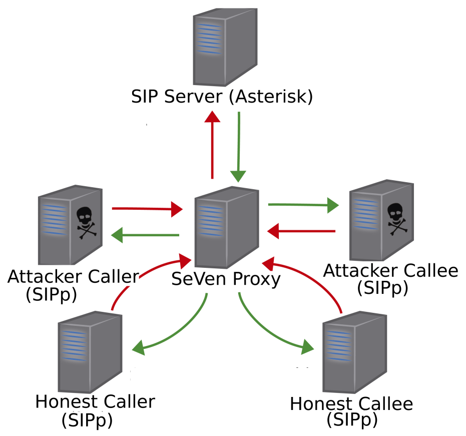


Figure 4. Experiment Topology

In practice, the values for $p_{\mathsf{IN}}, p_{\mathsf{WAIT}}$, the capacity of the server and the t_M will depend in the application. Moreover, the duration of call would be measured in seconds (or even milliseconds).

Remark: Other parameters besides call duration could be used in configuring the chances to drop clients. Dantas's Master thesis [14] details several other options, such as different classes of users (*e.g.*, Gold, Silver and Bronze), the type of connections, etc. Their use will depend on the particular situation. We leave the task to incorporate other parameters to future work.

4. Experimental Results

4.1. Set-Up

In our experiments, we used Asterisk version 13.6.0 which is a SIP server widely used by small and mid size companies for implementing their VoIP services. We assume there are honest users and malicious attackers which try to make the VoIP unavailable. Both the traffic of the honest users and the attackers are emulated using the tool SIPp [5] version 3.4.1. SIPp generates calls which may be configured as the caller or the callee. Thus, in our experiments, we used pairs of SIPp, one pair for generating the honest user calls and the other pair for generating the attacker calls. Finally, we developed the SeVen proxy in C++ which implements the selective strategy described in Section 3.1.

Figure 4 illustrates the topology of the experiments we carried out. To make a call, the pairs of SIPp send messages to the SeVen proxy which on the other hand forwards them to Asterisk. Similarly, any message generated by Asterisk is forward to the SeVen proxy which then forwards them to the corresponding users. Therefore, SeVen is acting as an *Outbound Proxy* for both Asterisk and the pairs of SIPp. For our experiments, it is enough to use a single machine. We used a machine with configuration Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz and 8 GB de RAM.

Finally, we set the duration of the calls generated by SIPp as follows:

- **Total Call Duration of Clients:** Whenever we generate a new call, we generate randomly with a uniform probability a natural number between $[1, 2 \times t_M]$. Thus legitimate user calls may at most take twice the average call duration. SIPp ends the call when its corresponding call duration is reached;

- **Call Duration of Attackers:** Following the Coordinated Call Attack, we do not limit the call duration of an attacker call. He call communicate for indefinite time.

Parameters We use the following parameters to configure our experiments:

- **Average Call Duration** (t_M) – We assume known what is the average duration of calls. This can be determined in practice by analyzing the history of calls. We assume in our experiments that $t_M = 5$ minutes;
- **Dropping Factor** – We assume the following values for the dropping chance function (Eq 1):
  - $p_{\mathsf{IN}} = 2$;
  - $p_{\mathsf{WAIT}} = 8$;
  - $\alpha = 1.89$.

  These values were chosen so that the probability of dropping increases in a reasonable fashion after the call duration is greater than t_M. Sample values are as follows, recalling that $t_M = 5$:

Call Duration (mins)	Dropping Factor
6	12.37
8	17.31
10	27.84

  That is, the chance of dropping a call with duration of 10 minutes is approximately 3 times greater than dropping a call whose status is WAITING (27.84/8). This is a reasonable ratio. However, according to the specific application other values can be set for $p_{\mathsf{IN}}, p_{\mathsf{WAIT}}$ and α. Finally, the choice of setting $p_{\mathsf{WAIT}} = 4 \times p_{\mathsf{IN}}$ was selected so that the calls with duration less than t_M have much less chance of being dropped than the calls that are still waiting for the responder.
- **SIP Sever Capacity** (k) – This is the number of simultaneous calls the SIP server can handle. We set $k = 50$ which is a realistic capacity for a small company allowing 100 users (2×50) to use the service at the same time.
- **Experiment Total Time** (T) – Each one of our experiments had a duration of 60 mins, that is, 12 times the average call of clients. With this duration, it was already possible to witness the damage caused by the Coordinated Call Attack as well as the efficiency of our solution for mitigating this attack.
- **Traffic Rate** (R) – Using a server with capacity of 50, we calculated using standard techniques [10] what would be a typical traffic of such a server. It is $R = 9.9$ calls per minute. In our experiments, we split this rate among clients and attackers. Recall that the Coordinated Call Attack does not generate traffic different from the expected traffic so that it can bypass usual defenses based on network traffic analysis. Thus, the total traffic in our experiments is always less or equal than R.

Quality Measures For our experiments, we used the following three quality measures for our calls:

- **Complete Call:** A call is complete whenever its status changed from WAITING to INCALL and *it is able to stay in status* INCALL *for its corresponding call duration*.

That is, the caller was able to communicate with the responder for all the prescribed duration;

- **Incomplete Call:** A call is incomplete whenever its status changed from WAITING to INCALL, but *it was not able to stay in status* INCALL *for its corresponding call duration*. That is, the caller was interrupted before completing the call;
- **Unsuccessful Call:** A call is unsuccessful if it did not even change its status from WAITING to INCALL. That is, the caller did not even have the chance to speak with the responder.

Intuitively, complete calls are better than incomplete calls which are better than unsuccessful calls. In order to support this claim, we also computed the average duration call of the incompleted calls, that is, the time that users in average where able to stay communicating before they were interrupted by SeVen.

4.2. Experimental Results

We carried out a number of experiments analyzing the attack and the effectiveness of our defense. Three types of experiments were carried out:

- **Type 1:** Attacking the Asterisk server without using SeVen to defend it;
- **Type 2:** Attacking the Asterisk server and using SeVen to defend it;
- **Type 3:** Not attacking the Asterisk sever but still using SeVen.

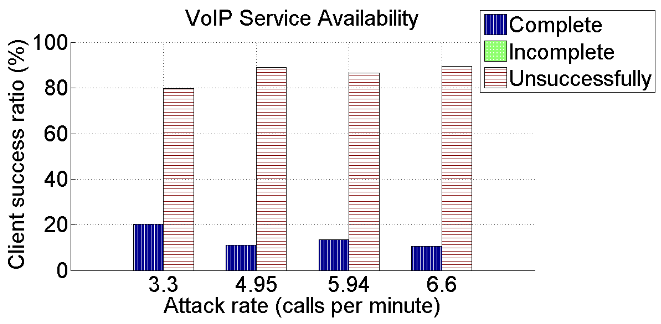
Figure 5 depicts the availability results for legitimate clients for each type of experiment. In our experiments with attack, we varied the attacker call-rate from 3.3 calls per minute until 6.6 calls per minute. The client call rate was adjusted in order to preserve the total rate of 9.9 calls per minute which is a typical call rate for the Asterisk server as detailed in Section 4.1. Thus for the scenario with an attacker call rate of 3.3 calls per minute, the client call rate was 9.9 - 3.3 = 6.6 calls per minute, which corresponds to a ratio of 2 client calls for each attacker call. Similarly, when the attacker call rate used is 6.6 calls per minute, we generated 3.3 calls per minute for legitimate clients, which corresponds to a ratio of 2 attacker calls for each client call.

When under attack and not running SeVen (Figure 5(a)), we observed that the ratio of unsuccessful calls was very high (80-90% of total client calls). This means that these clients were not even able to start a call. The remaining calls (10-20% of total client calls) were all successful, that is, each one could stay using the VoIP service for the total time assigned to them.

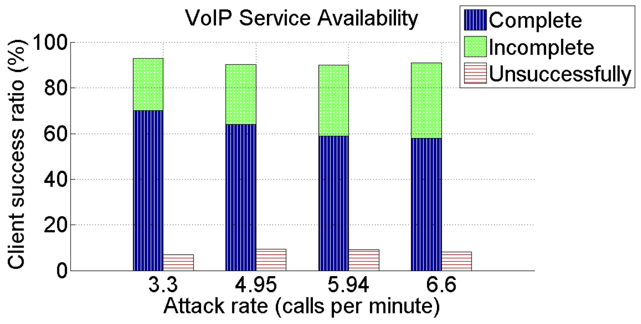
On the other hand, when under attack and using SeVen (Figure 5(b)), we observed that clients had many more successful calls (60-70% of total client calls), while some (0-10% of total client calls) were not able to even start the call. The remaining calls (20-30% of total client calls) were interrupted while calling due to SeVen's DDoS mitigation strategy. Therefore, around 90% of the clients were able to use (even if some in an incomplete fashion) the VoIP service.

Finally, we observed that SeVen did not affect the client success ratio when Asterisk is not under attack as depicted in Figure 5(c), where all client calls were successful.

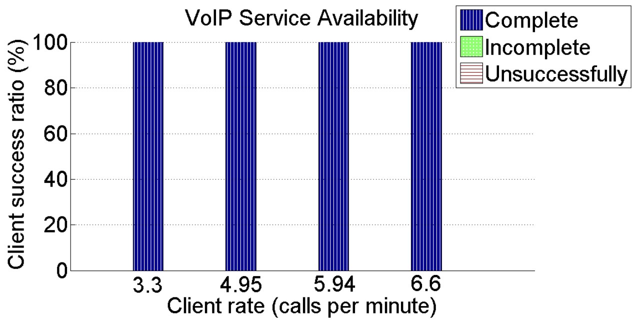
In order to understand better the profile of the incomplete calls in the scenario with attack and using SeVen, we measured the following ratio for each incomplete call: $\frac{t_D}{t_T}$, where t_D is the duration call, that is, the time when the call started until it was interrupted,



(a) Client Success when under attack and *not* running SeVen.



(b) Client Success when under attack and running SeVen.



(c) Client Success when not under attack and running SeVen.

Figure 5. Availability Results

and t_T is the intended total call duration time. Figure 6 depicts the average call duration ratio of the incomplete calls. The average was around 60% for each experiment with different attacker call rates. This means that the clients in incomplete calls still were able to communicate for more than half of the corresponding duration of the call. This supports our claim that incomplete calls are indeed much better than unsuccessful calls in which clients are not even able to start the call.

Finally, in order to understand better the behavior of the Coordinated Call Attack and in particular, how effective our defense is, we studied how many calls were occupied by attackers over time. Figure 7 depicts the results for the case when the attacker call rate is 4.95 calls per minute, that is, the same rate as the client call rate (9.9 - 4.95 =

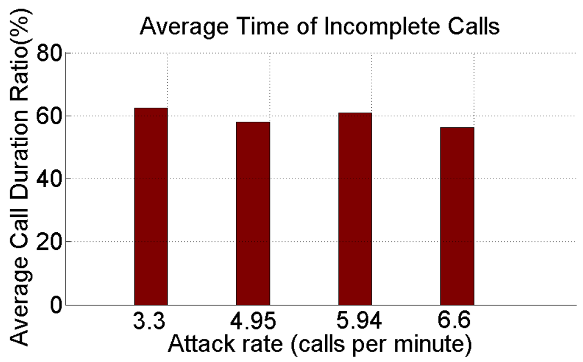
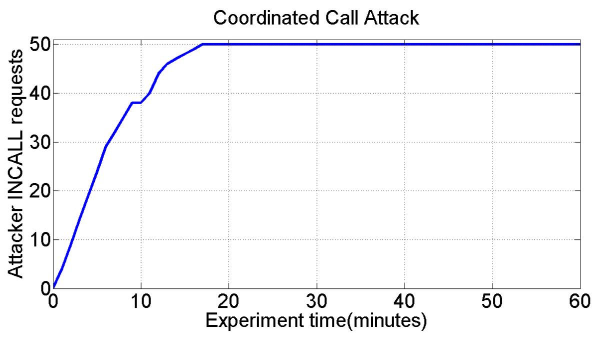
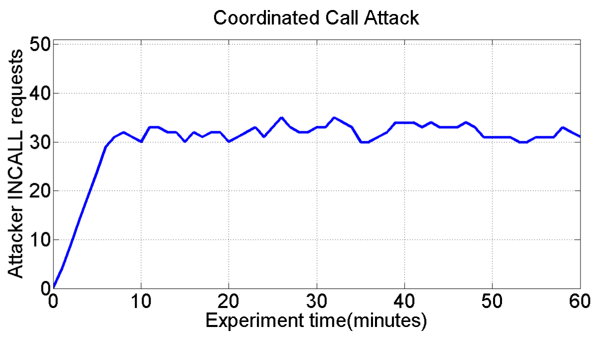


Figure 6. Average Duration of Incomplete Calls



(a) Without using SeVen.



(b) Using SeVen.

Figure 7. Attacker Occupancy over time when Attacker Call rate is 4.95 calls per minute.

4.95 calls per minute). The cases with other attacker call rates had similar results. We observed that when not using SeVen (Figure 7(a)), the attacker is able to occupy all the 50 call positions after 17 minutes. After this point, no legitimate client is able to use the VoIP service. This means that if we considered even longer experiments, the availability of clients (Figure 5(a)) would tend to zero as the clients are only able to access the VoIP service in the first 17 minutes of the experiment. *Indeed, we carried out experiments first initializing the attackers letting them occupy all available slots of the VoIP server and only then initialized the clients. The result of this experiment was 100% of unsuccessful calls.*

On the other hand, with SeVen (Figure 7(b)), we observe that the attackers were not able to occupy all positions of the VoIP server. After 8 minutes, the attackers were able to occupy a bit more than 30 slots, leaving the remaining slots available to legitimate clients. This remained so for the rest of the experiment. It demonstrates that the selective strategy indeed works as intuitively described in Section 3.

5. Conclusions, Related and Future Work

This paper introduced a new selective strategy for mitigating the Coordinated Call Attack on VoIP services. We implemented this defense and carried out a number of experiments and analyzes using an off-the-shelf VoIP server (Asterisk) demonstrating the effectiveness of our defense. Under an attack and using our defense, around 90% of legitimate clients were able to access the VoIP services, 63% of which used the service until full satisfaction. In contrast, without our defense, only 15% of clients had access to the VoIP service.

Most of the existing work [18, 28, 26, 27, 29, 31] on mitigating DoS attacks on VoIP services focuses on Flooding attacks, such as SIP-Flooding attack. They analyze the network traffic and whenever they observe an abrupt increase in the traffic load, they activate their defenses. The network traffic is usually modeled using some statistical approach, such as correlating the number of INVITE requests and the number of requests that completed the SIP initiation phase [18] or using more complicated metrics such as helling distance to monitor traffic probability distributions [28, 26, 27]. Other solutions place a lower priority on INVITE messages, which are only processed when there are no other types of request to be processed [29, 31].

As the Coordinated Call Attack emulates legitimate client traffic not causing an unexpected sudden increase in traffic, all these defenses are not effective in mitigating the Coordinated Call Attack. The few solutions we found in the literature for this type of attack are commercial tools that act as a firewall which monitors all the call traffic and the signaling [4, 9] or analyze audio samples [3] in order to differentiate the fraudulent calls from the legitimate ones. Less sophisticated mechanisms [8] monitors all the incoming requests and rejects those whose IPs do not belong to a list of trusted IPs. Clearly such approaches does not work well when the attackers are malicious users who's IPs are in the trusted list and are not using automation to make the calls. In addition, these commercial tools can be expensive for small businesses to purchase and maintain, and they do require technical expertise for proper installation.

One main advantage of our proposed solution is that it is not tailored using many specific assumptions on type of service. The only assumption used is a previous knowledge of the average call duration, which can be easily inferred from the service call history. Moreover, our solution can be easily integrated with other mechanisms such as the IP filtering approach used in [8].

We also point out that before carrying out the experiments in this paper, we formally modeled the defense and the Coordinated Call Attack in the computational tool Maude [13] and used statistical model-checking methods [21] to verify it using Monte Carlo simulations. This is reported in another work [15].

There are many directions for future work. We will investigate how one can add additional information, such as the request IP origin, call history of users, into account to improve the precision of our defense. We are also investigating other attacks on VoIP service that emulate legitimate clients, such as the prank call attack [6]. We are also incorporating other optimizations in SeVen investigating ways to better integrate it with Asterisk. Finally, we are also investigating how SeVen can be incorporated into existing services such as those used by Fone@RNP.

Acknowledgments: The authors were partially funded by the RNP. Lemos, Nigam and Fonseca were partially funded by CAPES and CNPq. Dantas was supported by the Hessian excellence initiative LOEWE at the Center for Advanced Security Research Darmstadt (CASED).

References

[1] Asterisk private branch exchange. http://www.asterisk.org//. Accessed: 2015-27-09.

[2] Cyber threat bulletin: Boston hospital TDoS attack. http://voipsecurityblog.typepad.com/files/cyber-threat-bulletin-13-06-boston-hospital-telephony-denial-of-service-attack.pdf. Accessed: 2015-27-09.

[3] Pindrop: Protecting your call centers against phone fraud social engineering. https://www.pindrop.com/wp-content/uploads/2016/01/pindrop_overview_whitepaper_fi_20141121_v2.pdf. Accessed: 2015-27-09.

[4] Securelogix: Telephony denial of service (tdos) solutions. http://www.securelogix.com/solutions/telephony-denial-of-service-TDoS.html. Accessed: 2015-27-09.

[5] Sipp: Sip traffic generator. http://sipp.sourceforge.net. Accessed: 2015-27-09.

[6] Situational advisory: Recent telephony denial of services (tdos) attacks. http://voipsecurityblog.typepad.com/files/ky-fusion_tdos_3-29-13-2.pdf/. Accessed: 2015-27-09.

[7] TDoS- extortionists jam phone lines of public services including hospitals. https://nakedsecurity.sophos.com/pt/2014/01/22/tdos-extortionists-jam-phone-lines-of-public-services-including-hospitals/. Accessed: 2015-27-09.

[8] Tdos attack mitigation. http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/voice/cube_proto/configuration/15-mt/cube-proto-15-mt-book/voi-cube-tdos-attack-mitigation.pdf. Accessed: 2015-27-09.

[9] Transnexus nexoss. http://transnexus.com/telephony-denial-service-attacks/. Accessed: 2015-27-09.

[10] *Designing Optimal Voice Networks for Businesses, Government, and Telephone Companies*. 1980.

[11] *The Surging Threat of Telephony Denial of Service Attacks*, (accessed Setember 28, 2015). http://voipsecurityblog.typepad.com/files/tdos_paper_4-11-13.pdf.

[12] Lawrence Brown, Noah Gans, Avishai Mandelbaum, Anat Sakov, Haipeng Shen, Sergey Zeltyn, and Linda Zhao. The title of the work. *Statistical Analysis of a Telephone Call Center: A Queueing Science Perspective*, (100):36–50, 2002.

[13] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All About Maude: A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer, 2007.

[14] Yuri Gil Dantas. Estratégias para tratamento de ataques de negação de serviço na camada de aplicação em redes ip. Master Thesis in Portuguese, 2014.

[15] Yuri Gil Dantas, Marcilio O. O. Lemos, Iguatemi Fonseca, and Vivek Nigam. Formal specification and verification of a selective defense for tdos attacks. In *11th International Workshop on Rewriting Logic and its Applications (WRLA)*, 2016.

[16] Yuri Gil Dantas, Vivek Nigam, and Iguatemi E. Fonseca. A selective defense for application layer ddos attacks. In *IEEE JISIC 2014*, pages 75–82, 2014.

[17] S. Ehlert, Chengjian Wang, T. Magedanz, and D. Sisalem. Specification-based denial-of-service detection for sip voice-over-ip networks. In *ICIMP '08*, 2008.

[18] Do-Yoon Ha, Hwan-Kuk Kim, Kyoung-Hee Ko, Chang-Yong Lee, Jeong-Wook Kim, and Hyun-Cheol Jeong. Design and implementation of sip-aware ddos attack detection system. In *ICIS '09*, pages 1167–1171, New York, NY, USA, 2009. ACM.

[19] Adam Lipowski and Dorota Lipowska. Roulette-wheel selection via stochastic acceptance. *CoRR*, abs/1109.3627, 2011.

[20] r-u-dead yet. `https://code.google.com/p/r-u-dead-yet/`. 2013.

[21] Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 2005.

[22] Ravinder Shankesi, Musab AlTurki, Ralf Sasse, Carl A. Gunter, and José Meseguer. Model-checking DoS amplification for VoIP session initiation. In *ESORICS*, pages 390–405, 2009.

[23] slowloris. `http://ha.ckers.org/slowloris/`. 2013.

[24] slowread. `https://code.google.com/p/slowhttptest/`. 2013.

[25] J. Stanek and L. Kencl. Sipp-dd: Sip ddos flood-attack simulation tool. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–7, July 2011.

[26] Jin Tang, Yu Cheng, and Yong Hao. Detection and prevention of SIP flooding attacks in voice over IP networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1161–1169, March 2012.

[27] Jin Tang, Yu Cheng, Yong Hao, and Wei Song. SIP flooding attack detection with a multi-dimensional sketch design. *Dependable and Secure Computing, IEEE Transactions on*, 11(6):582–595, Nov 2014.

[28] Jin Tang, Yu Cheng, and Chi Zhou. Sketch-based sip flooding detection using hellinger distance. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–6, Nov 2009.

[29] Xiao-Yu Wan, Zhang Li, and Zi-Fu Fan. A SIP dos flooding attack defense mechanism based on priority class queue. In *WCNIS 2010*, pages 428–431, June 2010.

[30] Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Communications Surveys and Tutorials*, 15(4):2046–2069, 2013.

[31] Fan Zi-Fu, Yang Jun-Rong, and Wan Xiao-Yu. A SIP dos flooding attack defense mechanism based on custom weighted fair queue scheduling. In *ICMT 2010*, pages 1–4, 2010.