Hidden in plain sight. SDP-based covert channel for Botnet communication

Zisis Tsiatsikas[1], Marios Anagnostopoulos[1], Georgios Kambourakis[1], Sozon Lambrou[1], and Dimitris Geneiatakis[2]

[1] Dept. of Inform. and Comm. Systems Engineering, University of the Aegean, Karlovassi, Greece
{tzisis,managn,gkamb}@aegean.gr
[2] Electrical and Computer Engineering Department, Aristotle University of Thessaloniki, GR541 24 Thessaloniki, Greece
{dgeneiat}@auth.gr

Abstract. Covert channels pose a significant threat for networking systems. In this paper, we examine the exploitation of Session Description Protocol (SDP) information residing in Session Initiation Protocol (SIP) requests with the aim to hide data in plain sight. While a significant mass of works in the literature cope with covert communication channels, only a very limited number of them rely on SIP to realize its goals. Also, none of them concentrates on SDP data contained in SIP messages to implement and evaluate such a hidden communication channel. Motivated by this fact, the work at hand proposes and demonstrates the feasibility of a simple but very effective in terms of stealthiness and simplicity SIP-based covert channel for botnet Command and Control (C&C). As a side contribution, we assess the soundness and the impact of such a deployment at the victim's side via the use of two different types of flooding attacks.

Keywords: SIP; Botnet; Covert Channel; C&C; SDP.

1 Introduction

During the last decade, VoIP services have exhibited a remarkable expansion. As a matter of fact, recent reports [1] indicate that IP multimedia communication services gain ground against the Public Switched Telephone Network (PSTN) ones. This is because VoIP services provide more flexible and inexpensive models, and thus they gradually dominate the market. Among others, multimedia session establishment and management constitutes a fundamental operation in VoIP networks. Nowadays, Session Initiation Protocol (SIP) has been adopted as the prevalent signaling protocol for handling multimedia sessions over the Internet and 3rd Generation partnership Project (3GPP) realms. On the downside, SIP is inherently susceptible to different kinds of attacks [2, 3]. One of them lies in its exploitation as a covert channel. Adversaries usually employ covert channels aiming to communicate information over legitimate data flows. In fact, the text-based nature of SIP fosters such types of attacks. An adversary could easily craft

specific parts of the message in order to deliver data with special meaning over legitimate SIP requests. The only requirement for such an attack would be to conform to SIP syntax, otherwise the message parser module at the receiver side would possibly drop the request as malformed.

So far, SIP-based covert channels are scarcely addressed in the literature, and to our knowledge, no implementation exists. That is, the majority of the existing works concentrates on the applicability of information hiding techniques in VoIP-related protocols in general. This includes SIP, Real Time Protocol (RTP) and RTP Control Protocol (RTCP). The delivered channels may be used in a variety of ways, aiming to establish secret paths of communication. In this paper, we examine the feasibility of exploiting SIP as a Command and Control (C&C) covert channel aiming to deliver commands to a SIP botnet and launch attacks. The main contributions of this paper are summarized as follows:

- We present a simple but powerful in terms of stealthiness covert communication protocol to exchange botnet C&C messages over SDP data in SIP requests.
- We evaluate the effectiveness of the covert channel by controlling several bots and launching two different Denial of Service (DoS) type of attacks.
- An assessment of the attack impact in terms of resource consumption at the victim side is also included.

The rest of the paper is structured as follows. Section 2 provides an overview of session establishment in SIP-based networks. It also presents background information for botnet networks and an overview of the threat model. Section 3 briefly describes the proposed architecture and details on the protocol used for realising the covert C&C channel. Section 4 evaluates the impact of the attack in terms of CPU, memory and network utilization at the victim's side. Similar work in the literature is addressed in section 5. The last section draws a conclusion and gives pointers to future work.

2 Preliminaries

2.1 SIP Architecture

This section briefly describes the basic SIP architecture, including message structure, the involved entities, and the process of session establishment. The two endpoints, namely the caller and calee also referred to as User Agents (UA), have to send a REGISTER request to a SIP Registrar in order to declare their presence in the service provider or to update their contact information. The latter entity is responsible for gathering and storing registration data into a database to provide location service. Whenever a user wishes to start a session, she sends an INVITE request to the local SIP proxy. The latter retrieves the calee's information from the location server and forwards the message to it. Either the caller or the calee are able to terminate an ongoing session anytime by sending a BYE request.

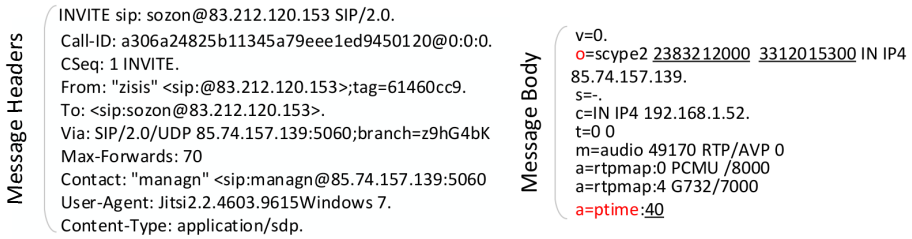

Fig. 1. A typical SIP INVITE message

A SIP message comprizes of several headers and a message body. It is text-based and presents similar structure to that of HTTP. Figure 1 depicts a typical INVITE request. As observed from the figure, the various headers contain information related to the sender and the recipient of the message, and also the communication path. Also, as seen in the figure, such a message is comprized of two parts; the left one containing the various headers, and the message body describing streaming media initialization parameters. The latter part is built following the SDP standard format [4]. Given the text nature of the message an adversary could straightforwardly manipulate the data contained in the SIP headers or SDP descriptors with the aim to build a covert channel over the legitimate information. Note that if this is done in a SIP-oriented (natural) way, the channel has many changes of going undetected. However, as explained further down, care must be taken in order not to alter important information that are required by the peers or the proxies to establish communication. Moreover, any manipulation in the various headers or parameters must be syntactically neutral; otherwise, the message could be dropped by the receiver's message parser. One may also think of encrypting the parts of the message to be used as the covert channel carrier. This however would require the provision of some key management process, and more importantly, will attract the attention of network defenses. So, the idea here is to hide the (C&C in our case) information in plain sight by simply mimicking the values contained in the fields of a normal SIP message.

2.2 Botnet Architectures

A botnet can be considered as a network consisting of infected and compromized computers, called bots, zombies or slaves, which are controlled by an attacker known as the botmaster or bot-herder. A bot agent obeys every command received by its botmaster ordering it to initiate or terminate an attack. Botnets pose a serious threat to the Internet, since they are capable of disrupting the normal operation of services, networks and systems at will of their botmaster.

For instance, botnets could be used for launching Distributed DoS (DDoS) attacks [5], sending spam emails on a massive scale, performing identity theft, distributing malware or even copyrighted material, and so forth.

Perhaps the most vital demand for maintaining control of the entire botnet is the ability for a bot to constantly stay in touch with its C&C infrastructure through a reliable and undetectable covert channel. That is, a bot will not be able to receive new instructions if the C&C cannot be located, and continue to probe the vanished C&C in vain. In this direction, botmasters employ a number of techniques to not only minimize the probability of bots losing contact with their C&C infrastructure, but also to render their botnet more agile to hijacking and stoppage attempts. Depending on how the bots are remotely controlled by their master, i.e., how the C&C channel is structured, one is able to classify them into centralized, decentralized or hybrid architectures.

The centralized infrastructure is based on the client-server model, where all bots are directly connected with one or few C&C servers. These servers undertake to coordinate the bots and instruct them to take action. Although a centralized botnet exhibits optimum coordination and rapid dissemination of commands, it also poses a single point of failure. From the moment the C&C server is detected and deactivated the entire botnet is turned off. Usually, a bot-herder conveys its command through a well-known protocol. This way, she is able to hide the C&C traffic into a legitimate one. As a rule of thumb, the communication channels in this approach are based on HTTP or IRC protocol [6]. In the first case, the communication is disguised inside the normal Web network traffic as the usage of Web is allowed in most networks, including corporate ones. On the other hand, in IRC-based architecture the bots are connected to IRC channels and waiting for commands from the bot-herder. Of course, the messages on the IRC channel are in an obfuscated custom dialect, e.g., encrypted or hashed to avoid disclosure. In our case, a centralized infrastructure is employed, where one or more SIP proxies are responsible for dispatching the commands to bots. Furthermore, we are not based on the aforementioned protocols, but rather we utilize SIP as a covert channel. Although, centralized approach seems easily detectable, the botmaster is capable of evading defence mechanisms by applying fluxing techniques. As explained further down, fluxing allows the aspiring botmaster to frequently change the IP and/or the domain name of the proxy.

Alternatively, a decentralized architecture may be selected to carry out the C&C mechanism. In this approach, there is not a central C&C server, but rather the various bots communicate with each other via Peer-to-Peer (P2P) protocols. In other words, the bots behave as C&C server and client at the same time. Therefore, if any of the bots is tracked down and deactivated, there are no implications to the robustness of the entire network [7]. The hybrid architecture combines the advantages of both the centralized and decentralized ones. That is, in this setting, the bot agents exhibit diverse functionalities. Some of them, temporarily undertake the C&C server role, with the aim to coordinate the botnet and disseminate the instructions, while the others wait for commands before springing to action [8].

2.3 Threat Model

As already pointed out, various vulnerabilities have been presented so far in the literature concerning SIP [2, 3, 9]. The formulation of a threat model in our case has to do with adversaries who try to capitalize on SIP as a covert channel. We consider two different cases depending on who controls the SIP Registrar with which the bots need to be registered.

In the first one, the botmaster controls the Registrar, e.g., she is the owner of this server or she has compromized it in some way. As a result, the botmaster is able of registering users with the SIP proxy. This way she solves the problem of randomly assigning and updating usernames to the bots. Moreover, she is capable of further eliminating the chances of getting detected by applying IP and Domain Fluxing to the SIP proxy without significant modification to the proposed architecture. In the case of IP flux, the botmaster would regularly alter the IP address pertaining to the Fully Qualified Domain Name (FQDN) of SIP Registrar by owning or controlling a group of PCs dedicated to that purpose. On the other, by applying domain fluxing, she continuously modifies and associates multiple FQDNs to the SIP Registrar. For example, every day, the botmaster could assign a new domain name to the SIP Registrar. These names might be generated by a hash function taking as input the current global date and a secret string. With the same way, the various bots could produce the domain name of a specific day.

The second scenario is the opposite of the former, i.e., the botmaster does not control the Registrar. In this case, the easiest workaround for the botmaster is to register the bots and herself to a SIP public service provider. A list of such providers is included in [10]. However, the problem of assigning usernames in this case may not be so trivial. The botmaster and consequently the bots must know which usernames are still available (not taken by other users). This requires either a public directory or a P2P protocol for sharing and updating a list which contains the already assigned usernames. Another more straightforward solution lies in exploiting SIP protocol requests to determine if a UA is alive. For example, an OPTIONS request could be used by the botmaster (or a bot) to identify if a username has already been assigned to another user. According to SIP RFC [11], this request is used by a UA for identifying the capabilities either of another UA or a SIP proxy. Therefore, one could take advantage of this functionality to build a list of the already occupied usernames. A third option is for the botmaster to assign totally randomly generated usernames for the bots, but this may attract the attention of the proxy administrator. Such a list can be shared between the botmaster and each bot beforehand. Generally, it can be argued that the more realistic the usernames the less the chances of being detected as malicious.

In our case, we assume that the Registrar is in the possession of the botherder. We also hypothesize that the bots have been installed in the host machines following an infection. Nonetheless, this infection phase remains out of scope of this work.

3 Architecture and Operation

3.1 SIP as a covert channel

To create a SIP-based covert channel one needs to choose specific parts of the message and use them as data carriers. In fact, several SIP headers or SDP descriptors contained in, say, a SIP request can be used to bear information with special meaning to the communicating parties. In any case, the selection must fulfil the next two requirements. On the one hand, it must be syntactically correct, otherwise the message will be most likely dropped by the parsing process. On the other, it must preserve the communication information at least regarding to the sender and the SIP proxy. Otherwise, the message may never be delivered correctly.

In this work, we concentrate on fields contained in the message body of a SIP request where the literature seems to be quite incomplete. As already pointed out and depicted in the right part of fig. 1, this part of the message follows the SDP data format. Precisely, these pieces of data contain information related to the media parameters of a session and are comprized of 5 mandatory and 15 optional fields [4]. We make use of only two descriptors namely as <*o*> and <*a=ptime*>. The first one is mandatory while the second is optional. The <*o*> descriptor carries information in regards to the session originator and it is composed of 5 fields. Among them, the first and the last one point out the username and the IP address of the caller ("skype2" in fig. 1), while the second and the third indicate a unique session id and the session version. The fourth field is a text string bearing the type of the network ("IN" (Internet) in the normal case). The creation of session id and version fields are up to the creating tool. The RFC [4] suggests that both these parameters must receive numerical string values of at least 10 digits each created based on a Network Time Protocol (NTP) [12] format timestamp in order to ensure uniqueness. Also, RFC states that the <*a=ptime*> descriptor bears the length of time in milliseconds represented by the media in a packet. So, for example, any decimal value representing time in milliseconds is considered normal. The selected fields are shown in red in fig. 1. The interested reader who wishes to get a deeper understanding of SDP can refer to the corresponding RFC [4].

For exploiting the above mentioned fields aiming to deliver a covert channel over legitimate SIP messages one has to set specific values. Table 1 summarizes these values in the context of this work. As observed from the table, the protocol relies on three simple commands related to the type, the parameters, and the execution and termination of an attack. That is, the *a=ptime:<packet time>* descriptor can receive three values 20, 30 and 40. The first one triggers the UA to extract attack parameters and wait for further commands. The other two values correspond to the initiation and termination of the attack respectively. As shown in the table, the second and the third fields of the <*o*> descriptor bear the first and the second half of the victim's IPv4 accordingly. In the example given in fig. 1, the second and the third values of this descriptor are equal to 2383212000 and 3312015300 respectively. So, the IP address can be extracted

by a bot as follows: Assuming a quad-dotted notation, the first two digits of each 10 digit number represent the number of digits that this half of the address is consisted of. In the example, the first two values of session id are 2 and 3 leading the bot to extract the first half of the IP address, i.e., 83.212. In the same manner, session version starts with 33, thus allowing the bot to extract the remaining half 120.153. The last two digits of the second field of session version instruct the bot about the type of the attack. Specifically, a value of 00 means a SYN flooding, while 11 designates a PING one. Special care has been taken for these values to appear as perfectly legitimate ones. To do so, both session id and version numbers are appropriately padded with zeros to reach 10 digits, which is the minimum length suggested by the SDP RFC [4].

Keep in mind that the selected SDP descriptors receive values that correspond to fields which do not affect the session establishment, and thus the covert channel remains functional. In this way, a botmaster is able to hide messages in plain sight without being exposed. On the downside, the use of one optional descriptor for the creation of the covert channel adds 10 extra bytes per message. However, it can be safely argued that this presents a negligible increase in the network traffic to be noticed by the underlying defense mechanisms. Even for a large population of bots, where the botmaster needs to send one SIP request per bot, this augmentation shall be in the order of some tenths of kilobytes (e.g., for 10,000 bots it would be ≈98 kilobytes).

It should be stressed out that the aforementioned descriptors and fields are not the only ones that can be exploited for secretly communicating information between the two ends. Several other selections and combinations are possible. However, each of them should be done in such a way that will attract the minimum attention. For instance, the *k=* descriptor is to be avoided because its use is not recommended by the RFC [4]. Also, the employment of a large number of SDP optional fields for the needs of the covert channel would not only raise suspicions, but also augment the volume of each SIP request. On the other hand, the information carried by the *a=ptime:<packet time>* descriptor in our protocol could be moved to the padded segment of the *<o>* descriptor as given above.

Another point of interest here is that the architecture is fully dynamic because it can be used both by static and mobile UAs. That is, due to SIP intrinsic operation, each bot is reachable from virtually anywhere. As explained in section 2, the IP of the Registrar may change but the bots can become aware of this shift via a domain fluxing scheme or otherwise by extending the C&C instruction repertoire. The reader would likely notice that the communication protocol between the bot master and the bots is one-way. That is, a bot does not send any messages toward its bot-herder. Actually, from the botmaster's point of view, this is not really a problem; as she is in control of the SIP registrar she always knows which bot is alive (i.e., has been registered with the Registrar). On the other hand, one can anticipate that this approach also contributes in keeping the communication channel as hidden as possible. Putting it another way, the

Table 1. Description of C&C protocol messages (character X corresponds to a single digit of the victim's IPv4 address, and Z refers to a digit used for another command or it is zero-padded)

Descriptor	Field	Value	Hidden Message
<o>	sess-id	33XXXXXXZZ	First half of Victim's IP
<o>	sess-version	33XXXXXXZZ	Second half of Victim's IP
<o>	sess-version	ZZZZZZZZ00	SYN Flood attack
<o>	sess-version	ZZZZZZZZ11	PING Flood attack
a=ptime:<packet time>	-	20	Save attack parameters & wait
a=ptime:<packet time>	-	30	Launch attack
a=ptime:<packet time>	-	40	Stop attack

less information are transmitted towards a single receiver (the bot-herder) the less the chances of revealing the channel.

4 Evaluation

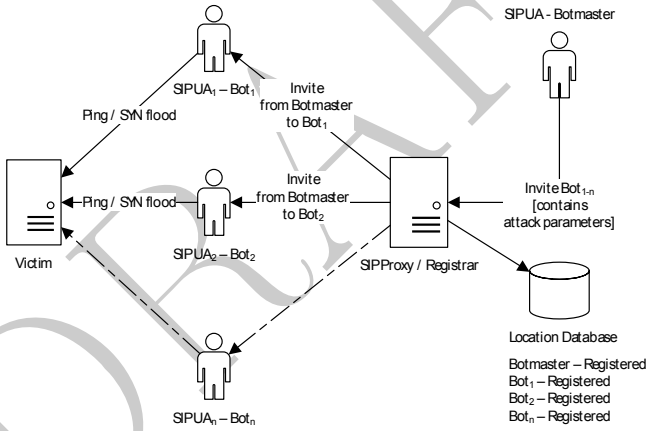

Fig. 2. Deployed test-bed and generic attack scenario

4.1 Test-Bed Setup

In order to evaluate the effectiveness of the C&C covert channel we created a test-bed depicted in fig. 2. We used 7 SIP UAs, one of which was used as the Botmaster and the rest as bots. The SIP UA were developed in JAVA language using the JAIN-SIP library [13]. Each UA runs on an Intel i3 3.3GHz processor with

4GB of RAM. The well-known SIP proxy Kamailio [14] has been employed in the cloud as both a SIP server and Registrar. The server machine was equipped with 1GB of RAM. Finally, the victim's machine was running on an Intel Pentium 4 2.8GHz processor having 1GB of RAM available.

We created six different scenarios each one employing a variant number of attack threads launched by each bot. We released both PING and SYN flooding assaults, each one with different number of attack threads. The first three scenarios correspond to a SYN flooding attack, while the rest to a PING one. For SYN flooding we used correspondingly 5, 15 and 30 attack threads per bot, while for PING 30, 80 and 160. This increased number of attack threads in the second type of attack was used in an effort to augment the impact of this particular attack. This is because a SYN flood is generally more powerful in contrast to a PING one. We used this simulation methodology aiming to grab a better understanding of the attack impact, especially when its volume augments. We employed three metrics to estimate the fallout of each type of attack on the victim's machine; network bandwidth utilization, memory consumption, and CPU usage.

4.2 Results

Figures 3 and 4 present snapshots of the received network traffic and CPU usage at the victim side under a SYN and PING flood attack respectively. From the left figure, one can easily observe that as the number of attack threads per bot remains low, the incoming traffic at the victim's side presents moderate fluctuations. On the downside, when the number of threads increases significantly the incoming traffic doubles. For example, when the threads per bot become equal to 30 the network volume doubles reaching 6MB/sec.

Figure 4 depicts CPU usage at the target machine for a different number of PING flooding threads. It is well perceivable that as the number of threads per bot increases, the CPU utilization percentage augments notably. For example, when the number of the attack threads per bot is set to 30 the CPU usage reaches a maximum value equal to 25%. On the other hand, when the number of threads per bot are sextupled (180), CPU usage reaches a peak value of 30%.

Regarding the memory consumption at the victim's side, we perceived a worst case increment of ≈100% (from 12% to 24%) in the case of PING flooding, and ≈118% (from 22% to 48%) for the SYN one.

5 Related Work

This section succinctly reports on works that have been presented in the literature so far regarding this particular topic. In the following, we group them into three categories. The first one includes a single contribution that relies on SIP to convey botnet singaling. The second embraces two works that identify SIP messages as possible carriers of spurious data. The last focuses on the use of RTP as a means of covert communication.

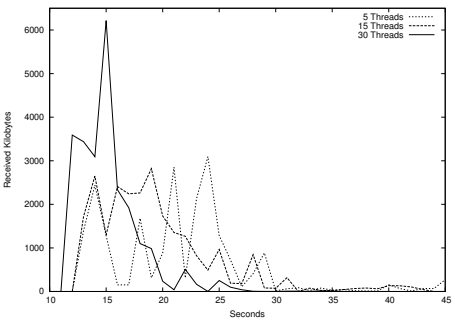

Fig. 3. Network Utilization at victim's side under a SYN flood attack

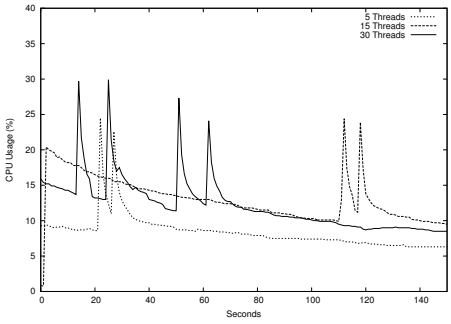

Fig. 4. CPU Usage at victim's side under a PING flood attack

Regarding the first category, the authors in [15] present a SIP-driven botnet. They rely on the well-known Storm botnet by encapsulating its P2P traffic (based on the Overnet Protocol) over SIP. They develop a test-bed composed of 30 bots claiming that the generated message rate resembles that of Storm's one. They correctly observe that their proposal introduces a significant overhead due to the use of SIP as the botnet's conveyor mechanism. This is caused by the need of continuously maintaining permanent connection paths between the various entities. Opposed to that, we simply exploit specific descriptors of the SDP data included in, say, a SIP INVITE request without further stressing the infrastructure.

To our knowledge, until now two works have been presented investigating the potential exploitation of SIP as a covert channel. In [16] the authors survey various steganographic techniques aiming to hide data in legitimate traffic of various networking protocols. They also address this potential for SIP, and for both message body and SDP data. Nevertheless, their analysis is at a high-level only without presenting any real implementation or results. This is in contrast to our work where we capitalize on SDP data to deliver C&C messages. The authors in [17] exploit 3 randomly generated strings included in SIP messages in order to create a covert channel. They rely in chaos theory aiming to analyze and reconstruct the random numbers included in the Call-ID header and the various Tags of a SIP message. This is undertaken by means of a time series analysis.

As already mentioned, the last category of works concentrate on RTP-based covert channels. The authors in [18] make use of four different data hiding algorithms at the RTP layer in order to assess the feasibility of covert channels in VoIP. Their approach is assessed in terms of bit error rates under the G.729 compression algorithm. Similar techniques at the RTP layer has been investigated by works in [19] and [20]. In the first one, the authors propose to build a covert channel over specific unused fields in RTP and RTCP by using steganography. Moreover, they introduce a method which relies on the intentional delay of packets for communicating data secretly. In [20] the authors create a covert channel over IP/UDP/RTP packets aiming to either to improve IP Telephony security

or change the behavior of existing protocols such as RTCP one. They propose two packet payload types characterized as either security or informational. They claim that their protocol is capable of ensuring authentication and integrity not only for the voice and its sender, but also for authenticating protocol parameters, including both the security and informational payloads.

6 Conclusions

This paper elaborates on the exploitation of SIP as a covert channel for building botnet C&C. We demonstrate that with little effort an aggressor is able to tinker with SDP data contained in SIP requests aiming to convey spurious information secretly. This is also done in an straightforward and simple way, perfectly in line with SIP/SDP standards, and without raising any suspicions or causing the messages to be dropped by the receiver as malformed. From a network defense view point, little can be done; the messages seem completely legitimate, they are sent only sporadically and do not augment the network traffic significantly (i.e., 10 additional bytes per bot is perceived). So, even deep and continuous packet inspection at the application or other layer would not reveal something suspicious. The only effective counteraction is to monitor SIP transactions for requests without a matching response. But on the other hand, this mismatch occurs for legitimate SIP transactions quite often too, thus it is to be assumed that it will cause a high false alarm rate at the proxy-side IDS. Overall, we argue that the simplest and more innocuous the covert channel the less the possibility of detecting it. As a secondary contribution, we provide results about the feasibility of such a covert C&C deployment by implementing two kinds of flooding attacks executed by the bots.

We are currently working on enriching C&C with more options for the botmaster. An idea is to find a way to dynamically change the pattern of communication, that is, the places (message headers or descriptors) where the bytes with special meaning are put. This way, the detection of the covert channel would become even harder. On the downside, upon change, this pattern must be communicated to the bot population. A second interesting issue to consider is the possibility of botnet partitioning. That is, in view of what has been discussed in the latter half of section 2.3, having some bots registered to a given public provider and the rest to another one(s). In this case, every bot needs to be informed to which Registrar (domain or IP) must register with, and the botmaster needs to keep and update a list of *{bot-username, Registrar-domain-name}* for being able to correctly dispatch the Invites. This naturally implies an extension of the covert channel to communicate a "Registrar shift" message to the bots.

Acknowledgements

This paper is part of the 5179 (SCYPE) research project, implemented within the context of the Greek Ministry of Development-General Secretariat of Research and Technology funded program “Excellence II / Aristeia II”, co-financed by the

European Union/European Social Fund - Operational program "Education and Life-long Learning" and National funds.

References

1. C. Mohr, "Report: Global voip services market to reach 137 billion by 2020," Nov. 2014. [Online]. Available: http://www.tmcnet.com/channels/hosted-softswitch/articles/393593-report-global-voip-services-market-reach-137-billion.htm
2. A. D. Keromytis, "A comprehensive survey of voice over ip security research," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 514–537, 2012.
3. D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis *et al.*, "Survey of security vulnerabilities in session initiation protocol," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 3, pp. 68–81, 2006.
4. M. Handley *et al.*, "Sdp: session description protocol." RFC 4566, US, 2006.
5. M. Anagnostopoulos, G. Kambourakis, P. Kopanos, G. Louloudakis, and S. Gritzalis, "Dns amplification attack revisited," *Computers & Security*, vol. 39, pp. 475–485, Nov. 2013.
6. R. C. P. Sergio S.C. Silva, Rodrigo M.P. Silva and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378 – 403, 2013.
7. P. Wang, L. Wu, B. Aslam, and C. C. Zou, "A systematic study on peer-to-peer botnets," in *IEEE ICCCN 2009*, Aug. 2009, pp. 1–8.
8. P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, no. 2, pp. 113–127, 2010.
9. D. Geneiatakis, G. Kambourakis, C. Lambrinoudakis, T. S. Gritzalis, "A framework for protecting a sip-based infrastructure against malformed message attacks," *Comput. Netw.*, vol. 51, no. 10, pp. 2580–2593, Jul. 2007.
10. "Sip service providers and carriers," 2015. [Online]. Available: http://www.cs.columbia.edu/sip/service-providers.html
11. J. Rosenberg *et al.*, "Sip: session initiation protocol," IETF RFC 3261, US, 2002.
12. D. Mills, "Network time protocol (version 3) specification, implementation," RFC 1305, US, 1992.
13. P. O'Doherty and M. Ranganathan, "JAIN SIP Tutorial - Serving the Developer Community," Tech. Rep., 2003.
14. "Kamailio – the open source sip server," 2014. [Online]. Available: http://www.kamailio.org/w/
15. A. Berger and M. Hefeeda, "Exploiting sip for botnet communication," in *IEEE NPSec 2009*, Oct. 2009, pp. 31–36.
16. W. Mazurczyk and K. Szczypiorski, "Covert channels in sip for voip signalling," in *Global E-Security*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2008, vol. 12, pp. 65–72.
17. H. Zhao and X. Zhang, "Sip steganalysis using chaos theory," in *IEEE CMCSN 2012*, July 2012, pp. 95–100.
18. T. Takahashi and W. Lee, "An assessment of voip covert channel threats," in *IEEE SecureComm 2007*, Sept. 2007, pp. 371–380.
19. W. Mazurczyk and K. Szczypiorski, "Steganography of voip streams," in *On the Move to Meaningful Internet Systems: OTM 2008*, ser. LNCS. Springer Berlin Heidelberg, 2008, vol. 5332, pp. 1001–1018.
20. W. Mazurczyk and Z. Kotulski, "Covert channel for improving voip security," in *Advances in Information Processing and Protection*. Springer US, 2007, pp. 271–280.