Analyzing Anomalies in Anonymized SIP Traffic

Jan Stanek, Lukas Kencl
Czech Technical University in Prague
Technicka 2, 166 27 Prague 6, Czech Republic
{jan.stanek, lukas.kencl}@fel.cvut.cz

Jiri Kuthan
Tekelec
Am Borsigturm 11, 13507 Berlin, Germany
jiri.kuthan@tekelec.com

***Abstract*—The Session Initiation Protocol (SIP) is a signaling protocol widely used nowadays for controlling multimedia communication sessions. Thus, understanding and troubleshooting SIP behavior is of utmost importance to network designers and operators. However, SIP traffic traces are hard to come by due to privacy and confidentiality issues. SIP contains a lot of personal information spread within the various SIP messages – IP addresses, names, usernames and domains, e-mail addresses etc. The known IP-address anonymization methods are thus insufficient. We present SiAnTo, an extended anonymization technique that substitutes session-participant information with matching, but nondescript, labels. This allows for SIP traces to be publicly shared, while keeping interesting traffic-session properties intact. We further demonstrate its usefulness by studying the problem of SIP NAT traversal as recorded in the anonymized traces. We analyze properties of the so-called "registration storm" incident and measure the influence of the active NAT traversal techniques on SIP traffic pattern, both only possible thanks to the preservation of session relationships inside the anonymized traces. As further benefit to the research community, we set up a public data-store with both the anonymization module and the anonymized traces available and invite other parties to share further SIP data using these open tools.**

I. Introduction

The Session Initiation Protocol (SIP) [1], [2] is a signaling communications protocol widely used nowadays for controlling multimedia communication sessions such as voice and video calls over Internet Protocol (IP) networks. It is also the core networking protocol used within the IP Multimedia Subsystem (IMS) [3] and is expected to become the dominant communication control protocol within LTE mobile networks [4]. With its ever higher proliferation, SIP and its control nodes, the SIP servers, attract much interest from the perspective of practical functioning within the Internet infrastructure, its traffic characteristics and general considerations of deployment. SIP is lightweight and has easily understandable and human-readable structure. However, it is neither easy nor standardized or obvious how to optimize its use in specific situations and environments.

A good way to design optimization techniques for SIP deployment would be to analyze SIP traffic from existing networks. Is SIP traffic stable and monotonous, or does it exhibit wildly fluctuating and skewed patterns? These are sample questions that the community might look to answer. However, publicly available analyses of SIP traffic are rare and thus not a lot of knowledge exists about typical behavior of a SIP server (as opposed to, for example, HTTP servers). Our aim is to promote analysis of real SIP-server behavior. SIP traffic traces are hard to come by due to privacy and confidentiality issues. By its nature, SIP contains a lot of personal information spread within the various SIP messages – IP addresses, names, usernames and domains, e-mail addresses and other information about the session participants.

ISBN 978-3-901882-58-6 © 2014 IFIP

Non-disclosure of IP addresses in publicly shared traces is typically solved by anonymization [5], [6]. Unfortunately, the classic IP-address anonymization methods are not sufficient for SIP traces since they do not handle the other types of sensitive information. Therefore, we present SiAnTo, an extended anonymization technique that substitutes session-participant information with matching, but nondescript, labels. This allows for SIP traces to be publicly shared, while keeping interesting traffic-session properties intact. To this end, we anonymize a sample of traces from a freely accessible and open SIP server with a worldwide user base.

We further demonstrate the usefulness of this technique by studying the well-known problem of SIP NAT traversal as recorded in the anonymized traces. NAT traversal is not well-solved in SIP as the server and clients typically need to support some other mechanisms (STUN [7], TURN [8], ICE [9] etc.) to maintain a connection. If the SIP server becomes unavailable for a time period, the clients keep trying to re-register periodically. Once the server becomes available again, all the clients try to register in a very short time interval and the ones that succeed submit their other requests immediately. This leads to a tremendous increase in SIP messages handled per second and may render the server incapable of answering all these requests timely, leading to delayed registrations and re-sending of time-outed requests. We analyze the properties of the registration-storm incident and discuss the influence of the active NAT traversal techniques on the overall SIP traffic, both only possible thanks to the preservation of session relationships inside the anonymized traces. As further benefit to the research community, we place the anonymization tool as well as the anonymized traces on a publicly available data-store available at `sipdata.org` and invite other parties to share further SIP data using these open tools.

The main contributions of this work are as follows:
(i) the SiAnTo anonymization method and implementation;
(ii) practical study showcasing its usefulness;
(iii) deep analysis of the registration-storm incident;
(iv) establishing an open platform for further SIP data exchange.

II. Related Work

SIP is very popular nowadays and with deployment in various new services, its popularity is only growing. Not surprisingly, many works have already focused on SIP and its various aspects. RFCs [1] and [2] describe the SIP standard as a whole. Papers by Sparks [10] and Prasad and Kumar [11] describe SIP basics and some typical extensions used in SIP environments. The well-known problem of NAT traversal in SIP and various approaches to solving this problem were described in numerous papers e.g by Yeryomin et. al. [12] and Song et. al. [13]. Even though there are many proposals how to solve the NAT traversal issues, none of these was adopted as a general solution since they typically impose strict requirements on the SIP devices and so active NAT keep-alive "pinging" methods are still used by SIP clients and servers. We locate and analyze these methods in real SIP traffic in this paper.

A lot of work was done on identification of SIP anomalies. From these we can name for example works by Heo et. al. [14] who focused on usage of statistical distances for respective SIP message types, Yunli et. al. [15] who proposed usage of Petri nets for SIP INVITE transaction description, Cortes et. al. [16] who used performance metrics to identify SIP processing times, Kang et. al. [17] who used some already known profilers on various real SIP datasets and Ehlert et. al. [18] who used decision modules to learn upon and then detect anomaly SIP traffic. While all the proposed methods are definitely useful, they require live access to the SIP traffic data or at least offline access to SIP traffic dumps. Obtaining such data is often complicated due to personal information contained in SIP traffic and threatening privacy breach issue. We try to overcome this issue by proposing SiAnTo designed specifically for this purpose and demonstrate that the anonymized traffic is still quite useful for anomalies identification and analysis.

The concept of hiding private or sensitive data but preserving some form of structural information has been studied in various sub-domains of ICT. Techniques concentrating on hiding the originator of information are routinely called *anonymization*. An important theoretical foundation for data anonymity and originator protection was laid in [19]. The k-anonymity model for protecting privacy allows holders to release their private data without being distinguishable from at least k-1 other individuals also in the release.

A well-known anonymization scheme over the network packet *IP addresses* called Crypto-PAn [5] preserves the prefix hierarchy of the original addresses, while making them computationally hard to reconstruct by using hashing. This in turn allows to share network traces (with packet headers only), with preservation of the prefix hierarchy. Similarly, in [20], structure of the router configuration files and data is preserved, while the actual values are obfuscated. A technique to process and transform the network *packet payload* has been proposed in [21]. This method uses dictionaries of important sequences that are valuable from the data mining perspective and should be preserved, while encrypting the rest with a cryptographically strong hash function. It is similar to SiAnTo and performs well in terms of data protection, however, its universality creates much overhead and prevents easy automation, a trademark of the lightweight SiAnTo.

SIP traffic anonymization is generally an unsolved problem. There are options allowing for privacy in SIP communication, such as practical and implemented SIPANON [22] introduced by Castleman or more general standard-based SIP privacy mechanism described in RFC 3323 [23] by Peterson et al. There are also various secure SIP proposals incorporating usage of encryption mechanisms such as RFCs 3329 [24] by Arkko et al. and 6216 [25] by Jennings et al. However, these mechanisms modify the SIP traffic directly and require deployment into the actual SIP network. Moreover, such traffic modification complicates the monitoring of SIP traffic and dependent service such as billing, troubleshooting etc. We aim to solve the problem of anonymization (privacy preserving) of captured SIP traffic, making the traces publishable without the fear of compromising privacy and without the need for any modification of the deployed SIP environment itself.

III. SIP Anonymization

Capturing a "real" SIP trace from a SIP deployment is not a difficult task, however, infrastructure owners are reluctant to share this data due to concerns about clients' privacy. The same problem arose in the past with web traces and was solved by anonymizing IP addresses and removing packet payloads. Such anonymization is not feasible in case of SIP, as SIP is an application layer protocol. By removing packet payloads, one would lose all the SIP-related information. Methods anonymizing SIP traffic "inline", during actual communication, such as [22], [23], require deployment inside the SIP environment. From the perspective of network administrators, they often modify SIP traffic in an undesirable manner too. We are not aware of any publicly available solution to anonymize an existing SIP traffic dump. There are two possible ways to tackle the problem of SIP anonymization: either extract only the "minimum" necessary information from each SIP message and keep and anonymize only this essential part, or leave the whole SIP message intact and anonymize only the potentially privacy leaking parts. Anonymization is easier to handle in the first approach as identification of personal data in smaller, clearly-defined parts is rather straightforward. However, implementing the first approach, it would be necessary to choose and define what data exactly are to form the essential part and some important property might be missed, complicating the utility of such anonymized capture. We follow the second approach. The SIP Anonymization Tool (SiAnTo) presented in this paper keeps SIP messages largely intact and leaves out only segments impossible to anonymize in principle (e.g. user-identifying SDP lines in the INVITE messages that could contain personal information in varying format). With regards to statistics and analyses, traffic anonymized with this tool is almost as useful as unanonymized traffic. Design, properties and implementation of SiAnTo are described in this section.

Schematic 1: Packet anonymization process inside SiAnTo

Input: Packet P to be anonymized
Output: Anonymized packet A
begin
  $A := P$
  foreach *defined object pattern* O **do**
    $S = find(O \text{ in } P)$
    foreach $x \in S$ **do**
      $y = O.anonymize(x)$
      $replace(x \Rightarrow y \text{ in } A)$
    end
  end
  $fix\_lengths\_and\_checksums(A \rightarrow header)$
  return A;
end

A. SiAnTo Architecture

The core problem was to identify the potential personal information leaking objects inside SIP messages and devise a way for their anonymization. After analyzing the SIP standard and SIP traffic, we concluded to use pattern matching for object identification, as the benevolent SIP architecture allows for optional loosely-defined header fields and message parts that potentially contain personal information too. Each object type has its own specific *anonymize* function. The object types share the mapping tables for username and domainname anonymization. The high-level schematic of SiAnTo per-packet processing is described in Schematic 1.

The three typical objects leaking personal information were identified as: 1) IP address 2) SIP URI 3) e-mail address

IP address anonymization was already handled well in many works targeting web traffic anonymization. We chose to use Crypto-PAn [5] for anonymization of both the IP addresses inside the IP header and IP addresses inside the SIP message. During anonymization, we keep track of the unanonymized-anonymized IP pairing in form of a 1-to-1 map. This way, the reverse process of deanonymization is possible. Additionally, deanonymization of IP addresses could also be achieved using the built-in Crypto-PAn feature if a key string is used.

SIP URI anonymization consists of three steps. First, we locate lines containing SIP URIs inside SIP messages by looking up the `sip:` string. Second, the SIP URI must be separated from the rest of the line using content-aware separator-oriented string tokenization. This is necessary due to the SIP URI form (see example in Figure 1). Note that the free-text username part is not effectively part of the SIP URI itself, however we need to anonymize it since otherwise it could leak user identification. The "extended" URI is therefore split into the free-text username part (everything before `sip:`), username part (everything after `sip:` and before `@`), domainname part (everything after `@` and before `:` or a delimiter) and the port part (if `:` is present, then everything from `:` to the next delimiter). Third, the username and domainname parts are anonymized separately. Each unique username is mapped to a unique sequence number and a table, mapping the username to a sequence number, is maintained. The same goes for domainname. An example of anonymized extended SIP URI is available in Figure 2. The secure variant of SIP URI identifier, `sips`, is handled the same way as `sip:`.

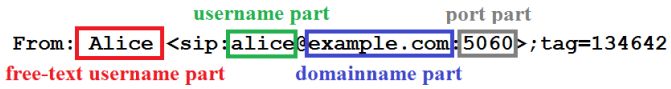

Fig. 1. Example of a `From:` line inside a SIP INVITE request. Division into tokens for anonymization process highlighted.

E-mail address anonymization is similar to SIP URI anonymization. We can locate e-mails by looking up the `mailto:` identifier or the `@` symbol. If it contains `@` and is not a SIP URI, then it probably is an e-mail. However, since e-mail addresses are not directly related to SIP, we simply remove them instead of anonymizing them. If reconsidered, anonymization could be done in the same way as for the SIP URI, just with separate map tables for usernames.

There is one other specific part requiring anonymization, though not part of the SIP protocol by itself. This is the Session Description Protocol (SDP) payload inside an INVITE request. While we consider only the signaling part (SIP traffic) anonymization and not the actual media traffic (is not part of the captured signaling traffic), lines starting with `u=`, `e=` and `p=` are interesting from the point of containing personal information. For simplicity, we delete these lines during anonymization, keeping all other SDP lines intact.

Further object types could easily be added to be handled by the anonymizer in the same manner as the existing object types and such extension would not affect the overall architecture.

B. SiAnTo Security and Privacy

While the proposed anonymization method is privacy-preserving on the basic level and not leaking any personal information directly, we cannot call it a secure solution from the formal security standpoint. There remain quite a few possibilities to exploit the unanonymized parts and properties to gain some knowledge about the participants in the SIP traffic captured – specific version of SIP clients (UA SIPAUA.build.bob0001 probably belongs to Bob), frequency analysis of SIP connections revealing timezone where the SIP network is deployed, pattern matching of known unanonymized traces to the anonymized ones etc. We are aware of these possibilities, however, all these attacks require some additional knowledge. After anonymization, the trace by itself does not provide sufficient information usable for mapping the anonymized sequence numbers to existing users/domains.

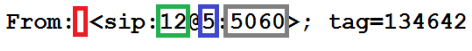

Fig. 2. Example of `From:` line inside a SIP INVITE request processed by SiAnTo. The free-text username part was deleted, username and domainname parts were anonymized and port part remains the same.

Since the traces do not contain the actual sessions (voice or video) and since to obtain the additional information required to derive the sequence-name mapping (for example about SIP user accounts and their use inside a company) the attacker would likely have to compromise the infrastructure of the service provider (which no anonymization can prevent), we consider the level of privacy sufficient for the initial release. We are looking forward to user feedback and plan to modify SiAnTo according to users needs. Removal of authorization fields and noonces during anonymization is currently under consideration.

C. SiAnTo Implementation

We implemented a prototype of SiAnTo in the form of a standalone C++ application. The prototype is available for using and testing at `sipdata.org`. To allow other researchers modifications, we publish the prototype as open source, under the GNU GPL license. The current version of the prototype (version 1.0) is capable of processing about 6500 packets per second on an average server machine (Intel Xeon@2.5GHz) and is single-threaded. Processing speed can be greatly improved (based on the available number of CPU cores) by splitting the capture file before processing and running the application multi-threaded.

To improve the utility of the tool, we also implemented some basic SIP statistics, generated during anonymization. These are among others: distribution of SIP messages to individual request and response types, traffic in "to server" and "from server" directions, number of individual user-names/domainnames in the trace, etc.. The user will therefore get not only his anonymized traffic dump but also a list specifying all basic SIP characteristics of the traffic processed. The computational cost of these statistics is very low as all the required information has to be processed anyway for the anonymization reasons. To view which statistics are currently provided, kindly visit `sipdata.org`.

IV. Description of the Testbed

This section describes the network in which we conducted our experiment.

To obtain realistic and representative results, we were running all our experiments including scheduled outage against a live SIP service hosted by the `iptel.org` site. The `iptel.org` site is an open SIP service. That means that any Internet user with a working SIP client can create an account and use the service for Internet telephony. As a result, the site is used by SIP clients of tens of different types and varying level of standard compliance and maturity. Some clients use the service from behind NATs, some use the service directly from the public Internet. The clients are geographically spread all around the globe with majority of users located in Europe and Northern America. For details see [26].

A. Network Traffic

In our experiments, there were two types of load: the ordinary `iptel.org` traffic and additional background load we introduced to cause congestion conditions.

Ordinary traffic is caused by about 3100 clients that are registered at any time of day with the service. For the sake of this study, we only consider SIP traffic and do not consider RTP traffic relayed through the site to facilitate NAT traversal. Ordinary traffic amounts to approx. 320 packets per second (pps). The biggest part of the SIP traffic, approx. 40%, is constituted by SIP REGISTER requests and responses to these requests. The reason is that the service is configured to force clients to re-register frequently. Frequent re-registrations help clients behind NATs to keep their NAT-bindings alive. If the bindings were not refreshed they would expire and incoming SIP traffic towards the client would fail.

The background load was generated by SIPp-DD [27], a modified version of the SIPp traffic generator [28]. It caused variable additional load and increased CPU load of the Session Border Controller in front of the `iptel.org` network to create the state necessary for congestion tests.

B. Network Topology

The `iptel.org` service consists of two key SIP elements: a Session Border Controller (SBC) and a SIP proxy server. The SBC is connected to both the public Internet and intranet, and its main task is facilitation of NAT traversal. The SIP proxy server is combined with a SIP registrar and located in a private network. The server performs all other SIP processing.

The SBC is "Frafos Adaptive Border Controller (ABC)", version 2.0.1.27 running on a linux-based rack PC – Dell Poweredge 1850 server constituting 3.2GHz Intel Xeon CPU with 2GB RAM. The SBC acts as a Back-to-back User Agent, i.e., it appears in SIP signaling as a User Agent to both upstream client and downstream server (see [29] for a more detailed description of the B2BUA concept). The SBC is configured to force clients to re-register in short intervals no longer than 180 seconds. This is achieved by forcing the value of "expires" parameter in Contact header field in answers to clients' REGISTER requests. This behavior dramatically increases registration traffic. To make sure that the traffic does not offend the infrastructure behind the SBC, the SBC caches the registrations and passes them downstream only in much longer intervals. Note that also other methods can be used to keep NAT bindings alive: clients sending empty packets or STUN requests to servers or servers sending unsolicited OPTIONS or NOTIFY requests to clients. However, the `iptel.org` configuration is using the frequent re-registration method so that it works with practically any SIP client, regardless what NAT traversal features it implements or not.

The proxy server is implemented using "sip-router", version 3.3.0, running on a linux-based rack PC – Dell Poweredge 1850 server with 3.2GHz Intel Xeon CPU and 2GB RAM. It authenticates incoming traffic against its subscriber database using SIP digest authentication, stores SIP registrations in user location database and performs basic call processing: user location lookups, call-forwarding, and routing.

REGISTER requests from the public Internet visit the SBC initially. The SBC decides whether it can process them locally using its cache or forwards them to the downstream SIP server.

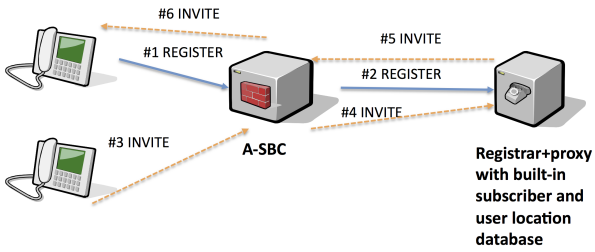

Fig. 3. Topology of the `iptel.org` service site.

If the request is forwarded, the SIP server authenticates the REGISTER and updates contacts upon success.

All other requests take a loop path through the network: they initially visit the SBC. The SBC forwards the requests to the proxy server. The proxy server looks up the request recipient in its user location database and forwards it to the registered User Agent through the SBC.

The message flow is depicted in the diagram in Figure 3. For the sake of brevity, the diagram only shows non-cached requests and omits authentication and responses.

V. Data Analysis

This section describes data analysis over the SIP data collected from network described in Section IV and anonymized using SiAnTo described in Section III. The traces were specifically captured to contain a SIP server outage period, to allow analysis of traffic anomalies caused by it. The outages were negotiated with our colleagues from `iptel.org` and caused artificially (by a firewall block). We stress that the outages were agreed upon beforehand and were timed and long enough only as necessary, to diminish the negative impact on the clients. To inspect the impact of heavy load on phenomena observed during outage, artificial load was generated using SIPp-DD [27], a modified version of the SIPp [28] traffic generator. Artificial traffic consisted of REGISTER requests with constant rate (simulating additional clients trying to register). We got the following traces for analysis:

- TR01 44 minutes of normal SIP traffic in 1128074 packets; time division [12 mins normal traffic; 21 mins outage; 11 mins after outage]
- TR02 41 minutes of enhanced SIP traffic (750 artificial REGISTER requests mixed with normal traffic from the 18th till the 30th minute) in 1941810 packets; time division [11 mins normal traffic; 10 mins outage; 20 mins after outage]
- TR03 34 minutes of enhanced SIP traffic (1500 artificial REGISTER requests mixed with normal traffic from the 18th till the 30th minute) in 2118021 packets; time division [11 mins normal traffic; 10 mins outage; 13 mins after outage]

Together with the traces, we got also the full log from the anonymization process. The concrete SIP statistics important for the analysis are described in detail later in this section. Worthwhile noting are also the processing times of the anonymization process, which took 150s for TR01, 264s for TR02 and 284s for TR03. This demonstrates ability of SiAnTo of processing even huge traces in reasonable time.

In the analysis we focused on the impact of presence of active NAT keep-alive methods on SIP traffic pattern and on the problem of registration storms. We also discuss the impact of anonymization on the measurements and the limitations imposed by anonymization on the analysis options.

A. Active NAT Keep-Alive Methods

Next to the standardized methods of NAT keep-alive defined for SIP (ICE, TURN), many clients and servers still use active repetitive messages to keep the NAT bindings alive. The reasons are purely practical – if the SIP device does not support any of the standardized methods, it can still keep its binding using periodical messages. The content of the messages is irrelevant, REGISTER or OPTIONS messages are used most often. Proprietary solutions such as empty UDP packets (with payload set to `0x0d0a`) and NOTIFY messages for the `keep-alive` event are also frequently used.

Instead of sending periodic SIP messages, the SIP server might choose to modify registration policy and force the clients to register more often. This can be done via setting the `expires` parameter in response to the REGISTER request (typical value is 3600s). Setting `expires` low enough, the server forces the clients to re-register often and thus the NAT bindings are kept alive using these enforced registers.

The drawback of active NAT keep-alive is the traffic overhead. SIP is by design very low-load and periodic messaging violates this property, as we demonstrate in the analysis.

B. Registration Storm in Theory

A "registration storm" is an incident occurring when many SIP clients try to register at the same SIP server at the same time. Depending on the implementation and configuration of the SIP server, the registration storm may negatively impact the SIP service and even lead to server failure in the worst case. Due to the nature of SIP – long default register expiration period of 3600s and often maximum thousands of users per a SIP server, this situation seems strange to occur, however, it was observed repeatedly in the SIP environment and leads to significant problems.

Registration storms seem to be caused by the SIP server stopping to serve requests, thus the clients becoming unregistered and trying to re-register. The reason for server unresponsiveness is typically a network failure. The SIP server becomes unreachable, however, as SIP is an application layer protocol, the SIP clients are unaware of the unreachability of the server and keep demanding registration. Interestingly, the rate of the registration traffic from clients increases quite drastically during server outage (impatient clients shorten their registration period, to be re-registered as soon as possible). After the server becomes reachable again, it starts serving requests, but due to the nature of SIP register handshake (after receiving response to the first REGISTER, sending a

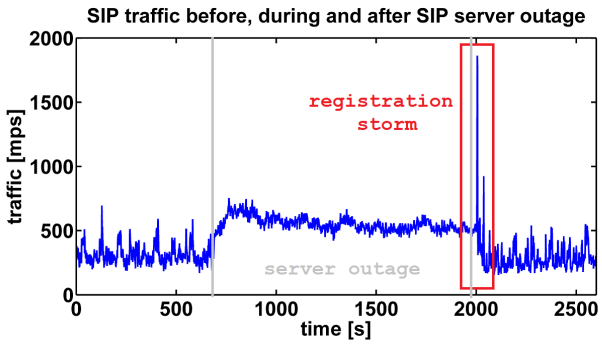

Fig. 4. The overall SIP traffic intensity monitoring before, during and after SIP server outage in a 1s resolution. A registration storm incident is emphasized.

second one containing the necessary authorization info) this only worsens the situation for a short time period as the clients generate additional register requests to finish the handshake. Additionally, successfully registered clients try to get access to other services using SUBSCRIBE, OPTIONS and NOTIFY requests right away, which just intensifies the load.

C. Analysis of the Actual Registration Storm

We use the TR0 trace in this analysis, graphical representation of the traffic is available in Figure 4.

To analyze the traffic in more detail, we split the trace into three different parts – normal traffic, outage traffic and registration storm traffic. Normal traffic term is used for traffic before the server outage and after the residues of the outage diminish and represents a casual load on the SIP server. As a representation of this traffic type we analyzed first 12 mins of the trace. Outage traffic starts when the SIP server becomes unreachable and ends when it becomes reachable again. Registration storm starts when the SIP server becomes available after the outage and ends when the level of registration traffic (REGISTER messages per second) stabilizes.

Properties of normal SIP traffic

number of registered users	fluctuating around 3152
average traffic load	323 pps (packets per sec.)
traffic to:from SIP server	55% : 45%
request types distribution	see Fig. 5

SIP traffic is generally assumed to be symmetric and of low load. Average load of 323 pps for 3152 registered users is a bit higher than expected, but this is caused by many active NAT keep-alive mechanisms observed in the traffic and by the fast-paced re-registration enforced by the server (`expires` set to 180s). The slight but notable asymmetry in the data indicates probability of minor problems with request handling for specific cases. After thorough analysis, we found a few of these problems, an interesting one being ignorance of requests having specifically malformed domain URI part (with two `@` signs inside the URI). The predominant request types are caused mainly by active NAT traversal methods – re-registration policy set up by the SIP server, active probing by clients using OPTIONS requests and `keep-alive` event related SUBSCRIBE and NOTIFY messages. Obviously, most

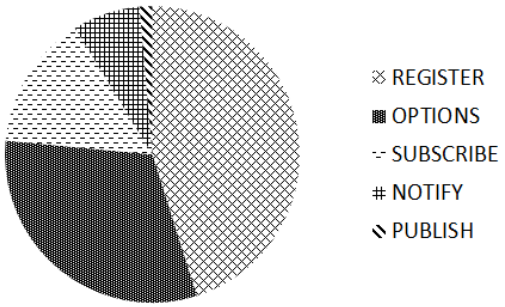

Fig. 5. Request type distribution in normal SIP traffic observed on the monitored server. Request types having less than 1% of the request traffic were left out.

of the SIP traffic is used just to keep the users registered and the NAT binding alive. Note that the analyzed snippet contained only 203 INVITE requests out of the total number of 90673 requests e.g. the actual session creation method constitutes only 2.2‰ of all the requests.

Properties of "outage" SIP traffic

number of registered users	decreasing 3152 $\rightarrow$ 0
average traffic load	549 pps
traffic to:from SIP server	100% : 0%
request types distribution	see Fig. 6

The traffic is notably higher than in the normal case, the biggest increase is in the number of REGISTER requests. This is caused by the clients who, after finding out that their register request was not responded to, start repeating the request more often. Depending on the type of client, the register message rate observed is around 1-17s (while in the normal case it is between 100-180s). Typical clients make a few tries (typically 8-20) in tight succession and then wait for a time period before trying again. The observed time periods of waiting vary – the shortest one observed was 30s, the longest 1200s. Some "smarter" clients are even able to dynamically prolong the waiting period, starting with 60s first waiting, 120s second etc. Figure 7 shows register requests of randomly chosen clients plotted throughout the whole trace. Note the different registration rate outside the outage and the different behavior during the outage. Interestingly, there are not only REGISTER requests being sent during the outage, as one would expect. All the observed request types are performing some kind of

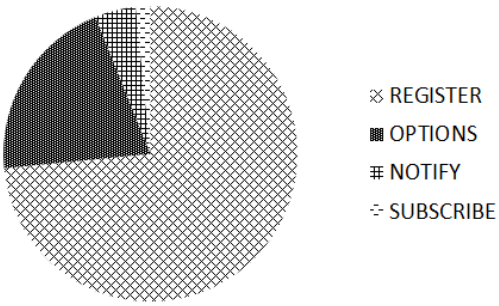

Fig. 6. Request type distribution in "outage" SIP traffic. Request types having less than 1% of the request traffic were left out.

Fig. 7. Register requests of a few users demonstrating different registration patterns and client behaviour. Note that client A stopped trying and did not register again in the analysed trace. This per-user analysis is only possible thanks to the session-preserving feature of the SiAnTo anonymizer.

NAT keep-alive – periodic OPTIONS requests, REGISTER requests (these are hard to discern due to many repeated "classical" register requests), SUBSCRIBE and NOTIFY with the `keep-alive` event. As observed, the clients try to keep their NAT bindings alive even if they are not registered. These requests are inherently useless and a small modification in the client software could optimize the SIP traffic by not issuing keep-alives when not registered.

Properties of the registration storm traffic

number of registered users	increasing $0 \rightarrow 1988$
average traffic load	481 pps
traffic to:from SIP server	54% : 46%
request types distribution	see Fig. 8

The moment the server becomes available again, the register requests start being processed, which leads to enormous increase of traffic load in a very short time. While the registration storm duration is about 40s in the observed case, the time interval of the excessive traffic is much shorter (about 3 seconds). The traffic increase is caused by the nature of SIP register handshake. The clients that manage to successfully register then immediately require additional services by issuing other requests. Therefore, instead of trying one message in a few secs (typical in the outage case), each client sends at least 2, but often 4-6, messages almost at once, generating more traffic. As can be seen in the request type distribution graph (Fig. 8), the distribution of requests is getting back from the outage distribution into the normal distribution.

A better view of the storm progress can be made looking at the traffic load progress in Fig. 9 (the traffic formed by

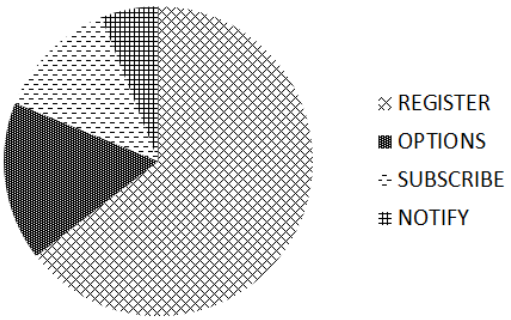

Fig. 8. Request type distribution during the registration storm.

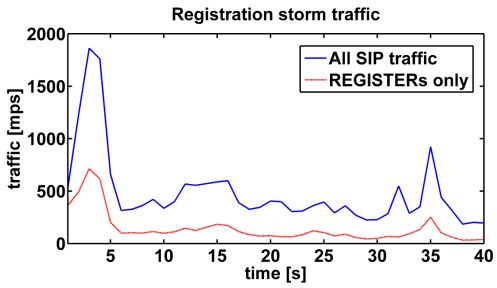

Fig. 9. Registration storm traffic progress.

REGISTER requests only is plotted in red). The spike visible in the 34th second was identified as a small group of clients that, upon receiving the `500 Server Internal Error` response during the storm started waiting for 30s before trying to register again. Presence of `500` responses during the registration storm was unexpected. We analyzed the responses and found out, that almost 17% were `500`. The cause was identified as a temporary SIP server problem that occurred during the highest load (between the second and third second of the storm). The server was able to overcome the error quite quickly, the majority of `500` responses was sent between the second and sixth second of the storm, yet, some `500` responses were still being sent even in the latter part of the storm. Closer inspection of the problem by our colleagues at the provider side showed that there was a memory error that occurred due to strange race conditions. They were able to track the problem and issue a fix thanks to the results obtained from the anonymized-trace analysis.

After the first wave of the register requests is handled, the traffic quickly returns to its normal pace and the properties become the normal SIP traffic ones. While the duration of the registration storm itself, characterized by extremely increased request traffic rate is quite short, the residues of the server outage in form of number of not yet successfully re-registered clients prevail for much longer. Note that after the registration storm ended, there were only 1988 successfully registered clients out of the original 3152 ones (see Figure 10). This slowed re-registration is caused by the "smarter" clients, that are waiting for longer periods instead of periodically sending their REGISTER requests in fast pace. While this behavior actually helps to lower the congestion during the registration storm (as these clients are not sending any requests), the question is whether it is desirable from the service point of view – until the clients successfully re-register, they are unreachable for other clients and thus cannot participate in sessions. Ping from the server side targeting the previously known client IP addresses after the outage might help to speed up the re-registration process quite notably (but should be arranged to be sent only after the registration storm fades out).

D. Registration Storm with Additional Traffic Load

In the previous analysis, the SIP server utilized only 18% of his processing capacity under normal load and was thus highly overprovisioned. To analyze the progress of the outage

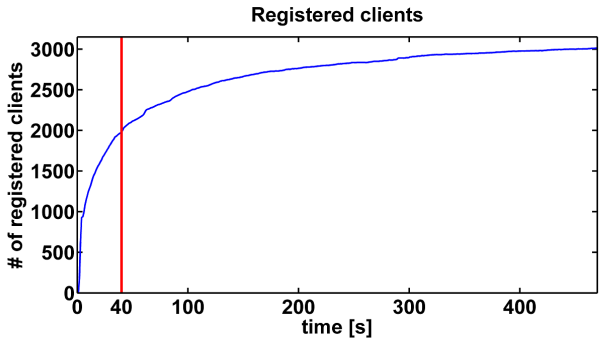

Fig. 10. Number of successfully registered clients during and after the registration storm.

and the consecutive registration storm for SIP server under higher load, we used traces TR02 and TR03 in this analysis.

Analyzing TR02, the observed registration storm as well as the re-registration time of the clients did not change notably from the TR01 case. The storm itself was 3s longer and the average re-registration time per client increased by some 0.2s (the re-registration time would probably increase more but the error producing `500 Internal Server Error` responses to REGISTER requests that we encountered in the previous analysis was already fixed and so there were no delays caused by it). Other than that there were no notable differences in the traffic pattern.

Before analyzing TR03, we were notified by colleagues from `iptel.org`, that this test had effectively incapacitated the SIP service. The analysis showed that after the outage ended, the clients were getting some responses, but only very irregularly and not a single client was able to re-register again. The incident would require additional analysis in cooperation with `iptel.org`, however, the observed impact was devastating for the SIP service in this case.

Our tests with artificial load demonstrate that if the storm is strong enough facing a heavily utilized SIP server, it could lead to critical disruption of service. However, the artificial load is just a simple substitution for the real situation and gaining access to traces containing real registration storms from deployed SIP environments will make it possible to conclude general statistics and metrics for this incident.

E. Mitigating Registration Storm

Thanks to the analysis, we now better understand the causes and progress of a registration storm. The inevitable question is how to defend against registration storms or, if impossible, how to diminish their negative impact?

If we do not wish to modify the registration handshake of the SIP standard itself and we cannot modify the behavior of all the different client types, we do not see any way how to prevent registration storms from happening. Decreasing the number of concurrent registrations during a storm by setting longer registration expiry period, thus allowing for longer server outage without all the clients registrations expiring is a possibility, however, while this might help a bit, it is not a very good idea due to the problem of NAT traversal and already mentioned active NAT keep-alive pinging, that is still very popular in SIP environments. It is also possible to use a transparent backup SIP server substituting during outage of the primary server, however, this is an expensive solution and cannot generally prevent outages caused by network failures.

If we accept the fact that server outages and subsequent registration storms are happening, we need some ways to diminish their impact. We now list some of the possibilities and discuss their properties.

The most straightforward solution is overprovisioning. If the SIP server is capable of handling 10-times higher load than its normal traffic, then it is very likely to survive the storm without problems, recovering from its impact in a few seconds. The disadvantage of this solution is price. Overprovisioning requires more expensive hardware and higher maintenance costs. It is also very ineffective since most of the time the overprovisioned capacity is wasted, as it cannot be used for other purposes.

Another solution is to distribute the load peak over time by dropping messages or asking clients to try later. Particularly, inserting a "Retry-After" header field inside the 5XX response message should force the clients to wait for the time specified inside this field. Using a randomized threshold-based variable as the "Retry-After" value should lead to efficient load spreading. The question is whether the SIP clients are implemented to use the time interval returned them inside this header field. We plan to address this question in future work.

Non-SIP based solution is to limit the number of requests per second on the network device before the SIP server. If we calculate the maximum number of requests per second the SIP server is able to handle and set a firewall limit for SIP port to this number, congestion at the SIP server will never occur. Unfortunately, due to the nature of SIP registration handshake, this solution might also lead to extreme registration prolonging. In the worst case, the first register request from one client gets processed, but the second one (with the actual authorization information) is over threshold and gets dropped. The client tries to re-register later, but the new registration request faces a chance of dropping again.

F. Anonymization Discussion

In Section III we introduced our anonymization technique together with an open source tool SiAnTo that implements it. We have used only anonymized data for the analysis introduced in this section, yet we have not specifically pointed out whether or how the anonymization itself impacts the progress or results of the analysis.

Concerning registration-storm analysis, the specific anonymization employed enables tracing of individual sources even after the anonymization and thus enables the vital *per-source analysis*. To prevent the need for guessing the SIP server address (unknown after anonymization) within the anonymized trace, we implemented SiAnTo so that it provides the new SIP server address on its output too. We recommend sharing the anonymization statistics together with

any shared captured SIP traffic data to make classification and search over the datasets easier.

When comparing the non-anonymized data obtained in our previous work [26] with their anonymized version, we identified the following limitations of our anonymization technique:

- geographical division based on IP address assignment is impossible due to the anonymized IP addresses;
- non-SIP messages get lost, even though they might be used for SIP-connected purpose (typically "empty" UDP NAT keep-alive pings);
- non-standard SIP messages can get malformed;
- SiAnTo currently does not handle IPv6 and has some issues with SIP over TCP.

We are currently unaware of other limitations. If any arise when SiAnTo is used on other SIP traces, we shall address these issues in future work and development of SiAnTo.

VI. Conclusion

With the growing popularity of SIP, the need for proper understanding of the engineering limits of its deployment is going to rise. Without open sharing of technical records within the engineering community this would prove difficult. Sharing of anonymized traces is thus *essential* to develop best practices in SIP deployment.

We have presented a first proposal for *full anonymization* of SIP traces, and shown that such traces reveal important traffic aspects, in particular relevant to the long-term problem of SIP, the NAT traversal. We have analyzed the incident of registration storm, described its progress, properties and discussed its eventual consequences. Our traffic analysis suggests that registration storms might after all not be as dangerous a phenomenon as perceived by the engineering community [30]. Nevertheless, we suggest a few remedies such as load distribution over time, request number limits or overprovisioning. Many other aspects can be analyzed in a similar manner, in particular various vulnerability exploits or events stemming form complex behavior of large SIP-user communities.

We make a further step of setting up `sipdata.org`, an open public SIP data-exchange platform similar to those already existing for pure IP traffic, and initiate it with the traces of this paper and the SiAnTo anonymization tool. While we have tested SiAnTo only on data provided by a single company, other problems that we did not encounter may arise. However, our plan is to maintain and regularly update SiAnTo to reflect the needs and feedback of its users.

We invite further interested parties to become involved and contribute to `sipdata.org`. This should spur many creative uses, understanding and best practices of SIP deployment, allowing researchers for testing their approaches and validating their assumptions on real SIP traffic samples and resulting in a marked improvement of all SIP-engineering aspects.

Acknowledgment

We thank iptel.org for generously providing access to SIP-server traces. We also thank CESNET z.s.p.o. for its kind project support through grant No. FR496/2013.

References

[1] Handley et. al., "SIP: Session initiation protocol (RFC 2543)," http://www.ietf.org/rfc/rfc2543.txt.

[2] Rosenberg et. al., "SIP: Session initiation protocol (RFC 3261)," http://www.ietf.org/rfc/rfc3261.txt.

[3] 3GPP, "3GPP Specification 23.228," http://www.3gpp.org/DynaReport/23228.htm.

[4] 3GPP , "3GPP Specification Series 36," http://www.3gpp.org/DynaReport/36-series.htm.

[5] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon, "Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme," *Computer Networks*, vol. 46, no. 2, pp. 253 – 272, 2004.

[6] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 29–38, 2006.

[7] Rosenberg, J. et. al., "Session traversal utilities for NAT (STUN)," http://tools.ietf.org/html/rfc5389.

[8] Mahy, R. et. al., "Traversal using relays around NAT (TURN): Relay extensions to session traversal utilities for NAT (STUN)," http://tools.ietf.org/html/rfc5766.

[9] J. Rosenberg, "ICE: A protocol for network address translator traversal for offer/answer protocols," http://tools.ietf.org/html/rfc5245.

[10] R. Sparks, "SIP: Basics and beyond," *Queue*, vol. 5, no. 2, pp. 22–33, Mar. 2007.

[11] J. Prasad and B. Kumar, "Analysis of SIP and realization of advanced IP-PBX features," in *ICECT 2011*, vol. 6, april 2011, pp. 218 –222.

[12] Y. Yeryomin, F. Evers, and J. Seitz, "Solving the firewall and NAT traversal issues for SIP-based VoIP," in *ICT 2008*, june 2008.

[13] M. Song, J. Chi, R. Pi, and J. Song, "Implementing an express SIP NAT traversal server," in *ICPCA 2007*, july 2007, pp. 527 –529.

[14] J. Heo, E. Chen, T. Kusumoto, and M. Itoh, "Statistical SIP traffic modeling and analysis system," in *ISCIT 2010*, oct. 2010.

[15] Y. Bai, X. Ye, and Y. Ma, "Formal modeling and analysis of SIP using colored petri nets," in *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011*, sept. 2011, pp. 1 –5.

[16] M. Cortes, J. R. Ensor, and J. O. Esteban, "On sip performance," *Bell Labs Technical Journal*, vol. 9, no. 3.

[17] H. J. Kang, Z.-L. Zhang, S. Ranjan, and A. Nucci, "SIP-based VoIP traffic behavior profiling and its applications," in *MineNet*, 2007.

[18] S. Ehlert, C. Wang, T. Magedanz, and D. Sisalem, "Specification-based denial-of-service detection for SIP Voice-over-IP networks," in *Internet Monitoring and Protection, ICIMP '08.*, 29 2008-july 5 2008.

[19] L. Sweeney, "k-anonymity: a model for protecting privacy," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[20] D. Maltz, J. Zhan, G. Xie, H. Zhang, G. Hjalmtysson, J. Rexford, and A. Greenberg, "Structure preserving anonymization of router configuration data," in *ACM IMC '04*, 2004.

[21] R. Pang and V. Paxson, "A high-level programming environment for packet trace anonymization and transformation," in *In Proceedings of ACM SIGCOMM*, 2003.

[22] M. Castleman, "SIPANON: A SIP anonymizer," *Columbia University Course Report (CS W3998)*, 2001.

[23] J. Peterson, "A privacy mechanism for the session initiation protocol (SIP)," 2002.

[24] J. Arkko, G. Camarillo, A. Niemi, T. Haukka, and V. Torvinen, "Security mechanism agreement for the session initiation protocol (SIP)," 2003.

[25] C. Jennings, B. Hibbard, K. Ono, and R. Sparks, "Example call flows using session initiation protocol (SIP) security mechanisms," 2011.

[26] J. Stanek, L. Kencl, and J. Kuthan, "Characteristics of real open SIP-server traffic," in *Passive and Active Measurements Conference 2013*.

[27] J. Stanek and L. Kencl, "SIPp-DD: SIP DDoS flood-attack simulation tool," in *ICCCN '11*. IEEE, 2011, pp. 1–7.

[28] R. Gayraud and O. Jacques, "SIPp open source traffic generator for SIP protocol," http://sipp.sourceforge.net/.

[29] H. Kaplan and V. Pascual, "A taxonomy of session initiation protocol (SIP) back-to-back user agents," IETF, Internet Draft, October 2012.

[30] C. Shen, H. Schulzrinne, and A. Koike, "A mechanism for session initiation protocol (SIP) avalanche restart overload control," IETF, Internet Draft, February 2014.