An efficient and easily deployable method for dealing with DoS in SIP services

Zisis Tsiatsikas[a,*], Dimitris Geneiatakis[b], Georgios Kambourakis[a], Angelos D. Keromytis[c]

[a]*Dept. of Inform. and Comm. Systems Engineering, University of the Aegean, Karlovassi, Greece*
[b]*Digital Citizen Security Unit, Joint Research Centre, European Commission, Ispra, Italy*
[c]*Department of Computer Science, Columbia University, New York, USA*

Abstract

Voice over IP (VoIP) architecture and services consist of different software and hardware components that may be susceptible to a plethora of attacks. Among them, Denial of Service (DoS) is perhaps the most powerful one, as it aims to drain the underlying resources of a service and make it inaccessible to the legitimate users. So far, various detection and prevention schemes have been deployed to detect, deter and eliminate DoS occurrences. However, none of them seems to be complete in assessing in both realtime and offline modes if a system remains free of such types of attacks. To this end, in the context of this paper, we assert that audit trails in VoIP can be a rich source of information toward flushing out DoS incidents and evaluating the security level of a given system. Specifically, we introduce a privacy-friendly service to assess whether or not a SIP service provider suffers a DoS by examining either the recorded audit trails (in a forensic-like manner) or the realtime traffic. Our solution relies solely on the already received network logistic files, making it simple, easy to deploy, and fully compatible with existing SIP installations. It also allows for the exchange of log files between different providers for cross-analysis or its submission to a single analysis center (as an outsourced service) in an opt-in basis. Through extensive evaluation involving both offline and online executions and a variety of DoS scenarios, it is argued

[*] Corresponding author
Email addresses: `tzisis@aegean.gr` (Zisis Tsiatsikas), `dimitrios.geneiatakis@jrc.ec.europa.eu` (Dimitris Geneiatakis), `gkamb@aegean.gr` (Georgios Kambourakis), `angelos@cs.columbia.edu` (Angelos D. Keromytis)

Preprint submitted to Journal of Computer Communications *February 8, 2015*

that our detection scheme is efficient enough, while its realtime operation introduces negligible overhead.

Keywords: Session Initiation Protocol, Entropy, Abnormal Traffic, Denial of Service, Anonymity.

1. Introduction

According to recent marketing analysis reports [1, 2], Voice over IP (VoIP) services are mushrooming on a daily basis. As in Public Switch Telephone Networks (PSTN), central role in VoIP communications plays a signaling protocol responsible for managing (establish, update, terminate) user sessions. Although various signaling protocols, including H.323 [3], SIP [4], Media Gateway Control Protocol (MGCP) [5] *etc.*, have been proposed, the Session Initiation Protocol (SIP) [4] seems to be the predominant one. This is because SIP inherits from the HTTP [6] model and structure, thus providing a high degree of freedom to develop easily new multimedia services and products.

Despite the advantages users enjoy due to SIP flexibility, various attacks have been identified against SIP-based VoIP services [7, 8, 9]. To alleviate, if not eliminate, such security flaws a diversity of detection and prevention solutions have been proposed in the literature [10, 11, 12, 13, 14]. However, while all these security mechanisms and countermeasures may be of considerable value, they do not capitalize on log files collected by the providers. So, it might be mistakenly taken for granted that the underlying services are secure, while in fact they are prone to security breaches, which have gone undetected. This especially applies to low-volume Denial of Service (DoS) attacks, which are lately on the rise and admittedly remain hard to detect and repel. In fact, the value of audit trail data in identifying security violations and flaws in applications has been highlighted by several researchers, security fora and organizations, including National Institute of Standards and Technology (NIST) [15].

On the downside, personal data contained in audit trails - and especially those stemming from the application layer as that of SIP - are subject to various legal restrictions and regulations. This fact alone makes the processing and exchange of audit trails among multimedia providers highly troublesome and problematic. This is because the exposure of sensitive personal information contained in audit trails to unauthorized entities is prone to several malicious acts that clearly violate the users' private sphere [16, 17, 18, 19]. For instance, an ill-motivated actor is able to learn the user's real identities and next eavesdrop on which services are being accessed by them, thus violating the principle of user anonymity [20, 21]. In the long term, when this kind of information is systematically gathered, the end-user can be profiled and sensitive information (e.g., preferred services) can be inferred. So, while audit trail analysis in VoIP ecosystems may be of great value, this needs to happen after a data-neutralization process takes place. This is necessary in order to obfuscate certain pieces of personal information contained in log files and preserve the privacy of the end-users.

Until now, various research works [22, 23, 24, 25, 26] have been dedicated to the identification of resource consumption attacks as a part of network Intrusion Detection Systems (IDS). However, as already pointed out, mainly for privacy reasons very few concentrate on the analysis of VoIP audit trails to identify and distinguish uncommon or suspicious traffic. Actually, a straightforward method to analyze audit trail data is to use entropy. For instance, the authors in [27] employ entropy theory to detect IP spoofing DoS attacks. This is done by monitoring the distribution of destination/source IP addresses of packets entering or leaving the network. Analogous methods can be utilized in VoIP ecosystem to analyse audit trails (or realtime traffic), but so far their scope is confined to the IP level only. On the other hand, data coming from the application layer are usually rich of information that can be processed towards identifying security incidents. The authors in [28] have identified this potential in theoretical level, but unfortunately the results they provide solely stem from offline analysis using predetermined patterns of SIP traffic.

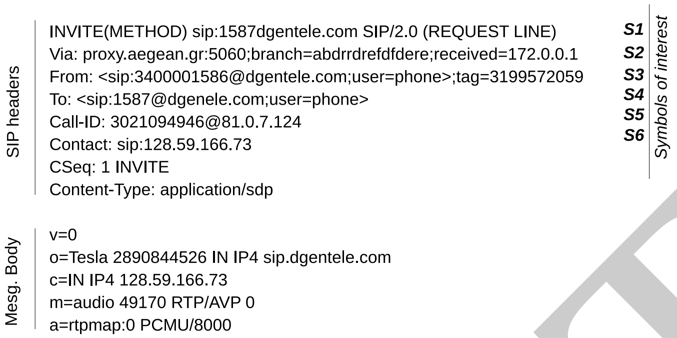
Contribution of this work: The work at hand builds over the results of [28], and details on an efficient and easily deployable method to analyze audit trail data from a security point of view in both realtime and offline fashion. On the top of that, our proposal enables VoIP providers to share their audit trails with trusted authorities in charge of analyzing its security status. This is possible because we mandate all data to be anonymized prior to being communicated between the different entities and get processed. In this respect, it is argued that our solution bridges the gap between the limitations of existing approaches to identify security flaws by examining the audit trails, while at the same time is orthogonal to the current defensive weaponry. Therefore, opposed to other works in the literature so far, our scheme not only is able to provide proofs of existing security flaws in a formal way as a public service, but also does so in a privacy-preserving manner from an end-user's viewpoint.

The main additional contributions of this work over that of [28] are:

- A threat model is introduced.
- A succinct analysis on the various anonymization techniques that can be of use with the log files is included.
- A new software module that enables realtime inspection of SIP messages is implemented.
- The efficiency of the proposed scheme is thoroughly evaluated in terms of detection rates for both offline and realtime operation. The performance of the latter in terms of service time (*i.e.,* the time needed to make a decision if an incoming message is malicious or not) is evaluated as well. This is done using several realistic scenarios involving a plethora of attack variations. So, in contrast to [28], the traffic used is not based on predetermined attack patterns, but follows a statistically fair distribution model for SIP calls.

Overall, the results show that our proposal is both privacy-preserving and practical. That is, it retains full compatibility with existing SIP deployments, is fast enough for realtime detection, and shows low to moderate rates of false positives and negatives upon execution.

The rest of the paper is structured as follows. The next two sections present background information with respect to SIP protocol and entropy theory correspondingly. The threat model is introduced in Section 4. Section 5 briefly addresses log file anonymization schemes with respect to our case, while Section 6 details on the proposed detection framework. Section 7 evaluates our solution in terms of effectiveness. The related work is discussed in Section 8. The last section concludes and provides pointers to future work.



```
INVITE(METHOD) sip:1587dgentele.com SIP/2.0 (REQUEST LINE)
Via: proxy.aegean.gr:5060;branch=abdrrdrefdfdere;received=172.0.0.1
From: <sip:3400001586@dgentele.com;user=phone>;tag=3199572059
To: <sip:1587@dgenele.com;user=phone>
Call-ID: 3021094946@81.0.7.124
Contact: sip:128.59.166.73
CSeq: 1 INVITE
Content-Type: application/sdp

v=0
o=Tesla 2890844526 IN IP4 sip.dgentele.com
c=IN IP4 128.59.166.73
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

SIP headers
Mesg. Body
S1
S2
S3
S4
S5
S6
Symbols of interest

Figure 1: A typical SIP INVITE request message

2. SIP-based VoIP Services

SIP [4] is an application-layer signaling protocol for creating, modifying, and terminating multimedia sessions among two or more participants. Actually, SIP is text-based with syntax similar to that of HTTP. SIP messages can be either a request or an acknowledgment to a corresponding request, consisting of the appropriate header fields and optionally a message body, depending on the nature of the request or response. An example of a typical SIP request message is given in Figure 1.

Whenever a user wishes to use a SIP service they should announce its presence by registering their current IP address to the registration service (registrar) through a SIP REGISTER message. After that, the user is able to initiate a session with another registered or interconnected User Agent (UA) by sending a SIP INVITE message to its local SIP proxy. After the call has been established, the two peers (the caller and callee) can start the multimedia session with the help of Real-time Transport Protocol (RTP) [29]. At any time, either the caller or the callee may terminate the call be sending a SIP BYE message toward the other end. These procedures are succinctly illustrated in Figure 2.

Note that this kind of network logistic data pertaining to SIP calls are kept by default by all VoIP providers in order to fulfill important tasks including billing, network management and planning, security assessment, etc. So, independently of the file for-

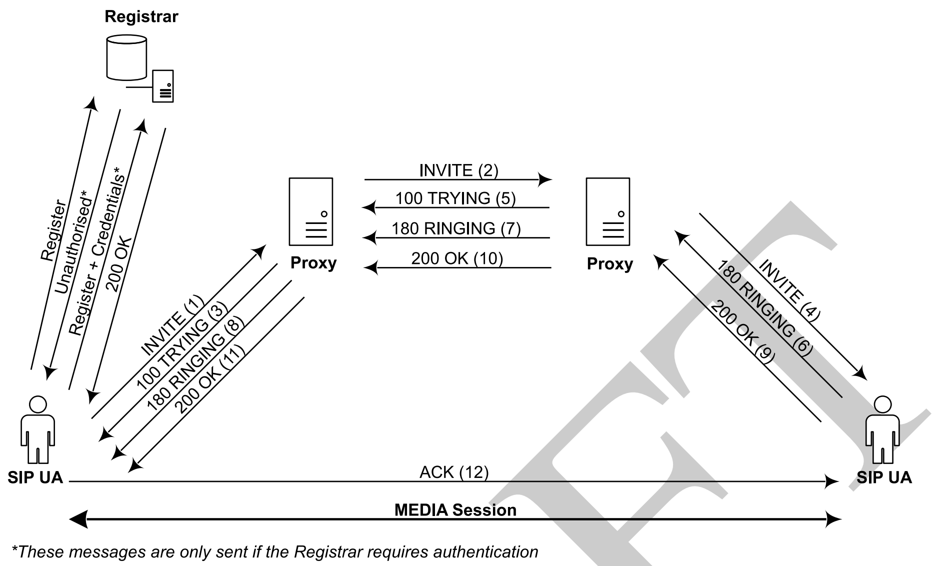


Figure 2: A simplified SIP message flow example

mat each provider uses to store them, these raw data are solely consisted of SIP requests and responses.

3. Use of entropy theory in SIP

3.1. Overview

Entropy is a metric of uncertainty based on the mathematical theory of communication [30] introduced by Shannon. Putting it another way, entropy quantifies the expected value of the information contained in a message. That is, a reduced uncertainty is quantified in a lower entropy and vice versa. Hence, the probability of occurrence (certainty of an outcome) of a symbol contained in a SIP message can provide one with knowledge about hidden redundancy in the information received.

Specifically, considering that a symbol S_i in a specific message set (M_{set}) has probability P_{S_i}, then the *itself information* included in this symbol is by definition:

$$I_{S_i} = -\log_b P_{S_i} \tag{1}$$

The average of *itself information* with reference to the message set (M_{set}) is called entropy and is computed using the following formula:

$$H(M_{set}) = -\sum_{i=1}^{n} P_{S_i} * \log_b P_{S_i} \tag{2}$$

The entropy of a source set M_{set} maximizes when all instances (e.g., messages) contained in that set have equal probability of occurrence ($P_{S_i} = 1/n$). This means that the uncertainty of the outcome is augmented, while the redundancy in M_{set} is reduced. With respect to *itself information* this fact indicates that all messages (or symbols corresponding to certain fields of a given message) contain the same amount of information. Note that the greater the probability of a specific message the less information is included in it. Furthermore, in case where two symbols are independent of each other, then the itself information and the entropy metrics are calculated using the formulas (3) and (4), respectively.

$$I(A, B) = I(A) + I(B) \tag{3}$$

$$H(A, B) = H(A) + H(B) \tag{4}$$

3.2. Symbol Definition

To apply the above mentioned principles of information theory in the context of a SIP auditing service, we define certain parts of a SIP message (headers) as the symbols of interest, as shown in the right side of Figure 1. The selection of these symbols reflects the different parts of a SIP message that an attacker could craft in order to launch a resource consumption or other type of attack. In fact, this method of assault is well-documented and evaluated in various researches so far [7, 8, 9, 31]. For instance, a malicious actor could fabricate different SIP messages by modifying some of their parts such as <Via>, <From>, <To>, <Call-ID> headers or even the First Line (corresponding to symbols S2 – S5, and S1 in Figure 1) depending on the situation at hand. The last two headers, namely <CSeq>, <Content-Type> shown in Figure 1 are left out because they bare minimum information regarding message entropy. That is, their values remain the same across different messages. For example, the latter header will always get the same values corresponding to the session ("application/sdp") parameters of the SIP phone in use. It is to be noted that excluding malicious SIP message tampering, replicated traffic can be also generated due to device misconfiguration or any other random cause. However, this situation is anticipated to happen mostly in small-scale, have short duration, and produce low-volume of extra traffic.

4. Threat Model

The formulation of a threat model in our case has to do with two types of adversaries; external and internal ones. The former category includes malicious entities trying to cause DoS or collect information about the service. Such adversaries will act from the perimeter of the network, meaning that they have no direct access to the resources of the service itself. On the other hand, internal adversaries are assumed to be honest-but-curious and reside either in the service provider or within a third party to whom the provider has outsourced one of its security-related services. Prior to explaining further, we make the hypothesis that the integrity of the log files is assured by the use of some well-accredited method [32, 33]. In fact, this requirement is a *sine qua non* for any service provider and it is also mandated by law in most countries worldwide [34, 35]. More specifically, the following assumptions are made:

Malicious external adversaries: The flexibility in message coding SIP offers can be of great advantage to an adversary when planning and executing, say, a flooding attack. Therefore, in this case, it is reasonable for one to assume a Dolev-Yao threat model [36] in which the adversary is among others able to eavesdrop, forge, and replay messages, and the only constraint is that of the cryptographic methods used. The latter, however, is not the case here as the tunneling of SIP traffic over, say, TLS or IPSec is not a widely-used practice, mainly due to the need of some sort of Public Key Infrastructure (PKI). So, for instance, the aggressor is able to launch a SIP INVITE or BYE flooding attack with the aim to paralyze the victim as reported in [31, 37] or execute a low-volume DoS to silently consume valuable network resources. This is for sure to gradually increase user discontent, which in turn leads to reducing provider's market share. Such type of assaults, especially the low-volume ones, may go totally unnoticed. In any case, however, the traces of the attack will remain hidden in the corresponding audit trails. Note that this kind of threats has to do with availability and integrity of the VoIP service itself, and do not focus on the (de)anonymization of log files.

Honest-but-curious third parties: While log files have special worth to multimedia providers for managing their network and billing purposes, they do not include only personal data but subscribers call history as well. Hence, due to the added value that such raw data have in terms of profit for different types of organizations, it can tempt any insider into gaining access to them. So, regarding the privacy preservation of the log files, we consider honest-but-curious (also known as semi-honest) third parties to which the service provider has outsourced the security analysis of its log files. Collaborating service providers who exchange log files in pursuit of shared goals also fall in this category. Insiders, that is, individuals working for the provider itself or a collaborating third party can also behave this way. This category of adversaries is supposed to have access to some version of the data and behave in a semi-honest manner. Namely, they might arbitrarily try to infer some additional information from the log files, but they obey the bilateral agreement in place as the case may be. Note that this category cannot be regarded as malicious because any insider attempt to corrupt the detection service is generally detectable if the assumption on the integrity of the log files holds. The capabilities of such an adversary are included in the following:

- They might learn which services are being accessed by the end-users of some provider by just observing the information contained in <From> and <To> headers. This information can be used towards profiling certain users. As already mentioned in the previous section such privacy breaches clearly violate the principle of user anonymity [38].

- They might copy (steal) log files with the intention to sell them to, say, advertising companies for profit.

- An adversary working for a given provider has access to the audit trails of another provider, and/or the employee of a public analysis (detection) service is able to snoop on records contained in the raw data of one or multiple VoIP providers. This is of significant importance as the confidentiality protection of audit trails is of matter to the VoIP provider itself. For example, many providers would be interested in hiding data about which their most popular (accessed) service is.

- Following the previous issue, an adversary working for the detection service or a provider can correlate pieces of data contained in log files of different providers in order to deduce more information about the end-users of interest.

From the previous analysis it becomes obvious that the sharing of network logistic data among providers and between providers and third-parties (for outsourcing purposes) remain highly questionable due to the type of data contained in them. As already pointed out, this is of utmost important here as the security analysis service may be outsourced to an external entity and/or some collaborating providers may share their data based on a common agreement. This situation becomes even more complex, assuming service providers operating in different countries or continents (mainly due to diverse legislation and legal requirements applying to each particular country). Cloud-based operation, which is currently on the ascend, adds one more dimension to worry about.

5. Anonymization of log files

The privacy and security requirements of the proposed scheme are strongly related to the robustness of the anonymization process conducted on the log files. Therefore, there is a need for a fast-performing solution able to cope with real-time detection, but also strong enough to deter de-anonymization attacks when offline analysis is performed.

In this context, various techniques have been proposed in the literature for anonymizing data that include private information [39, 40]. Since anonymization and privacy cannot be considered as binary properties for any system, each solution is suitable for employment to a specific architecture and context depending on the requirements at hand. In the following, a brief description of such approaches is offered showing their advantages and disadvantages pertaining to our case. Note however that a comprehensive analysis of all possible anonymization schemes for preserving the privacy of network log files remains out of scope of this work.

Randomization was introduced in works [41, 42] for defending against zero-day attacks. A similar approach can be followed for data anonymization. To do so, data are transformed via the use of a randomized function. However, the transformation pattern should be also known to the third party that wishes to analyze the transformed data.

Generalization [43] divides data (e.g., sensitive headers per SIP message) into Quasi-Identifier (QI) groups, and changes their QI-values into less explicit forms, in a way that data in the same QI-group are indistinguishable by their QI-value (corresponding to the number of SIP message in our case). The philosophy of this method has been embraced and evolved toward forming more advanced ones including anatomy [44], permutation anonymization [45] and others.

K-anonymity is a privacy preserving approach [46] that constitutes k records indistinguishable. In this scheme a set of k data records are k—anonymized if for any data record with a given set of attributes there are at least k-1 other records that match with those attributes. To achieve this, sensitive attributes are hidden in order to obstruct leakage of real identities.

Symmetric encryption can be used to provide not only confidentiality services, but also data anonymization [47]. Sensitive record's attributes are encrypted using a secret key. The output looks random, while decryption is computationally infeasible, unless you know secret key. Symmetric encryption schemes can be used either in a deterministic way, i.e., the same input produces the same output for the same key, or in a semantically secure mode, i.e., a publicly known input modifies the output per encryption for the same key.

Message Authentication Codes (MACs) are symmetric schemes that are used for data integrity protection. When the plaintext space is much smaller than the tags space, the tag can be used as pseudonym. Similarly to symmetric encryption schemes, MACs are keyed constructions that can be either deterministic or non-deterministic.

Hash functions are instances of one way functions. They are very efficient keyless schemes that are computationally difficult to invert. Secure hash functions possess several nice cryptographic properties, like pre-image, second pre-image and collision security.

Searchable encryption has gained a lot of attention the last few years. It enables keyword search on encrypted data [48] by employing either symmetric or asymmetric cryptography. The asymmetric schemes are based mainly on homomorphic encryption and functional encryption, and so far they add excessive overhead. On the other hand, symmetric solutions are more practical. They are based on a combination of data structures and symmetric encryption algorithms. More precisely, the user, for a specific collection of documents, creates a corresponding index of terms (keywords) with the help of which one can execute queries. This process has inherently at least linear preprocessing complexity on the number of files.

To sum up, although each of the aforementioned techniques could be employed to impose log files anonymity, the selected anonymization technique must fulfill the following requirements:

- Users' anonymity, including SIP URI, IP address, etc. must be preserved.
- Perform fast both in realtime and offline.
- The entropy of the original data after anonymization must remain intact.
- It must be computationally expensive to execute de-anonymization attacks.

Based on the above requirements the anonymization technique must be a property-preserving scheme and more precisely an entropy preserving one. While randomization and generalization kind of solutions perform fast, they produce negative effects on symbols frequency, thus affecting entropy. Furthermore, to perform any analysis on the anonymized data requires the transformation method to be exchanged between the peers. Consequently, these schemes can be regarded as more complex, and naturally vulnerable to attacks, as this additional information is required to be stored in a secure manner. For example, in [49] it is noted that anonymization can have severe undesirable outcomes if implemented incorrectly. This negative effect is discussed by the authors in [50], showing that the correlation of anonymous data with publicly available Internet movie database information made possible to reveal the real identities of many of its customers. In this line, K-anonymity and its variants can be used to hide sensitive data effectively, but affect also the entropy of the anonymized data. Finally, non-deterministic symmetric schemes (encyrption and MACs) are not entropy preserving as well. The outputs are based on some input randomness and for the same input different outputs are produced.

Thus, among the proposed solutions, we have to choose one of the deterministic algorithms, *i.e.,* hash functions, deterministic symmetric encryption schemes and deterministic MACs. All three of them are considered among the fastest cryptographic primitives and they are adequate for real time applications.

Regarding privacy protection, hash functions are the weakest ones. In our setting, the plaintext space is rather small. Thus, the keyless nature of hash functions make them vulnerable to brute force attacks. Let us assume an adversary who possess parts of the exchanged traffic (corresponding to a dictionary) and the anonymized data. Then, she is in position to execute a brute force attack aiming to match the dictionary records with the ciphertext and reveal, say, the URI of the end-users. To examine this possibility, we performed such an attack assuming that the adversary has built a dictionary of about 1 million records of different SIP INVITE requests for the same provider obtained, say, by eavesdropping. Using a laptop machine incorporating an i7 2.20 GHz processor and 6 GB RAM we managed to reveal a SHA-256 hashed header in 12.892 secs. On the other hand, the use of keyed solution, like a deterministic symmetric encryption or MAC algorithm, renders the application of a brute force infeasible.

In terms of performance, depending on the platform each technique excels. For instance, encrypting the following SIP request *INVITE sip:zisis@195.251.161.166 SIP/2.0* with AES-128 takes approximately 0.38 msec. For the same header, a keyed-hash message authentication code (HMAC)-SHA256 requires 0.29 msec, and a SHA256 digest 0.13 msec. We consider the mean value of 1,000 iterations executed on the same machine described above.

However, it is known that using deterministic algorithms makes the encrypted data vulnerable to frequency analysis.To minimize the effect of such an attack, we modify the parameters, *i.e.,* the secret key, of the MAC algorithm per outsorced log file. Thus, using the HMAC-SHA256 scheme parametrised with a specific secret key for a set of log entries, the identical headers will produce the same digest and the entropy will be preserved. For the next log file, a new randomly chosen key will be used for anonymization. In this way, an adversary is still able to process anonymous log records, but she cannot correlate different anonymised data sets.

Of special interest are searchable encryption schemes and especially, the symmetric ones. The last few years, several symmetric searchable encryption (SSE) algorithms have been proposed that have practical implementations [51]. However, the most efficient among them by design allow some leakage. More precisely, the leakage contain both the search pattern and the access pattern, *i.e.,*, as soon a keyword is queried, the server knows when the same keyword is accessed again. Thus, these schemes cannot protect from frequency analysis. In order to hide access pattern, SSE must use Oblivious RAM solutions, which are inadequate for real time applications, since they add polylogarithmic overhead (around $O(\log^2(N)$ per search, where N the number of data blocks) [52].

Summarizing, one needs to make a trade-off between performance and the level of anonymity offered. In this respect, the use of an HMAC scheme seems to be a fast-performing, well-respected and tested choice. Further analysis on the appropriateness of an HMAC scheme over that of a hash function in terms of server overhead pertaining to a real time detection service is given in section 7.4.

6. Classification of traffic

6.1. Overview

According to the entropy theory, symbol redundancy indicates lower entropy values. This means that symbols having greater frequency occurrence correspond to less “disorder” compared to others coexisting in the same set of messages. In the ideal case, an audit trail should not contain message redundancies, except those that take place due to retransmissions, subscribers peculiarities or habits (*e.g.,* a given user calls another very often), and/or other random causes as noted in section 3.2. Following this observation, we rely on entropy to quantify the level of disparity among messages belonging to the same audit trail as well as ones recorded in different log files, and determine whether they contain intrusive records or not. This is because entropy provides a single-quantity measurement, which corresponds to the randomness of a given audit trail. Note that this quality allow us to even compare audit trails belonging to different multimedia providers, which may be very handy when investigating large-scale distributed DoS (DDoS).

Figure 3 offers an overview of the proposed solution to identify DoS incidents in SIP networks. As observed from the Figure 3, all log files are anonymized before any processing can take place (1). Next, for each message contained in the audit trail, an entropy value is computed based on the headers of interest and compared against that derived from a reference (attack-free) file (2). If a predetermined threshold is exceeded, then the message is classified as malicious (3). Additionally, if one needs to obtain an estimate on whether the examined audit trail as a whole is suspicious, she can calculate its overall entropy and have it compared with the value corresponding to the reference file. This means that our scheme allows for auditing the security level of a given audit trail in both per message and per set basis.

It is therefore implied that before we are able to decide whether an audit trail contains a DoS, we require that one of the cross-evaluated audit trail sets is attack-free. Putting it another way, this set is used as a reference (training set) during the detection phase. Hence, the existence of such an attack-free audit trail along with the quality of the latter in terms of proper characterization of the service is of major importance. Actually, this is a well-known issue in intrusion detection, as semi-supervised anomaly detection techniques are able to detect anomalies after being trained by a dataset that has only the normal instances labeled. Generally, (semi)-supervised detectors rely on the existence of pre-labeled data to build their predictive models for both normal and anomalous classes during the training phase. Newly fed data instances are evaluated against this pre-constructed model, in deployment phase, so that they are categorized in normal or one of the intrusive classes. Realistically, data labeling is not always possible, either due to the high cost of the labeling process, or the sensitive nature of the data. Generally, it is much easier to construct a dataset with normal instances as in most situations normality is the rule. In the case of SIP the correct labeling (identification) of messages can be decisively seconded by the billing service, because these logs are supposed to be accurate and valid.

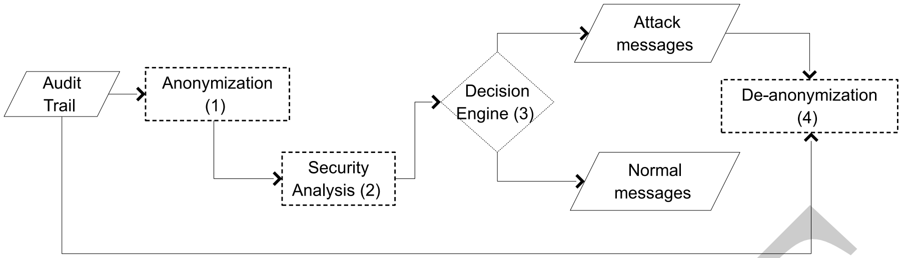


Figure 3: Abstract view of the proposed audit-trail analysis model

6.2. Metrics Definition

This section details on the metrics used by the proposed detection scheme.

Actual Information (AI): quantifies the randomness of a message in relation to all the other ones contained in the same set. Recall that in the proposed model, a SIP message is consisted of S1 to S6 symbols and the itself information of each one is calculated using equation (1). So, we compute the randomness of each individual message using formula (5).

$$AI(M) = \sum_{i=1}^{n} I_{S_i} \tag{5}$$

Theoretical Maximum (TM): defines the theoretical maximum randomness value that a message can hold vis-a-vis a particular set of messages (M_{set}). This value is computed using formula (6), where k is the number of symbols defined (6 in our case) and n is the total number of messages existing in that set (M_{set}). Keep in mind that in the case of SIP ACK and BYE messages, the corresponding number of symbols is 5 due to the absence of the <Contact> header.

$$TM = -k * \log_b 1/n \tag{6}$$

Normal Average Distance (NAD): represents the average randomness distance of an attack-free traffic from its theoretical maximum value. Its value is computed using formula (7).

$$NAD = Avg(TM - AI) \tag{7}$$

Actual Information Distance (AID): measures the distance of an examined audit trail message from its theoretical maximum. Its value is computed by formula (8).

$$AID = TM - AI \tag{8}$$

Normal Threshold (NT): defines the threshold that should not be exceeded by the AID of an examined message in order this message to be classified as normal. The NT value relies on NAD adjusted by a parameter δ, which in turn relies on the characteristics of the examined traffic, *i.e.,* NT = NAD + δ.

Audit Trail Entropy (ATE): this metric represents the overall randomness included in an audit trail. It is based on the sum of AI values of all the messages contained in a given M_{set}, and it is computed using formula (9).

$$ATE(M_{set}) = \sum_{i=1}^{n} H_{S_i} \tag{9}$$

6.3. Example

To exemplify the above metrics, in Table 1 we present a simple artificially-created audit trail consisted of 10 messages with different symbols. For instance, the M1 message is comprised of symbols A1, E1, I1, K1, Q1 and U1 corresponding to some of SIP headers existing in the left side of Figure 1. In this case, the itself information for A1, as calculated by equation (1), is 3.32. Also, by using formula (5), the AI for M1 is $3.32 + 1 + 0.73 + 0.15 + 1.73 + 2.32 = 9.25$. If we consider this audit trail to be attack-free, and considering that TM is $-6 * \log_2(0.16) = 15.84$, then NAD will be $(15.84 - 9.25)/10 = 0.65$ according to equation (7). Moreover, the entropy values for each symbol S1 to S6 are $3.32, 1, 1.77, 0.46, 1.89$ and 1.92 respectively, while the entropy calculated over the whole set of messages contained in the audit trail is 10.36. The lower the ATE value is, the less the randomness contained in the examined set.

So, it is straightforward that whenever one wishes to identify whether a message contained in any given audit trail is malicious or not, requires her to compute its AI and AID (subsequently) and then compare the latter against the NT computed over the attack-free audit trail.

Table 1: Example based on artificial audit trail

Msg.	**Symbols**						**AI**
	S1	**S2**	**S3**	**S4**	**S5**	**S6**	–
M1	A1	E1	I1	K1	Q1	U1	9.25
M2	A2	E2	I1	K1	Q2	U2	9.25
M3	A3	E1	I1	K1	Q3	U3	9.25
M4	A4	E2	I1	K1	Q1	U4	8.25
M5	A5	E1	I1	K1	Q2	U4	8.25
M6	A6	E2	I1	K1	Q3	U4	8.25
M7	A7	E1	I2	K1	Q1	U4	8.25
M8	A8	E2	I3	K1	Q2	U1	11.84
M9	A9	E1	I4	K1	Q3	U2	11.84
M10	A10	E2	I5	K2	Q4	U3	16.6
ATE	3.32	1	1.77	0.46	1.89	1.92	–

6.4. The proposed scheme

Initially, the data contained in the audit trail files are anonymized to obfuscate user specific information contained in SIP headers. As already pointed out in section 5, this is done by creating a keyed hash [53] (HMAC) value for each header of interest. Bear in mind that this information corresponds to the symbols S1 to S6 defined in Figure 1. This (keyed) hashing phase allows one to retain symbols frequency while obscuring the initial values. That is, due to the intrinsic properties of HMAC functions, the chances for someone to reverse the hash is almost non-existent, while the output produced allows for exchanging the data among providers or between a provider and a public data analysis service. Also, it should be noted that even if the key (k) is compromised the original values (x) cannot be reversed engineered from $HMAC(k, x) = value$. However, such a situation will constitute the data vulnerable to brute force attacks. This observation also applies to cases where the access to the original data and the secret key used to create the corresponding HMAC is managed by different entities.

Thus, as shown in Figure 3, in case there is a need to identify each one message classified as malicious by the framework, the service provider searches for its hash in the already anonymized audit trail. If a match is found, then the initial message can be retrieved ~~in cleartext~~ from the original data for further examination. Of course, another possibility is for the detection framework to send back to the service provider only the serial number of the messages detected to be suspicious. This way, we can publicize information and outsource data for security related analysis that otherwise would remain private.

Recall from section 5, however, that while this HMAC-powered anonymization scheme is very fast, it is only fair when it comes to unlinkability [54]. That is, the hash values of two headers pertaining to the same user, say, the caller in an INVITE request, will be identical. Therefore, a person having access to the anonymized data is able to deduce certain relationships among the data, even if the real identities of the corresponding persons remain hidden.

After the obfuscation phase, the randomness included in the audit trail of interest is quantified. Having in mind the metrics defined in section 6.2, we calculate (a) the AI per message, (b) the AID for the examined set, and (c) the ATE over the whole set of messages contained in the audit trail file. As already pointed out, to identify abnormalities (which may denote a DoS attack) it is assumed that there exists at least one attack-free audit trail, to be used as a training set for calculating the NAD and (consequently) the NT metric. The latter value is used to decide if a given message is intrusive or not. This procedure is presented in detail in pseudocode in Appendix I, Algorithm 1.

7. Effectiveness & Performance Evaluation

7.1. Implementation

The proposed detection scheme was implemented as two independent software modules; one for contacting offline analysis of audit trail data, and the other for inspecting SIP messages in realtime. Both modules were programmed in C language and are capable of processing any type of SIP message either request or response. The realtime module is built as an extension of the well-known SIP server Kamailio [55].

For both modules, before doing any security analysis, the data are anonymized through the HMAC-SHA256 function [56] and stored in a hash table. In fact, the hash table structure stores a different record per unique SIP symbol found in the examined messages. The SIP symbol is the alias-key which is used to retrieve the corresponding value from the hash table.

As detailed in Section 7.4 and summarized in Algorithm 1, in the case of realtime detection, the data remain in the hash table for a predefined message-window M_w. As soon as the M_w expires, the module calculates the NT, NAD metrics over the received messages up to that moment and compares the AID of every next incoming message against that of NAD.

For offline analysis the exact same metrics are calculated, but this time over the whole traffic corresponding to the audit trail at hand. This procedure is summarized in Algorithms 2, 3. However, if one needs to obtain a security assessment of the messages received during a specific period of time, the audit trail can be split into segments over which the metrics can be calculated. By doing so, one is able to achieve a similar approach to the M_w used in realtime analysis. Algorithm 5 exemplifies this procedure.

7.2. Test-bed setup

To evaluate the effectiveness of the proposed scheme in detecting abnormalities in SIP traffic we used a properly designed testbed as shown in Figure 4. To simulate different types of both legitimate and attack (flood) traffic we employed *sipp v.3.2*[1] and *sipsak*[2] tools respectively. Table 2 summarizes all the scenarios used for the evaluation of our scheme. The main scenarios, namely SN-1 to SN-5, serve as references for attack-free traffic and therefore are used for calculating the base-value of NAD and subsequently the NT metric. The latter will be compared against that of AID calculated per message for every sub-scenario. To simulate legitimate traffic in terms of incoming calls in a realistic manner we employed an exponential inter-arrival time distribution (λ = 100), similar to that used in evaluating SIP server performance [57].

[1] `http://sipp.sourceforge.net/`

[2] `http://sipsak.org/`

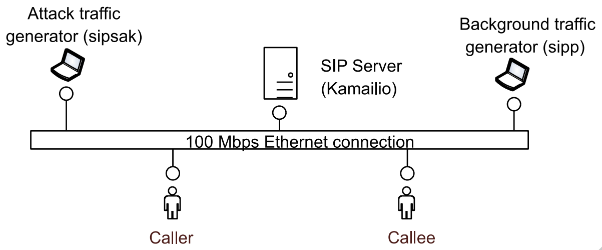

Attack traffic generator (sipsak)
SIP Server (Kamailio)
Background traffic generator (sipp)
100 Mbps Ethernet connection
Caller
Callee

Figure 4: Deployed test-bed

Apart from assessing the effectiveness of the modules to detect suspicious traffic, we also measure the introduced overhead of the one destined to realtime operation. For the latter, the SIP server has been configured to operate in single-thread mode, representing a worst case scenario. The server was running on an i7 2.2 GHz machine having 6 GB of RAM.

7.3. *Off-line Analysis*

Figure 5 illustrates a snapshot of the AI metric distribution for the main scenarios SN-1 and SN-2. It can be observed that this metric follows a nearly similar pattern for both scenarios. Analogous distributions are perceived for the remaining attack-free scenarios as well. In fact, this slight difference is due to the disparate rate in calls per second each scenario incorporates. This call distribution represents the calls initiated by legitimate users in a typical SIP ecosystem as explained in [58, 59]. Contrary to that, as observed from Figure 7.3, the AI produced in attack scenarios SN-1-2 and SN-2-2 exhibits high fluctuations. The calculated statistical results for all the scenarios are summarized in Table 3.

It should be noted that in all the attack sub-scenarios the AI obtains greater values due to the higher number of messages existing in the corresponding audit trails. However, when an attack unfolds, the AI for messages belonging to the attack traffic receives lower values. Thus, the NAD metric needs to be adjusted according to the parameter δ (see section 6.2) for calibrating it to the current traffic pattern. In our case, the δ parameter is equal to the *Standard Deviation* value calculated over the messages

Table 2: Description of scenarios

Scenario Number	Description
SN-1	This attack-free scenario simulates 30 legitimate users establishing 5 calls/sec.
SN-1-1, SN-1-2, SN-1-3	These sub-scenarios use the background traffic of SN-1 and simulate a single source SIP INVITE flood attack with a rate of 20, 40, 80 calls/sec respectively.
SN-2	It simulates 30 legitimate users establishing 2 calls/sec. This is an attack-free scenario as well.
SN-2-1, SN-2-2, SN-2-3	These sub-scenarios use the background traffic of SN-2 and simulate multiple sources of SIP INVITE flood attack with rates of 50, 175, 350 calls/sec respectively.
SN-3	It simulates 50 legitimate users establishing 120 calls/sec. This scenario contains no attack traffic.
SN-3-1, SN-3-2	These sub-scenarios employ the background traffic of SN-3 and simulate a single source SIP INVITE flood attack of 400, 1200 calls/sec respectively.
SN-4	This attack-free scenario incorporates 50 legitimate users establishing 120 calls/sec.
SN-4-1	It relies on the background traffic of SN-4 and simulates 24 single source SIP INVITE floods each one with 800 calls/sec.
SN-5	This last attack-free scenario incorporates 50 legitimate users establishing 20 calls/sec.
SN-5-1	It relies on background traffic of SN-5 and simulates 16 single source SIP INVITE floods each one with 266 calls/sec.

that consist the corresponding normal traffic set. This means that in order to deduce if a particular message is part of an uncommon traffic stream, we compute its AID (according to equation 8) and compare it against the NT value. If the latter is exceeded, the message is characterized as suspicious.

Generally, the existence of excessive symbol repetition in any given audit trail is a

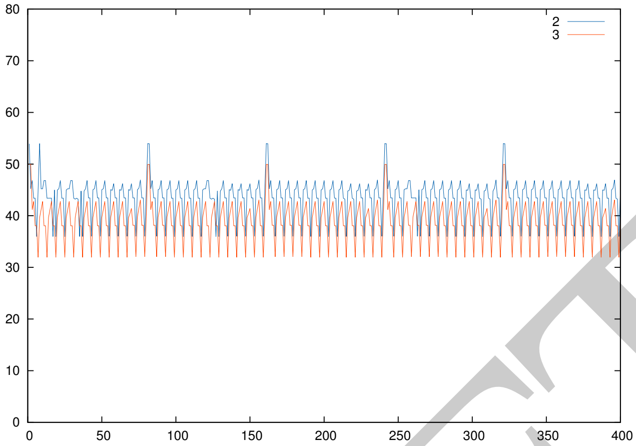


Figure 5: A snapshot of the AI for scenarios SN-1, SN-2 (normal traffic only)

high indication of uncommon user behavior because legitimate users are not capable of generating high volumes of traffic in a short period of time, unless, for example, their device is infected by a malware. In opposite to that, low values in symbol (S_i) recurrence is a strong indication of normal traffic, and thus it can be used in cases where an attack-free audit trail is not available. However, in such an unusual case, one should take into account either the theoretical users' behavior or employ other techniques (as in [60, 58]) aiming to estimate the appropriate threshold.

To evaluate the accuracy of this detection module in identifying DoS attacks we use legacy IDS error assessment metrics, namely False Positive (FP) and False Negative (FN) [61]. The first one is related with messages detected as abnormal but they belong to the legitimate traffic, while the latter involves messages detected as normal but they belong to abnormal traffic. To further validate the outcomes at a later stage, we mandate all the attack traffic to be recorded in a separate log file.

The FP and FN results for all the scenarios are summarized in Table 4. It is observed that the FP percentage varies from 0.3% to 10.6%, while that of FN reaches 1.8%. For instance, in SN-1-1 to SN-1-3 the FP fluctuates between 7.5% and 10.6%. This is because aggressors usually do not solely rely on random messages to cause DoS, but employ smart spoofing schemes where the attack messages are crafted to contain legiti-

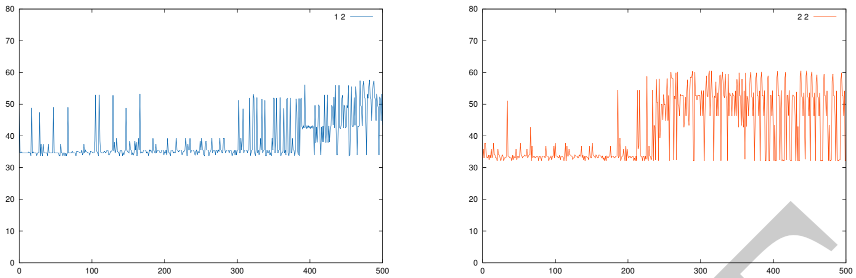


Figure 6: A (random) snapshot of the AI for scenario SN-1-2 (left) and SN-2-2 (right)

mate SIP addresses/headers. So, some of these specially manipulated ~~crafted~~ messages would remain hidden under the normal traffic, while other pertaining to normal traffic are being classified as malicious, generating FP alarms. In this respect, it can be said that such an FP is rather expected since it corresponds to spoofing attacks which is very difficult to defend at least in their early stages or in cases the attacker uses "low and slow" attack techniques. On the other hand, the FN percentage is almost negligible, around 2% in the worst case. This result is particularly encouraging as it makes harder for an attack to go totally undetected.

It is also worth noting that if we solely consider the whole message as an independent symbol, then the AI obtains the maximum theoretical TM value. This happens because every SIP message, either legitimate or not, always presents some additional fields or parameters that uniquely differentiate it from any other. Obviously, if doing so, one will end-up believing that the audit trail under investigation is attack-free. Thus, our model makes use of the AI metric which involves information stemming from different, but clearly defined, symbols of the SIP message structure. Of course, more SIP headers can be added or removed depending on the case.

7.4. Real-Time Analysis

This section reports on the results obtained from the realtime detection module running on Kamailio SIP server. The evaluation scenarios remain the same as in the case of offline analysis in an effort not only to assess the accuracy of the realtime module but

Table 3: Offline analysis metrics

Scenario	**TM**	**St. Deviation (δ)**	**Threshold**	**AI mean value**
SN-1	69.94	3.83	33.3	43.47
SN-1-1	80.75	7.14	43.94	43.95
SN-1-2	81.85	7.01	44.12	44.74
SN-1-3	83.80	6.60	44.92	45.48
SN-2	61.93	3.83	26.87	38.89
SN-2-1	79.67	7.08	45.43	41.32
SN-2-2	83.43	6.83	47.86	42.40
SN-2-3	86.71	5.32	47.18	44.85
SN-3	96.88	4.02	45.88	55.02
SN-3-1	108.92	5.16	56	58.08
SN-3-2	117.15	4.48	61.65	59.98
SN-4	99.80	3.85	46.87	56.78
SN-4-1	117.23	4.95	61.2	60.98
SN-5	89.73	4.09	44.23	49.58
SN-5-1	105.51	4.77	55.31	54.97

also to cross-evaluate their outcomes. It is to be noted that the main difference between the realtime and offline modules is the amount of traffic that is available to the realtime module at a given time. This is because the offline module works based on statistical metrics derived from all the available traffic, which naturally is not the case for the realtime one. Thus, as already pointed out, a M_w is used as a training phase to provide an estimation of NT and NAD metrics. This procedure is illustrated in Figure 7. Of course, these metrics can be automatically readjusted as further traffic is available, similar to the offline approach. However, the adjustment of TM and NAD metrics needs to take into account the M_w parameter as well. This also means that the selection of M_w is critical to the accuracy of realtime detection. To our knowledge, there is no direct approach to formally define these parameters, mainly because they are highly contextual, *i.e.,*, closely bound to the characteristics of the service and underlying network.

Table 4: False alarm ratio (offline analysis)

Scenario	**Traffic**		**FP**		**FN**	
	Total calls	**Attack calls**	**Instances**	**%**	**Instances**	**%**
SN-1	3,230	0	0	0	0	0
SN-1-1	11,263	7,204	1,195	10.6%	0	0
SN-1-2	12,790	8,523	1,132	8.8%	0	0
SN-1-3	16,019	11,763	1,203	7.5%	0	0
SN-2	1280	0	0	0	0	0
SN-2-1	9,936	8,291	391	3.9%	0	0
SN-2-2	15,340	13,657	458	2.9%	0	0
SN-2-3	22,420	20,799	496	2.2%	0	0
SN-3	72,553	0	0	0	0	0
SN-3-1	291,634	228,432	3,104	1.0%	4,111	1.4%
SN-3-2	754,510	705,936	12,182	1.6%	7,274	0.9%
SN-4	101,762	0	0	0	0	0
SN-4-1	761,963	694,788	2,479	0.3%	6,873	0.9%
SN-5	31,775	0	0	0	0	0
SN-5-1	196,761	168,470	5,835	2.9%	3,696	1.8%

Therefore, similar to other anomaly-based approaches [62], we adopted an error-trial approach to balance between the Mw parameter and the false alarm rate.

In a nutshell, as the M_w expires, the module starts to monitor every incoming SIP message to decide if it is normal or not. This is done by computing its AID according to equation (8) and comparing it against the NT value.

Table 5 overviews the average processing time per message introduced in SIP server as well as other statistical metrics (min, max, st.dev) for realtime operation for all the scenarios. Note that the table contains the overhead induced when a SHA256 or HMAC-SHA256 operation is involved for data anonymization. This is done to acquire a cleaner view of the applicability of each method in terms of performance. As illustrated in the table, for the HMAC-SHA256 case, this time penalty remains low at

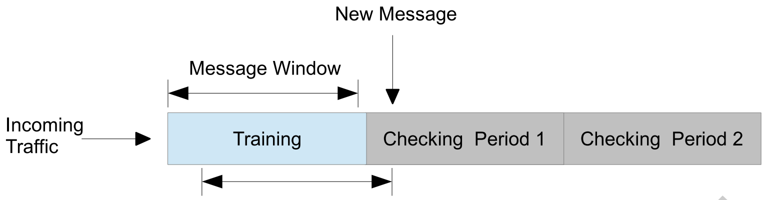


Figure 7: Abstract view of the *Message window (M_w)*

about 3.9 and 1.8 ms considering the first two attack-free scenarios, namely SN1, SN2. This is expected as these scenarios have the lowest call rate of 5, 2 calls per second respectively.

However, as the volume and the arrival rate of messages increases, the introduced overhead is expected to augment as well. For example, the average overhead in SN-4 is ≈16.3 ms, while for SN-5 is ≈14.9 ms. Even in the worst case scenario of SN-3-2, the mean value of this metric is ≈16.8 ms. This observation is further supported by the standard deviation per scenario, which fluctuates between 2.8 and 9.7 ms. Therefore, it can be safely argued that the use of HMAC-SHA256 does not generate a significant increase in overhead. So, as already discussed in section 5, this stronger anonymization method is definitely to be preferred over that of a hash function.

To assess the effectiveness of the realtime module in terms of false alarms we use different values for the M_w, ranging from 100 to 1000 messages. This will allow for fine-tuning of the M_w parameter based on the recorded FP, FN metrics, and provide a general estimation on how the M_w affects these metrics. Figure 8 illustrates the variation of FP for all the scenarios. Taking SN-1-1 to SN-1-3 as examples, no FP is observed when the M_w varies from 100 to 1000 messages. However, for scenarios SN-2-1 and SN-2-2, when the M_w is set to 1000 messages, a rather negligible FP of ≈1.9% is perceived. A similar FP distribution is recorded for scenarios SN-4-1 and SN-5-1, showing in all cases an FP percentage below 2%.

It is also important to note that while in the case of offline analysis the FP was approximately 10%, here is much smaller (≈2%). This betterment is because the realtime module is able to detect bursts of DoS traffic taking place within the predefined M_w (due to the decrease of the AI that in turn triggers the threshold). This includes spoof-

Table 5: SIP server overhead (in ms)

Scenario	SHA256 / HMAC-SHA256			
	Min	**Max**	**Average**	**St. Deviation**
SN-1	0.440 / 0.423	36.079 / 44.711	3.812 / 3.899	6.929 / 7.274
SN-1-1	0.325 / 0.413	40.869 / 40.800	15.354 / 15.700	9.311 / 9.251
SN-1-2	0.330 / 0.295	48.262 / 43.710	15.201 / 15.627	9.274 / 9.156
SN-1-3	0.338 / 0.312	43.873 / 43.324	15.658 / 15.951	9.186 / 9.383
SN-2	0.464 / 0.422	32.233 / 28.354	2.124 / 1.837	2.731 / 2.867
SN-2-1	0.396 / 0.342	59.128 / 65.678	15.113 / 15.426	9.290 / 9.429
SN-2-2	0.395 / 0.449	44.651 / 51.038	15.315 / 15.328	9.384 / 9.303
SN-2-3	0.395 / 0.344	178.276 / 58.846	15.476 / 15.510	10.645 / 9.745
SN-3	0.346 / 0.373	45.720 / 124.347	15.874 / 16.028	8.385 / 8.765
SN-3-1	0.308 / 0.276	72.268 / 68.205	16.601 / 16.597	8.537 / 8.678
SN-3-2	0.295 / 0.349	60.860 / 62.937	16.806 / 16.841	8.365 / 8.516
SN-4	0.295 / 0.330	155.355 / 511.134	16.167 / 16.256	8.849 / 8.173
SN-4-1	0.209 / 0.440	300.404 / 56.473	16.511 / 16.744	8.560 / 8.067
SN-5	0.307 / 0.291	36.196 / 47.506	14.734 / 14.894	8.236 / 8.383
SN-5-1	0.319 / 0.320	55.762 / 65.981	15.534 / 15.668	8.722 / 9.090

ing attacks exploiting IP addresses belonging to legitimate users. Contrary to that, the offline module cannot detect such cases as the attacks are spread along the entire audit trail. In any case, as mentioned previously, if the audit trail is split into multiple data segments one can achieve similar results for the offline module as well. Nevertheless, in that case, the threshold needs to be adjusted to the appropriate segment following a normalization procedure.

Figures 9 and 10 offer an overview of FN percentage under different configurations of the M_w parameter. As illustrated in the Figure 9 when the M_w is configured to a low value it produces high FN percentages in almost every scenario. For instance, in SN-2-* sub-scenarios as the M_w augments from 100 to 1000 messages causes the FN to be decreased rapidly. Particularly, in these sub-scenarios the FN is decreased

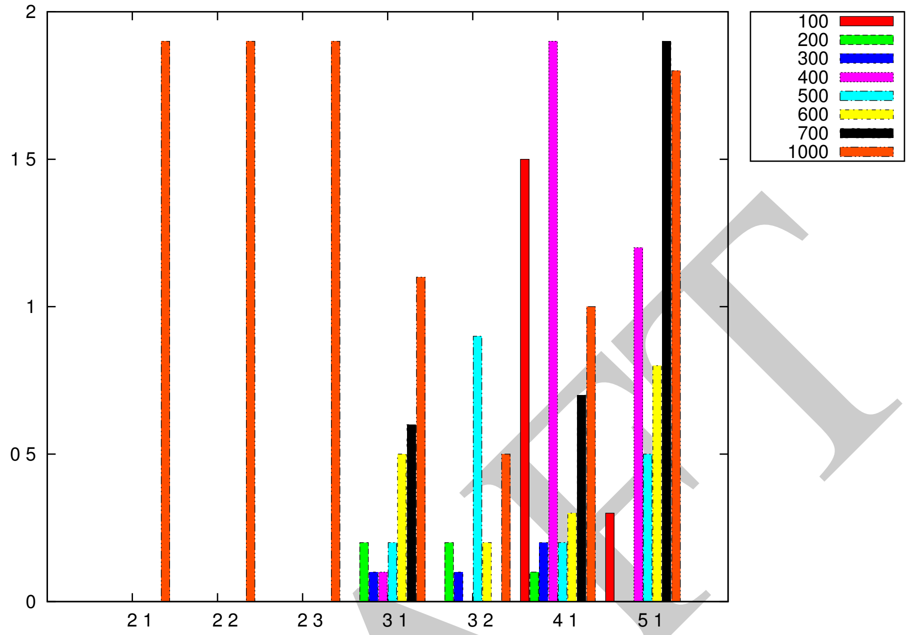


Figure 8: FP distribution under different M_w (all scenarios)

highly from almost 30% down to ≈2% as the M_w increases. In SN-1-*, where the legitimate traffic is slightly increased, compared to that of SN-2-*, the FN mean value is ≈5.5% for all the variations of M_w. Note that as the legitimate traffic accumulates, e.g., in SN-3-*, SN-4-* and SN-5-*, the FN is also increased considering the same values of M_w. This behavior is actually expected because the detection engine makes decisions based on a limited volume of incoming traffic. To further verify this result, we experimented with sizes of M_w greater than 1000 messages. It is therefore not surprising that for scenario SN-4-1 when the M_w is set to 4000 messages, an FN of 3.7% is generated (an improvement of 10.6%). Overall, for all the scenarios included in Figure 10, when the M_w is equal to 1000 we perceive a minimum and maximum FN percentage of 4.8% and 24.5% respectively. On the other hand, when the M_w is increased to 2000, the corresponding values drop to 4.2% and 19.1% respectively. Generally, the results designate that if the M_w is adjusted to the corresponding normal traffic, then the generated FP and (especially) FN alarms can be greatly optimized.

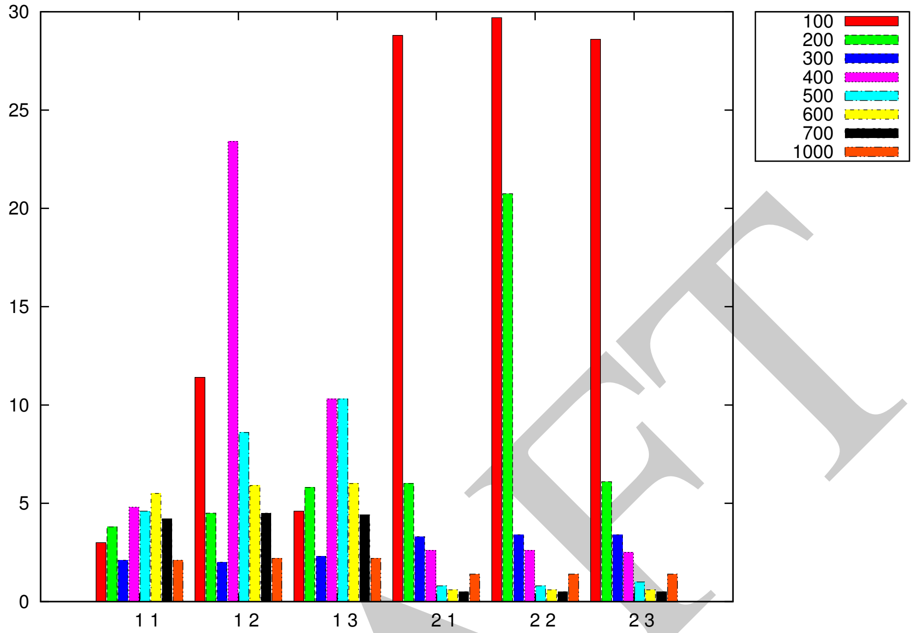


Figure 9: FN distribution under different M_w (scenarios SN-1-* to SN-2-*)

When comparing the FN values outputted by this module against those of the offline one, we can make the following remarks: (a) the minimum value in both cases is very low at about 1%, (b) the maximum values however present a significant difference of ≈27% (29-1.8), thus further verifying the above mentioned rule "the larger the M_w the better the FN, (c) elaborating on the latter point, from the results obtained, offline detection seems to be far more robust due to the great mass of information available; on the downside, the online detection module is easily adjustable, performs sufficiently well, and comes very handy in cases where a timely reaction is desired.

8. Related Work

Audit trails analysis is a well-known method used in different realms for identifying a variety of problems such as misconfiguration, frauds, etc. [63]. For instance, an audit trail analysis combining data from different network components have been introduced in [64] to detect performance issues in the provided services. In other

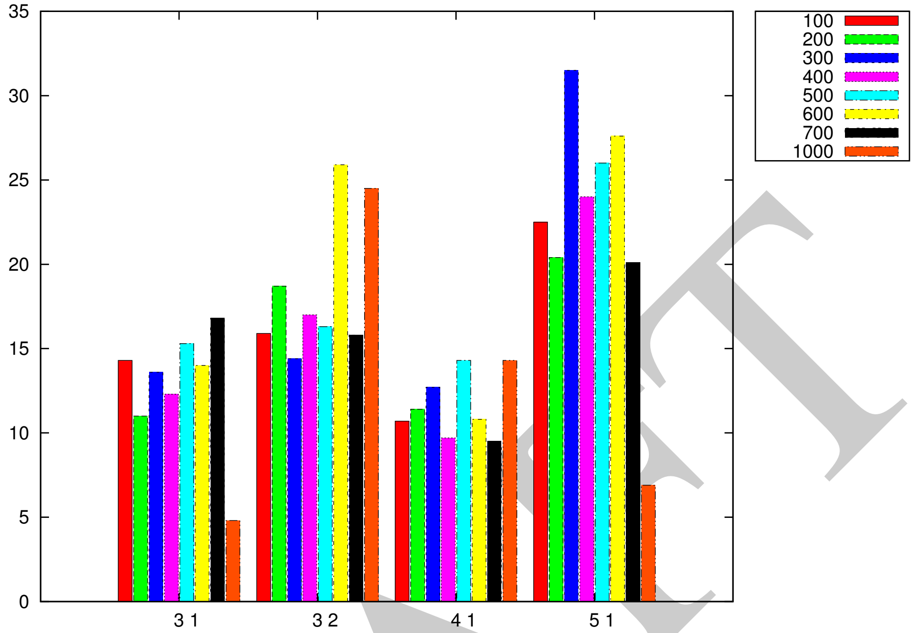


Figure 10: FN distribution under different M_w (scenarios SN-3-* to SN-5-*)

cases, audit trails have been used to detect attack incidents [65]. In the context of the current work emphasis is given to contributions dealing with DoS attacks and especially those capitalizing on entropy information stemming from network logistic data. Hence, other works on the general filed of intrusion detection in SIP and related protocols [66, 67, 12, 68, 69] remain out-of-scope. The interested reader can also refer to [70] for a comprehensive review on DoS attacks in general and SIP in particular [7].

The authors in [27] propose a system that analyzes the level of entropy in the distribution of source and destination IP addresses toward protecting IP services against spoofing attacks. According to the authors, all active (TCP/UDP) sessions are examined if they follow the normal entropy distribution. In case a violation of the normal pattern is detected the session is dropped. In more detail, the authors point out that after identifying upper and lower entropy thresholds for each link under normal conditions, and comparing them with current source and destination entropy values, different flavors of DoS incidents can be identified. The authors in [71] propose a solution for protecting web services against distributed DoS. Their scheme is based on attack-tree model and utilizes entropy to identify anomalies. Initially, an attack-tree is built to obtain an abstraction of the router-level Internet graph. Next, for each router, the entropy is calculated having as input immediate packet flows. Finally, an alert is raised every time the entropy falls below a threshold.

Lately, the importance of audit trails in security analysis has motivated researchers to propose various methods of log anonymization and security analysis [72, 73, 74, 75, 76, 77]. This is in contrast to our solution which preserves anonymity. For instance, the work in [72] introduces a data anonymization method as a way to surpass legal restrictions in audit trail analysis. This is achieved by replacing all identifiers existing in transaction messages by equivalent pseudonyms. The latter are constructed using Shamir's cryptographic approach of secret sharing [78]. Moreover, the work in [75] introduces practical tools that can be used toward the pseudonymization of log files in Unix systems. The authors in [77] conduct a survey of current research attempts on sharing log files and log anonymization tools. They elaborate on the problem and present a detailed road-map to cope with the issues germane to large-scale log sharing. The authors in [73] propose a scheme for collaborative security analysis which is based on two main entities, namely threat analysis centres and alert repositories. In their analysis, they identify the lack of privacy for actors contributing alert data in security repositories. This happens because the sensitive nature of the information which is encapsulated in alerts requires the sanitization of the alert data. Motivated by this fact, they propose a set of data sanitization techniques that facilitate community alert aggregation and correlation, but also maintain privacy for those willing to contribute alert data. Under the same context, the authors in [74] present a hierarchy-based solution for protecting private information contained in alert data in an effort to balance between privacy requirements and the need for intrusion analysis. Specifically, their approach consists of two phases. The first one is an entropy-driven alert sanitization, while the second is coined sanitized alert correlation. The authors concentrate their efforts on forming similarity functions between sanitized attributes and constructing attack scenarios from sanitized alerts. The work in [76] deals with the challenging issue of security analysis in virtual organizations, and argues that local intrusion detection and audit practices are incapable of detecting and repelling distributed attacks. The authors correctly identify that a distributed approach able to coordinate information inputted by each member is probably more suited for these environments, but privacy concerns often hamper this significant security information exchange process between the members. In this direction, they propose a privacy-preserving framework to facilitate distributed audit, without requiring the release of excessive private information among members.

Recently, the authors in [62] proposed an interesting SIP flood detection scheme based on the sketch data structure [79] to monitor and record specific features of SIP traffic, and the Hellinger distance to detect DoS in realtime. The authors assume that the attacker cannot deduce the normal sketch, which is stored securely on the SIP server after a training period. However, the attacker is in position to monitor the traffic during the training period and identify the normal sketch. So, naturally in this case, she is able to bypass the detection mechanism. To eliminate such a possibility the authors suggest additional modifications in the SIP client to be exercised during the registration phase in order to hide the normal sketch distribution. On the other hand, these amendments involve certain implications with reference to already deployed SIP ecosystems. So, while this contribution is perhaps the most relevant work to ours, it presents two major differences. First off, the work at hand is able to operate in dual-mode; both offline and online. Secondly, our scheme does not require any modifications to the SIP protocol but instead relies on information already bore by any SIP message. Also, in this point it should be mentioned that while the results of [62] are not directly comparable to ours (as the detection methods employed are different in nature), an effort is made to extract some useful observations that can be used to match the recorded results against those of the current work. First off, [62] considers an attack-free call generating rate that is distributed from 25 to 75 per second, having a mean of 50 per second. As described in Table 2, in the current work, the call rate varies between 2 and 120, depending on the number of users employed in every basic scenario. Similar differences are spotted in the call rate and number of users used to represent attack scenarios between the two works. Having the aforementioned differences in mind, in [62] one can observe an attack detection probability of 100% for sketch-based detection and a false alarm percentage that fluctuates from zero to nearly 8% or more for DDoS detection depending on the configuration. Lastly, the computational cost in terms of CPU time induced by [62] seems to be significantly higher than that of our solution, as it is reported to be 632.3 and 820.6 ms for the two scenarios tested.

A last work focusing on the analysis of security incidents and audit trails in converged networks is given in [80]. More specifically, the authors rely on a predetermine attack pattern to identify malicious activity. This is done by combining information stemming from multiple sources. Unfortunately, the analysis conducted by the authors is confined mostly to theoretical level and the results provided in terms of detection accuracy are based on predetermined attack patterns. Last but not least, their scheme only considers offline processing leaving realtime detection for future work.

9. Conclusions And Future Work

The proliferation of VoIP services, and especially those based on SIP, is expected to significantly augment in the years to come. However, SIP needs to confront several security issues mainly due to its open and text-oriented nature. In this direction, researchers are seeking novel proposals that are able to promptly identify security breaches and apply effective methods of control.

In this paper, we capitalize on an idea proposed in [28], that is, the use of entropy theory to detect abnormalities in raw application data. Specifically, through extensive experimentation, we extend, calibrate and throughly assess the effectiveness of the initial idea, thus offering a complete formalized framework that can be used to trace and detect DoS attacks in SIP ecosystems. The proposed solution can be used to assess and certify the security level of VoIP provider with reference to DoS attacks, based on the logged traffic without trampling on end-user's privacy. We assert that our framework exhibits several advantages over those in earlier work. That is, it is lightweight, practical, privacy-preserving, and retains full compatibility with the SIP standard operating effectively in both offline and realtime fashion. It also presents a high degree of flexibility (through the tuning of its parameters) depending on the everyday VoIP traffic each provider has to cope with.

Currently, we are working towards providing a more robust data anonymization scheme for use especially in cases where the exchange of log files among providers or between a provider and a data analysis center is deemed necessary. Particularly, focusing on offline analysis, we are interested in schemes that can offer unlinkablity along with anonymity. The employment of machine learning techniques on the audit trail data as a second more comprehensive layer of analysis is another direction worthy of investigation in a future work.

Acknowledgements

The authors would like to thank Dr. Panagiotis Rizomiliotis and the anonymous reviewers for their valuable comments. 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] A. Cox, Mobile voip users to reach 1 billion by 2017, or one in seven mobile subscribers (2012).
URL http://www.juniperresearch.com/viewpressrelease.php?pr=355

[2] IBISWorld, Voip in the us: Market research report (2013).
URL http://www.ibisworld.com/industry/default.aspx?indid=1269

[3] C. K. Yeo, S. C. Hui, I. Y. Soon, L. M. Ang, H.323 compliant voice over ip system, Int. J. Comput. Appl. Technol. 16 (4) (2003) 143–153. doi:10.1504/IJCAT.2003.000321.

[4] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, Sip: Session initiation protocol (2002).

[5] F. Andreasen, B. Foster, Media gateway control protocol (2003).

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext transfer protocol – http/1.1, (rfc 2616) (1999).

[7] S. Ehlert, D. Geneiatakis, T. Magedanz, Survey of network security systems to counter sip-based denial-of-service attacks, Vol. 29, 2010, pp. 225 – 243. doi:10.1016/j.cose.2009.09.004.

[8] D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, S. Ehlert, D. Sisalem, Survey of security vulnerabilities in session initiation protocol, Vol. 8, 2006, pp. 68–81. doi:10.1109/COMST.2006.253270.

[9] A. D. Keromytis, A comprehensive survey of voice over ip security research, IEEE Communications Surveys and Tutorials 14 (2) (2012) 514–537. doi:10.1109/SURV.2011.031611.00112.

[10] D. Geneiatakis, G. Kambourakis, C. Lambrinoudakis, T. Dagiuklas, S. Gritzalis, A framework for protecting a sip-based infrastructure against malformed message attacks, Comput. Netw. 51 (10) (2007) 2580–2593. doi:10.1016/j.comnet.2006.11.014.

[11] S. Ehlert, G. Zhang, D. Geneiatakis, G. Kambourakis, T. Dagiuklas, J. Markl, D. Sisalem, Two layer denial of service prevention on sip voip infrastructures, Comput. Commun. 31 (10) (2008) 2443–2456. doi:10.1016/j.comcom.2008.03.016.

[12] D. Seo, H. Lee, E. Nuwere, Sipad: Sip-voip anomaly detection using a stateful rule tree, Comput. Commun. 36 (5) (2013) 562–574. doi:10.1016/j.comcom.2012.12.004.

[13] J. Fiedler, T. Kupka, S. Ehlert, T. Magedanz, D. Sisalem, Voip defender: Highly scalable sip-based security architecture, in: Proceedings of the 1st International Conference on Principles, Systems and Applications of IP Telecommunications, IPTComm ’07, ACM, New York, NY, USA, 2007, pp. 11–17. doi:10.1145/1326304.1326307.

[14] J. Tang, Y. Cheng, Intrusion Detection for IP-Based Multimedia Communications over Wireless Networks, Springer Publishing Company, Incorporated, 2013.

[15] M. Swanson, B. Guttman, Generally accepted principles and practices for securing information technology systems [electronic resource], National Institute of Standards and Technology, Technology Administration, U.S. Dept. of Commerce, 1996.

[16] P. Ohm, Broken promises of privacy: Responding to the surprising failure of anonymization, Tech. Rep. ID 1450006, Social Science Research Network, Rochester, NY (Jul. 2012).

[17] O. Tene, Privacy: The new generations, SSRN Scholarly Paper ID 1710688, Social Science Research Network, Rochester, NY (Nov. 2010).

[18] L. Sweeney, Uniqueness of Simple Demographics in the U.S. Population, LIDAP-WP4, Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, 2000.

[19] P. Golle, Revisiting the uniqueness of simple demographics in the us population, in: Proceedings of the 5th ACM workshop on Privacy in electronic society, WPES '06, ACM, New York, NY, USA, 2006, pp. 77–80. doi:10.1145/1179601.1179615.

[20] A. Pfitzmann, M. Hansen, A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management (2009).

[21] F. Pereniguez, R. Marin-Lopez, G. Kambourakis, S. Gritzalis, A. Gomez, Privakerb: A user privacy framework for kerberos, Vol. 30, 2011, pp. 446 – 463. doi:10.1016/j.cose.2011.04.001.

[22] S. Ehlert, G. Zhang, D. Geneiatakis, G. Kambourakis, T. Dagiuklas, J. Markl, D. Sisalem, Two layer denial of service prevention on sip voip infrastructures, Vol. 31, Elsevier Science Publishers B. V.,

Amsterdam, The Netherlands, The Netherlands, 2008, pp. 2443–2456. doi:10.1016/j.comcom.2008.03.016.

[23] J. Ioannidis, S. M. Bellovin, Implementing pushback: Router-based defense against ddos attacks, in: Proc. Internet Society Symposium on Network and Distributed System Security, 2002.

[24] R. Mathew, V. Katkar, Survey of low rate dos attack detection mechanisms, in: Proceedings of the International Conference, Workshop on Emerging Trends in Technology, ICWET '11, ACM, New York, NY, USA, 2011, pp. 955–958. doi:10.1145/1980022.1980227.

[25] J. Mirkovic, P. Reiher, D-ward: A source-end defense against flooding denial-of-service attacks, Vol. 2, IEEE Computer Society Press, Los Alamitos, CA, USA, 2005, pp. 216–232. doi:10.1109/TDSC.2005.35.

[26] G. Oikonomou, J. Mirkovic, P. Reiher, M. Robinson, A framework for a collaborative ddos defense, in: Proceedings of the 22nd Annual Computer Security Applications Conference, ACSAC '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 33–42. doi:10.1109/ACSAC.2006.5.

[27] W. Ehrlich, K. Futamura, D. Liu, An entropy based method to detect spoofed denial of service (dos) attacks, in: S. Raghavan, B. Golden, E. Wasil (Eds.), Telecommunications Modeling, Policy, and Technology, Vol. 44 of Operations Research/Computer Science Interfaces, Springer US, 2008, pp. 101–122.

[28] Z. Tsiatsikas, D. Geneiatakis, G. Kambourakis, A. Keromytis, A privacy-preserving entropy-driven framework for tracing dos attacks in voip, in: Availability, Reliability and Security (ARES), 2013 Eighth International Conference on, 2013, pp. 224–229. doi:10.1109/ARES.2013.30.

[29] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, Rtp: A transport protocol for real-time applications (2003).

[30] C. E. Shannon, A mathematical theory of communication, Vol. 27, 1948.

[31] D. Geneiatakis, N. Vrakas, C. Lambrinoudakis, Utilizing bloom filters for detecting flooding attacks against sip based services, Computers & Security 28 (7) (2009) 578–591.

[32] J. E. Holt, Logcrypt: Forward security and public verification for secure audit logs (2005).

[33] B. Schneier, J. Kelsey, Cryptographic support for secure logs on untrusted machines, in: Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7, SSYM'98, USENIX Association, Berkeley, CA, USA, 1998, pp. 4–4.

[34] A. Mitrakas, Assessing liability arising from information security breaches in data privacy, International Data Privacy Lawdoi:10.1093/idpl/ipr001.

[35] E. Commission, Directive on privacy and electronic communication - 2002/58/ec (2002).
URL http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:2002L0058:20091219:EN:PDF

[36] D. Dolev, A. C. Yao, On the security of public key protocols, Information Theory, IEEE Transactions on 29 (2) (1983) 198–208. doi:10.1109/TIT.1983.1056650.

[37] G. Zhang, S. Ehlert, T. Magedanz, D. Sisalem, Denial of service attack and prevention on sip voip infrastructures using dns flooding, in: Proceedings of the 1st International Conference on Principles, Systems and Applications of IP Telecommunications, IPTComm '07, ACM, New York, NY, USA, 2007, pp. 57–66. doi:10.1145/1326304.1326314.

[38] G. Kambourakis, Anonymity and closely related terms in the cyberspace: An analysis by example, Journal of Information Security and Applications, Elsevier.doi:dx.doi.org/10.1016/j.jisa.2014.04.001.

[39] S. Coull, F. Monrose, M. Reiter, M. Bailey, The challenges of effectively anonymizing network data, in: Conference For Homeland Security, 2009.

CATCH '09. Cybersecurity Applications Technology, 2009, pp. 230–236. doi:10.1109/CATCH.2009.27.

[40] R. Dewri, I. Ray, I. Ray, D. Whitley, Exploring privacy versus data quality tradeoffs in anonymization techniques using multi-objective optimization, J. Comput. Secur. 19 (5) (2011) 935–974.

[41] G. S. Kc, A. D. Keromytis, V. Prevelakis, Countering code-injection attacks with instruction-set randomization, in: Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS '03, ACM, New York, NY, USA, 2003, pp. 272–280. doi:10.1145/948109.948146.

[42] E. G. Barrantes, D. H. Ackley, T. S. Palmer, D. Stefanovic, D. D. Zovi, Randomized instruction set emulation to disrupt binary code injection attacks, in: Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS '03, ACM, New York, NY, USA, 2003, pp. 281–289. doi:10.1145/948109.948147.

[43] P. Samarati, Protecting respondents' identities in microdata release, IEEE Trans. on Knowl. and Data Eng. 13 (6) (2001) 1010–1027. doi:10.1109/69.971193.

[44] X. Xiao, Y. Tao, Anatomy: Simple and effective privacy preservation, in: Proceedings of the 32Nd International Conference on Very Large Data Bases, VLDB '06, VLDB Endowment, 2006, pp. 139–150.

[45] X. He, Y. Xiao, Y. Li, Q. Wang, W. Wang, B. Shi, Permutation anonymization: Improving anatomy for privacy preservation in data publication, in: Proceedings of the 15th International Conference on New Frontiers in Applied Data Mining, PAKDD'11, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 111–123. doi:10.1007/978-3-642-28320-8_10.

[46] L. Sweeney, K-anonymity: A model for protecting privacy, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10 (5) (2002) 557–570. doi:10.1142/S0218488502001648.

[47] K. Seppanen, Method, apparatus and computer program for anonymization of identification data, uS Patent App. 12/168,041 (Nov. 20 2008).
URL http://www.google.com/patents/US20080287118

[48] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: Improved definitions and efficient constructions, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, ACM, New York, NY, USA, 2006, pp. 79–88. doi:10.1145/1180405.1180417.

[49] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, G. Karypis, Privacy risks in recommender systems, Vol. 5, IEEE Educational Activities Department, Piscataway, NJ, USA, 2001, pp. 54–62. doi:10.1109/4236.968832.

[50] S. Bruce, Why 'Anonymous' data sometimes isn't.
URL https://www.schneier.com/essay-200.html

[51] E. Shi, E. Stefanov, C. Papamanthou, Practical dynamic proofs of retrievability, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security, CCS '13, ACM, New York, NY, USA, 2013, pp. 325–336. doi:10.1145/2508859.2516669.
URL http://doi.acm.org/10.1145/2508859.2516669

[52] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, S. Devadas, Path oram: An extremely simple oblivious ram protocol, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security, CCS '13, ACM, New York, NY, USA, 2013, pp. 299–310. doi:10.1145/2508859.2516660.
URL http://doi.acm.org/10.1145/2508859.2516660

[53] N. institute of standards, technology, FIPS PUB 198-1, The Keyed-Hash Message Authentication Code , Federal Information Processing Standard (FIPS), Tech. rep., DEPARTMENT OF COMMERCE (Jul. 2002).

[54] A. Cooper, M. Hansen, R. Smith, H. Tschofenig, Privacy terminology and concepts.

[55] Kamailio the open source sip server (2014).
URL http://www.kamailio.org/w/

[56] D. Eastlake, T. Hansen, Us secure hash algorithms- (sha and sha-based hmac and hkdf) (2011).

[57] R. Krishnamurthy, G. Rouskas, Evaluation of sip proxy server performance: Packet-level measurements and queuing model, in: Communications (ICC), 2013 IEEE International Conference on, 2013, pp. 2326–2330. doi:10.1109/ICC.2013.6654877.

[58] A. N. Hussain, Measurement and spectral analysis of denial of service attacks, Ph.D. thesis, Los Angeles, CA, USA, aAI3196820 (2005).

[59] P. O. S. V. De Melo, L. Akoglu, C. Faloutsos, A. A. F. Loureiro, Surprising patterns for the call duration distribution of mobile phone users, in: Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III, ECML PKDD'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 354–369.

[60] P. Barford, J. Kline, D. Plonka, A. Ron, A signal analysis of network traffic anomalies, in: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, IMW '02, ACM, New York, NY, USA, 2002, pp. 71–82. doi:10.1145/637201.637210.

[61] G. Gu, P. Fogla, D. Dagon, W. Lee, B. Skorić, Measuring intrusion detection capability: an information-theoretic approach, in: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, ASIACCS '06, ACM, New York, NY, USA, 2006, pp. 90–101. doi:10.1145/1128817.1128834.

[62] Y. Cheng, Y. Hao, wei song, J. Tang, Sip flooding attack detection with a multidimensional sketch design, IEEE Transactions on Dependable and Secure Computingdoi:10.1109/TDSC.2014.2302298.

[63] R. T. Mercuri, On auditing audit trails, Commun. ACM 46 (1) (2003) 17–20. doi:10.1145/602421.602436.

[64] K.-I. Pun, Y.-W. Si, Audit trail analysis for traffic intensive web application, in: Proceedings of the 2009 IEEE International Conference on e-Business Engineering, ICEBE '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 577–582. doi:10.1109/ICEBE.2009.91.

[65] K. H. Lee, X. Zhang, D. Xu, Loggc: Garbage collecting audit log, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security, CCS '13, ACM, New York, NY, USA, 2013, pp. 1005–1016. doi:10.1145/2508859.2516731.

[66] G. Karopoulos, G. Kambourakis, S. Gritzalis, Privasip: Ad-hoc identity privacy in sip, Computer Standards & Interfaces 33 (3) (2011) 301–314. doi:http://dx.doi.org/10.1016/j.csi.2010.07.002.

[67] Y. Bouzida, C. Mangin, A framework for detecting anomalies in voip networks, in: Availability, Reliability and Security, 2008. ARES 08. Third International Conference on, 2008, pp. 204–211. doi:10.1109/ARES.2008.205.

[68] S. Ehlert, C. Wang, T. Magedanz, D. Sisalem, Specification-based denial-of-service detection for sip voice-over-ip networks, in: Internet Monitoring and Protection, 2008. ICIMP '08. The Third International Conference on, 2008, pp. 59 –66. doi:10.1109/ICIMP.2008.14.

[69] K. Rieck, S. Wahl, P. Laskov, P. Domschitz, K.-R. Mller, A self-learning system for detection of anomalous sip messages, in: H. Schulzrinne, R. State, S. Niccolini (Eds.), Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks, Vol. 5310 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 90–106. doi:10.1007/978-3-540-89054-6_5.

[70] S. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks, Com-

munications Surveys Tutorials, IEEE 15 (4) (2013) 2046–2069. doi:10.1109/SURV.2013.031413.00127.

[71] G. Gandhi, S. Srivatsa, An entropy algorithm to improve the performance and protection from denial-of-service attacks in nids, in: Computer and Electrical Engineering, 2009. ICCEE '09. Second International Conference on, Vol. 1, 2009, pp. 603 –606. doi:10.1109/ICCEE.2009.266.

[72] J. Biskup, U. Flegel, Threshold-based identity recovery for privacy enhanced applications, in: Proceedings of the 7th ACM conference on Computer and communications security, CCS '00, ACM, New York, NY, USA, 2000, pp. 71–79. doi:10.1145/352600.352611.

[73] P. Lincoln, P. Porras, Privacy-preserving sharing and correlation of security alerts, in: Proceeding of USENIX Security Symposium, 2004, pp. 239–254.

[74] D. Xu, P. Ning, Alert correlation through triggering events and common resources, in: Proceedings of the 20th Annual Computer Security Applications Conference, ACSAC '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 360–369. doi:10.1109/CSAC.2004.5.

[75] U. Flegel, Pseudonymizing unix log files, in: Proceedings of the International Conference on Infrastructure Security, InfraSec '02, Springer-Verlag, London, UK, UK, 2002, pp. 162–179.

[76] A. J. Lee, A privacy-preserving interdomain audit framework, in: Proceedings of the Workshop On Privacy In The Electronic Society, 2006. doi:10.1145/1179601.1179620.

[77] A. Slagell, W. Yurcik, Sharing computer network logs for security and privacy: a motivation for new methodologies of anonymization, in: Security and Privacy for Emerging Areas in Communication Networks, 2005. Workshop of the 1st International Conference on, 2005, pp. 80 – 89. doi:10.1109/SECCMW.2005.1588299.

[78] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613. doi:10.1145/359168.359176.

[79] B. Krishnamurthy, S. Sen, Y. Zhang, Y. Chen, Sketch-based change detection: Methods, evaluation, and applications, in: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, IMC '03, ACM, New York, NY, USA, 2003, pp. 234–247. doi:10.1145/948205.948236.

[80] J. Pelaez, E. Fernandez, Voip network forensic patterns, in: Computing in the Global Information Technology, 2009. ICCGI '09. Fourth International Multi-Conference on, 2009, pp. 175 –180. doi:10.1109/ICCGI.2009.53.

APPENDIX I

Algorithm 1: EntropyDetectionModule/RealtimeAnalysis

```
Input: SIPmessage
Output: MessageClassification
$AlocateMemory($HashTable$)$;
$NAD \leftarrow Avg(AID)$;
$SipHeaders \leftarrow SplitSIPMessage($SIPMessage$)$;
while $(SipHeaders \neq NULL)$ do
    $H_{S_i} \leftarrow$ HMAC-SHA256$(SipHeaders)$;
    InsertToHashTable$(H_{S_i})$
end
$AID \leftarrow ComputeMetricsHashTable(SipMessage, HashTable)$;
if $AID$ is greater than $(NAD + \delta)$ then
    $print(ALERT - AttackMessage)$
else
    $print(LegitimateMessage)$
end
```

Algorithm 2: ComputeMetrics/OfflineAnalysis

```
Input: DataFile
Output: EntropyFile
OpenToRead(DataFile);
$EntropySumArray[S1, S2, S3, S4, S5, S6]$;
$i \leftarrow 1$;
$SymbolCounter \leftarrow 1$;
$SUM \leftarrow 0$;
while DataFile $\neq$ EOF do
    if Search($H_{S_i}$, $DataFile$) then
        $Temp \leftarrow readLine$ /*Locate Ap in line*/;
        /*Ap(HSi) : Appearances of HSi*/;
        Read(Ap);
        $Prob(H_{S_i}) \leftarrow (Ap/N)$;
        /*N : overall number of occurences*/ ;
        Write $I(H_{S_i})$ in EntropyFile;
        /*I(HSi):Itself Information of HSi*/ ;
        $SUM+ \leftarrow I(H_{S_i})$;
        $EntropySumArray[S_i]+ \leftarrow Prob(H_{S_i}) * I(H_{S_i})$;
    end
    if symbolCounter equals 4 or 6 then
        $ActualInfo(CurrentMessage) \leftarrow SUM$;
        $symbolCounter \leftarrow 1$;
        $SUM \leftarrow 0$;
        $i \leftarrow 1$;
    end
end
```

Algorithm 3: ComputeSymbolFrequency/OfflineAnalysis

```
Input: TrafficFile
Output: DataFile
$i \leftarrow 1$;
$symbolCounter \leftarrow 1$;
OpenToRead(TrafficFile);
while TrafficFile $\neq$ EOF do
    $OSi \leftarrow readLine$;
    /*OSi : Original Symbol*/;
    if Search($O_{S_i}$, $HeaderSymbol$) or Search($O_{S_i}$, $symbol[s1 - s6]$) then
        $H_{S_i} \leftarrow$ HMAC-SHA256($O_{S_i}$);
        /*HSi : Hash Symbol*/;
        $symbolCounter+ \leftarrow 1$;
        InsertToHashTable($H_{S_i}$);
        /*For insertToHashTable see algorithm 4*/
        Write(Hash($O_{S_i}$) : $H_{S_i}$);
    else
        if symbolCounter equals $4$ or $6$ then
            $i \leftarrow 1$
            $symbolCounter \leftarrow 1$
        end
    end
end
OpenToWrite(TrafficFile);
while TrafficFile $\neq$ EOF do
    if SearchLine equals $H_{S_i}$ then
        if LookUpHashTable(HashTable) $\neq$ 0 then
            Write($H_{S_i}$ : $Ap$ :: LookUpHashTable($HashTable, H_{S_i}$))
        end
    end
end
```

Algorithm 4: InsertToHashTable

```
Input: SIPHeader
Output: UpdateHashTable
if LookUpHashTable(SIPHeader, HashTable) equals $0$ then
    Insert$(SIPHeader, 0)$
else
    Insert$(SIPHeader,$ LookUpHashTable$(SIPHeader, HashTable) + 1)$
end
```

Algorithm 5: Timestamps/OfflineAnalysis

```
Input: TrafficFile
Output: EntropyFile/Interval
$Interal \leftarrow userChoice$;
$timestamp \leftarrow$ Read(TrafficFile);
while $(timestamp \neq timestamp + Interval)$ do
    TraficFile $\leftarrow$ InsertToFile(HMAC-SHA256(SIPHeader))
end
ComputeMetrics(ComputeSymbolFrequency(TrafficFile))
```