Computer Communications 32 (2009) 203–212

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

OntoSPIT: SPIT management through ontologies

S. Dritsas, V. Dritsou, B. Tsoumas, P. Constantopoulos, D. Gritzalis *

Information Security and Critical Infrastructure Protection Research Group, Dept. of Informatics, Athens University of Economics and Business (AUEB), 76 Patission Ave., 10434 Athens, Greece

A R T I C L E   I N F O

Article history:
Received 23 March 2008
Received in revised form 6 October 2008
Accepted 6 October 2008
Available online 18 October 2008

Keywords:
SPIT (SPAM over Internet telephony)
Ontology
VoIP
SPAM

A B S T R A C T

VoIP enables new ways for communication. At the same time, it provides new means, in terms of transmitting bulk unsolicited messages and calls, namely SPam over Internet telephony (SPIT). In this paper, we propose a conceptual model, based on an underlying ontology, which describes the SPIT domain. The ontology provides capabilities, such as modeling the SPIT phenomenon in a SIP-based VoIP environment, a common understanding of SPIT domain, as well as reusable SPIT-related knowledge interoperability, aggregation and reasoning. We demonstrate that the proposed ontology, combined with a set of SPIT identification criteria, as its underlying axioms and rules, could enhance the correlation and management of SPIT incidents. It could also support SPIT detection, thus facilitating the better protection of VoIP environments in a holistic, cooperative, and effective way.

1. Introduction

Voice-over-IP (VoIP) increasingly penetrates the telephony market, as it appears to be an attractive alternative compared to traditional telephony. VoIP seamless integration with the existing IP networks (and especially the Internet), its lower costs compared to PSTN telephony, the use of computer-based soft-phones, as well as the existence and the implementation of sophisticated end-user services, in terms of portability, accessibility and convergence of telephone networks, are some of the reasons that make VoIP an increasingly attractive and advantageous network application.

However, the use of VoIP also facilitates the exploitation of new threats and vulnerabilities that Internet protocols pose [22,18]. For example, low-cost calls and zero-cost instance messages, combined with the pervasiveness of Internet, could be used by malicious users to make bulk unsolicited calls and/or send bulk unsolicited instant messages. This is a new form of SPAM, in the context of VoIP environments, which is called SPAM over Internet telephony (SPIT). SPIT is expected to cause stronger impact on users, than email SPAM, as it appears to be more intrusive [12,20].

SPIT has received low attention until now, mainly due to its rather embryonic stage of employment. However, several anti-SPIT frameworks have been already proposed, focusing on countering the SPIT phenomenon by adopting concepts, approaches, and techniques mainly used for fighting email SPAM [13]. The effectiveness of these tools is usually considered inadequate, mainly due to their ad hoc nature, as well as due to the real-time nature of VoIP communications.

In this paper, we show that the effectiveness of an anti-SPIT technique depends on several factors, and relates to different elements that are incorporated within a VoIP infrastructure. Next, we propose an ontology that describes the SIP (session initiation protocol)-based SPIT (or anti-SPIT) domain, and we show that this ontology (called ontoSPIT) can provide a common understanding of the SPIT domain, facilitate the reuse of domain knowledge, and, finally, help solve several domain interoperability issues, mainly through making domains' SPIT assumptions explicit.

The paper is organized as follows. First, we briefly present the necessary background, regarding the VoIP environment and the SPIT phenomenon and we refer to a number of issues regarding the ontology paradigm. In the sequel, we introduce the SPIT-related conceptual model together with the essential terms that define the SPIT domain. Next, we present a set of SPIT identification criteria, which can facilitate the detection of SPIT traffic in a VoIP environment. Based on the above criteria, we build an event-condition-action (ECA) model, which acts as the basis for a detailed SIP-oriented SPIT ontology (ontoSPIT). Then, the ontoSPIT model is presented in detail and comment upon. After presenting the model, we demonstrate the applicability of it, through the use of the semantic web rule language (SWRL). Finally, we conclude by providing the reader with an overview of our findings.

2. Background

In this section, we provide the reader with a brief description of the SPIT phenomenon, together with some comments on ontologies and their applicability.

* Corresponding author. Tel.: +302105810116.
E-mail addresses: sdritsas@aueb.gr (S. Dritsas), vdritsou@aueb.gr (V. Dritsou), bts@aueb.gr (B. Tsoumas), panosc@aueb.gr (P. Constantopoulos), dgrit@aueb.gr (D. Gritzalis).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.10.004

2.1. SPIT phenomenon overview

VoIP is a technology that takes advantage of existing data networks and provides the establishment of voice sessions and sending voice as data packets over IP networks. Currently, VoIP implementations are usually based on the SIP, which tends to be the dominant underlying technology for providing services, such as IP telephony, instant messaging and presence [16]. SIP is an application layer protocol used to create, maintain, and terminate multimedia sessions. The basic SIP entities that support its functionality are: (a) user agents (UA), which act as communication end-points, and (b) SIP servers, which help and support the SIP sessions. Fig. 1 presents the basic operation model of SIP protocol including its main components.

The rapid adoption of VoIP technologies is expected to introduce new types of threats. Such a new threat is the SPIT phenomenon, which is defined as a set of bulk unsolicited voice calls or instant messages. Currently, three different types of VoIP SPAM types have been recognized, namely [17]: (a) *call SPIT*, which is bulk, unsolicited session initiation attempts to establish a multimedia session, (b) *instant message SPIT*, which is bulk, unsolicited instant messages, known as SPIM, and (c) *presence SPIT*, which is bulk, unsolicited presence requests, enabling a malicious user become a member of the address book of one or more users.

Nowadays, SPIT is not yet a major issue. However, it is expected to become soon attractive to malicious users (“spitters”), thus making the further growth of VoIP technology practically challenging. On the other hand, the anti-SPIT frameworks, which have been introduced so far, are often considered inadequate [13]. Also, the existing methods of defeating SPAM filtering, combined with the enhanced capabilities of spammers, may result in more complex and effective SPAM attack strategies, which could be really difficult to detect and fight. Moreover, the real-time nature of VoIP services, in contrast with the store-and-forward communication pattern of email technology, led us to consider that it is more efficient to handle SPIT in the signaling phase, where SIP protocol is applied, than to filter the payload of a session (i.e. the content of a call).[1]

Having the above in mind, we suggest that an efficient SPIT handling process involves three distinct steps (see Fig. 2), namely: (a) *prevention*, where specific actions are required, so as to impede a SPIT call or message to be transmitted and leave the callers’ domain, (b) *detection*, where appropriate actions should be applied, in order to detect a SPIT call or message in the callee’s domain, and (c) *reaction*, where appropriate actions for handling a SPIT attack (e.g. place the specific caller/sender to the domain’s black list) should be used.

2.2. Ontologies and their applicability to the SPIT problem

An ontology is an explicit specification of a conceptualization, which can be used to describe structurally heterogeneous information sources, helping both people and machines to communicate in a concise manner [7]. Ontologies are discussed in the literature as a means to support knowledge sharing and reuse [4]. The reusability approach is based on the assumption that if a modeling scheme – i.e. ontology – is explicitly specified and mutually agreed upon by the parties involved, then it is possible to share, reuse, and extend knowledge.

An ontology is comprised by three major building blocks: concepts, relationships, and axioms. Concepts are abstract terms, which are typically organized in taxonomies. Hierarchical concepts are linked with an *“is-a”* relationship. Furthermore, concepts can have properties, which help in establishing relationships between either hierarchically classified or completely unrelated concepts. Axioms, which are expressed by integrity constraints, are rules that are valid in the modeled domain, finally constraining the possible (i.e. meaningful) interpretations for the defined concepts.

Due to their beneficial characteristics, ontologies have been used in several different domains, so as to provide information normalization and in cases where the scope is to represent the necessary knowledge of a specific domain [15]. For example, ontologies express the knowledge of experts on a specific domain explicitly, by supplying and analyzing all the necessary information. As this information is expressed by an explicit way, it can be well understood by both humans and machines, thus making the knowledge mutually comprehensible. This latter fact enables the reuse of the domain knowledge from different parties, thus avoiding the development of new ontologies from scratch. Ontologies can be used in any system, regardless of its architecture and technology given the semantics of the ontology are clearly defined. Moreover, decision-making logic can be embedded in a natural way via the use of ECA rules which can lead to implicit knowledge acquisition, thus enriching the ontology knowledge. Finally, the ontology paradigm follows the open world assumption (OWA), which makes it a perfect choice for simulating uncertain real-world scenarios where incomplete knowledge is present.

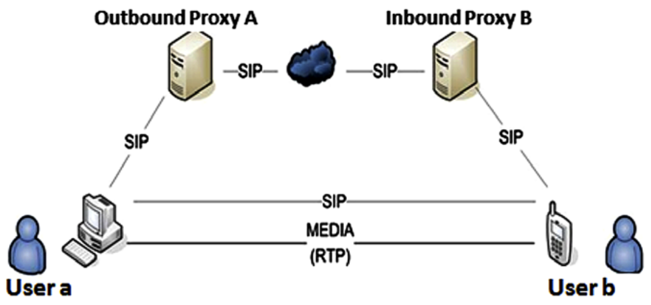

Fig. 1. SIP basic operation (“Trapezoid”).

3. Related work

In this section, we will refer to some related work. Most of this work focuses on detecting email SPAM, and this is done for two main reasons. First, because email SPAM and SPIT have certain similarities (and differences, as well), therefore such a brief review makes sense for better understanding the situation in SPIT, and second because SPIT detection is dealt with in really few publications.

A system that uses an ontology to determine whether an email should be deleted or not is used in [19]. The approach takes as input a set of email messages and, based on predefined requirements, decides whether a message should be retained or not. The system is implemented using Protégé-2000 [14]. It identifies the special features of each message (expressed by a set of rules), which are then categorized by an underlying Bayesian classifier. Probabilities regarding the categorization of the messages are assigned to them, as well as for further training the system. A similar approach analyzes each incoming mail in order to identify its purpose [3]. This is accomplished by developing an ontology that classifies the messages in terms of the intended scope of the email sender (e.g. make a request, propose a meeting, etc.). The delivered messages are then compared to the existing classes of the ontology, so as to identify and categorize them.

[1] It is theoretically possible to filter the payload of a session (i.e. the RTP traffic), but this requires devices with high computational resources, so as to realize the real-time nature of VoIP communications. In this paper, we deal with the signaling phase of a SIP-based VoIP session.

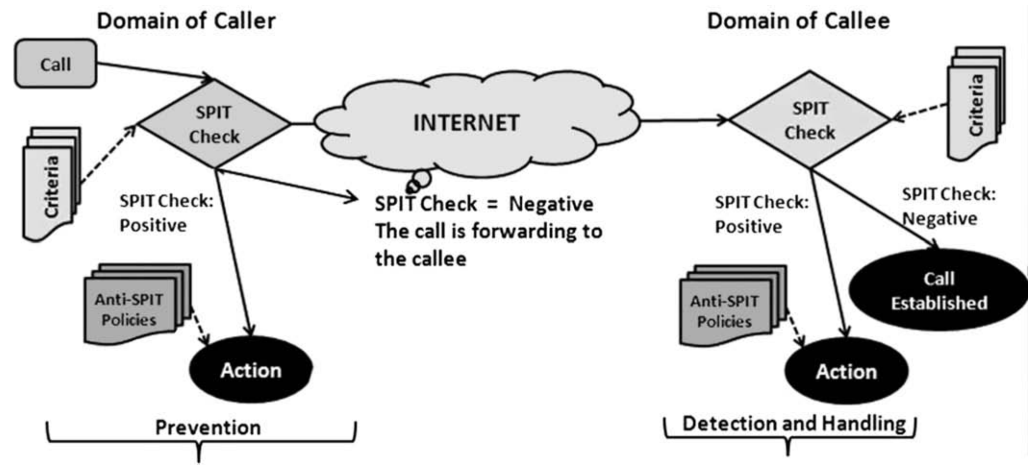

Fig. 2. A macroscopic view of SPIT management process.

Brewer et al. [2] propose an extension to existing SPAM filters, where the semantics of email messages are used as an additional parameter for classification. The system takes as input the user's interests and creates the "user's schema". Important relationships are defined and the schema is then populated, while assigning to each class and relationship an initial score. Incoming messages are mapped to entities of the ontology, relationships based on the schema are explored, and an aggregate score of the message that depends on the predefined relationships is computed. Each time a message exceeds the user's specified threshold value, the message is classified under this specific interest. A similar view is proposed in [11], where the users' interests are used to detect SPAM mails. Important concepts for each user are identified by collecting each one's characteristics (e.g. age, gender, job) and their responses to previous messages. Each email category is then labeled by a human expert regarding this information. Through the study of the preferences and responses of several users, specific relationships are detected, which are then eliminated and translated to axioms of the proposed ontology. Thus, the system detects the associations between the incoming messages and the users' interests, so as to categorize the messages and to estimate user' actions (e.g. reply, delete, store).

Youn et al. [23] suggest a framework to filter SPAM, based on a decision tree that could be considered an ontology. All the rules that are concluded from the training dataset build a decision tree. The tree is then transformed in attribute-relation file format (ARFF), an ASCII text file that describes a list of instances sharing a set of attributes. By this way they map the initial decision tree to a formal ontology, which may be queried when new emails arrive. The system gives as output the decision whether the email is SPAM or not.

Ontology is also used in order to conceptualize the security flows of the SIP protocol [5]. Each "SIP message" constitutes its main concept, followed by the attributes of the first line of the message and its header (according to the specifications in RFC 3261). Using the information contained in the first line of a SIP message, different types of requests can be identified. These are, in turn, described by five methods followed by their uniform resource identifiers (URI). Each URI constitutes a rule that should not be violated by the structure of the message. The header of the message is then checked for the validity of its structure. The target of the threat (in case of a malformed message), as well as the consequence of the attack, are also conceptualized, offering all the necessary information to the interested parties.

4. SPIT domain description

Wishing to develop a SPIT-oriented knowledge base and enhance the sharing of its information and components, we propose an approach based on ontologies (*ontoSPIT*). By adopting ontologies, we can use their advantages in an effort to help stakeholders of SPIT domain (i.e. administrators and end-users) make decisions regarding anti-SPIT actions. These actions should be based on predefined assumptions or on tacit knowledge entailed by the amount of SPIT related information generated by different sources, such as anti-SPIT systems, VoIP domains, SPIT attack patterns, etc.

In this context, we define a generic SPIT Ontology (ontoSPIT), as "*an ontology that elaborates on the SPIT-related aspects of a system*" in an effort to manage SIP-based SPIT attacks.[2] In order to build the ontology, we have to first outline the basic terminology of the domain and then to present a conceptual model that depicts the relations between the terms of the anti-SPIT domain. Furthermore, having in mind that there is no standard method for ontology development [15], we followed the collaborative approach for ontology design [9]. During design, SPIT-emerging frameworks, well-known security standards, email anti-SPAM approaches, and the design criteria in [7] were taken into account. More specifically, we used this method as a starting-point, elaborating it in the context of SPIT management. The output of this approach is a five-step process (detailed in Section 5).

Fig. 3 depicts the conceptual model of the SPIT domain. In line with this model, which acts as a domain ontology [8], we define the basic terms that are used:

- *User*: Conceptualizes a legitimate actor (i.e. user) that has (possess) a VoIP device (User Agent, in terms of SIP definition, i.e. a soft-phone), who will be the potential target of a SPIT attack.
- *Asset*: Describes the VoIP-related assets (including the VoIP devices, network resources, bandwidth, etc.), that support the VoIP service provision to the user and has a specific value for the user.
- *Domain*: Describes the domain that a legitimate user belongs to. For instance, the user alice@example.com belongs to the domain *example.com*. The domain is responsible for the registered users that belong to it, therefore it is in charge of SPIT management and handling.

[2] In the sequel, the terms "SPIT Ontology" and "Ontology" will be used interchangeably.

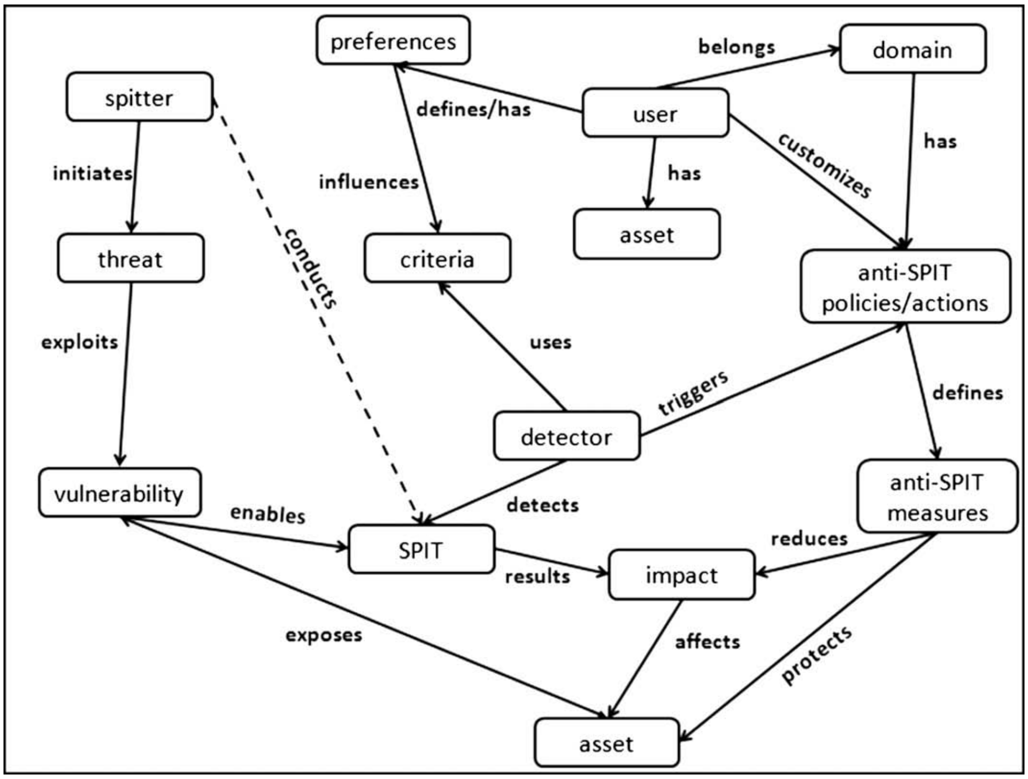

Fig. 3. Conceptual model of the SPIT domain.

- *Spitter*: Defines the potential attacker (spitter) who is responsible for conducting SPIT attacks and, therefore, expects to get benefits from them.
- *Threat*: Defines the possible conditions and/or events that facilitate a SPIT attack to take place.
- *Vulnerability*: Defines the specific conditions that could be exploited by a spitter, in order to realize an SPIT attack. For example, the use of open relays servers in a VoIP environment defines a SIP vulnerability that could be used for conducting SPIT.
- *SPIT*: Defines any attempt made by an attacker, in order to make bulk unsolicited calls or to send bulk unsolicited instance messages to a legitimate user.
- *Impact*: Describes the impact of the successful realization of a SPIT attack.
- *anti-SPIT Policies/Actions*: Defines the generic actions adopted by a domain or by a user, in order to handle SPIT attacks. For example, if a user has a SIP URI that has as domain the string *example.org*, then the call should be blocked.
- *anti-SPIT Measures*: Represent the mechanisms and techniques adopted by a domain and/or by a user, in order to counter SPIT attacks. They are enabled under specific circumstances, in line with domains' anti-SPIT policies. For example, if a policy defines that for any unknown user the usage of a puzzle test is obligatory, then the puzzle mechanism should be enabled and used in order to fulfill the policy.
- *Preferences*: Defines the interests of the legitimate user, i.e. refers to the circumstances under which a user can label a call or a message as SPIT. For example, if a message or a call has as its subject the word "crime", a user might label this message as SPIT and wishes this message not to be delivered, while another user might not label it so.
- *Criteria*: Describes the circumstances under which an action and/or condition could be labeled as SPIT. They are defined by the domain that a user belongs to. The criteria may be fine-tuned by the user preferences, which are applied in a second level, e.g. after the domain-defined global SPIT criteria. For example, if a user makes a lot of call attempts in a specific time period, then this action should be labeled as SPIT and the user as spitter.
- *Detector*: Defines the entity that will be responsible for detecting a potential SPIT attack. The entity compiles the appropriate information, in order to enable a specific set of actions for handling a SPIT attack after its occurrence.

Regarding the conceptual model of the domain (Fig. 3), we should mention that the ontoSPIT approach deals with the SPIT detection–identification criteria, as well as with the actions that should be performed by all the agents. As a result, in this paper we do neither focus on threats and vulnerabilities that SPIT suffers by, nor on their corresponding impacts.

5. SPIT identification criteria

SPIT management requires, together with strict identification and authentication of every SIP-participating domain, the use of appropriate means ("criteria") for identifying SPIT calls and/or messages. In this section, we present a list of such criteria. They will be used as conditions rules, so as to identify a potential SPIT attack.

Furthermore, we classify them in terms of their importance, as well as of their place of applicability (i.e. at source or target domain).[3] In this context, we propose three generic categories of SPIT identification criteria, which we describe in the sequel.[4]

[3] For demonstrating the applicability of ontoSPIT, we use the "trapezoid" of the SIP operation.

[4] Although the criteria are generic and could be used in order to check all the traffic (independently of the protocol used), our analysis is focused on the SIP protocol.

Table 1
Headers used for checking caller/sender and domain trustworthiness.

	Header field	Description
Headers	From	Indicates the initiator of a SIP request
	To	Specifies the "logical" recipient of the request
	Contact	Contains a list of URLs used to redirect future requests
	Reply to	Contains a logical return URI that may be different from the header field
	Organization	Conveys the name of the organization to which the entity issuing the request or response belongs

5.1. Caller/Sender properties criteria

This category includes the criteria that are related to the attributes and/or characteristics of the caller or the sender of an instant message (e.g. sender's SIP User Agent) in terms of the addresses of the senders/callers (i.e. SIP URI or IP address), as well as the domain that the caller/sender belongs to. These criteria are:

- *Caller/Sender trustworthiness*: The address (i.e. SIP URI and/or IP address) of the caller/sender is analyzed, so as to determine if she belongs to a specific list of potential spitters. The analysis could be done through the examination of either the global spitter list that a domain controls or through the list that each callee/receiver maintains.
- *Domain trustworthiness*: The domain address (logical or IP address) is analyzed, in order to determine if the specific domain is a potential source of SPIT calls/messages. The analysis is done by examining the potential SPIT domains list, which each domain holds.

Table 1 depicts the SIP session headers fields, which could be used for checking the applicability of the proposed criteria.[5]

5.2. Message properties criteria

This category includes criteria that analyze specific call or message characteristics and patterns, in order to determine whether a call (message) is a possible SPIT.

- *Path traversal*: A call or a message might pass through many intermediates before reaching its final destination. In the SIP protocol this path is mainly denoted by the *Via* header. Thus, if a SPIT-suspicious domain is recognized in these headers, then the call or the message may be SPIT. Table 2 summarizes the headers that could be checked for path traversal checking.
- *Number of calls–messages sent in a specific time frame*: It analyzes the number of call (messages) attempts made in a specific time period by a user (e.g. the amount of INVITE requests). If this number is greater than a pre-defined threshold, then the call (message) is characterized as SPIT.
- *Receivers' address patterns*: If the receivers' addresses follow a specific pattern (e.g. alphabetical SIP URI addresses), then the call (message) can be SPIT.
- *Small percentage of answered/dialed calls*: It indicates the number of successful call completions from this caller per a pre-defined time period, which is relative to the number of failed ones.
- *Large number of errors*: When a user sends a large number of INVITEs messages and the SIP protocol returns a large number of error messages (e.g. *404 Not Found*), then this user may be a spitter.
- *Size of SIP messages*: A set of SIP messages sent by a user to other users is analyzed. If the messages have a specific size, then they may have been sent by an automated ("bot") software, therefore the call is considered as SPIT.

5.3. Semantics analysis criteria

This category includes criteria that are based on the semantic analysis of specific SIP packet headers. In more detail, we analyzed specific headers of a SIP packet, as well as the messages that might contain a message body, by using well-known content analysis techniques (e.g. Bayesian filters). These message bodies, which could be used in order to carry a SPIT call and/or message, might be encapsulated into specific *SIP Requests* and *SIP Responses* messages or in specific *Reason Phrases*. The list of the specific headers that might carry a body as a SPIT is depicted in Table 3.

5.4. SPIT criteria applicability and hierarchy

An important issue regarding the suggested criteria is in which place is more suitable for them to be applied. In this respect, we present a set of criteria that will be used in the source domain and in a preventive manner, while others will be used in the target domain and in a detective manner.

Moreover, although it is reasonable all criteria be used on both sides of a communication channel, we suggest that the identification of the exact place, where a criterion should take place, is often a better choice, in terms of bandwidth economies ensuring the QoS of VoIP communications. Table 4 provides our view regarding the applicability of the criteria.[6]

Another important issue, regarding the criteria, is the sequence they apply. This has to be dealt with, not only because not all of them can run in parallel, but mainly because of their differences, in terms of accuracy, simplicity, speed, and economy. For example, if we detect a SPIT call based on the identity of the caller (i.e. SIP URI), it is not necessary to enable other criteria. On the other hand, if specific criteria cannot identify a SPIT call, we should activate more of them. In Fig. 4, a criteria application hierarchy is suggested, based on their complexity and cost of applicability.

6. Ontology development

In this section, we will develop, in detail, the suggested ontology, so as to better facilitate the knowledge sharing amongst different SIP domains. The method we have adopted for developing the ontology includes the following six steps:

Step 1. Definition of the SPIT phenomenon, in terms of an ECA model and based on (a) the conceptual model of the domain (Fig. 2), and (b) the identification criteria.

[5] This set of criteria is similar to the while/black listing techniques used for SPIT detection.

[6] Again, we use the "trapezoid" operation of the SIP protocol.

Table 2
Headers used for checking path traversal.

	Header field	Description
Headers	Via	Indicates the path taken by the request so far
	Route	Is used to store the route set of a transaction
	Record-Route	Is used to establish a route for transactions belonging to a session

Step 2. Utilization of the ECA model information for the development of a data flow diagram (DFD) that depicts the input and output of the process. For each rule identified, we describe how information flows through this DFD.

Step 3. Identification of the main concepts and relationships, which are essential to detect and handle SPIT attacks regarding the DFD model.

Step 4. Development of event-centric parts of the ontology. All events of the ECA model are treated as individuals. For each one of them, a partial ontology is developed by employing the outcome of the previous step. Semantic relationships between the identified concepts are also added.

Step 5. Integration of the partial models in a SPIT-oriented ontology; in this step we integrate each part into a wider ontology and extend the model with additional attributes and rules, if any.

Step 6. Refinement of vocabulary and normalization of the ontology prototype; we revise the vocabulary and adjust accordingly concept attributes and relationships, so as to avoid redundancies.

For obvious reasons, we have not included in this paper a complete description of all the steps of the method. Instead, we present just one example of the ECA rules recognized in Step 1, for demonstration purposes. The complete ontoSPIT model is depicted as it was finally formed after performing all the intermediate steps.

6.1. SPIT ECA rules

The list of criteria is the kernel of the ontoSPIT approach. They will be used in such a manner, so as to lead the system to specific actions (or set of actions), when a SPIT call/message will be detected. These criteria, through the use of the ECA modeling approach, are converted to specific conditions, under which the model has to react.

ECA modeling is based on a set of specific ECA rules (/patterns) [1,6]. These rules specify actions that will be triggered when particular events take place and if certain conditions are fulfilled. The structure of the ECA rules, as well as their main components are: (a) *Event*: refers to a specific situation taking place and triggers the evaluation of a specific rule, (b) *Condition*: describes the situation, in which the actions of the rule are enabled, when specified events take place, and (c) *Action*: characterizes the operations of the rule that determine what will be the outcome (action) of the occurrence of a specific event (or a set of events).

In this context, we use ECA modeling principles to consider all criteria as specific conditions, hence we develop the rules that should be applied for detecting and handling SPIT. This means that each criterion is a condition, in which an agent has to be able to decide whether a message/call is considered as SPIT or not. Then, for each individual condition we identify the events that can generate it. The process is concluded by the description of the desired actions. We adopted specific actions to take place when a SPIT call is detected, according to [20,21].

To demonstrate this approach, we present an example of one ECA rule. Suppose that a user (alice@domain1.org) sends from her domain and through her proxy an INVITE message to another user (bob@domain2.org), who belongs to another domain. Let us suppose that the detector entity (see Section 4) is enabled in the source domain, so as to check the INVITE message before it enters in the core network. If the message leaves, then another instance of the detector entity is enabled in the target domain, in order to check if it is SPIT or not. In the source domain the holding ECA rule appears on Table 5.

Table 3
Headers, request–response messages and reason phrases for SPIT detection through semantic analysis.

SIP properties	Header field, message and reason phrase name	Description
Headers	Subject	Might be parsed by the user's software and displayed to the user's terminal
	Retry After	The optional textual comment parameter might be used to convey a SPIT message
	Call-Info	Some parameters of the specific header, i.e. purpose, icon, info and card, could be used for sending SPIT messages
	Alert-Info	Specifies an alternative ring back tone that could be used as a pre-recorder audio SPIT message
	Error-Info	The pointer to additional information, in relation to the error status response, could be set by an attacker to point to a SPIT message
	Warning	The warning text that this header contains could be replaced by a SPIT message
Messages containing a payload (message bodies). The list contains request and response messages.	INVITE	Might contain a message body
	ACK	Might contain a message body
	180 Ringing	Might contain a message body providing textual information, or conveying a ring tone (e.g. an audio file), or even animations
	183 Session Progress	Is used to convey information about the progress of the call that is not otherwise classified. The Reason-Phrase or message body could provide additional details
	200 OK	Contain returned information that depends on the method used in the request
	300 Multiple Choices	Might include a message body
	380 Alternative Service	Contain a message body in which the alternative services are described
	488 Not Acceptable Here	Might contain a message body with a description of media capabilities
	606 Not Acceptable	Might contain a message body
Reason phrases used only in response messages	182 Queued	Might contain a reason phrase, illustrating further details about the status of a call, or a message body that play an audio file, from an on-hold music palette
	183 Session Progress	Is used to convey information about the progress of the call that is not otherwise classified. The Reason-Phrase or message body could provide additional details
	200 OK	Contain returned information that depends on the method used in the request
	400 Bad Request	Is used to identify a syntax problem in more detail
	480 Temporarily Unavailable	Might contain a reason phrase indicate a more precise cause, such as why the callee is unavailable
	484 Address Incomplete	Might contain a reason phrase

Table 4
Where SPIT identification criteria should be applied?

Identification criteria	Source domain	Target domain	Comments
Caller/Sender properties criteria			
Caller/Sender trustworthiness		✓	As we check the caller/sender address, this should take place in the target domain
Domain trustworthiness		✓	As we check the sender's domain address, this should take place in the target domain
Message properties criteria			
Path traversal	✓		The messages used in this case are completed in the source domain, so we should impede them in the source domain
Number of calls-messages sent in a specific time frame	✓		If a caller/sender tries to send many *Request* messages during a specific time period it is better to impede the message to insert the core network
Receivers' address patterns	✓		If the receivers' addresses follow specific pattern, then it is better to impede the traffic to leave the source domain
Small percentage of answered/dialed calls	✓		We check the history of calls per SIP URI. It is better to control this in the source domain, because if we put it in the target domain the range of URIs that receives potential calls may be very large, so keeping history for that could be costly
Large number of errors	✓		The error messages are received by the sender, thus the control must take place in the source domain
SIP Message size	✓		If the size of many SIP Request messages is stable, then we may have a suspicious call and thus we should impede the traffic to leave the source domain
Headers semantics analysis criteria			
Headers	✓		The list of the headers presented in Table 3 is checked and is better to run this check on the source domain
Request messages	✓		These messages are sent by source domain, thus the analysis better takes place in the source
Response messages		✓	These messages are sent by target domain, thus the analysis better takes place in the target
Reason phrases		✓	These messages are sent by target domain, thus the analysis better takes place in the target

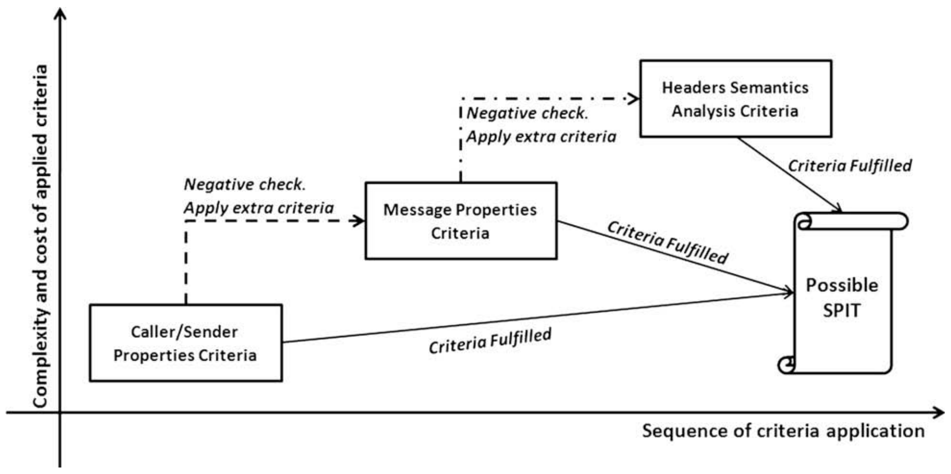
Complexity and cost of applied criteria
Sequence of criteria application
Caller/Sender Properties Criteria
Message Properties Criteria
Headers Semantics Analysis Criteria
Possible SPIT
Negative check. Apply extra criteria
Negative check. Apply extra criteria
Criteria Fulfilled
Criteria Fulfilled
Criteria Fulfilled

Fig. 4. Anti-SPIT criteria application hierarchy.

6.2. SPIT-oriented ontology: the detailed model

The conceptual model of the proposed ontology is depicted in Fig. 5. The definitions of the concepts that appear in it follow:

- *Agent-User-Proxy*: A *User* or a *Proxy* can be considered as *Agent*, as they both share common properties. Thus, they both represent subclasses of the concept *Agent*. All *Agents* belong to a *Domain* and have an *ID*, so as to be identified by humans and machines. They are individuals, which are able to perform several *Actions*.
- *Listing*: *Whitelist* and *Blacklist* are represented by this concept. Since they are both managed by *Agents* and they both contain specific *IDs* (either of *Domains* or individual *Users*), they have been conceptualized as subclasses of the concept *Listing*.
- *Action*: As every event requires at least one *participating Agent*, they are presented as *Actions*. This is a super-class of the actions *Request*, *Response*, *Messaging* and *Processing*. To support the order of *Actions* we assign the attribute *previous*.
- *Messaging*: It refers to performing both a SIP Call, as well as a SIP Instant Message. In both cases the SIP Message describes the content of the action of *Messaging*, whereas other types of *Messaging* can also refer to this *Action*. No matter the type the action follows, each message should have *sender* and *receiver*, both *Agents*.
- *Request-Response*: They represent the actual actions of requesting and responding and they are performed by *Agents*. Hence, these concepts do not stand for the SIP Request and Response packets; these packets are handled as the contents of the actions *Request* and *Response*, i.e. these SIP Messages include the needed information whenever one of these *Actions* takes place. A *Request* is done by a *User* having as target another *User* and is usually achieved through intermediate *Proxies*. Accordingly, a *Response* is initiated by an *Agent*, having as a receiver a *User* and is also achieved through *Proxies*.
- *SIP message*: Represents packets that conform to the SIP protocol. Constituent elements of these messages are *SIP Header* and *Body*.
- *Processing*: It refers to a subclass of *Action*, assigning to it two further attributes. Each *Processing* may *have parts*, which are also *Processing* actions. As this is a super-class of actions *Checking* and *Decision*, each *Processing* may consist of any combination of sequential *Checkings* and *Decisions*. What makes this action distinguishable from its broader is that for each *Processing*, i.e. for

each *Checking* and *Decision*, we need to keep track of the *Agent* it is performed by, in order to know who makes the check and/or decides whether the message is SPIT or not. This information is inferred by the entity *Stage* through their relationship *has stage*.

- *Stage*: It represents a counter that infers the *Agent* that performs each specific *Processing*. It can take three possible values, each of them indicating a specific type of performer that processes a request.
- *Checking*: It refers to the checking of *SIP Messages* performed by *Agents* in order to detect and handle possible SPIT attacks. It is possible to be preceded either by *Request* or *Allow* actions. The value of the counter *Stage* plays also an important role here, as it specifies the *Agent* that is responsible for this action and, thus, the analysis that needs to be performed to the message (see further *Criteria*/*Preferences*).
- *Criteria*: The suspicious circumstances, under which a message is considered as SPIT by a *Proxy,* are described through this entity. The analysis of the message is based on these *Criteria* whenever the *Stage* of the *Processing* indicates that it is performed by a *Proxy* server.
- *Preferences*: Contrary to the aforementioned condition, in cases when a *User* checks a message this is always done regarding her peculiar interests, i.e. her *Preferences*. A *User* is required to check the message only when the latter has successfully gone through the proxies checking without having been detected as possible SPIT and the *User* is the final receiver of the *Request*.
- *Decision*: In order for a *Decision* to take place, either a *Checking* or a *Testing* must have preceded. This concept is a super-class of the possible *Actions* that can be performed regarding the delivery of a message, i.e. *Allow*, *Block*, *Add to whitelist*, *Add to blacklist,* and *Check further*.
- *Allow*: At this point the checked message has not been detected as SPIT and so the *Agent* that has just performed the Checking, inferred once again by the value of the counter *Stage*, allows it to continue through the network, in order to reach its recipient. This action may also give a *Response*.
- *Block*: Contrary to the previous action, it conceptualizes the action of blocking of a request. It indicates that the latter is forbidden to reach its final recipient, therefore a *Response* is provided to the initial sender.
- *Add to whitelist*:It can only be performed if it is preceded by an *Allow* decision, therefore the attribute *previous* is here mandatory. As a consequence, the *Whitelist* of the *Agent* that has taken this *Decision* is updated, so as to add the sender's *ID* to it. Once again, the counter *Stage* indicates the *Agent* whose list is to be updated
- *Add to blacklist:* This can be decided only when a *Block* action precedes it, which means that its attribute *previous* is mandatory. This time the *Blacklist* of the *Agent* that has taken this *Decision* is updated, so as to add the sender's *ID* to it. The counter *Stage* indicates the *Agent* whose list is to be updated.
- *Check further-Testing*: In some occasions the *Agent* cannot make a decision about the validity of the message with certainty. A common handling of this situation is the *Agent* to decide to *Check further* the message, triggering by this way a variety of possible extra checks to be performed, e.g. challenge the sender to solve a puzzle. These further checks are initiated by the action *Testing*, a subclass of *Messaging* as shown in Fig. 5. In this case, the *sender* of the *Messaging* action is the *Agent* indicated by the entity *Stage* and the *receiver* is the *Agent* that made the initial *Request*, whose *ID* is *contained in* the *SIP Header*. The attribute *previous* is mandatory, since a *Testing* cannot take place if the *Agent* has not previously decided to make additional check.

6.3. OntoSPIT integrity constraints

We now describe a set of integrity constraints that should hold, so as the ontology be valid.

Stage: It is a counter from which we can conclude who is the *Agent* that processes each time the *Request*. Whenever the process *Allow* takes place, the value of this counter increases by 1 unit. Regarding the basic trapezoid SIP function (see Fig. 1), which depicts the simplest case of the route that each *Request* has to follow to reach its destination, the possible values of this counter are:

- *Stage = 1:* Processes take place between *User a* and *Proxy A*. All possible actions are decided by *Proxy A*.
- *Stage = 2:* Processes take place between *Proxy A* and *Proxy B*. All possible actions are decided by *Proxy B*.
- *Stage = 3:* Processes take place between *Proxy B* and *User b*. All possible actions are decided by *User b*.

Relationships previous and has previous: These relationships originate from the entities *Checking*, *Decision*, *Allow*, *Check further*, *Add to whitelist*, *Add to blacklist* and *Testing*. They are sub-relationships of relationship *previous* that has as domain and range the entity *Action*. All these relationships are mandatory, except for the one that has as domain the entity *Decision* and range the entity *Testing*.

Relationship performed by:
The *Agent* that performs the *Processing* each time is defined by considering the value of the counter *Stage*. Each value corresponds to one particular category of *Agent* that is taking *Actions.* Moreover, it can take only one value each time, i.e. the *Processing* cannot be performed by two different *Agents* at the same time.

Relationships from and to of entity Response:
From refers to the *Agent* that makes the specific *Decision* and is defined by the entity *Stage* as mentioned above. Relationship *to* always refer to the *Agent* that sent the initial *Request*, i.e. the *User* that is defined by the relationship *from* of entity *Request*.

Relationships gives:
This is originated by the entity *Block* is mandatory, i.e. action *Block* and will always give a *Response* to the sender. The relationship that originates from the entity *Allow* is applied only when this action is taken by the final receiver.

7. The OntoSPIT framework in practice

We will now transform the SPIT identification criteria into practical decision-making means. This is done with the use of

Table 5
Example of an ECA rule.

	Source domain ECA rule	Target domain ECA rule
Event	Sending INVITE message	Receiving INVITE message
Condition	User is authenticated *and* path-traversal is OK *and* headers are OK	Domain is OK *and* user is OK *and*
Action	Forward the INVITE	Establish the call

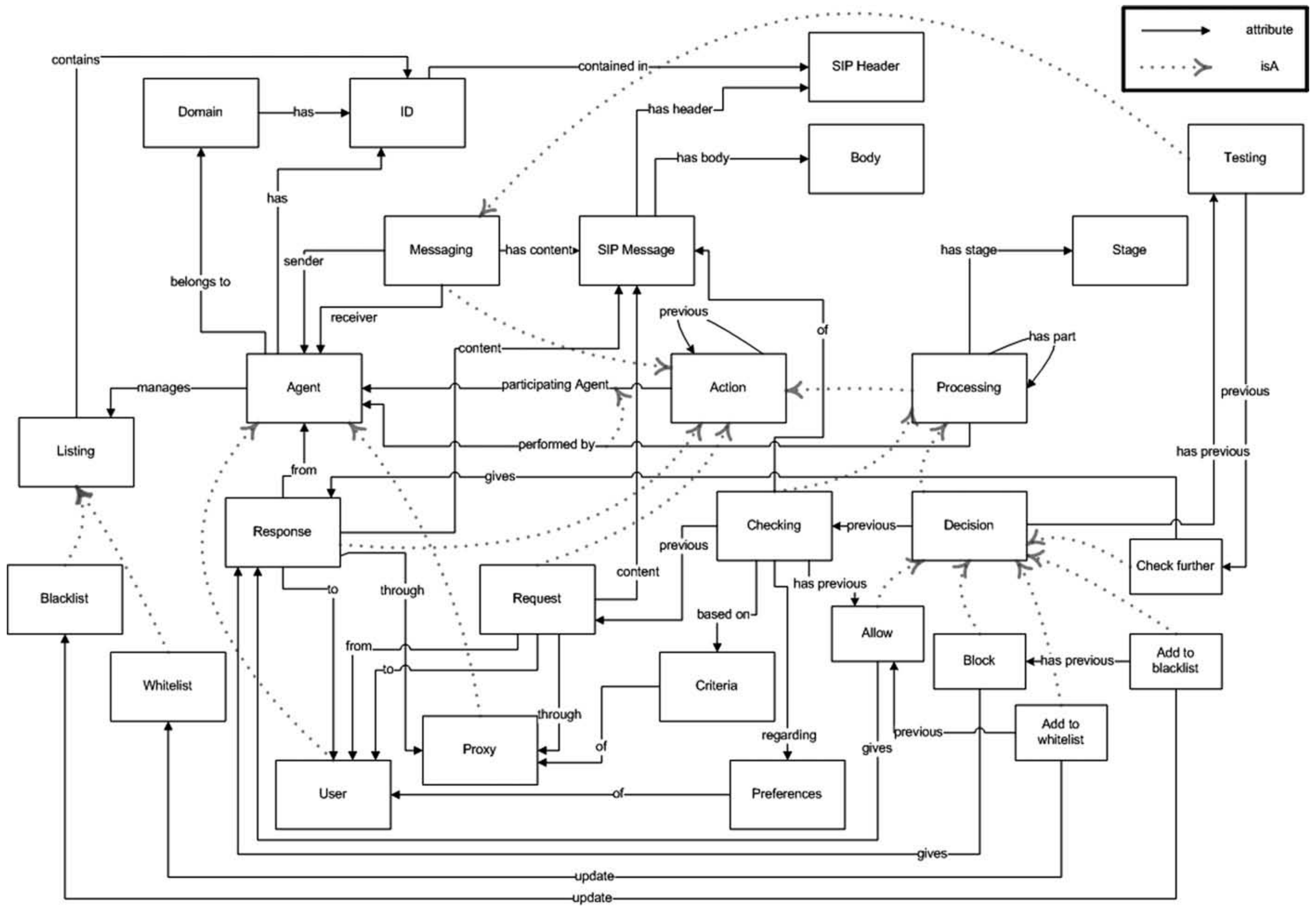

Fig. 5. Anti-SPIT ontology.

the SWRL rules (semantic web rule language) [10]. We assume that the ontology is implemented in the OWL language[7]; the SWRL allows users to write rules to reason about OWL individuals, to infer new knowledge about those individuals and also provides a rich toolbox for expressing the ECA model in an ontology environment since it supports the triggering of *Events*, *Conditions* to be satisfied and relevant *Actions*. Furthermore, the SWRL has built-in support in the Protégé environment [14]. In this context, we will transform a sample set of the SPIT identification criteria into SWRL SPIT identification rules, in order to give working examples. The decision-making process may include decision points of more than one level, thus resulting effectively to a sequence of decision points.

In the following example, the ECA rule presented in the Section 6.1 is implemented. To recall, this rule defines that when an INVITE message is sent (event), this message is legitimate if the following conditions hold true: (1) the user is successfully authenticated, (2) *AND* the Path-traversal check is OK, (3) *AND* the message headers are OK. Since SWRL does not currently support sub-queries, every condition of (1), (2), (3) has to be translated into a different rule storing the result to an intermediate variable; at a second step, a final rule will consolidate the results and decide if the message is to be transmitted or not.

The three sub-rules are depicted below, followed by the final decision rule. The most important SWRL variables in the rules (*u*, *SIP_Header* and *msg*) stand for the user identity, the SIP message header, and the overall SIP message, respectively, whereas the rest of the variables names are self-explanatory.

1. The sub-rule "*user is successfully authenticated*" is expressed as:
   SystemUser(?u)// (1.1)
   ^SystemUserAuthenticated(?u) // (1.2)
   -> SIP_UserAuthenticatedOK(?u),
   where the routine *SystemUser()* returns true if the user *u* is a legitimate user of the source system, and the routine *SystemUserAuthenticated()* returns true if the user *u* has successfully authenticated to the source system; both routines accept a parameter *u*, which refers to the user identity. The result of this sub-rule is that the *u* user is a legitimate sender at the system level; this information is stored in an intermediate variable, let it be the *u-> SIP_UserAuthenticatedOK* with the value true if the overall rule is satisfied.
2. The sub-rule "*Path-traversal is OK*" is expressed as:
   SIP_HeaderViaIsLegitimate(?SIP_Header, ?via) // (2.1)
   ^SIP_HeaderRouteIsLegitimate(?SIP_Header, ?route) // (2.2)
   ^ SIP_HeaderRecordRouteIsLegitimate(?SIP_Header, ?record-route) // (2.3)
   -> SIP_HeaderPathTraversalIsLegitimate(?msg),
   where the routine *SIP_HeaderViaIsLegitimate()* returns true if the field *Via* is not included in a list of non-permitted hops, the routine *SIP_HeaderRouteIsLegitimate()* returns true if the field *Route* is not included in a list of non-permitted routes, and the routine *SIP_HeaderRecordRouterIsLegitimate()* returns true if the field *Record-Route* is not included in a list of

[7] OWL web ontology language reference (available at www.w3.org/TR/owl-ref/).

non-permitted record-routes. The result of this sub-rule is that the path traversal details of the SIP message header are legitimate ones; this information is stored in the *msg->SIP_HeaderPathTraversalIsLegitimate* intermediate variable with the value true if the overall rule is satisfied.

3. The sub-rule "*HeadersOK*" is expressed as:
 SIP_HeaderFromIsLegitimate(?SIP_Header, ?from) // (3.1)
 ^SIP_HeaderToIsLegitimate(?SIP_Header, ?to) // (3.2)
 ^SIP_HeaderContactIsLegitimate(?SIP_Header, ?contact) // (3.3)
 ^SIP_HeaderReply-ToIsLegitimate(?SIP_Header, ?reply-to)// (3.4)
 ^ SIP_HeaderOrganizationIsLegitimate(?SIP_Header, ?organization)// (3.5)
 -> SIP_HeadersAreLegitimate(?msg),
 where the routine*SIP_HeaderFromIsLegitimate()* returns true if the field *From* is not included in a list of non-permitted source adresses, the routine *SIP_HeaderToIsLegitimate()* returns true if the field *To* is not included in a list of non-permitted destinations, the routine *SIP_HeaderContactIsLegitimate()* returns true if the field *Contact* is not included in a list of non-permitted contacts, the routine *SIP_HeaderReply-ToIsLegitimate()* returns true if the field *Reply-To* is not included in a list of non-permitted reply addresses, and the routine *SIP_HeaderOrganizationIsLegitimate()* returns true if the field *Organization* is not included in a list of non-permitted organizations. The result of this sub-rule is that the address-related details of the SIP message header are legitimate ones; this information is stored in an intermediate variable, let it be the *msg->SIP_HeaderAddressesAreLegitimate* with the value true if the overall rule is satisfied.Finally, the *overall rule* is evaluated as follows:
 (?u->SIP_UserAuthenticatedOK)
 ^(?msg->SIP_HeaderPathTraversalIsLegitimate)
 ^(?msg->SIP_HeaderAddressesAreLegitimate)
 -> Forward_INVITE_msg(?msg)
 Having similar rules, an automated approach can decide whether the SIP message under question is a SPIT message or not. The reader can observe the scalability of the approach, thus giving the rule developer the capability to build as complex rules as necessary, so as to achieve the message filtering goal by effective reasoning.

8. Conclusions and further research

VoIP technology and its services seem to gain serious attention, especially after the introduction of the SIP protocol. In addition to the advantages introduced by VoIP technology, there are specific concerns. Amongst them, the SPIT problem is one that has been identified as a serious drawback for the VoIP environments. In order to better counter and manage the SPIT phenomenon in VoIP environments, it is important to address and evaluate its issues in a holistic approach.

In this paper, we set the foundations for establishing a knowledge-based, ontology-centric framework (ontoSPIT) with respect to the SPIT management issues. The proposed framework may support VoIP stakeholder's activities with respect to SPIT detection issues, as well as to the selection of certain actions and controls for countering SPIT attacks.

In addition, our approach provides a common understanding of SPIT domain, and reusable SPIT-related knowledge interoperability, aggregation, and reasoning. Moreover, and in contrast to the majority of the existing anti-SPIT mechanisms and approaches, the proposed ontology provides reusability and knowledge sharing amongst humans and machines.

Regarding future work, we plan to extend our model, so as to include specific SPIT related threats and vulnerabilities, in an effort to provide a complete framework regarding SPIT management. Furthermore, we are looking forward to incorporate the ontology in more general security ontologies, so as to enhance the existing models, in order to also incorporate in them SPAM/SPIT management processes.

References

[1] J. Bemmel, P. Dockhorn, I. Widya, Paradigm: event-driven computing, White Paper, Lucent Technologies, December 2004.
[2] D. Brewer, S. Thirumalai, K. Gomadam, K. Li, Towards an ontology driven spam filter, in: Proc. of the 22nd IEEE International Conference on Data Engineering Workshops (ICDEW'06), USA, April 2006, 79 pp.
[3] W. Cohen, V. Carvalbo, T. Mitchell, Learning to classify email into "speech acts", in: Proc. of Empirical Methods in Natural Language Processing (EMNLP 2004), Spain, July 2004, pp. 309–316.
[4] S. Decker, M. Erdman, D. Fensel, R. Studer, Ontobroker: ontology based access to distributed and semi-structured information, in: Proc. of the 8thWorking Conference on Database Semantics, Kluwer Academic Publishers, New Zealand, 1999, pp. 269–351 (January).
[5] D. Geneiatakis, C. Lambrinoudakis, An ontology description for SIP security flows, Computer Communications 30 (6) (2007) 1367–1374 (January).
[6] N.H. Gehani, H.V. Jagadish, O. Shmueli, Event specification in an active object-oriented database, in: Proc. of the ACM International Conference on Management of Data (SIGMOD '92), USA, 1992, pp. 81–90.
[7] T. Gruber, Toward principles for the design of ontologies used for knowledge sharing, in: Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, Dordrecht, 1993 (March).
[8] N. Guarino, Understanding, building, and using ontologies: a commentary to "using explicit ontologies in KBS development, International Journal of Human and Computer Studies 46 (3/4) (1997) 293–310.
[9] C. Holsapple, K. Joshi, A collaborative approach to ontology design, in: Com. of the ACM 45 (2) (2002) 42–47.
[10] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, SWRL: a semantic web rule language combining OWL and RuleML, The DARPA Agent Markup Language Homepage (November) (2003).
[11] J. Kim, D. Dou, H. Liu, D. Kwak, Constructing a user preference ontology for anti-spam mail systems, in: Proc. of the 20th Conference of the Canadian Society for Computational Studies of Intelligence, Canada, May 2007, pp. 272–283.
[12] J. Mallios, S. Dritsas, B. Tsoumas, D. Gritzalis, Attack modeling of SIP-oriented SPIT, in: Proc. of the 2nd IEEE-IFIP International Workshop on Critical Information Infrastructures Security (CRITIS'07), Spain, October 2007.
[13] G. Marias, S. Dritsas, M. Theoharidou, J. Mallios, D. Gritzalis, SIP vulnerabilities and antiSPIT mechanisms assessment, in: Proc. of the 16th IEEE International Conference on Computer Communications and Networks (ICCCN'07), IEEE Press, USA, 2007, pp. 597–604. August.
[14] N. Noy, R. Fergerson, M. Musen, The knowledge model of Protégé-2000: Combining interoperability and flexibility, in: Proc. of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2000), France, October 2000, pp. 17–32.
[15] N. Noy, D. McGuiness, Ontology development 101: a guide to creating your first ontology, Technical Report SMI-2001-0880, Stanford Medical Informatics, Stanford University, USA, March 2001.
[16] J. Rosenberg et al., Session initiation protocol, RFC 3261, June 2002.
[17] J. Rosenberg, C. Jennings, Session initiation protocol (SIP) and spam, RFC 5039, Network Working Group, January 2008.
[18] S. El Sawda, P. Urien, SIP security attacks and solutions: a state-of-the-art review, in: Proc. of the IEEE International Conference on Information & Communication Technologies: From Theory to Applications (ICTTA'06), Syria, vol. 2, April 2006, pp. 3187–3191.
[19] K. Taghva, J. Borsack, J. Coombs, A. Condit, S. Lumos, T. Nartker, Ontology-based classification of email, in: Proc. of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2003), USA, April 2003, pp. 194–198.
[20] H. Tschofenig, T. Froment, D. Wing, H. Schulzrinne, Requirements for authorization policies to tackle spam for internet telephony and unwanted traffic, Internet Draft (June) (2007).
[21] H. Tschofenig, H. Schulzrinne, T. Froment, G. Dawirs, Anti-SPIT: a document format for expressing anti-SPIT authorization policies, Internet Draft (February) (2007).
[22] VOIPSA, VoIP security and privacy threat taxonomy, October 2005.
[23] S. Youn, D. McLeod, Efficient spam email filtering using adaptive ontology, in: Proc. of the 4th IEEE International Conference on Information Technology (ITNG'07), USA, April 2007, pp. 249–254.