A Formally Verified Mechanism for Countering SPIT

Yannis Soupionis[1], Stylianos Basagiannis[2], Panagiotis Katsaros[2], DimitrisGritzalis[1]

[1] Information Security and Critical Infrastructure Protection Research Group,
Dept. of Informatics, Athens University of Economics and Business (AUEB)
76 Patission Ave., Athens, GR-10434, Greece
{jsoup, dgrit}@aueb.gr

[2] Dept. of Informatics, Aristotle University of Thessaloniki
Thessaloniki, GR-54124, Greece
{basags, katsaros}@csd.auth.gr

Abstract. Voice over IP (VoIP) is a key technology, which provides new ways of communication. It enables the transmission of telephone calls over the Internet, which delivers economical telephony that can clearly benefit both consumers and businesses, but it also provides a cheap method of mass advertising. Those bulks unsolicited calls are known as SPam over Internet Telephony (SPIT). In this paper we illustrate an anti-SPIT policy-based management (aSPM) mechanism which can handle the SPIT phenomenon. Moreover, we introduce a formal verification as a mean for validating the effectiveness of the aSPM against its intended goals. We provide model checking results that report upper bounds in the duration of call session establishment for the analyzed anti-SPIT policy over the Session Initiation Protocol (SIP) and prove the absence of deadlocks.

Keywords: Spam over Internet Telephony (SPIT), Policy management, Model checking, Formal verification, Voice over IP (VoIP).

1 Introduction

VoIP (Voice over Internet Protocol) is the general term that describes two-way transmission of voice over the Internet in real (or near-real) time. Adoption of VoIP as a mainstream communication mean brings huge benefits to organizations, like a reduced call cost as well as smooth integration with the current internet infrastructure and the provided services. In VoIP, the voice signal is digitized and then transmitted over the Internet in packets, as it happens when sending an email with a sound file attachment.

Therefore, VoIP can be thought as an evolution of the email service and it seems that it also faces the same problems and challenges as its predecessor. The main problem of the email is SPAM. The term used in VoIP instead of SPAM is SPIT [1, 2] and it is derived from ‘‘Spam over Internet Telephony’’. SPIT [2] refers to all unsolicited and massive scale attempts for initiating a session that establishes voice communication with an oblivious user. If the user answers the call, the “spitter” broadcasts his message in real time. SPIT comes in three different types, namely: (a) call SPIT, (b) instant message SPIT and (c) presence SPIT. SPIT is expected to become a serious threat to the spread of VoIP in the forthcoming years. This threat stems from the well known email spam problem and existing botnets that in addition to SPAM emails are re-programmed for initiating VoIP calls. This is the reason why companies, like NEC

and Microsoft, invest into developing mechanisms for tackling SPIT [3, 4]. Currently, several SPIT prevention methods have been proposed, but research on SPIT prevention is still in its infancy.

In a recent article [5], an anti-SPIT Policy Management (aSPM) mechanism for detecting and handling SPIT calls was proposed. This paper introduces formal verification as a mean for validating the effectiveness of an anti-SPIT Policy against its intended goals. Moreover, we provide model checking results with upper bounds in the duration of call session establishment, which show that the aSPM mechanism does not affect dramatically the time needed for session establishment and prove the absence of deadlocks.

Model checking communication protocols like SIP is based on a finite state model representing at a suitable level of abstraction the behavior of a system where the protocol runs in one or more concurrent protocol sessions. Correctness properties are expressed as assertions or temporal logic formulae that are algorithmically validated by state exploration across all possible execution paths. Operational errors or security flaws can be detected in the form of safety or liveness property violations that reflect unexpected behavior. For a violated property, the analyst gets a counterexample, i.e. an execution path to the detected invalid state that provides valuable feedback for redesigning the system. To the best of our knowledge, our work is the first attempt towards formal verification of anti-spamming countermeasures.

In Section 2 we describe the methodology for formally verifying the anti-SPIT policy mechanism at hand. Section 3 outlines the analyzed anti-SPIT policy called aSPM. In section 4 we report results obtained from a real-time SIP session experiment, in order to formulate valid modeling assumptions and property specifications. Section 5 introduces the SIP-aSPM model developed in the SPIN model checking tool [6] and the correctness properties that are validated. In section 6 we provide the obtained verification results and the paper concludes with a critical view of the outcomes and a discussion on future research prospects.

2 Methodology

Figure 1 provides an overview of the adopted methodology for formally verifying the anti-SPIT policy at hand.

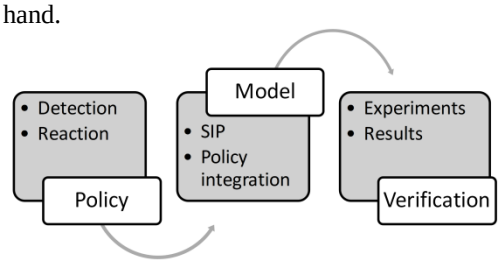

Figure 1: Methodology for formally verifying anti-SPIT policy

The first step towards the development of an anti-SPIT policy is the detailed design of the detection and the reaction processes that comprise the policy. It is also necessary to describe how the policy is integrated into a SIP VoIP environment.

The next step involves the development of a finite state model for the SIP protocol that in our case takes place within the SPIN model checking tool [6]. SPIN is an automated model checker that aims to efficiently verifying (distributed) software systems. Its success has been proved through several case studies [7, 8], where SPIN is used to trace design errors in distributed systems. It provides the capability of reporting flaws like model deadlocks, unspecified receptions, flags incompleteness, race conditions, unwarranted assumptions about the speeds of processes [9] and others.

In our problem, one of the research goals is to verify that the anti-SPIT policy does not create "invalid" states in the SIP communication process and for this reason our model reflects in detail the call session establishment and the SIP message exchanges.

The anti-SPIT policy is then integrated into the model by implementing the behavior of entities that enforce the policy, as well as the message overload incurred by the policy reaction process. Model parameters concerning the cost in time for the exchanged messages have been derived from the measurements made in the conducted call session experiments.

Finally, we verify property specifications like the absence of deadlock in execution scenarios with one or more concurrent protocol sessions and the requirement of timely session establishment.

3 Policy description

While SIP is considered to be one of the most widespread protocols for multimedia session maintenance there are reports [10] that describe SIP security errors and therefore raise the need for a security policy mechanism. Our work on the anti-SPIT Policy Management (aSPM) mechanism [5] is based on identifying potential threats that can exploit protocol weaknesses. The SIP protocol provides a connection-based technology, i.e. a call setup phase has to take place before any voice traffic is carried across the IP network. Signaling commands are used in establishing and terminating a call, as well as in providing some special features such as call forwarding and call waiting. Therefore, the nature of SIP only allows for proactive countermeasures against SPIT, like for example policy rules, which can properly adjust the reaction of the elements that participate in the negotiation and the communication process. Based on this assumption, the anti-SPIT policy takes the form of an obligation policy [11] that includes a set of rules and the appropriate countermeasures obeying the condition-action model [12]. Therefore, aSPM consists of the detection and the reaction processes.

The detection process should be able to detect a SPIT call or message, when it reaches the callee's domain or User agent client (UAC). SPIT detection depends on pre-identified criteria and it is influenced by the preferences-interests of the callee, in terms of the attributes of the call or message, or the anti-SPIT policies of the callee's domain. In order to define the detection rules (conditions) the SIP-targeted SPIT threats were identified by an in-depth analysis. The result of the SIP analysis was a number of well-defined SPIT-related threats and vulnerabilities, in accordance with the SIP RFC [13]. Afterwards, attack scenarios were created based on SPIT-oriented

attack graphs. The attack scenarios were converted into specific attack signatures, which formed the base for the condition creation (policy key element).

The reaction process applies specific actions in case a call or a message has been detected as SPIT. These reactions, i.e. the application of specific anti-SPIT measures, are enforced by the anti-SPIT policies of the callee's domain. Most of them are SIP messages [14, 15], because the policy should: (a) be transparent to the administrators and users and (b) keep to a minimum the participation of other applications during message handling. An example of a proposed condition - action is illustrated in Table 1.

Table 1 A condition and its suggested countermeasures

Caller's device (UAC) receives a response with message code 300 (Multiple Choices) and there is a SIP header in *Contact* field which is not part of the *From*
1. The UAC rejects the call and returns a message 403 (Forbidden) 2. The UAC rejects the call and returns a message 606 (Not Acceptable)

The countermeasures that are taken into account in this research work are the proposed SIP messages. The SIP message response codes, which are used by the policy, are the 4xx request messages and the 6xx global failures. These messages direct the caller or his/her domain to resend the previously dispatched message so as to meet the necessary requirements of the callee or his/her domain.

3.1 Policy Integration in a VoIP environment

In this section we describe how the anti-SPIT policy can be integrated in a SIP infrastructure in order to measure the anti-SPIT policy impact to the overall SIP signaling phase. For this purpose, the policy is formally described with an XML schema that includes the attack scenarios (conditions) and the applied countermeasures (actions).

The implementation approach is depicted in Figure 2 and includes two additional modules:

1. The XML parser which reads the XML policy instance into memory and provides easy access to tag values of the document.
2. The policy enforcement, which has as input the parsed xml document, together with the message attributes. The module checks all the policy conditions, so as to find out which are fulfilled (first a SIP message is received and parsed, and then the message attributes are checked against the policy element) and if one or more conditions are met, then the associated action (described in the fulfilled policy element) takes place.

The proposed policy integration will obviously incur time delays to call establishment. Therefore in the next section we report the results obtained in an experiment scenario, which illustrate that the additional time needed due to the aSPM infrastructure is a minor effect on the call establishment procedure.

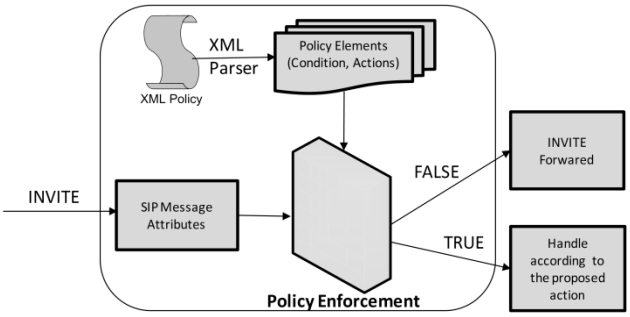

Figure 2: The aSPM Architecture

4 Experiments

In order to evaluate the impact of policy enforcement in terms of the cost in time for the SIP negotiation process, a laboratory environment was deployed. The SIP server where the policy was implemented is the SER server [40] one of the most widely used SIP servers. It is an open source software product, currently used by organizations including Columbia University, Swiss Federal Institute of Technology, etc. [16, 17].

The laboratory (Figure 3) that we have installed consists of the following fundamental entities:

1. Two (2) SIP SER servers. They have been customized in order to register users, redirect SIP messages and establish calls. The PCs used for setting up the SER servers were a Pentium 4, 2.8GHz with 1GB RAM running the Fedora 9 operating system.
2. A policy module. It has been installed in the SIP server, at the "logical" entrance of the network environment and includes the domain XML policy.
3. Two (2) soft phone clients that have been part of the VoIP callee's domain. These clients are active, which means that they are ready to interact with incoming calls. The used soft-phone software is called twinkle [18] and it is an open-source product.
4. An external client. This client is programmed to make new calls to the internal clients. The calls are initiated by using SIPp [19] which establishes calls as well.

The experimentation scenario is related to our goal to verify all possible execution paths in a SIP setting with one to two concurrent protocol sessions. Since our policy module accepts all the incoming and outgoing SIP traffic of the VoIP infrastructure, we made two tests: a) the aSPM was enabled and b) the aSPM was disabled.

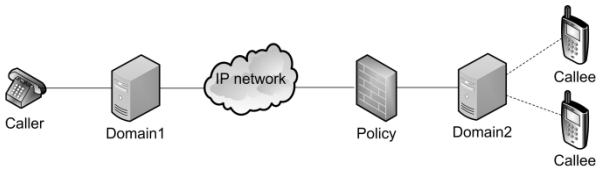

Figure 3: Laboratory Environment

The total number of initiated calls was 10.000 and the needed time for the message exchanges between the participating entities is illustrated in Table 2. The participating entities are the two domains (D), the User Agent Client (UAC) and the policy module (aSPM). The first two columns show the time costs when the aSPM is disabled, and the last two when the aSPM is enabled. The third column shows the time needed for the caller's UAC to send the final CK message, where the policy does not interfere.

The main assumption is that the time needed to send a message between two communication entities does not depend on the role of the involved entities in the communication, i.e. whether they are the sending or the receiving entity. For example the same time is needed to send a message from Domain1 to Domain2 with the time needed to send a message from Domain 2 to domain 1.

Table 2: Time needed for exchanging messages

Channel / **Time (msec)**	UAC – D	D –D	UAC –UAC	UAC – aSPM	D- aSPM
Minimum	97	153	97	286	321
Average	102	154	98	387	428
Maximum	106	162	98	432	642

The times shown have been used as parameters in the formal verification model in order to assess how the message exchange affects the requirement of timely session establishment. The reported times may be altered depending on the hardware features that are assumed in the described experiment. However, the proposed verification approach is still effective for other parameter sets.

5 Formal Verification

Formal verification took place in five successive steps:

1. First, we developed a model for a single SIP protocol session according to the protocol specification in [13]. While SIP entities may interact with a significant number of messages, we aimed to an abstract representation of the protocol operation. More precisely, we omitted message manipulation functions that can have a negative impact to the model's state space, without being within the scope of the intended aSPM analysis.
2. From the experiment of section 4, we collected data values for the parameters representing time in the modeled SIP message exchanges. These values were attached to executable actions that reflect the expected behavior in a single protocol session.
3. We created a second protocol entity (Callee) that participates in a parallel SIP session with the protocol initiator (Caller).
4. The aSPM policy was then integrated into the SIP model.
5. SPIN's state exploration functions allowed model checking for a possible deadlock, for reachable states that potentially violate SIP functional properties (there-

fore called invalid states), as well as for a Linear Temporal Logic [9] formula encoding the requirement of timely session establishment.

5.1. Assumptions and property specification

We assumed that in all cases SIP messages are delivered to the intended recipient, thus excluding the presence of a man-in-the-middle intruder entity, which is an open research prospect. The initiator of the communication (Caller of the SIP) belongs to Domain1 and both responders (Callees of the SIP) belong to the same domain, namely Domain2.

Since we intended to verify a system execution scenario with two concurrent protocol sessions and in order to avoid the possible state space explosion [23], we focused only on the SIP messages that are related to the behavior of the aSPM policy. Thus, we omitted the message manipulation functions after having checked that they do not affect the SIP execution outcome and they are not affected by the policy reactions. For each protocol participant, we encoded the messages that can be sent and might be received in all protocol steps. These messages are:

1. `INVITE` messages that are created by the Caller
2. `2xx` successful response message created by the Callee
3. `3xx` redirection response messages created by the Domain 2 server
4. `4xx` request failure messages created by the Domain 2 server
5. `6xx` global failure messages created by the Domain 2 server
6. `ACK` messages created by the Caller

The developed protocol model was verified for the absence of deadlocks, i.e. the resulting executions either terminate with successfully completed initiated sessions or with failed sessions, due to dispatched messages that declare an error. Thus, the produced state space includes error execution paths that are possible in a real SIP communication. Formal analysis can either take into account these paths or not, depending on property specifications that are model checked.

Apart from the absence of deadlock we also studied call establishment timeliness for all error-absent execution paths and for both versions of the model, i.e. the SIP model and the model with the aSP policy. In this way, we aim to verify that the aSP mechanism does not incur unacceptable message overload that can undermine call establishment timeliness.

5.3. SIP-aSPM model description

The SIP-aSPM model was encoded in SPIN's model description language called PROMELA [9] and comprises six interacting processes (`proctypes`): i)the Caller (`UAC`) who initiates one or two protocol sessions, ii) the `Domain1` where the Caller belongs to, iii) two (2) Callees namely `UAC_1` and `UAC_2` that communicate with the initiator, iv) the `Domain2` where the Callee belongs to and (v) a `proctype` stopwatch `Timer` that measures the time for SIP call establishment with and without the aSPM policy. The model also defines `rendezvous` communication channels,

in order to allow synchronous message passing between the aforementioned processes.

Based on the experiment results of Table 2 we defined discrete time values for each message delivery action. Values `time_i` with i={1,2,3,4,5} determine the global timer updates implemented by proctype `Timer` that take place in every message exchange. For example, every message exchange between the UAC and UAC_1 or UAC2 results in a time_3 increase (98 msec) of the global timer. By tagging the modeled message exchanges with time values we can derive verification results for all possible execution paths that depend on the decisions of the aSPM policy.

The model allows model checking properties over the combined execution of up to two parallel sessions, which can be non-deterministically initiated at any time by the caller's UAC. In the two-session execution scenario shown in Figure 4, the UAC entity eventually selects both UAC_1 and UAC_2 for establishing distinct SIP media sessions. More precisely, UAC non-deterministically selects the first Callee (UAC_1 or UAC_2) and dispatches an INVITE message (named INIVTE_1 if the recipient is UAC_1 and INVITE_2 if the recipient is UAC_2). The caller's Domain 1 simply forwards the message to Domain 2. Upon arrival of an INVITE message to Domain 2, there are three possible responses, namely: (i) redirection of the Callee entity (message 3xx), (ii) request failure (message 4xx) or (iii) global failure (message 6xx). Redirection involves reform of the received INVITE message, in order to incorporate the new Callee's address. Caller responds only to the two mentioned failure messages (4xx and 6xx) by resending a new INVITE message, but this happens only when he has not already received more than three (3) error messages. In the latter case Caller drops the call and session establishment fails.

If there is no error, Domain 2 forwards the INVITE message to the Callee's address and waits for the callee's approval. Callee produces a 2xx response message (shown as m200_OK in Figure 4) and sends it to the Caller via the Domain 2 server. Domain 2 forwards the message to Domain 1 and consequently the server handles it to the Caller.

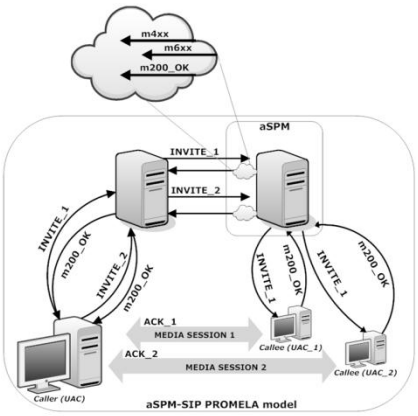

Figure 4: Message exchanges in the SPIN SIP-aSPM model

When Caller receives the expected 2xx message, he sends an ACK message directly to the Callee, thus establishing a new media session. At any time of the described call establishment procedure, the Caller may non-deterministically initiate a second SIP session with the other Callee. Both sessions may be completed successfully or otherwise any of them can terminate due errors reported from Domain 2, upon receipt of the dispatched INVITE messages.

5.4. Verification Results

In order to verify the previously specified properties (section 5.1) we have, at first, to define them using the appropriate temporal logic. SPIN model checker [6] is an automated on-the-fly model checker that is used to trace logical errors in distributed systems, i.e. communication protocols. It also supports the verification of user-defined properties as formulations of the Linear Temporal Logic (LTL). Using LTL we are capable of model checking the developed SIP-aSPM model for correct-ness requirements such as, properties relating timeliness. In detail, we a) create appropriate LTL formulas describing the desired SIP properties, and b) define system process invariants (using assertions) that are validated throughout the verification process.

Timely completion of media session establishment was checked by formula Q1, where:

```
Q1: [](q → p)
```

with → representing the left associative implication, `[]` for the temporal operator `always` and `p, q` user-defined symbols with the following values:

```
#define p time<4000, #define q (sessions==0)
```

Model variable sessions – with initial value 2 – changes with the number of sessions that are successfully completed. If sessions==0 then both SIP sessions end with success. Value 4000 represents the allowed duration in msec for successful call establishment. Formula Q1 is therefore interpreted as:

"*If in a reachable state both sessions have finished with success, this happens in less than 4.000 msec*".

For Q1, SPIN generated the corresponding never claim and verified that the property holds in all possible executions. Table 3 reports the obtained model checking results for a series of properties that are expressed as variants of formula Q1. Shown results are accompanied by state space statistics under various state exploration strategies with or without partial order reduction (an optimization for ignoring all superfluous interleavings). We observe that when the antiSPIT mechanism is enabled, the verification analysis produced no error across execution paths, where at least one session is successfully completed in less than 6,5 sec. Similarly, there were no errors across paths where both sessions are successfully completed in less than 10 sec.

When aSPM is disabled, the provided results report upper time boundaries for successfully establishing one or two call sessions.

Apart from the Q1 LTL formula, we also used assertions (defined as active monitor proctypes), in order to derive additional verification results. With assertions we can detect invalid end-states, like when sending more than two error messages of type 4xx or 6xx. We detected an invalid end state, which is reported in Table 3 in the state space with 13172 states, thus realizing that the Domain 2 can send more than two 4xx or 6xx messages. This result validates the error prone design of our model, i.e. a realistic representation of all cases where there is an unexpected termination of the SIP protocol due to multiple error messages 4xx or 6xx. If no error messages are produced by entity Domain 2, the SIP-aSPM model terminates correctly.

Finally, we successfully verified the absence of deadlocks by using the corresponding SPIN model checking option.

Table 3: Results of the SIP-aSPM Formal Verification

Property Description	States	Transitions	Memory (MB)	Property Definition	Verification Result
At least one (1) Session Successful Completion before 6500 msec (aSP Enabled)	3.8e+06	7.181e+06	585.309	Q1	Valid
Parallel Session Successful Completion before 10000 msec (aSPM enabled)	3.8e+06	7.246e+06	616.11	Q1	Valid
Full State Space with no errors (Deadlock absence)	3.8e+06	7.181e+06	585.309	-	Valid
Partial State Space Search before 6000 msec (with invalid end-states)	13172	13172	0.924	Assertion	Invalid
Full State Space Search for one (1) successful session completion before 3000 msec (aSPM Disabled)	3.8e+06	9.238e+06	600.712	Q1	Valid
Parallel Session Successful Completion before 6000 msec (aSPM Disabled)	3.8e+06	7.246e+06	616.114	Q1	Valid
Partial State Space Search with more than two (4xx or 6xx) message dispatches (unexpected errors)	956	2163	0.149	Assertion	Invalid

6 Related work

In this section, we briefly survey some relevant techniques and approaches, which focus on modeling policy infrastructure and verification of SIP protocol properties.

Zave [20] has presented three formal, state-oriented models for SIP and has discussed five cases where the SIP standards are incomplete. She proposed solutions to all detected problems, but her work concerns different SIP messages from those studied in our work (e.g. the update message that concerns the re-negotiation session establishment).

Liu [21] has modeled and analyzed SIP INVITE transactions over an unreliable medium. Then by examining the state space of the model it was found that the INVITE transaction is not free of livelocks and dead codes, as it is in the case of a reliable medium. The results of this work are not directly connected to our work, since with the SIP messages involved it is not possible to produce SPIT. As a consequence, this approach cannot be used for studying the behavior of our policy implementation.

Finally, Schaeffer-Filho et al. [22] have modeled a specific policy interaction, but for a totally different system. They defined a formal model for the design of Self-Managed Cells (SMCs) with a consistent policy which assists to the collaborations across SMC. The created model allowed them to verify the correctness of the anticipated SMC interactions based on policy decisions before these interactions are implemented or deployed in physical devices (e.g. PDAs, mobile phones, sensors).

7 Conclusions

The Session Initiation Protocol has become a popular communication system in Voice over IP applications, as it is widely used for establishing and maintaining multimedia sessions over the Internet. One of the obvious potential problems of VoIP applications is the growth of the SPIT phenomenon [2]. This work proposes a formally verified SPIT policy mechanism over the SIP protocol. We used the SPIN model checking environment, in order to fully analyze the temporal behavior of the policy and protocol interactions.

The paper discusses the proposed aSPM methodology and the SIP-aSPM model development. We define the basic properties where the verification was focused. The obtained model checking results provide evidence that the aSPM mechanism does not affect dramatically the time needed for call establishment in two concurrent SIP sessions. We note that the parameters used depend on the hardware features assumed in the conducted laboratory measurement.

The proposed analysis can be used for deriving upper bounds in the duration of media session establishment for SIP-aSPM systems. Also, it is open to extensions that will provide additional verification results. We consider studying error scenarios that will be generated with a powerful intruder model entity over the SIP session establishment. In this way, we will be able to validate whether the proposed aSPM mechanism over SIP is vulnerable to intruder attacks that may subvert the SIP protocol's functionality. Further experimentation and improvements in the proposed policy may be done, but in any case the formal verification method seems to be a valuable mean for the design of effective antiSPIT policies.

References

1. S. Sawda, O. Urien, “SIP security attacks and solutions: A state-of-the-art review”, in *Proc. of the IEEE International Conference on Information and Communication Technologies: From Theory to Applications* (ICTTA ’06), Vol. 2, pp. 3187-3191, April 2006.
2. J. Rosenberg, C. Jennings, *The Session Initiation Protocol (SIP) and Spam*, Network Working Group, RFC 5039, January 2008
3. J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, T. Ewald, “Detecting SPIT Calls by Checking Human Communication Patterns”, in *Proc. of IEEE International Conference on Communications* (ICC’07), pp. 1979-84, United Kingdom, 2007.
4. D. Graham-Rowe, *A Sentinel to Screen Phone Calls Technology*, MIT Review, 2006 (accessed November 8, 2009).
5. Y. Soupionis, S. Dritsas, D. Gritzalis, "An adaptive policy-based approach to SPIT management", in *Proc. of the 13th European Symposium on Research in Computer Security* (ESORICS 2008), Lopez J., Jajodia S. (Eds.), pp. 446-460, Springer, Malaga, October 2008.
6. G. Holzmann, “The model checker SPIN”, *IEEE Transaction on Software Engineering*, 5/23 (1997) 279–295.
7. The SPIN model checker website, at http://spinroot.com/ (last access: 23rd of May 2010)
8. S. Basagiannis, P. Katsaros, A. Pombortsis, "Intrusion Attack Tactics for the Model Checking of e-Commerce Security Guarantees", in *Proc of the 26th International Conference on Computer Safety, Reliability and Security* (SAFECOMP ’07), Lectures Notes in Computer Science 4680, 238-252, Springer Verlag, Germany, September 2007.
9. G. Holzmann, *The SPIN Model Checker - Primer and Reference Manual*, Addison Wesley, 2003.
10. T. Walsh, D. Kuhn, *Challenges in securing voice over IP*, National Institute of Standard and Technology (NIST), USA.
11. M. Sloman, E. Lupu, “Security and management policy specification”, *IEEE Network*, Special Issue on Policy-Based Networking 16(2), pp.10–19, 2002
12. M. Strembeck, “Embedding policy rules for software-based systems in a requirements context”, in *Proc. of the 6th IEEE International Workshop on Policies for Distributed Systems and Networks* (POLICY ‘05), June 2005
13. J. Rosenberg, et al, *Session Initiation Protocol (SIP)*, RFC 3261, June, 2002.
14. Cisco Systems, Session Initiation Protocol gateway call flows and compliance information SIP messages and methods overview (http://www.cisco.com/application/pdf/ en/us/guest/ products/ps4032/c2001/ccmigration_09186a00800c4bb1.pdf)
15. Cisco Systems, SIP Messages and Methods Overview (http://www.cisco.com/univercd/cc/ td/doc/product/software/ios122/rel_docs/sip_flo/preface.pdf)
16. SER server version 2.0, (www.iptel.org/ser), (retrieved Mar. 20, 2009)
17. Example SER deployments, http://mit.edu/sip/sip.edu/deployments.shtml
18. Twinkle softphone(http://www.twinklephone.com) retrieved 25/08
19. SIPP traffic generator for the SIP protocol (http://sipp.sourceforge.net/) (retrieved 30.09.09)
20. P. Zave, “Understanding SIP through model-checking”, in *Proc. of the 2nd International Conference on Principles, Systems and Applications of IP Telecommunications*, pp. 256-279, Springer-Verlag LNCS 5310, 2008.
21. L. Liu, Verification of the SIP Transaction Using Coloured Petri Nets, in *Proc. of the 32nd Australasian Computer Science Conference*, pp. 63-72, New Zealand January 19-23, 2009.
22. A. Schaeffer-Filho, E. Lupu, M. Sloman, S. Eisenbach. “Verification of Policy-based Self-Managed Cell Interactions Using Alloy”, in *Proc. of the 10th IEEE International Symposium on Policies for Distributed Systems and Networks* (Policy 2009), UK, July 2009
23. P. Godefroid, *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*, pg. 142, 1996.