A SPIT Avoidance Workflow for SIP-Provider

Nicolas Rüger, Sebastian Hübner, Bettina Schnor
Institute of Computer Science
University of Potsdam
Potsdam, Germany
{rueger,huebners,schnor}@cs.uni-potsdam.de

***Abstract*—Voice-over-IP (VoIP) replaces traditional telephony network infrastructures in growing numbers. Along with this infrastructure change, Spam over Internet Telephony (SPIT) is likely to spread massively, similiar to spam in e-mail infrastructures. Thus, it is necessary to develope appropriate countermeasures. Since the request for a call, usually indicated by the ringing of the phone, might already be a disturbance or annoyance of the called party, traditional content-based preventive and defensive measures, used to avoid e-mail spam, are not applicable anymore. This paper presents a new SPIT Avoidance Workflow for the detection and prevention of unsolicited calls and its implementation. The majority of the presented preventive security measures are applied on side of the provider using the Session Initiation Protocol (SIP). The workflow therefore is implemented for the widely used Kamailio SIP Server that has become a de facto standard in the field of SIP telephony because of its high-performance and robustness. Further, this paper gives an evaluation of the overhead introduced by the different workflow modules. The measurements of the prototype confirm that it is possible to filter and rate call attempts in a larger scale.**

Keywords-Session Initiation Protocol (SIP); Voice over IP (VoIP); Spam over Internet Telephony (SPIT); Security

I. INTRODUCTION

Spam over Internet Telephony (SPIT) might become a serious problem due to the continuous infrastructure change of traditional telephone networks to Voice-over-IP (VoIP) infrastructures. The Session Initiation Protocol (SIP) [14] has prevailed for signalling in VoIP infrastructures. Signalling includes establishment, modification and termination of a media session between the communication endpoints. During the signalling all necessary data for a call, like identities of the communication partners, represented by Uniform Resource Identifier (URI), are exchanged.

So far, a concept is missing, which should combine different approaches for the avoidance of SPIT, and thereby enables a reliable SPIT detection and prevention, without ignoring the requirements of existing communication infrastructures.

In [9] Liske et al. already point out that the known and adapted methods for the prevention of spam are mostly inappropriate for SPIT and therefore fail. SPIT disturbs the called user already by ringing the phone. Therefore, a telephone call needs to be filtered before any voice content is received, while e-mail spam can be analyzed and filtered after receiving the content. Hence, known filter methods are not applicable for VoIP traffic.

When Internet telephony replaces the traditional telephony, we will face VoIP infrastructures, which are managed by different providers. Here, we focus on SIP based infrastructures. Proprietary protocols (e.g., Skype [8]) are not considered. Our approach is aimed at a VoIP infrastructure that offers services comparable to the traditional telephony. Skype uses a restrictive approach with the introduction of buddylists that limit communication to the number of known contacts. This violates the principle of traditional telephony where it is possible to call everybody.

In this paper, we present a mechanism for the detection and avoidance of SPIT that we implemented on side of the provider by extending the widely accepted and used software *Kamailio* [12]. For that purpose, we have analyzed, extended and combined different approaches and results from real VoIP traffic analysis [5] in our overall concept of a SPIT Avoidance Workflow.

This paper is structured as follows: In Section II, we discuss several approaches related to the detection and avoidance of SPIT. The SPIT Avoidance Workflow is presented in Section III. In Section IV, our prototype implementation is described. The results of a detailed investigation of the overhead introduced by the different modules of the SPIT Avoidance Workflow is given in Section V.

II. RELATED WORK

There exist different approaches that relate to the detection and prevention of SPIT including consequences on suspicious call attempts.

A. Authentication

As SPIT detection is necessarily to be done during the call initiation, it has to focus on the available information, e.g., the caller's identity. Especially the appliance of actions with long term effects, e.g., blacklisting a certain user, desires for reliability about the user's identity. Therefore, authentication is an essential requirement for most approaches regarding SPIT prevention as mentioned by Hansen et al. in [7] and Liske et al. [9].

In [13], Mueller and Massoth further describe a basic approach that validates the existence of a calling user during

 ISBN: 978-1-61208-183-0

the initiation of calls. Therefore, at least non-existing faked identites can not be used to initiate malicious calls.

B. Filter Mechanisms

While the known traditional content filter methods, used to detect spam e-mails, are not useful for identifying SPIT, some adapted filter methods will apply for VoIP traffic. Therefore, new filter methods have to be applied during the call initiation for certain attributes, like the caller's identity, as there is no content to analyze. In [7] Hansen et al. introduced a concept for several SPIT filter mechanisms, e.g., whitelists including a web of trust, statistical blacklists or greylists. But, an implementation of the described mechanisms and a performance evaluation was not done.

C. Micro Payment

The use of a payment mechanism initiated by the called party in case of uncertainty about the caller's trustworthiness is introduced in [10], along with the necessary SIP extensions for the micro payment. The use of micro payment seems to be an effective method to prevent SPIT, as the initiater of spit is not willing to pay any amount, due to fact that these calls are initiated en masse.

D. Reputation

In [9], Liske et al. present a way of building a reputation system on base of a micro payment system. The payment requests and their corresponding responses are analyzed to calculate a caller's reputation. Thus SPIT detection by reputation benefits from SPIT prevention by payment. The authors explain that only a single header extension of the underlaying protocol is necessary in order to pass the reputation to the callee. Hence, the approach can be easily integrated in a SIP network that already integrates payment functionality.

In [1], Balasubramaniyan et al. introduce a defense mechanism that is based on the duration of calls between certain users. Therefore, a network of relations is spanned between single users of a VoIP-System. The characteristics of every user, regarding the call duration, is observed during a longer time period. Based on the resulting history of this behaviour analysis, a rating for every user is generated. The rating reflects the reputation of the rated user within the VoIP-System.

E. Behavioral Analysis

In [17], Sengar et al. present two approaches based on the anomaly detection of the distributions of selected call features (inter-arrival time between calls and call duration). The first approach is to detect individual SPIT call and has similarities to some modules presented in this paper. The second approach is designed to detect groups of (potentially collaborating) VoIP spam calls, e.g., a botnet used for sending SPIT. The authors analyze the call behaviour and compare it to theoretical reference pattern to detect unsusual call behaviour. Overhead measurements with prototypes are not given.

F. Consequences on suspicious call attempts

Once a call attempt is suspected by the provider to be a SPIT call, consequences need to follow. According to legal regulations [2], the provider must not drop a call. Therefore, the call is forwarded to the callee. The callee needs to handle the call attempt appropriately. For this purpose several actions may be taken, e.g., reject the call, answer the call, forward the call to a mailbox. In [7] Hansen et al. explain different options like voice menus or announcements of alternative reachability for the handling of an unsolicited call where they emphasize the use of mailboxes.

To sum up, a powerful Anti-SPIT solution has to combine different approaches for an effective detection and prevention mechanism. Just the combination of different approaches for avoiding SPIT will lead to a successful solution as Mueller and Massoth mention in [13]. Especially, the interactions between methods applied by the provider and methods implemented by the client are important.

III. SPIT Avoidance Workflow

We propose a SPIT Avoidance Workflow with a modular structure. The workflow is applied at the provider's side during the call initiation. The modular structure of our overall approach allows easy customizing and experimental combinations of single modules. Furthermore, the solution is easily extendable and can be changed to fit future requirements. The overall architecture of the SPIT Avoidance Workflow is shown in Figure 1. Details for the boxes *Check Filterlits* and *SPIT-Estimation* are shown in Figure 2 and Figure 3.

In addition to the implemented workflow, we will describe an approach for some consequence modules that need to be applied on side of the client in order to evaluate results given by the provider.

A. Check Syntax Module

Certain fields of an incoming message (e.g. caller address, callee address, etc.) are checked, whether or not the fields are filled with valid values, according to the RFC 3261 [14].

If the check is not passed, the response *484 Address Incomplete* or *400 Bad Request* is sent and the call attempt is canceled as it can not be processed. Otherwise, the Filterlist Modules are applied. The list of syntax checks is optionally extendable.

B. Filterlist Modules

Filterlists provide an effective means to categorize incoming messages during the call initiation. Certain fields of an

 ISBN: 978-1-61208-183-0

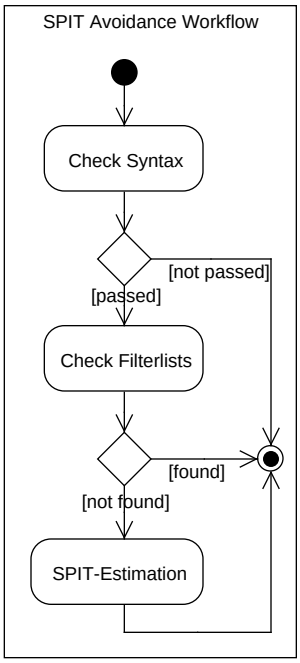

Figure 1. SPIT Avoidance Workflow

incoming message (e.g., the caller's identity in the *From* header) are compared to lists with previously stored values. If a match is present, a corresponding action is executed.

Simple concepts like the black- or whitelisting of users are already well-known from e-mail infrastructures. By using such lists, it is important to be aware about the priority of the interpreted lists, e.g., personal vs. global lists or manually vs. automatically managed lists. Implementing our approach, we set a higher priority to personal and to manually managed lists. In our opinion, the personal settings like the manually and therefore intentionally set entries, should be preferred.

Therefore, we introduce global and callee specific lists for black- and whitelisting. In addition, we propose a global delay list that is to indicate suspicious callers whose calls are delayed during the initiation. This idea is based on the approach that such timeouts will lead callers, that initiated calls en masse, to hang up the phone before the call attempt is actually forwarded to the callee.

The detailed part of the workflow for the Filterlist Modules is shown in Figure 2. A blacklisted user, e.g., receives *403 Forbidden* and the call attempt is canceled as the callee decided so in advance by putting the caller on the blacklist. Once a caller can not be categorized by any filterlist, the Estimation Modules will apply.

C. Estimation Modules

The Estimation Modules realize the main concept of SPIT detection. The modules focus on undesired calls that have

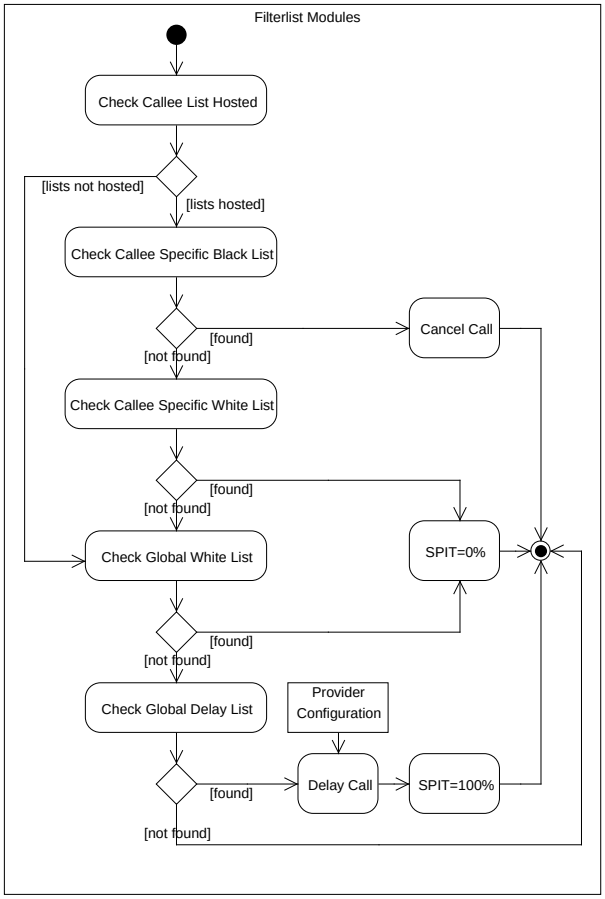

Figure 2. Filterlist Modules

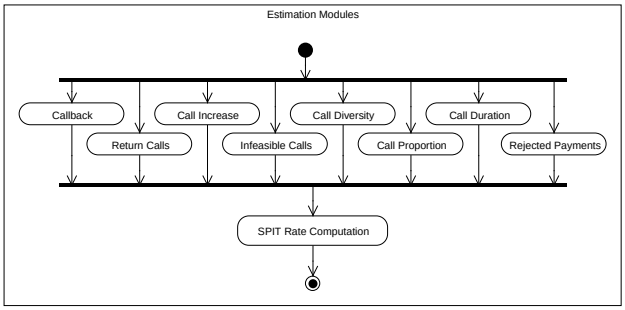

Figure 3. Estimation Modules

 ISBN: 978-1-61208-183-0

been initiated in a larger scale. Both criteria characterize SPIT. Various methods that are managed modular, evaluate the caller's behaviour in the past. The approach is based on the assumption that it is possible to identify users with criminal or malicious intentions because of their specific behaviour related to different criteria. Parameters, such as the number of calls initiated by the user, the duration of calls, the number of different contacts during a certain period of time etc. are analyzed by the provider. Based on this data, the probability of SPIT related to a special criterion is estimated by each method and therefore realized by a single module in each case (see Figure 3).

The detailed mathematical formulas of the *SPIT Estimation Modules* are given in [15].

1) Callback: The module Callback returns a low SPIT probability if a caller was contacted by the callee already before. The idea given by Seifert in [16] is that a user who returns someone's call is a legitimate and desired contact. Therefore, also past call attempts of the callee of the ongoing call are rated.

2) Return Calls: This module returns a low SPIT probability if a user gets called back frequently. This approach is based on the consideration that a user who is never called back by others, is probably to be classified as an unwanted dialog partner.

We now propose that only those interactions between users should be considered, where the relevant past calls have been successful. These calls must have been confirmed by both partners and there must have been a conversation with submitted voice content. Both parties need to confirm at least one call request from the other party. This ensures that both participants really wanted to interact with each other. Such a interaction is rated as a successful *returned call*.

3) Call Increase: The module Call Increase recognizes increasing call numbers of a single user in comparison to himself during a short time period. In [16] Seifert describes a mechanism our module is based on. The module is designed to identify accounts that already exist for a longer time and have now been compromised. The number of outgoing calls of the current caller, in several elapsed intervals, is set into relation to his initiated calls during the last interval. If the number of calls has risen too high, the module returns a high SPIT probability. Balasubramaniyan et al. in [1] and Sengar et al. in [17] assume that a significant divergence from the normal call behaviour is a signal for outgoing SPIT.

4) Infeasible Calls: Based on an approach in [16], our module Infeasible Calls evaluates the number of calls that have not been successful because a recipient with the selected SIP address was nonexistent. Therefore, the module controls which response codes the caller has received on his call attempts in the past. If, too often in the past, the caller of the current call tried to attain identities that did not exist, the module returns a high SPIT probability.

Therefore, we do now propose to extend the message *404 Not Found* by a SIP conforming string extension. In the communication between various providers it should be distinguished between *404 Not Registered* and *404 Not Existing*, while the client still only receives *404 Not Found*.

Thus, it is possible for the providers, to separate infeasible calls from the calls to currently not available users. In addition the approach eliminates the chance of creating a file of existent addresses, as these could possibly be used for sending SPIT in the future.

5) Call Diversity: This module rates the number of calls to different SIP URIs as proposed by Liske et al. in [10]. For that purpose, the module compares the number of made calls to the number of diverse dialed numbers.

We propose that callers who call too many different SIP URIs are suspected to spit.

6) Call Proportion: The module Call Proportion assesses the number of calls made by a user, compared to the average number of calls of all users in an observation period. If in the past the caller of the current call has initiated significantly more calls than other users in average, our module returns a high SPIT probability. This again bases on the assumption made by Balasubramaniyan et al. in [1] and Sengar et al. [17] that a strong divergence from the normal call behaviour is probably SPIT and has been mentioned by Seifert in [16] as well.

7) Call Duration: This module considers the duration of current caller's past calls as Liske et al. mention in [9]. A caller who initiates many very short calls is suspected to be an unwanted caller. Therefore, we compare the number of short calls (e.g., $\leq$ 5sec.) to the overall number of phone calls, the user has initiated.

8) Rejected Payments: The module evaluates the number of calls of the caller, which have not been successfully concluded because the caller did not agree to pay the fees that were caused by his call. In [9], Liske et al. suppose that a caller is suspected to spit if he did reject such requests too often. Thereto, the caller is rated with a high SPIT probability by the module.

Through *SPIT Rate Computation*, the results of the individual modules are combined. The result is represented within a single value for the SPIT probability. In [9], Liske et al. already mention a related reputation system.

The detailed mathematical formulas of the *SPIT Rate Computation* are given in [15].

Finally, the *Consequence Modules*, not shown in the figures due to their client side's deployment, generate an appropriate reaction to the determined *SPIT Rate*. In [7], Hansen et al. already emphasize the importance of mailbox mechanisms to prevent SPIT. Our proposal is to provide certain capabilities to the client, to evaluate the *SPIT Rate* that has been forwarded by the provider.

Thus, our Consequence Modules decide by *time of day*

 ISBN: 978-1-61208-183-0

and by transmitted *SPIT Rate* about how to respond to the call. Possible consequences are: signal an incoming call (e.g., by ringing or vibrating the phone), forward the call to a mailbox, request micro-payment from the caller or reject the call. The Consequence Modules are not part of our prototype implementation as they should be located within the client's VoIP (soft-)phone.

IV. Implementation

The implementation presented in this paper contains the *Syntax Check Module*, the *Filterlist Modules* and the *Estimation Modules* including the *SPIT Rate Computation* as described in Section III.

Our implementation extends the functionality of the *Kamailio SIP Server* [12]. It is complying with the Kamailio project guidelines and compatible with the existing code. Kamailio is de-facto standard in the field of telephony via SIP due to its open source code, modularity, world-wide usage, high-performance, robustness and its active developer community.

The Estimation Modules access the information in the database, previously collected by a *Call Trace Update Functionality*. Therefore, the Call Trace Update Functionality logs all relevant data (e.g., start and end time, caller, callee, etc.) of every call. Based on this information, the behaviour of the caller gets evaluated and expressed as module-specific SPIT probability.

As a final result, the *SPIT Rate* is calculated using the SPIT probabilties from all SPIT Estimation Modules. Different strategies can be applied to include the single SPIT probabilities in the computation, as described by Seifert in [16]. It is possible to incorporate the ratings of all modules in the same degree in the reported value as we did for our prototype implementation. For future use, we propose a weighted accumulation to underline the greater importance of certain modules. As the number of modules that contribute to the SPIT rating may vary, the calculation should be implemented dynamically. If, for example not enough information about the caller is available, not all input parameters for all modules are applicable.

We assume that the relevant data for the single modules is available when it needs to be analyzed. In case no relevant data is found during the evaluation of the available data, the corresponding module is not able to compute a result and can not be involved in the *SPIT Rate Computation*. Our implementation considers this dynamic computation.

The final result of the computation is added as an additional header field *Spit Rate* to the initial *Invite* message. Thus it is transmitted to the client as a compressed value. This extension is conformant with the RFC 3261 [14].

For our implementation, we have chosen a MySQL database [3] as backend because the link is well supported by Kamailio and the database is Open Source like Kamailio itself. The entity relationship model is shown in Figure 4.

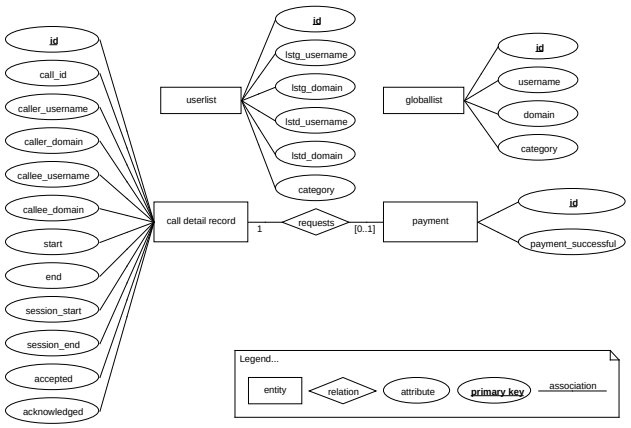

Figure 4. Database Entity Relationship Model

Within the shown database structure, all relevant data needed by the modules described in Section III is stored.

We have implemented our SPIT Avoidance Workflow by extending the Kamailio routing logic in *kamailio.cfg*. We have chosen the interface SQLOps [11] provided by Kamailio. It allows interaction with the database backend directly from the routing logic.

Analysis of different available interfaces arrived at the conclusion that SQLOps shows the best performance. It holds and reuses one database connection that is established on Kamailio startup. The *Perl* interface [4] seemed to be an alternative because it allows the execution of arbitrary Perl scripts, but it etablishes a new database connection for each running script. Therefore, it is not scalable for growing numbers of calls per second.

V. Measurements

The measurements were made to compare the performance of the Kamailio SIP server with and without the extensions for SPIT prevention. Thus, it is possible to evaluate the overhead of our solution and its impact on the number of processed calls per second. Therefore, telephony traffic was simulated, while the SPIT prevention modules have been active or inactive.

A. Testbed and Scenarios

We used three nodes (each with 2 x AMD Opteron 244 CPU, 1.8GHz, 4GB RAM, Gigabit Ethernet Interconnection) to setup one SIP proxy (Kamailio [12], v3.1.1), two user agents (SIPp [6], v3.1) that generated (resp. processed) a various number of SIP calls and a MySQL [3] database (v.14.12 distrib. 5.0.51a) to store the collected connection data. Kamailio has been configured to use 1024MB of memory, to create eight processes and its log level was set to zero.

We measured the default configuration of the proxy in comparison to the behaviour of the proxy with one single

 ISBN: 978-1-61208-183-0

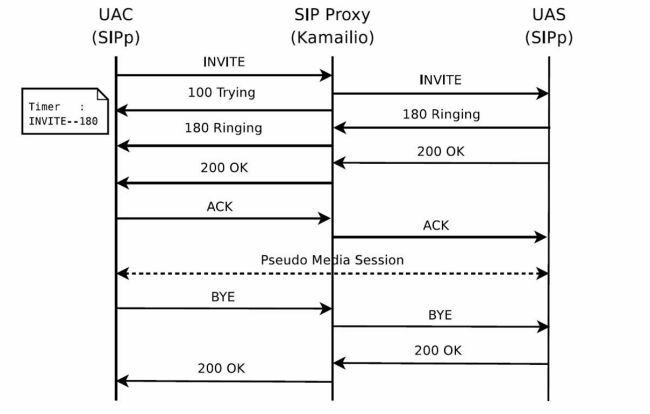

Figure 5. Measurement of Round Trip Time

module switched on in each case. Thereby, we wanted to figure out which module causes the most overhead.

Each call generates a time value. The round-trip time (RTT) represents the delay of a user agent server's response, including Kamailio action (processing the activated modules). We focused on this part of the session initiation as it measures the most expensive part. It represents the time interval from the start of the session initiation to the first ringing (see Figure 5).

For each single module, we measured the RTT in different series. Each series used a constant call frequency (number of calls per second) for 60 seconds. We have chosen different call frequencies for each series in steps of 1 from the range of 1 to 10 calls per second (cps), steps of 10 from the range of 10 to 100cps and steps of 100 from the range of 100 to 1000cps. That sums up to 28 series of different call frequencies per module lasting 60 seconds each (see Table I). The proxy and the user agents have been restarted for each measurement.

RANGE (CPS)	STEPS	NO. OF SERIES
1 - 10	1	10
20 - 100	10	9
200 - 1000	100	9
		$\sum$ 28 a 60 sec.

Table I. Measurements for each module

B. Results

For all modules and each call frequency we calculated the corresponding median values for the RTT.

1) Syntax Check and Filterlist Modules: The module Global List includes the global delay- and global whitelist whereas the module User List includes the callee's personal black and white list. The lists were filled with 50 sample entries each. As shown in Figure 6, the Kamailio SIP server only shows slightly higher response times of maximum (~0.25ms) with the Syntax Check or the Filterlist Modules switched on, than without any modules for SPIT detection.

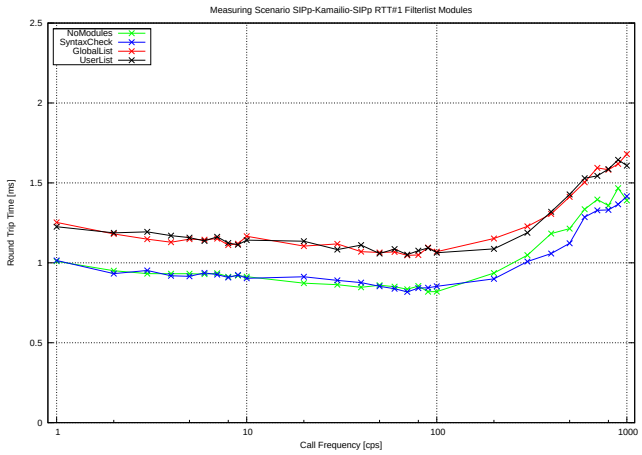

Figure 6. Median RTT with activated Filterlist Modules

Slight deviation of 0.1ms is due to normal measurement fuzziness using the described measurement environment. The reached accuracy of 0.1ms is sufficient to show the impact of the modules.

2) Estimation Modules: Figure 7 shows the RTT for the Estimation Modules, with exception of module Call Increase as it does not fit within the chosen scale, in comparison to the proxy's plain behaviour. The modules Callback, Infeasible Calls and Rejected Payment show just a slightly higher response time of ~0.25 to 0.5ms.

From a call frequency of 100cps the response time for the modules Call Diversity, Return Calls, Call Duration and Call Proportion reaches unacceptable high values, whereas the module Call Increase shows this behaviour already from a call frequency of 10cps.

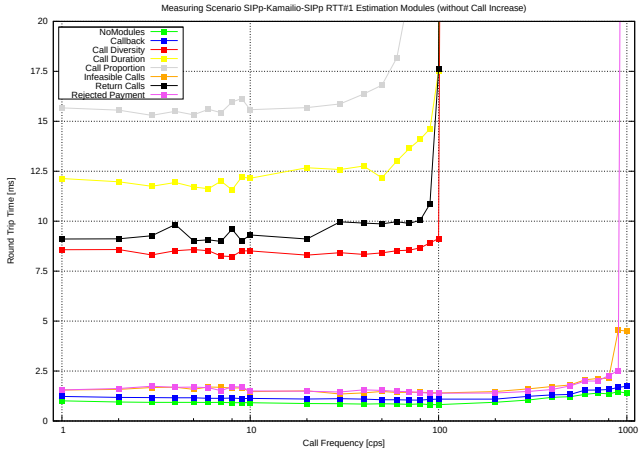

Figure 7. Median RTT with activated Estimation Modules

The measured delays for some modules are resulting from the queries to the database, initiated by the Kamailio routing

 ISBN: 978-1-61208-183-0

logic during the processing of messages. Further tests have shown, that the delays do not occur if the requests on the database are not performed, even though the remaining program code (e.g., decisions, branches, variable assignments, adding the SPIT-header fields etc.) of the SPIT Avoidance Modules kept unchanged. Therefore, the delays result from the database queries only. This conclusion is backed by the measurement results, which show that especially the modules with more frequent requests to the database scale poorly.

3) Call Trace Update Functionality: The Call Trace Update Functionality does not scale well as it accesses the database very often during the call initiation to log all necessary data. Figure 8 shows that the functionality causes high response times already from a call frequency of 10cps.

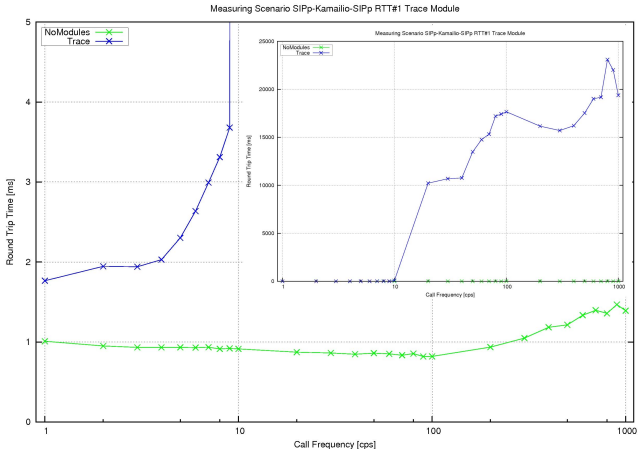

Figure 8. Median RTT with activated Call Trace Update

As the Call Trace Update Functionality provides the basis for the Estimation Modules, some effort needs to be made to improve the performance of this basic functionality.

C. Result Summary and Suggestions for Improvement

In summary, six modules scale satisfying and five modules do not. In addition, the trace mechanism for data collection does not scale very well.

Since we have identified the database queries as the reason for the measured delay, the following actions might lead to better response times and thus, to a higher performance of the SPIT Avoidance Modules that did not scale and finally to a better Kamailio scalability. The selected MySQL database was used in the standard configuration. We have made no improvements for our measurements. Therefore, indexing of database entries, detailed analysis and optimization of the individual database queries, the usage of numeric data types instead of strings, increasing the cache size of the database and the number of possible child processes of Kamailio might lead to a much better performance.

Further, the usage of more powerful hardware as well as separating the database system will improve the overall system performance in addition.

VI. Conclusion and Future Work

We presented a SPIT Avoidance Workflow consisting of filterlists, call detail record analysis, and a rating system based on this analysis. Between modules known from prior works, we also proposed new modules like a global delay list.

We have shown that the workflow can be integrated within the Kamailio routing logic to detect and prevent SPIT. For some of our modules, the measurements with the prototype confirm that it is possible to filter and rate call attempts in a larger scale without increasing the response time or the scalability of the Kamailio SIP Server. Already six from eleven SPIT Avoidance Modules worked very well without any database optimization or any special hardware.

Future work has to be done to optimize the SQL queries and to improve the underlying database structure for a better performance.

References

[1] Vijay Balasubramaniyan, Mustaque Ahamad, and Haesun Park. CallRank: Combating SPIT Using Call Duration, Social Networks and Global Reputation. In *CEAS'07*, 2007.

[2] Translation of the German Criminal Code provided by Prof. Dr. Michael Bohlander Bundesministerium der Justiz. German Criminal Code - Strafgesetzbuch (StGB). http://www.gesetze-im-internet.de/englisch_stgb/german_criminal_code.pdf, 2010.

[3] Oracle Corp. MySQL v.14.12 distrib. 5.0.51a. http://downloads.mysql.com/archives.php?p=mysql-5.0&v=5.0.51a, 2008.

[4] Bastian Friedrich. Kamailio (OpenSER) Perl Module. http://www.kamailio.org/docs/modules/3.1.x/modules_k/perl.html, 2007.

[5] Stefan Gasterstädt and Bettina Schnor. What VoIP-CDR can tell us (not)? Technical Report ISSN 0946-7580, TR-2010-2, Potsdam University, Germany, May 2010.

[6] Richard Gayraud, Olivier Jacques, et al. SIPp: An Open Source Performance Testing Tool for SIP [v3.1]. http://sipp.sourceforge.net, March 17^{th}, 2009.

[7] Markus Hansen, Marit Hansen, Jan Moeller, Thomas Rohwer, Carsten Tolkmit, and Henning Waack. Developing a Legally Compliant Reachability Management System as a Countermeasure against SPIT. In *Third Annual VoIP Security Workshop*, Berlin, Germany, June 2006.

[8] Skype Limited. Skype Technologies (Microsoft). http://www.skype.com, 2011.

 ISBN: 978-1-61208-183-0

[9] Stefan Liske, Klaus Rebensburg, and Bettina Schnor. Implicit Reputation in a Payment Integrated SIP Network. In *Proceedings of the 14th Annual Workshop of HP Software University Association (HP-SUA)*, pages 161–170, Munich, Germany, July 2007.

[10] Stefan Liske, Klaus Rebensburg, and Bettina Schnor. SPIT-Erkennung, -Bekanntgabe und -Abwehr in SIP-Netzwerken. In U. Ultes-Nitsche, editor, *Proceedings of KiVS – NetSec 2007, Workshop „Secure Network Configuration“*, pages 33–38, February 2007.

[11] Daniel-Constantin Mierla. Kamailio (OpenSER) SQLOps Module. http://www.kamailio.org/docs/modules/3.1.x/modules_k/sqlops.html, 2008.

[12] Ramona-Elena Modroiu, Bogdan Andrei Iancu, Daniel-Constantin Mierla, et al. Kamailio (OpenSER) [v3.1.1]. http://www.kamailio.org/, December 02^{th}, 2010.

[13] Juergen Mueller and Michael Massoth. Defense Against Direct Spam Over Internet Telephony by Caller Pre-Validation. In *2010 Sixth Advanced International Conference on Telecommunications (AICT 2010), Barcelona, Spain, 9-15 May 2010*, pages 172–177, Washington, DC, USA, 2010. IEEE Computer Society. Best Paper Award.

[14] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922.

[15] Nicolas Rueger. Konzeption und Implementierung providerseitiger Sicherheitsmechanismen zur Erkennung und Abwehr von SPAM over Internet Telephony (SPIT) in VoIP-Netzwerken. Diploma thesis, University of Potsdam, Institute for Computer Science, 2011.

[16] Juergen Maximilian Seifert. Klassifizierung und Implementierung von SPIT (Spam over Internet Telephony) Abwehrmassnahmen in SIP-Netzen. Diploma thesis, University of Potsdam, Institute for Computer Science, 2006.

[17] Hemant Sengar, Xinyuan Wang, and Art Nichols. Thwarting Spam over Internet Telephony (SPIT) Attacks on VoIP Networks. In *Proceedings of the Nineteenth International Workshop on Quality of Service*, IWQoS '11, pages 25:1–25:3, Piscataway, NJ, USA, 2011. IEEE Press.

 ISBN: 978-1-61208-183-0