Progressive Multi Gray-Leveling: A Voice Spam Protection Algorithm

Dongwook Shin, Master Key Consulting
Jinyoung Ahn, Towson University
Choon Shim, Qovia Inc.

Abstract

As voice over IP gains popularity, it is easy to imagine that Internet hacking and other security problems we currently face with email spam will attack the VoIP environment. In contrast to email protocols represented by SMTP, VoIP does not tolerate any negotiation in its signaling steps or screening content due to the delay limitation. Thus, it is difficult to implement voice spam protection or control algorithms by just mimicking popular email spam protection algorithms. This article proposes a voice spam control algorithm called Progressive Multi Gray-Leveling (PMG) that fits in VoIP settings. PMG progressively calculates the "gray level" of a caller and determines whether the call will be connected or blocked based on previous call patterns. Based on experiments, it turns out that PMG can learn a spammer's calling pattern and block massive spam attacks from them effectively.

As voice over IP (VoIP) applications spread over the Internet, it is not difficult to imagine it developing the same problems in the near future as we face in email spam, Internet hacking, and other security problems. In the same way as the current Internet portal and users are disturbed by spam emails, call managers are likely to face spam voice problems just as most email servers have spam mail problems. It may cause more complicated problems in VoIP than spam mails in the Internet [1].

In an email setting, we have a significant chance to quarantine incoming emails to check if they are spam or a virus. In fact, as Simple Mail Transfer Protocol (SMTP) allows a repeated negotiation step between the sender and receiver, there are plenty of chances to tap into the negotiation steps and quarantine emails. At the same time, we can also check the content of incoming emails and filter those that contain suspicious contents using filtering techniques, including a Bayesian network [2]. As opposed to email protocols represented by SMTP, VoIP does not tolerate any negotiation in the signaling step or screening of content because of its slim delay limitation. Thus, it is harder to implement any Spam over Internet Telephony (SPIT) protection or control algorithms in a VoIP setting.

Ram Dantu and Prakash Kolan [3, 4] proposed a voice spam protection algorithm utilizing user feedback. Their algorithm calculates the reputation value of a caller using a Bayesian inference function, taking the past history of the caller. If the caller's reputation is below a certain threshold, the call is blocked. In this method the user feedback is essential since all the spam history of a caller comes from user feedback. Y. Rebahi and D. Sisalem [5] presented a reputation based spam blocking algorithm where reputation network manager is built from SIP repository. A reputation from a source *S* to destination *D* is computed as the product of each path reputation that the SIP message is traversing from *S* to *D*. However, they did not address how to compute the unit reputation of each SIP server from which all global reputation comes.

In this article we propose a SPIT protection algorithm called Progressive Multi Gray-Leveling (PMG) that calculates the gray level of a caller (the level that determines if the caller is a likely spam source or not), progressively in multiterm levels, and determines if the call is to be connected based on previous call patterns rather than user feedback. The notion of a gray level stems from a spam email protection algorithm called *graylisting* [6], where the term *gray* indicates a level existing between white and black. Opposed to the white and black that always allow or block incoming mail from a certain sender, a gray level determines the legitimacy of a sender depending on the current situation. Depending on the policy, PMG can also work with white and black lists if PMG is set to have a higher (or lower) priority than white and black lists. For instance, if you set the black list to have higher priority over PMG, incoming calls from senders in the black list are always blocked regardless of the result of PMG computation. Similarly, if the white list is set to have higher priority over PMG, the senders in the list are always guaranteed to make calls, no matter what their call patterns look like.

PMG monitors only call patterns from each caller and determines the spam voices based on these patterns. The essence of the algorithm is that as a caller continues to make calls through the call server in a certain time span, the gray-level (the opposite of the credit) becomes higher and makes the caller much likely to appear as a spam source. If the level is higher than a threshold that determines the caller is a spam source, the caller is no longer able to make calls. Unlike the black list, the caller will not stay in that status permanently. If the spam call is not initiated in the next time period, the gray

Dongwook Shin and Jinyoung Ahn were at Qovia at the time the work discussed in this article was done.

0890-8044/06/$20.00 © 2006 IEEE

	Email spam	Voice spam
Protocol	SMTP/POP3	SIP/Skinny/RTP
Session	Asynchronous	Synchronous
Source of Spam	SMTP clients	Softphones
Danger	Annoying/email folder full	Annoying/call server outage/ voice mailbox full
Deleting	Deciding to delete after seeing subject line	Deciding to delete after hearing message

Table 1. *Email spam vs. voice spam.*

value decreases below the threshold, enabling the caller to resume calling.

We implemented the algorithm in two environments, one in a Cisco call manager and the other in the Vovida Open Communication Application Library (VOCAL) [7] server and IPTEL SIP Express Router (SER) [8] using stateful proxy [9]. Java Telephone API (JTAPI) [10] is used in the Cisco call manager, whereas stateful proxy is employed in both VOCAL and IPTEL SER. We carried out several experiments to test how PMG preformed and demonstrated that PMG works well in spam voice protection if we set parameters correctly.

Spam Voice Call and Protection

Voice spam is similar to email spam in that senders use an IP network to target a specific user, or group of users, to generate an abundance of calls for marketing purposes or to disrupt a user's normal activities. However, voice spam is quite different in several aspects from email spam, and the implication may be more dangerous to a VoIP environment than email spam is to the Internet.

As easily seen in Table 1, SPIT can be more malicious than E-mail spam, and thus a major source of trouble. Email spam is able to disrupt application service providers (ASPs) and fill email folders, compelling recipients to filter through them. Likewise, voice spam will be able to produce call server outages and cause voice mailboxes to overflow, obligating recipients to sort through messages. Fortunately, sorting email spam is now a fairly simple task in that it can be sorted out by subject, for example, and multiple emails can be deleted at once. However, with voice spam, a receiver has to listen to each voice mail message and delete unsolicited ones.

In the spam protection aspect, voice is more difficult to protect than email, due to its synchronicity. Since email is delivered asynchronously, users and administrators have a lot of chances to check and quarantine them. One well-known email spam protection algorithm, Bayesian Spam Filtering [2], is able to analyze the content of emails and detect spam mails at a certain checkpoint. Because email is delivered asynchronously, it does not really matter if email delivery is delayed for a short period of time. However, in the case of SPIT, we have to decide if the incoming call is spam or not within the connection time. Once the connection is set up, it is too late to take action because the voice spam is already disrupting the call server and recording voice mail.

Progressive Multi Gray-Leveling

The notion of gray-leveling stems from an email spam protection algorithm by Evan Harris called graylisting [6]. Whereas blacklisting denies all mails from a given source and whitelisting accepts all from the source, graylisting determines the legitimacy of a sender depending on the current situation. However, different from graylisting that investigates email content, PMG as proposed in this article monitors call patterns from each caller and determines SPIT based on these patterns.

The essence of the algorithm is that as a caller attempts to make numerous calls through the call server in a certain time span, their gray level will increase, thus classifying the caller as a potential spam source. Once the gray level becomes higher than the given spam source threshold, the caller will not be able to make any more calls. However, the graylist differs from a black list in that the caller will not permanently stay in "spam status." If the spam caller stops initiating SPIT within a certain time period, the gray level will decrease and eventually stay below the threshold.

PMG computes the two levels based on call patterns, short-term and long-term, whenever an incoming call is received. PMG determines that a call is a SPIT if the summation of the two levels exceeds the given threshold. If the summation is less than the threshold, the caller is considered a regular user and his/her call is connected. If the summation is greater than the threshold, the caller is regarded as a spam generator and is therefore blocked. Figure 1 illustrates how these two work in PMG.

In this computation, the short term represents a short period of time (e.g., 1 min) during which a voice spam source is able to generate many calls to attack a server. The short-term level increases very quickly in this case and protects the server from those intensive calls in a short period of time. The short-term level decreases as soon as the caller stops sending calls. If the caller does not make a call in a couple of hours, the short-term level returns back to zero. Hence, with only a short-term level, a spammer is able to send spam calls over again after a relatively short period of time.

To compensate for the short-term level issue, we use the long-term level that considers the call patterns over a rather long period of time (e.g., several hours or days). The long-term level increases more slowly than the short-term level and also decreases at a much slower pace. Therefore, the long-term value remains a lot longer than the short-term level. At the same time, it also takes into account the history of a caller that has been detected as a spam generator. If a caller has ever been detected as a spam source, the long-term value is multiplied by the number of times it was detected as a spammer and increases much faster than other regular users. Hence, a spam generator is able to make only a fraction of the calls it originally produced in its previous trial.

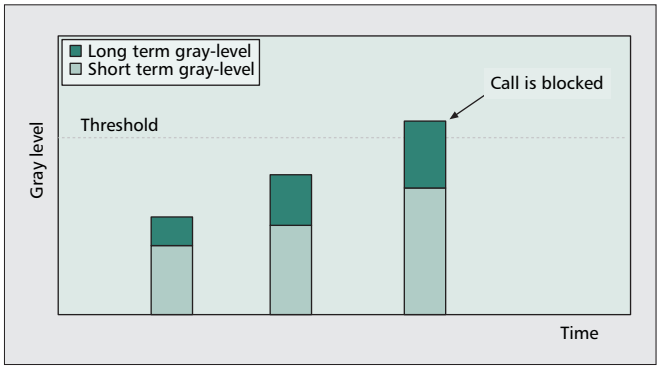


Figure 1. *The relationship between gray levels and threshold.*

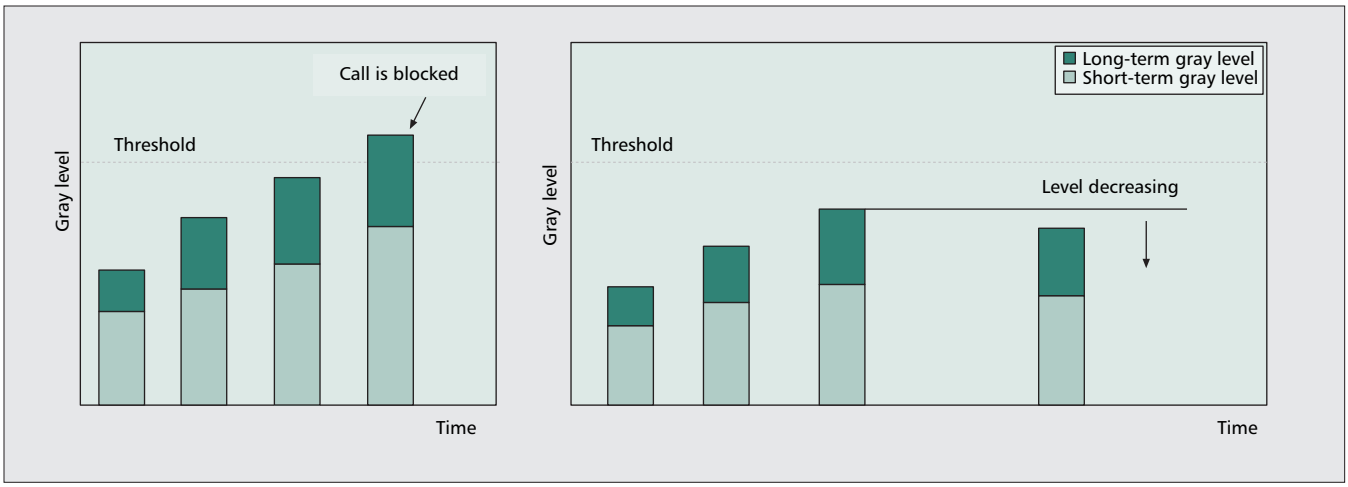


Figure 2. *Short interval vs. long interval calls.*

Figure 2 illustrates a case where a caller generates many calls in short intervals (left graph), whereas the right graph displays calls are made in long intervals. As seen in the short interval case, both short-term and long-term gray levels increase quickly, allowing PMG to promptly block the series of calls coming in short intervals. On the other hand, the long interval case shows that when intervals between calls are long, the short-term level does not increase quickly and will even decrease if the interval is longer than the given time span (i.e., 10 min). The resulting summation of the two levels does not increase very fast and therefore allows a larger number of calls to connect.

One more thing to note is that PMG maintains the caller's identity based on a unique caller address (like SIP URI), not IP address. This is because PMG targets a user who generates spam calls, not the hardware used by the spammer. Thus, if a spammer is able to get a new identity easily, PMG does not work for this case since PMG has to recalculate the gray level for the new identity. Hence, to achieve maximum benefits, PMG has to work together with strong authentication and a strict user management mechanism.

Algorithm Detail and Analysis

The gray level of a caller in the PMG algorithm is an important element that must be determined when deciphering SPIT. We use two parameters in its algorithm to calculate the whole gray level of a caller: one for short-term gray level (S or STGL) and one for long-term gray level (L or LTGL). The decision to connect a call is decided by calculating the two levels and making a summation of them. If the summation falls below the threshold (T), the connection is made; otherwise, the connection is blocked. Figure 3 illustrates the flowchart of the PMG algorithm.

As shown in the flowchart, the PMG uses a relational database and keeps track of the quadruple (*PS*, *PL*, *PT*, *SH*) for every distinct caller. This quadruple is updated every time the caller makes a call. Algorithm 1 shows a precise description of PMG. All the variables are computed as in Algorithm 1 except *PT* is the time that the same caller tries to make a call for the last time. If *I*, the interval of the current call and the previous call (represented as *PT*), from the same caller is becoming small, the *S* and *L* are getting high.

To further explain this algorithm, three points need to be

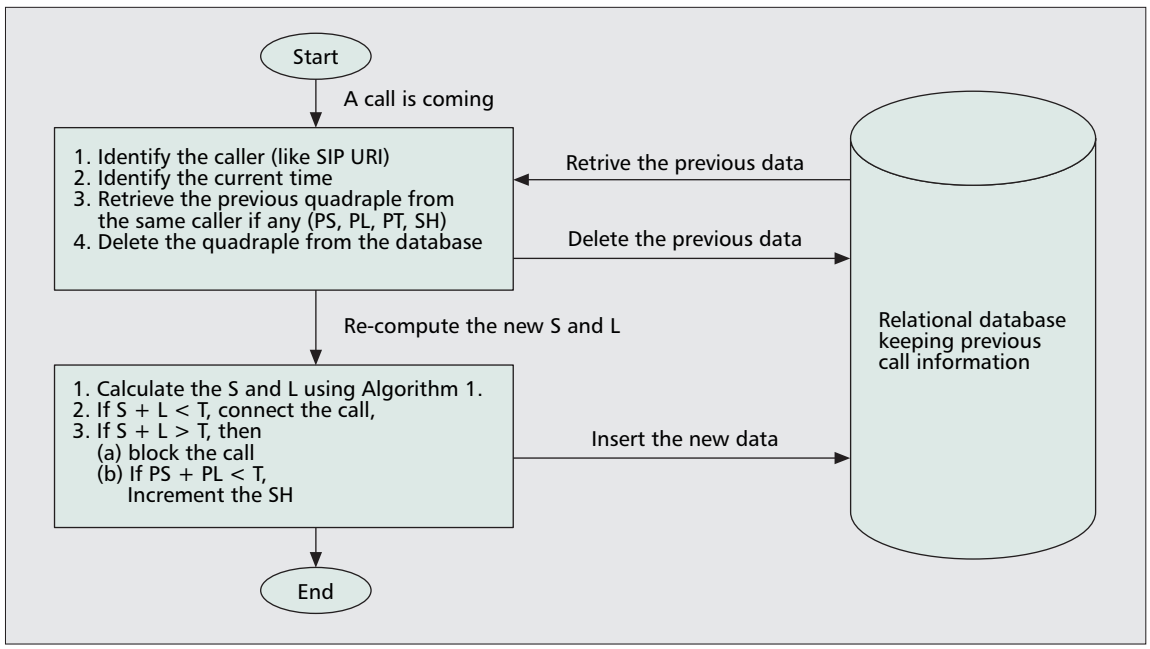


Figure 3. *Flowchart of PMG.*

Description: There are two parameters for calculating the whole gray level of a caller: One for S (short-Term gray level) and L (long-term gray level). The decision whether to put through the call from the caller is to calculate the two levels and measure the summation of the two. If the summation is below the threshold **T**, the connection is put through. Otherwise, the connection is blocked.

The levels are calculated in the following ways.

a. 1. If TL2 – I > 0, L = PL + **C2***(SP+1) * (TL2 – I)/TL2.
   2. If TL2 – I <= 0, L = PL + **C2*** (1-SP)/(SP+1)) * (TL2 – I)/TL2.
   3. If L < 0, then set L to 0

b. 1. If L < T, S = PS + **C1***(TL1 – I)/min(max(I,1), TL).
   2. If S < 0, then set S to 0.
   3. If S >= T, **let L = S and S = 0**;

c. If PS + PL < T and S + L > T, then increment SH.

* *PS* and *PL* are the *S* and L value respectively computed at the last call coming from the same caller.

* *TL1* is a short-term time period (e.g., 1 min.)

* *TL2* is a long-term time period (e.g., a day). But as TL2 is a lot longer than TL1, L increases in a much slow pace than S increases.

* I is the interval between the last call and the current call coming from the same caller.

* *C1* and *C2* are constants that can adjust the weight of S and L in calculating the whole gray-level. If C1 is bigger than C2, then S is increasing faster than L and makes the algorithm run well for the short-term attacks. On the other hand if C2 is bigger than C1, the algorithm memorizes the past history for a longer period and handles the long-term attack more effectively.

* *T* is the threshold value that judges if the coming call is a spam call or not.

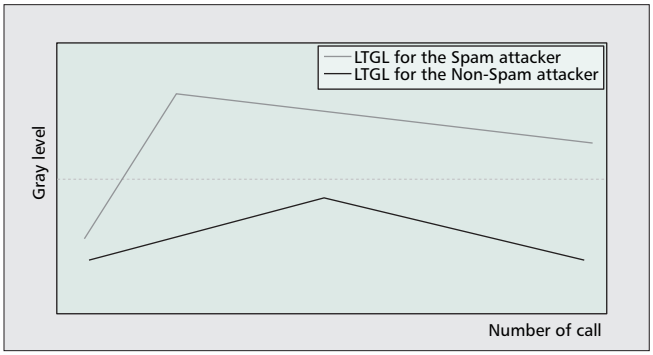
* *SH* is the recorded history of a caller that has been recognized as spam voice attacks. As a caller is increasingly detected as spam trials, he has more weight on L creating a longer period for the L to decrease. So as a previously identified spam caller tries another spam voice attack, he is more likely to be detected earlier than previous attacks and eventually considered as blacklist.

■ Algorithm 1. *The pseudo code of PMG.*

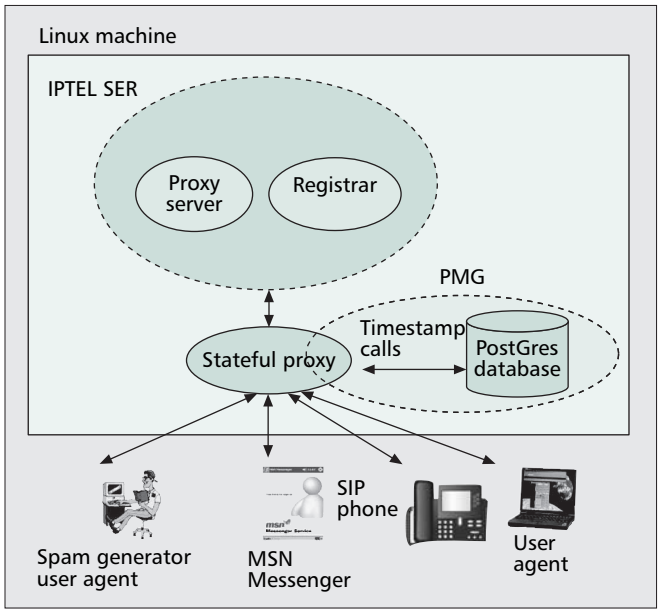
addressed. First, L is calculated differently depending on the sign of value of the expression (a.1 and a.2), $TL2 - I$. When it is positive (a.1), L is increased as a multiple of the previous spam history. That is, if a caller has made a spam attack twice, L is supposed to increase three times as fast as that of one who does not have spam history. If it is negative (a.2), it is decreased more slowly for the spam attacker than for the normal caller. Figure 4 illustrates this graphically.

Second, S increases more quickly as the intervals of the incoming calls get shorter. It is because the denominator of the expression b.1 ($min(max(I,1), TL1)$) is a value ranging from 1 and TL1 and equivalent to the interval of the two consecutive calls. The reason to limit the range of the denominator of the expression within 1 and TL1 is to adjust the steepness of the increase. If there is no limitation and the denominator is given as I alone, STGL can be any high number (e.g., 60,000 for the two calls at intervals of 0.001 s when TL1 is 60 s) for only two calls.

Third, once S passes the threshold, it does not increase in the same way as it did below the threshold. Rather, S delivers its value to L and resets itself to zero. After this moment, L takes the responsibility of tracking the gray level and runs its course, before S increases over the threshold once again. The idea behind this is that S is designed to protect the call server from spam calls very quickly. Once S does its job and denies the spam calls, L takes the value of S and decreases it very slowly to block the spam calls over a long period of time until it falls below the threshold.



■ Figure 4. *LTGL — spam vs. non-spam attacker.*



■ Figure 5. *The implementation of PMG on an IPTEL server.*

PMG is based on a kind of crediting mechanism as is Dantu's method [3, 4]. It differs from Dantu's in that PMG automatically computes the gray level and automatically blocks SPIT, whereas Dantu's method depends on user feedback. If a caller has tried SPIT attacks previously and stops doing that for a long time, he/she is allowed to make calls again. However, if he/she tries SPIT attacks again, the calls are blocked much more quickly than before. This is because their gray values are higher than before, and L will increase much faster than before multiplied by the number of previous spam attacks.

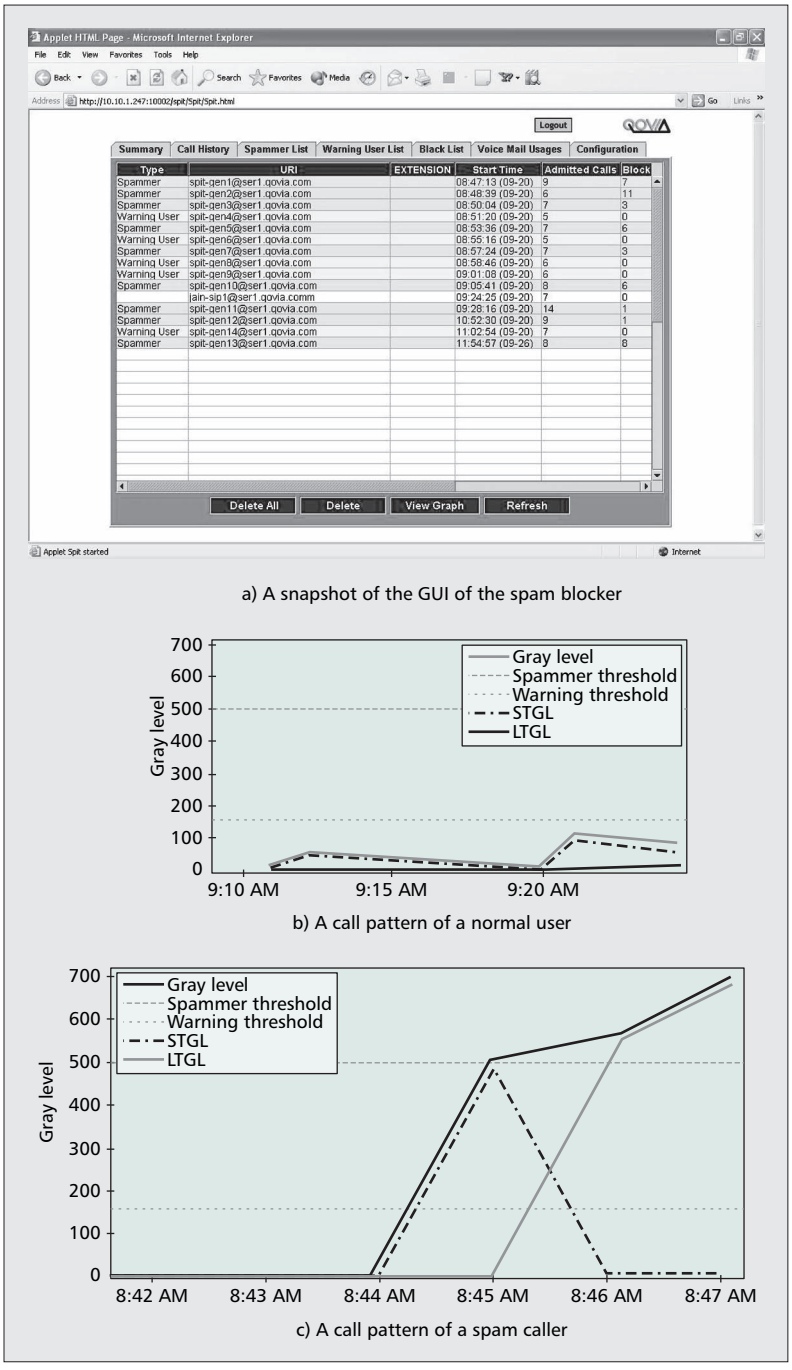
Implementation

We have implemented PMG on three platforms: one for a Cisco Call Manager using the Java Telephone application programming interface (JTAPI) [10], and the other two for VOCAL [7] and IPTEL SER [8] using stateful proxy [9]. A stateful proxy keeps track of all signals and has the call state. Figure 5 describes the implementation of PMG in a platform, IPTEL SER.

In Fig. 5 PMG is running as part of stateful proxy and timestamps incoming calls to the PostGresSQL [11] database. Whenever a caller makes a call to IPTEL SER by sending an INVITE message, PMG in the stateful proxy catches it and analyzes the call pattern. If the gray level is lower than the spammer threshold, PMG passes the INVITE message to a proxy server and lets the call through. Otherwise, it simply blocks the INVITE message. This prototype system employs another concept called a *warning threshold*. The warning threshold is lower than the spammer threshold and alerts the administrator that the current caller can be a potential spammer in the future and suggests tracking of the call. Figure 6 shows snapshots of the Web-based user interface that shows the call status of all the callers.

As shown in Fig. 6a, the first column represents the type of a user depending on the call patterns. The type Spammer is one whose gray level reaches above the spam threshold, and the latest calls have been blocked. The type Warning User is a caller whose gray level is above the warning threshold but below the spammer threshold. Note that all the incoming calls from warning users were admitted, not blocked at all; but the administrator can watch to see if their behaviors perform the appropriate actions and respond to potential attacks appropriately.

Figures 6b and 6c show typical call patterns of a normal user and spammer, respectively. A typical normal user tends to make calls during a certain period of time, but does not call in other periods, such as late night or early morning. So the gray level of the normal user increases when he/she keeps calling, but decreases when they are not making any calls. In particular, the call interval is not that short (if we consider that it includes some time to dial, waits until the called party answers, and has a conversation, whether it is a person or an answering machine). So it is very likely that the gray level is maintained under the warning and spammer thresholds at most. However, as Fig. 6c demonstrates, the gray level of the spammer is likely to increase quickly as it keeps calling very often and goes above the spammer threshold. Note that in original PMG, when STGL reaches the spammer threshold, it resets to zero and LTGL takes its value after that. But in Fig. 6c, all the values are recalculated every minute, and each line (STGL, LTGL, and gray level) is drawn between two endpoints at each minute. Hence, it looks like STGL linearly drops and LTGL linearly increases from 8:45 a.m. to 8:46 a.m. In reality, STGL drops immediately to zero when it reaches the spammer threshold, and LTGL goes the opposite way.



■ Figure 6. *Snapshots of the web-based GUI of the spam blocker: a) a snapshot of the GUI of the spam blocker; b) a call pattern of a normal user; and c) a call pattern of the spam caller.*

Experiment

We carried out experiments with different settings on both implementations. The results were the same on both since PMG does not depend on implementation details. We generated three kinds of call patterns, regular, random, and Gaussian, and sent the calls to the available numbers in a round-robin fashion. The regular pattern generates calls in a fixed interval. The random pattern generates a random number ranging from 0 to a maximum number equal to twice the fixed interval of the regular pattern, so the average of the interval can be approximately the same as the fixed one, and the same number of calls can be generated in an observed period no matter which pattern is used. The Gaussian pattern is the same as the random pattern, except the Gaussian function is used instead of a random number generator.

First, we set **TL1** to 1 min, **TL2** to 1 h, **C1** to 3, **C2** to 1, and **T** to 1000. We generated calls every second from the same source and checked how PMG works. The reason to set C1 three times higher than C2 is that we hoped to make the STGL grow more quickly and block short-term attacks more efficiently. Table 2 shows how PMG worked when we generated 200 calls.

Trial	# of connect	# of disconnect	STGL	LTGL
1st trial	6	194	0	1484
2nd trial after 1 day	0	200	0	1474
2nd trial after 10 days	0	200	0	1365

Table 2. *The result of the first experiment (a call every second).*

In this experiment PMG accepted the first six incoming calls, but rejected the calls after that because the summation of two levels was higher than 1000. We did the second trial one day later and another 10 days later. Note that LTGL decreased much more slowly than STGL. In the second and third trials it did not accept any incoming calls from the same source since it takes many days for LTGL to come down below the threshold value of 1000.

In the second experiment we set the parameters to the same numbers as in the first except calls were generated every 5 s. Table 3 shows the results of the experiment.

Trial	# of connect	# of disconnect	STGL	LTGL
1st trial	32	168	0	1350
2nd trial after 1 day	0	200	0	1338
2nd trial after 10 days	0	200	0	1230

Table 3. *The result of the second experiment (a call every 5 s).*

The number of connected calls (32 calls in the first trial) in the experiment is higher than that in the first one (six calls in the first trial). This is because the interval, 5 s, is longer than the first, 1 s, and STGL and LTGL grow a bit more slowly. This coincides with the design rationale of PMG in that PMG is designed to respond and block malicious attacks more quickly than less malicious incoming calls. In fact, as the interval between calls becomes less, the more malicious the calls become because they can incur the denial of service, or outage of the call server. The numbers of connected calls in the second and third trials do not change at all.

The next two experiments have been carried out to find which is the better setting for **TL2**, makes PMG take effect longer and block spam calls in longer period of time. In these experiments, we set **TL1** to **10 mins**, **C1** to 1, **C2** to 1 and **T** to 1000. This time, we set the C1 equal to C2 and make the STGL grow less quickly than the first two experiments.

We also generated 200 calls every second for the first spam attacks in both experiments. The first experiment sets the TL2 to **100 mins** and the second sets it to **1 day**. Table 4 summarizes the results.

Trial	# of connect	# of disconnect
1st trial (TL2 = 100mins)	18	182
2nd trial 10 days later (TL2 = 100 min)	28	172
2nd trial 30 days later (TL2 = 100 min)	42	158
1st trial (TL2 = 1 day)	18	182
2nd trial 10 days later (TL2 = 1 day)	11	189
2nd trial 30 days later (TL2 = 1 day)	22	178

Table 4. *An experiment with different setting of TL2.*

As shown in Table 4, PMG worked better for the next spam attacks when we set TL2 long enough that LTGL did not decrease quickly. That is, when the second spam attack was tried 10 days later, the PMG with **TL2** set to 100 min passed 28 calls, but the PMG with **TL2** set to 1 day passed only 11 calls. When the second attack came 30 days later, the first PMG passed 42 calls, whereas the second PMG accepted 22 calls. However, if TL2 is set at a too high value, LTGL is not likely to decrease below the threshold for a long period of time, whicht may result in putting the spammer in the blacklist forever.

To summarize, how to set the configuration (or parameters) in PMG depends on the situation and objectives of each VoIP setting. Also, we can customize the configuration to each individual user or group of users so that PMG can perform optimally for them. These can be done manually by the administrator or automatically if we can calculate an optimal parameter setting after we observe the traffic patterns over a certain period of time.

Conclusion

While voice spam is not as well known to users as email spam, it must be taken as a serious threat to any VoIP environment. Marketers and spammers are already harassing users by sending unsolicited calls that consume not only our time, but also our precious bandwidth. While source IP names and addresses can be blacklisted, determined senders can easily use false information and thus route around fixed address blocks. It is

time to deploy a means to protect the VoIP environment from spam and stay one step ahead of the game.

This article proposes a SPIT protection algorithm called PMG, and shows that it can monitor call patterns and determine voice spam based on those patterns. When a call is received, PMG calculates the two gray levels from previous calling intervals. If the summation of the two levels is less than a given threshold, the caller is considered a regular user and the call is connected. If the summation is greater than the given threshold, the caller is regarded as a spam generator and blocked. Once a given threshold within a specified interval is reached, the caller can no longer place any calls. PMG, however, gives callers multiple chances to atone for their undesired behavior. If a blocked caller stops producing voice spam in a specified period of time, their block will eventually be removed. However, if they try spam attacks over again, PMG can block them at a quicker rate than before since they have a spam history.

PMG uses a caller's address such as a SIP URI for its identity. So if a spammer can get or change its address easily, PMG does not work well since it has to restart the gray level computation over again. Therefore, it has to be paralleled with a strong authentication mechanism that enables the provider to maintain a list of VoIP users in a secure way.

References

[1] T. Nolle, "Thinking Outside the VoIP Box," *IEEE Network*, no. 3, vol. 19, 2004, pp. 80.

[2] A. Dornan, "Bayesian Spam Filtering," *IEEE Network*, no. 3, vol. 19, 2004, pp. 64–65.

[3] R. Dantu and P. Kolan, "Preventing Voice Spamming," *VoIP Security — Challenges and Solution, IEEE GLOBECOM 2004*, Dallas, TX, Dec. 2004.

[4] R. Dantu and P. Kolan, "Detecting Spam in VoIP Networks," *USENIX SRUTI '05 Wksp.*, Cambridge MA, July 2005, pp. 31–37.

[5] Y. Rebahi and D. Sisalem, "SIP Service Provides and the Spam Problem," *2nd Wksp. Securing Voice over IP*, Washington DC, June 1–2, 2005.

[6] E. Harris, "The Next Step in the Spam Control War: Graylisting," http://projects.puremagic.com/graylisting

[7] L. Dang, C. Jennings, and D. Kelly, *Practical VoIP Using VOCAL*, O'Reilly, 2002.

[8] J. Kuthan, J. Janak, and Y. Rebahi, "SIP Express Router v0.11.0 — Admin Guide," http://www.iptel.org/ser/doc/seruser/seruser.html, 2001.

[9] J. Rosenberg *et al.*, "SIP: Session Initiation Protocol," IETF RFC 3261, http://www.ietf.org/rfc/rfc3261.txt, June 2002.

[10] Sun Microsystems, Java Telephony API Specification 1.4, http://java.sun.com/products/jtapi/

[11] E. Geschwinde and H.-J. Schonig, *PostgreSQL: Developer's Handbook*, 2002.

Biographies

DONGWOOK SHIN (dongwookshin@hotmail.com) was a research leader at Qovia. He graduated from Seoul National University and received a Ph.D. degree at KAIST in 1990. He was at Chungnam National University, Korea, as a faculty member. After that, he did XML related research at the National Library of Medicine, NIH. When he was at the National Library of Medicine, he developed two advanced XML information retrieval systems, XRS and XPERT, which were referenced broadly. During his tenure at Qovia, he focused on research on voice spam and denial of service protection, Cisco's VoIP solution, XNMP, and SIP and VoIP security, and filed several patents related to these areas. He also won the 2004 Maryland Innovator award hosted by the *Daily Record*. He is now a development lead at Master Key Consulting.

JIN YOUNG AHN (jinyoung123@gmail.com) is currently pursuing a Master's in computer science at Towson University. She has a Bachelor's degree in computer science from the University of Maryland, College Park. She has worked as a research intern at Qovia Inc. and was involved in developing a VoIP security application. She currently works as a research assistant at Towson University. Her previous research work includes VoWLAN capacity estimation and throughput measurement for UDP traffic in an 802.11 network. She is currently working on design and implementation of a P2P VoIP application. Her area of research is wireless networks, network security, and VoIP.

CHOON SHIM (cshim@adelphia.net) is responsible for Qovia's technology direction and development of the Qovia product line. He was previously president at Widearea Data Systems, where he designed and developed collaboration platform software. Prior to joining Widearea Data Systems, he was the senior development manager and principal engineer for Merant. He is a successful technology leader with over 20 years of experience architecting, building, and delivering large-scale infrastructure software products. He has extensive hands-on technical development skills and has successfully managed software teams for well-known enterprise software companies including BMC Software and EMC Corporation. He is a frequent speaker at VoIP and networking conferences for academia and industry; he recently chaired the VoIP Security Panel at SUPERCOMM '05. He has received numerous industry recognitions, including Innovator of the Year from the Department of Business and Economic Development of Maryland, and Executive of the Year from the Tech Council of Maryland. He has a B.S. in computer science and an M.S. in electrical engineering.