RepuScore: Collaborative Reputation Management Framework for Email Infrastructure

Gautam Singaraju and Brent ByungHoon Kang – University of North Carolina at Charlotte

ABSTRACT

We propose RepuScore, a collaborative reputation management framework over email infrastrucure, which allows participating organizations to establish sender accountability on the basis of senders' past actions. RepuScore's generalized design can be deployed with any Sender Authentication technique such as SPF, SenderID and DKIM. With RepuScore, participating organizations collect information on sender reputation locally from users or existing spam classification mechanisms and submit it to a central RepuScore authority. The central authority generates a global reputation summary which can be used to enforce sender accountability. We present the algorithms for reputation score calculation and share our findings from experiments based on a RepuScore prototype using a) our simulation logs and b) a 20 day log from a non-profit organization with five collaborating domains.

Introduction

In an effort to prevent sender address spoofing and phishing attacks, about 35% of all emails over the Internet are authenticated using sender authentication systems [13] such as DKIM [1, 23], SPF [22] and SenderID [11]. These systems allow receivers to authenticate the sender's mail server before email delivery to a mailbox.

Authentication schemes alone, however, do not provide the organization with the capability to differentiate between a credible sender and an unscrupulous one. Indeed, it has been noted that spammers have been the early adopters of these systems. This shows that a sender's identity does not necessarily guarantee their trustworthiness because trusting a sender can only be possible after verifying their past adherence to best mail practices. Currently, sender identification techniques are being used as the basis for determining the sender's history of adherence to best mail practices [21]. A reputation management system deployed at a single organization [3] has demonstrated that the history of the sender's adherence can provide an effective email classification mechanism.

We believe that organizations would benefit from sharing senders' reputation information that is individually collected at each domain. By collecting reputation scores from multiple organizations, the email receivers could access a complete history of a sender's past actions. Such a global perspective of a sender's reputation would allow receivers to trust a sender that they have no prior information about. As with receiver collaboration, a sender's spamming activity would be reported to all receivers: the onus is on the senders not to transmit unsolicited emails to any reputation-sharing receiver.

In this paper, we propose RepuScore, a reputation management framework for email infrastructure that uses receiver collaboration to compile global reputation for a sender. RepuScore helps create and maintain a trusted group of organizations. We discuss the deployment of RepuScore with sender authentication techniques. The design considerations for RepuScore are as follows:

First, the RepuScore framework can be used to collect, compute and share reputation among organizations. To keep track of the sender's history of adherence, RepuScore takes into account the reputation of the sender in the previous time frame along with the spam rate in the present time frame. Towards this, RepuScore employs the Time Sliding Window Exponentially Weighted Moving Average (TSW-EWMA) algorithm.

Second, RepuScore eases the overhead of reputation collection and computation with the help of a distributed architecture. Such architecture allows each organization to collect votes from its users. However, distributing the reputation management creates additional challenges.

Since RepuScore employs a distributed reputation framework, it is susceptible to Sybil attacks [20, 26]. In Sybil attacks, a malicious receiver manipulates the rating mechanism by creating multiple identities to give a higher rating to emails sent from the colluding senders and a lower rating to legitimate senders. Sybil attacks are thwarted by valuing a reputable participant's rating more highly than that of a less reputable participant. RepuScore employs the Weighted Moving Algorithm Continuous (WMC) [24] to thwart Sybil attacks. RepuScore introduces a participant voting threshold, a minimum threshold required by organizations to participate in global reputation computation, to mitigate Sybil attacks.

Third, RepuScore supports a centralized reputation scoring mechanism with minimal overhead. This centralized mechanism creates a trusted group of reputable senders. The lack of centralized enforcement has been cited as the main obstacle in tying email fraud to a particular user or organization [10].

The remainder of this paper is organized as follows. In the following section, we discuss the related work. We then describe the design issues for a reputation management framework and present the RepuScore design in the next section followed by its prototype implementation. We then discuss our results and conclude in the last sections.

Related Work

In this section, we discuss the reputation management frameworks that have been designed for email infrastructure, followed by a discussion on sender identity systems.

Reputation Systems for Email Infrastructure

SenderPath's Sender Score [19] and Habeas' SenderIndex [8] provide reputation for a sender's IP address. SecureComputing's TrustedSource [4] provides a global reputation system with the help of deployed mail servers in different organizations. Reputation based on IP addresses is not effective, as an IP address cannot be bonded to a specific organization [5]. For instance, when multiple organizations share an IP address, spammers in a single domain can affect the reputation of users in other organizations. Moreover, if organizations move to another service provider, their past actions would no longer be attributed to them. We believe that a reputation should be more closely associated with the organization, possibly utilizing the domain name of the organization.

Project Lumos [9] was proposed as an effort to provide reputation among collaborating ISPs. The receivers provided feedback as to whether a sender was a spammer or otherwise. Reputation was based on the activity of the previous 180 days. Project Lumos was designed to consider the weighted average of previous and present reputation of the senders. We believe that to thwart Sybil attacks and provide an open reputation management system, the reporter's reputation should also be taken into account in order to provide an accurate summary of a sender's reputation.

Google's reputation service [3] identifies the senders using best-guess SPF [22] or DKIM [1, 23] and computes the sender's reputation based on the inputs from users. This system demonstrated a high accuracy in classifying Google's emails. The paper also points out the need for a third party reputation framework.

Certification Systems for Email Infrastructure

Systems like SenderPath's SenderScore Certified [18], Habeas' Safelist [7] and Goodmail's Certified Email [6] are certification and accreditation services. These services allow bulk senders to obtain third party certification to be able to send bulk emails. They are not really reputation systems, as the sender maintains the reputation and not the receivers.

Identity Based Email Classification

Receivers can identify senders based on the sender's email ID, IP address or domain. For instance, PGP [15, 27] is an email Id-based authentication technique where a third-party server maintains individual users' public keys. The receivers verify the senders' signed emails by retrieving the sender's public key.

IP addresses are used to identity spammers in systems such as Blacklist IP and Real-time Blackhole List (RBL) [17] that keep a list of IP addresses that propagate spam. Though several RBLs are available, a recent study has shown that only 50% of spam is correctly identified by combined use of two or more lists [16].

We believe that maintaining a group of high-spam-propagating domains is more difficult than maintaining a group of non-spam-propagating domains. Spamming senders usually do not exist for long periods of time, whereas non-spamming senders usually exist for long periods of time.

SenderID [11] verifies the IP address presented by the email against that of the sender's registered mail servers. Using Sender Policy Framework (SPF) [22], a receiver thwarts sender forgery by identifying the sender's mail server through DNS entries. Domain Key Identified Mail (DKIM) [1, 13] publishes the mail server's public keys as a part of DNS records. Each email is signed by the sender's mail server. The signature is used by the receiver's mail server to verify the sender.

Accredited DomainKeys adds a central authority to DomainKeys architecture [12]. The centralized authority, called the Accreditation Bureau, maintains the sender domain's public key. The users should conform to a specified usage policy and adherence to the policy is checked periodically. We suggest that a reputation based mechanism where the receivers can vote on whether the senders adhere to the specified usage policy would help in enforcement of the usage policy.

RepuScore Design

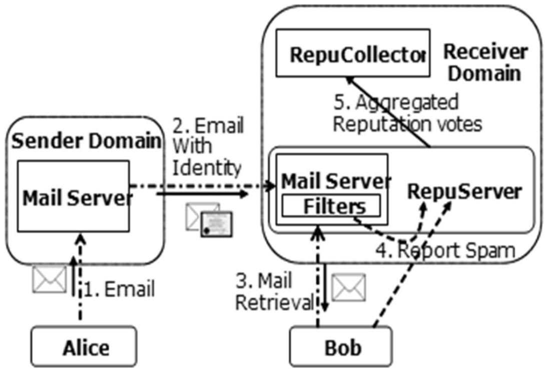
In this section, we describe the design considerations for a reputation framework. We note that authentication techniques and a reputation framework work together to create a trusted group of reputable senders. A verified identity (through an existing authentication mechanism) is a required basis for maintaining sender's reputation. Moreover, a reputation service is able to guide a receiver through the process of validating the sender before the sender's emails are accepted

Sender Identity Techniques

Email Id-identity systems, such as PGP, can be used to maintain reputation. However, using email ids entails maintaining a huge overhead for vote collection, storage and reputation computation. Instead of using email ids as identities for reputation management, we use domain authentication schemes, thereby decreasing the number of identities needed. We believe that this approach is more scalable than the email id based reputation system.

As mentioned, about 35% of all authenticated email over the Internet is authenticated using SPF, DKIM or SenderID [13]. A reputation management system can be built to help evaluate the senders who are being authenticated using these mechanisms. Such a mechanism will help evaluate the domains that adhere to a common guideline.

The lack of a centralized authority [10] has been noted as a main reason for the inability to tie email forgery to a single user or the organization. A central authority can maintain a trusted group of reputable senders where each sender needs to maintain a high reputation. Such a mechanism allows a common best email practice to be enforced among senders.



Listing 1: Collection of reputation votes by different RepuServers in an organization. The sender's identity could be used with any domain authentication technique. The reputation votes can be submitted either by users such as Bob or using any currently available filtering mechanism. Each domain maintains a single RepuCollector that collects the reputation votes from the multiple RepuServers in the organization.

Design of RepuScore Framework

A reputation management framework should only accept a single reputation vote from each organization. A large global organization might have multiple mail servers, each situated in different geographic locations, for example, in different countries. If a reputation management framework considered votes from mail servers, an organization with a huge number of mail servers would have greater say than organizations with a single mail server. Hence, each organization should be given a single vote that should be the aggregate of all the mail servers in the domain.

We define RepuServer as a mail server with the capability of verifying the users and collecting the reputation votes from them. Each local RepuServer at a domain collects votes from its users and email filters, aggregates the votes locally, and forwards them to the RepuCollector of the domain. We define a RepuCollector as an organizational level service that aggregates the votes from the local RepuServers and participates in a global reputation of peer RepuCollectors.

Each RepuServer records the total number of emails received during a given period and counts the ones that are considered to be spam by the currently available spam-filtering mechanism or by the user's input. Figure 1 demonstrates the mechanism for the receivers to report spam to their local RepuServer. Such reports can also be performed by currently available email classification techniques without user involvement. In the event that the report is conflicting, a user's input can be used to increase the reputation of the sender.

The RepuCollector's reputation should decrease for bad behavior and increase in the absence of bad behavior. For example, if spam is reported, the sender's reputation should decrease. If no spam is reported, the reputation should increase.

An ideal initial reputation is a requirement for building the reputation of a new RepuCollector or RepuServer. An improper initial reputation would give high spam propagating domains an unfair advantage as their reputation would stay high for a long time. In contrast, a low initial reputation would be unfair to a new domain as its emails would not be accepted by peers.

Since the sender's reputation changes over time and is computed after receiver collaboration, the reputation is computed in every time period. We define this period as the Reputation Aggregation Interval. A RepuCollector should invest a significant number of reputation aggregation intervals to be considered a good sender. Such a mechanism would make spamming unviable for a spammer as it would require a significant investment of resources, including both time and money. In addition, a quick reduction in reputation for non-adherence to the policy removes spammers from the trusted group of senders.

Finally, the reputation framework should guard against Sybil attacks [20, 26] where users with multiple identities attempt to change the reputation of the senders [24]. We believe that domains which transfer high amounts of spam would attempt to unfairly increase their reputations in order to be considered part of the trusted group of senders. To thwart such attacks, RepuCollectors having a reputation lower than a given threshold, which we refer to as the Participation Threshold, would not be allowed to participate in the voting mechanism.

RepuScore Prototype Implementation

Our framework, RepuScore, is a generic design that can be employed by sender identity systems. As discussed in the related work section, RepuScore creates a trusted group of reputable senders. We believe that a reputation framework should facilitate creation of such a group rather than just maintaining a group of blacklisted senders. In this section, we describe a generalized architecture that can be used with RepuScore. We generalize the architecture so that any sender authentication scheme can be used along with RepuScore. The architecture describes vote collection, reputation computation and also centralized reputation computation. Finally, we discuss the RepuScore algorithm used for reputation computation.

RepuScore differs from other approaches because of the collaborative reputation based on the scores suggested by peers. As RepuScore is designed as a collaborative mechanism, it has been designed to protect against Sybil Attacks, where a single attacker can take multiple identities. Towards this, RepuScore takes into account the reputation of the reporting server along with the reputation they report for a peer.

RepuScore Architecture

RepuScore's hierarchical architecture is designed so that the reputation collection and computation is manageable as the number of participating domains increase. The RepuScore framework computes reputation based on the votes collected by each RepuServer. While collecting reputation votes, a RepuServer checks the validity of the reporting users. The user's votes are based on their evaluation of the sender's adherence to best practices. We outline three major steps in RepuScore's architecture:

a) Reputation Vote Collection
b) Reputation Computation
c) Reputation Lookup Service

Reputation Vote Collection

As the definition of spam is subjective, an email regarded as spam by one user might not be considered so by another. Therefore, a global blacklist or white list would not be ideal as it would fail to represent the conflicting views of multiple users. RepuScore employs a social rating mechanism to consider the conflicting views of the users.

The receiver's RepuServer can maintain the number of emails received and the emails marked as spam for each sender RepuServer. The vote collection mechanism should require minimal participation from the users. For example, RepuServer collects the users' votes based on the users' implicit inputs. Users only need to mark an incorrectly filtered-email as non-spam or to report a spam email that was not correctly filtered by the spam classifiers. (Many email services provide similar mechanisms for their users to report a spam email or an incorrectly filtered email.) Figure 1 demonstrates the mechanism in which the RepuServer collects the votes from multiple users. Before accepting votes from the users, the RepuServer should validate the users.

The spam classifiers are also used along with users' input in collecting votes. A negative vote for a sender is entered when the spam filters determines an email as spam. Likewise, a positive vote for a sender is automatically made when the sender's email is not considered spam. In the event that the spam filter marks a legitimate email as spam, the users can mark the email as non-spam, submitting a positive vote for a sender to the RepuServer.

Reputation Computation

Based on the number of spam and emails collected, each RepuServer calculates the reputation of the sender RepuServer. RepuServer Reputation is defined as the weighted average of its reputation in the previous reputation aggregation interval and the reputation computed in the present reputation aggregation interval.

RepuScore calculates the reputation of a RepuCollector based on the reputation of the RepuServers maintained by the RepuCollector. We define the RepuCollector Reputation as the aggregate reputation of the RepuServers in their domain in the present reputation aggregation interval.

Each RepuCollector calculates the local reputation for each peer RepuCollector. The computed reputation is digitally signed by each RepuCollector to maintain the integrity of the data. To provide a global perspective, the locally computed RepuCollector's reputations should be collected by the Central Authority.

RepuScore introduces a central authority that collects reputation votes from all the RepuCollectors and computes the global reputation for all RepuCollectors. The central authority verifies the RepuCollector's votes based on the digital signature. The central authority should make sure that the reputation collection is conducted once every reputation aggregation interval. The central authority calculates a global reputation for each RepuCollector based on the change in its reputation as reported by peer RepuCollectors. The central authority takes into account the reputation of the RepuCollectors to compute the global reputation of the peer RepuCollectors. If the reporting RepuServers' reputation is below the participation threshold, their reputation votes are not factored into the global reputation.

Reputation Lookup Service

A reputation Lookup service can be provided with the help of a third party lookup service. The reputation lookup service can be similar to Realtime Black Lists. Such a reputation look up service can also provide a mechanism for the receivers to lookup the reputation of a sender's RepuCollector as reported by peers.

An alternate way for receivers to determine reputation is by associating the reputation with a sender identity that can be vouched for by a third party. For example, in the case of Accredited DomainKeys, the reputation can be embedded as the part of the seal that is supplied to the MTAs. When the client checks the DNS entries, the seal can be verified for the reputation.

RepuScore Algorithm

In this section, we discuss RepuScore's algorithm. The RepuCollector's reputation is calculated based on the reputation of all the RepuServers it maintains. We first demonstrate how each RepuCollector calculates the reputation of peer RepuServers. We then discuss the reputation computation of peer RepuCollectors. Finally, we demonstrate how a global reputation is calculated.

With the help of reputation, administrators in an organization can evaluate the compliance of the domain by checking their organization's reputation services. If the domain's reputation is lower than expected that would imply that there might be bots on the server [14].

RepuServer Reputation Calculation

As discussed in the above sections, a RepuServer's reputation is calculated by peer RepuServers. The reputation in RepuScore is always in the open interval (0, 1). A score of 1 indicates a highly reputable sender whereas a score of 0 indicates a sender with a low reputation. For all sender RepuServers, each receiving RepuServer maintains the number of emails received and the number among those marked as spam. The reputation of a RepuServer is computed as the number of good emails over the number of emails sent by a RepuServer in a particular interval. The reputation is calculated based on the modified time sliding window exponentially weighted moving average (TSW-EWMA) algorithm [2].

Equation 1 displays the weighted moving average. The RepuServer Reputation is based on the reputation in the previous interval and the reputation in the present interval. Correlation factor α indicates the amount of previous reputation considered for computation of the RepuServer's reputation in the new interval. If the correlation factor is high, the reputation of a sender takes a long time to increase or decrease, as a lot of weight is given to the previous reputation. However, if the correlation factor is low, the reputation increases or decreases very quickly since current actions are given additional weight. We demonstrate the effect of the correlation factor on reputation in our experiments.

RepuCollector Reputation Calculation

Based on the change in a RepuServers' reputation, the RepuCollector's reputation can be updated. Equation 2 shows the local reputation computation of a RepuCollector. The local RepuCollector reputation is the average reputation of all of its RepuServers. Each RepuServer transmits the reputation of the peer RepuCollectors to the central authority. As discussed in the above sections, the central authority considers the votes only from the RepuCollectors whose reputation is greater than the participation threshold. We demonstrate that such a mechanism helps in the creation of a trusted group of reputable senders.

Given:

- E_m is the set of emails received by RepuServer in reputation aggregation interval m
- S_i is the RepuServer i

For all $\boldsymbol{RS_i \in}$ ***Set*** $\boldsymbol{E_m}$:

- $NewRep(S_i, m) = 1 - \dfrac{n(spam_m)}{n(TotalEmails_m)}$
- $Rep\ (S_i, m) = \alpha \times Rep(S_i, m-1) + (1-\alpha) \times NewRep(S_i, m)$

Where:

- $Rep\ (S_i,\ m)$ is Sender RepuServer's Reputation in the interval m.
- $n(spam_m)$ is the number of spam received in the interval m.
- $n(TotalEmails_m)$ is the total number of emails received in the interval m.
- α $(0 \le \alpha \le 1)$ is the correlation factor between the previous and the present value.

Equation 1: Local RepuServer reputation.

Given:

- RepuCollector reputation is the average of the reputation reported to a RepuCollector's by its RepuServers.

For all RepuCollectors:

$$LocalCollectorRep_{co}(m) = \frac{1}{n(S_p)} \sum_{i=0}^{n(RW)} repu(S_i, m)$$

Where:

- $LocalCollectorRep_{co}$ is the local reputation of RepuCollector c reported by RepuServer o in the organization.
- $n(S_p)$ is the number of Sender RepuServer seen by a RepuServer o.

Equation 2: Local RepuCollector reputation.

Given:

- The RepuCollector reputation is weighted moving average continuous of local reputation computed in the mth Interval.

For all RepuCollectors, Central Authority calculates:

$$CollectorRep_l(m) = \sum_{n=0}^{i} \frac{CollectorRep_n(m-1) \times Local\ Collector_{ln}(m)}{\sum_{n=0}^{i} CollectorRep_n(m-1)}$$

Where:

- $RD\text{-}Repu_l$ is the global reputation of RepuCollector$_i$.

Equation 3: Global RepuCollector reputation.

The central authority calculates the global reputation of the RepuCollectors based on a modified Weighted Majority Algorithm (WMA) called WMA Continuous (WMC) proposed by Yu et al [24]. The WMC algorithm has been used in peer-to-peer systems to detect deception. We provide the participation threshold as a mechanism to remove domains that propagate spam and increase reputations of other senders.

Equation 3 demonstrates the Global RepuCollector reputation as the reputation-weighted average of the local RepuCollector reputation computed by each peer. The new reputation is computed once every reputation aggregation interval and is valid for one Aggregation Interval.

Experiments and Results

In this section, we demonstrate the effectiveness of RepuScore through experiments. We accomplish this with the help of a) simulated logs to demonstrate specific properties of RepuScore and b) real logs from a non- profit organization. The logs from the organization were 20-day logs collected by five domains that they maintained. The log contained information about 45K domains to which about 450K emails were sent, 55% of which were marked as spam by RBLs or rejected since the sender domain were determined not to exist through DNS reverse lookup.

Effect of α on Reputation of a Trusted RepuCollector With Sudden Increase in the Amount of Spam it Transmits

Spammers might attempt to thwart RepuScore by building reputation and then suddenly transmitting huge amounts of spam. In such cases, it is expected that the reputation of the sender would decrease and the spammer would be removed from the trusted group within a minimal number of reputation aggregation intervals.

To demonstrate the effectiveness of RepuScore, we created logs with 100 RepuCollectors spanning 45 reputation aggregation intervals. We selected a random number of RepuServers which reported to their local RepuCollectors. The number of emails and spams that were transmitted to and from an organization was perturbed using a random number; for example, since RepuScore creates a trusted group of reputable senders, the spam rate among them was set at under 20%, whereas a spamming domain's spam rate was set at greater than 95%. (We see this trend in the logs from the non-profit organization.)

Figure 2 demonstrates the reputation of a RepuCollector from which the amount of spam suddenly increased as a function of α. For the first 30 reputation intervals, the RepuCollector built its reputation and attempted to be a part of the trusted group. After reputation interval 30, the spam rate from the RepuCollector increased to 95%. The RepuCollector's reputation is based on the reputation of all its RepuServers. The jump in the value of reputation is due to the value of α and the initial reputation value of RepuDomain that was set at 0.5. Therefore, the reputation of the RepuCollector for α = 0.9 decreased from 0.7 after the first reputation aggregation interval. In cases where the sender does not propagate spam, the reputation should increase slowly, which indicates a long past history. Hence the high value of α implies an association for a long history of good actions. If the sender propagates spam, the reputation should decrease immediately, reflecting the current actions of the sender. A low value of α guarantees an immediate reduction when the sender propagates spam. Equation 4 demonstrates our change in the reputation algorithm to accommodate this behavior. Figure 3 demonstrates the change in reputation by employing the modified algorithm. For a high α, the reputation increases gradually but decreases more rapidly.

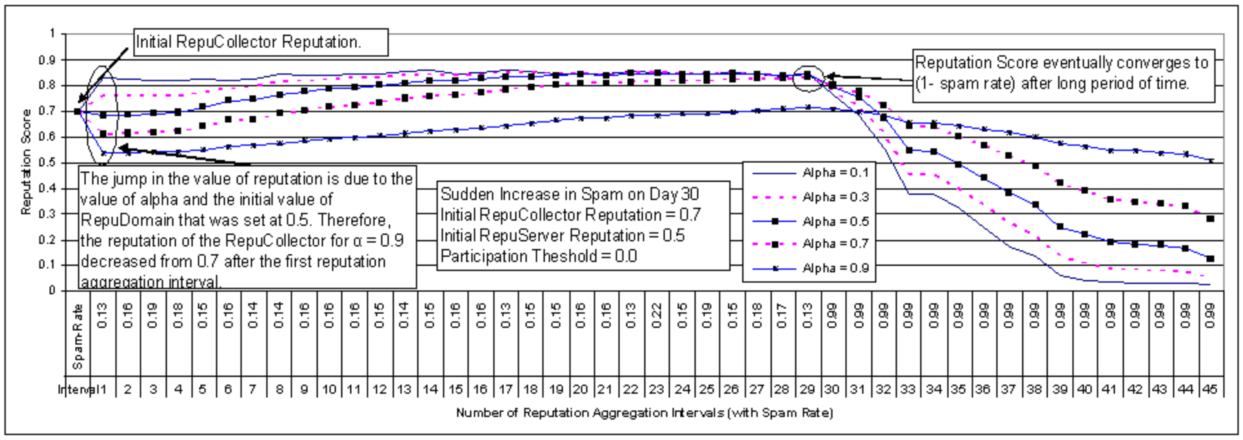


Figure 2: demonstrates the change in the reputation score of a trusted domain that transmits spam after reputation interval 30 as a function of α. The reputation eventually converges to (1 - average spam rate) over multiple reputation intervals. High α puts more weight to previous reputation score, whereas low α puts more weights to current score. Thus, for high values of α, it takes long time for the reputation to be built up whereas for low α value the decrease (or increase) in reputation is faster. The sudden drop from the initial score to the first interval is due to the effect of α. The RepuCollector's reputation has been set at 0.7. In the future intervals, the RepuCollector reputation is based on the reputation of all RepuServers which starts at 0.5. Therefore, for α = 0.9, the reputation of RepuDomain is around 0.55.

$$\begin{aligned}
&If\ (Rep\,(S_i, m-1) \geq NewRep\,(S_i, m))\\
&\qquad Rep\,(S_i, m) = \alpha \times Rep(Si, m-1) +\\
&\qquad\qquad (1 - alpha) \times NewRep\,(S_i, m)\\
&Else\\
&\qquad Rep(RS_i, m) = (1 - alpha) \times Rep\,(RS_i, m-1) +\\
&\qquad\qquad \alpha \times NewRep\,(S_i, m)
\end{aligned}$$

Equation 4: Local RepuServer reputation.

Figure 4 shows the modified RepuScore algorithm with collaboration among multiple domains using the 20 day logs from the non-profit organization. The reputation of the spamming domain decreased, but the reputation of a good domain increased.

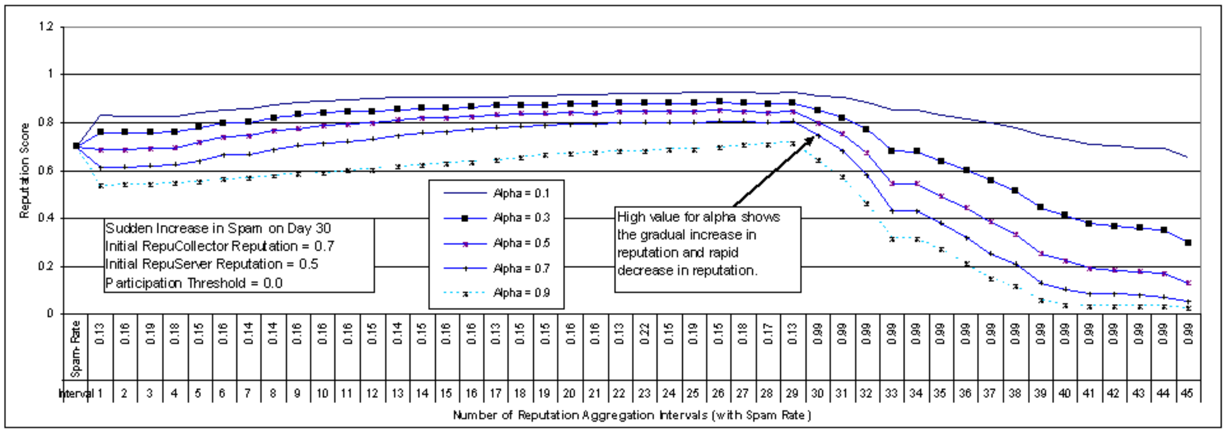
Participation Threshold and Initial Values for RepuCollector

Having an appropriate initial value for RepuCollector's reputation is extremely important to maintain a trusted group of reputable senders. For instance, if the initial reputation scores for the RepuCollector and RepuServers are set too high, it would take a long time for the reputation to decrease. On the other hand, if the initial reputation is set too low, it would take a long time for the reputation of a non-spamming RepuCollector to increase.

Our experiments show that an ideal initial reputation value for the RepuServer and the RepuCollector is between 0.5 and 0.7. With different initial values we noted that the average reputation of all the domains using the logs from the non-profit organization converged to about 0.6 for $\alpha = 0.1$, 0.47 for α=0.5 and 0.36 for $\alpha = 0.9$. Hence, an ideal initial reputation should be equal to the average reputation of all domains in the system after a long period of time. In order for the new reputation domains to participate in the reputation aggregation intervals, the threshold should be 0.1-0.3 below the initial reputation.

Resilience to Sybil Attacks

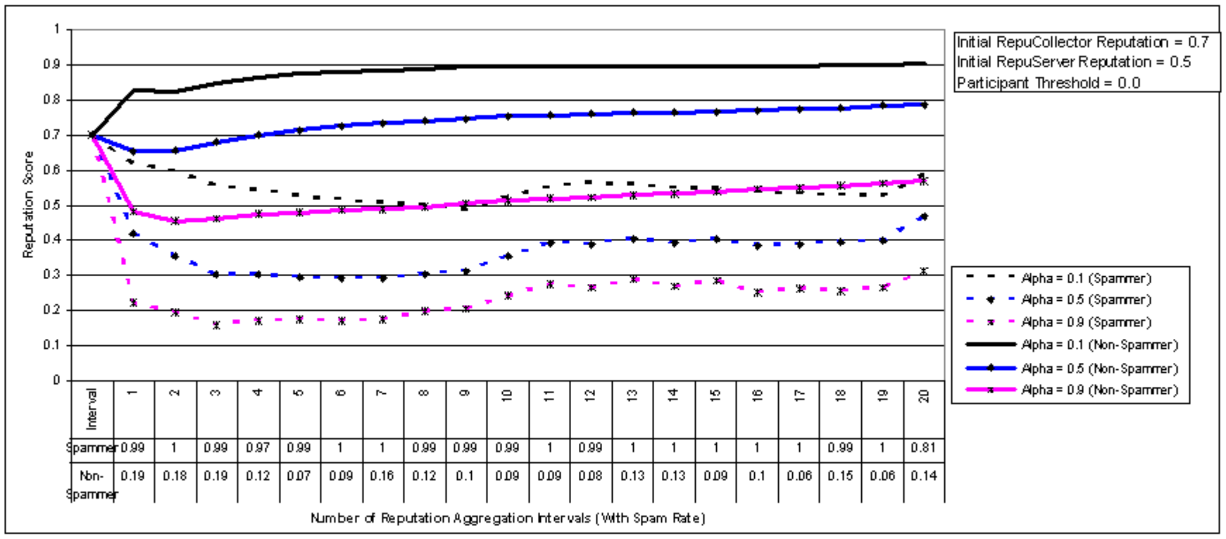
We increased the percentage of malicious RepuCollectors from 10 to 30% to demonstrate RepuScore's



Interval	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Spam-Rate	0.13	0.16	0.19	0.18	0.15	0.16	0.14	0.14	0.16	0.16	0.16	0.15	0.14	0.15	0.16	0.16	0.13	0.15	0.15	0.16	0.16	0.13	0.22

Interval	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
Spam-Rate	0.15	0.19	0.15	0.18	0.17	0.13	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Figure 3: In the modified RepuScore algorithm, a high value of α (other than 1.0) implies gradual increase, but fast decrease in reputation when the domain starts to spamming.



Interval	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Spammer	0.99	1	0.99	0.97	0.99	1	1	0.99	0.99	0.99	1	0.99	1	1	1	1	1	0.99	1	0.81
Non-Spammer	0.19	0.18	0.19	0.12	0.07	0.09	0.16	0.12	0.1	0.09	0.09	0.08	0.13	0.13	0.09	0.1	0.06	0.15	0.06	0.14

Figure 4: Using modified RepuScore algorithm with 20 days log from a non-profit organization. A number of new RepuCollectors were introduced at different reputation aggregation intervals.

resilience to Sybil attacks. Each RepuCollector transmits a high amount of spam (> 95%) for the first 30 reputation aggregation intervals. After 30 reputation intervals, we had the Sybil attacker to start increasing the reputation of its own Sybil domains and decrease the reputation of other domains. Figure 5 demonstrates our results. The reputation of the Sybil domains steadily decreased, but the reputation of the non-Sybil domains increased.

Conclusion

RepuScore is a collaborative reputation framework that collects votes from multiple organizations in order to collectively compute the reputation of a sender. We believe that RepuScore is a step toward enforcing sender accountability through collaboration among domains.

Simply blacklisting spammers is ineffective because spammers continue to easily create new sender identities. In contrast, a legitimate sender's identity typically exists for long periods of time. Thus, we believe a reputation framework such as RepuScore will be more effective in blocking spam email by maintaining a group of reputable trusted senders rather than identifying spamming domains.

RepuScore distributes the overhead for reputation collection and computation by using a distributed architecture while allowing a centralized authority to collectively calculate the global reputation for each sender domain.

Our experiments using simulated logs and an actual log from a non-profit organization demonstrated RepuScore's effectiveness and its ability to thwart Sybil attacks. We also presented the algorithms for reputation score calculation and demonstrated the effect of the correlation factor α where a sender's reputation increases gradually when it does not propagate spam but decreases immediately when it transmits spam.

Availablity

RepuScore will be an open-source effort aimed to provide participating domains with the ability to contribute information about senders and also lookup the collected reputation about them. RepuScore will be made available from http://isr.uncc.edu/RepuScore .

Acknowledgement

We would like to thank our shepherd, Peter Galvin, and the anonymous reviewers for their insightful comments. We would also like to show our appreciation to our LISA copy-editor, Rob Kolstad, for his excellent help. Finally, we thank our ISR lab member Sumeet Jain, for his help from the early stage of this paper.

Author Biographies

Gautam Singaraju is a fifth year doctoral student at University of North Carolina at Charlotte and is advised by Dr. Kang. Previously, he completed M.S in Computer Science at UNC Charlotte and a B.Tech in Electronics and Communication Engineering at JNTU, India. Since 2003, he has also been a volunteer System Administrator for a global non-profit organization. During the summer of 2007, he has worked at VMware with the performance group. Gautam can be reached at gsingara@uncc.edu.

Brent Hoon Kang received his Ph.D in Computer Science from the University of California at Berkeley, working on the Berkeley Digital Library and OceanStore project. Prior to Berkeley, he received an M.S in Computer Science from the University of Maryland at College Park, and a B.S in Computer Science and Statistics from Seoul National University. Since Fall 2004, he has been an assistant professor at the University of North Carolina (UNC) at Charlotte, and has been leading the Infrastructure Systems Research (ISR) Lab. As part of his research efforts, he recently worked on IT

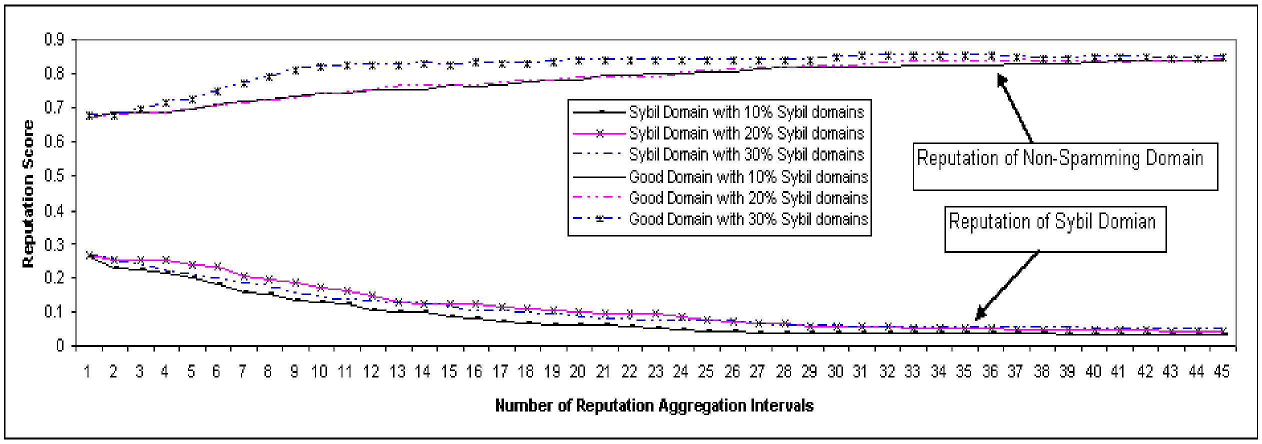


Figure 5: Multiple spamming domains (under a Sybil attacker's control) increase their votes for each other after the reputation aggregation interval 30. The domains give each other high reputation scores and attempt to decrease the reputation of other domains. However, our RepuScore framework was resilient to Sybil attack. The participation threshold was set to 0.

infrastructure design and administration issues related to protecting infrastructure against security threats such as the bots/malwares and the email spam/phishing problems. As part of his efforts on Information Assurance (IA) education program, he has been developing the hands-on cyber exercise components that foster students' creativeness and problem solving skills for IT systems design and defense. Hoon can be reached at bbkang@uncc.edu.

Bibliography

[1] Allman, E., *DomainKeys Identified Mail (DKIM): Introduction and Overview*, 2005, http://mipassoc.org/dkim/info/DKIM-Intro-Allman.html .

[2] Biswas, S. and R. Morris, "ExOR: Opportunistic Multi-Hop Routing for Wireless Networks," *Proceedings of ACM SIGCOMM '05*, Philadephia, 2005.

[3] Taylor, Bradley, "Sender Reputation in a Large Webmail Service," *Third Conference on Email and Anti-Spam (CEAS 2006)*, 2006.

[4] CipherTrust, *TrustedSource: The Next-Generation Reputation System*, White Paper, 2006.

[5] Dewan, P. and Dasgupta, P., "Pride: Peer-to-Peer Reputation Infrastructure for Decentralized Environments," *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, pp. 480-481, ACM Press, New York, NY, USA, May 19-21, 2004.

[6] Goodmail Systems, *Certified Email*, http://www.goodmailsystems.com/certifiedmail .

[7] Habeas, *Habeas Safe List*, http://www.habeas.com/en-US/Senders/Safelist/ .

[8] Habeas, *Habeas SenderIndex*, http://www.habeas.com/en-US/Receivers/SenderIndex/ .

[9] Brondmo, Hans Peter, Margaret Olson, Paul Boissonneault, *Project Lumos: A Solutions Blueprint for Solving the Spam Problem by Establishing Volume Email Sender Accountability*, 2003.

[10] Jakobsson, Markus, Steven Myers, *Phishing and Countermeasures, Understanding the Increasing Problem of Electronic Identity Theft*, Wiley, 2006.

[11] Microsoft Corporation, *Sender ID Framework – Executive Overview*, 2004.

[12] Goodrich, Michael T., Roberto Tamassia, Danfeng Yao, "Accredited DomainKeys: A Service Architecture for Improved Email Validation," *Proceedings of the Second Conference on Email and Anti-Spam (CEAS)*, 2005.

[13] Peterson, Patrick, "SIDF and DKIM overview Scorecard," *Authentication Summit II*, 2006, http://www.aotalliance.org/summit_archive/pdfs/2_Summit_Scorecard_final.pdf .

[14] Proofpoint, *High-performance Email Reputation and Connection Management*, March, 2007, http://www.proofpoint.com/products/dynamic-reputation.php .

[15] Price, W., *Inside PGP Key Reconstruction, A PGP Corporation White Paper*, 2003.

[16] Ramachandran, Anirudh and Nick Feamster, "Understanding the Network-level Behavior of Spammers," *Proceedings of ACM SIGCOMM*, Pisa, Italy, 2006.

[17] Realtime Blackhole List, Mail Abuse Prevention System LLC, California, 2002, http://www.mail-abuse.org/rbl/.

[18] Sender Score Certified, *Return Path Management*, http://www.senderscorecertified.com .

[19] Return Path, *Sender Score Email Reputation Management*, http://www.returnpath.com/delivery/senderscore .

[20] Srivatsa, M., L. Xiong, L. Liu, "TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Networks," *14th World Wide Web Conference (WWW 2005)*, Japan, 2005.

[21] Jordan, Stephanie, Matt Blumberg, Des Cahill, Richard Gingras, "Accountable Email: Building on Authentication," *Authentication Summit II*, 2006, http://www.aotalliance.org/summit_archive/pdfs/7_building_on_authentication.pdf .

[22] Wong, M. W., *Sender Authentication: What To Do*, Technical Document, 2004, http://www.openspf.org/whitepaper.pdf .

[23] Yahoo Inc., *DomainKeys: Proving and Protecting Email Sender Identity*, http://antispam.yahoo.com/domainkeys .

[24] Yu, Bin and Munindar P. Singh, "An Evidential Model of Distributed Reputation Management," *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2002.

[25] Yu, Bin and Munindar P. Singh, "Detecting Deception in Reputation Management," *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, Melbourne, ACM Press, 2003.

[26] Yu, H., M. Kaminsky, P. B. Gibbons, and A. D. Flaxman, "Defending Against Sybil Attacks via Social Networks," *Proceedings of ACM SIGCOMM Conference*, 2006.

[27] Zimmermann, P., *The Official PGP User's Guide*, MIT Press, Cambridge, 1995.