CLEI ELECTRONIC JOURNAL, VOLUME 11, NUMBER 2, PAPER 1, DECEMBER 2008

A Trust Model for a Group of E-mail Servers

Leonardo B. de Oliveira and Carlos A. Maziero
Graduate Program in Computer Science
Pontifical Catholic University of Paraná
Curitiba – PR, Brazil
{bispo,maziero}@ppgia.pucpr.br

Abstract

E-mail services are essential in the Internet. However, the current e-mail architecture presents problems that open it to several threats. Alternatives have been proposed to solve some problems related to e-mail services, offering reliability and scalability to such systems. This work presents a distributed trust model, allowing to create dynamic and decentralized trusted server lists, through the exclusion of servers recognized as spreaders of malicious messages. The trust model uses a social network approach and defines strategies for trust information update, propagation, and storage. A prototype was built to evaluate the proposed model's effectiveness.

Keywords: e-mail service, spam, trust networks.

1. I

E-mail systems are commonly used due to their simplicity, flexibility, and low costs for their implementation and usage. However, such systems suffer problems caused by fragilities in the protocols involved in the communication. Amongst them, there are problems concerning the absence of a robust mechanism for sender authentication, poor confidentiality and integrity mechanisms for message delivery, and also the lacking of a consistent reputation mechanism for users and e-mail servers.
Recent research works in the e-mail domain are proposing methods to solve such fragilities. Content filtering and classification algorithms are able to scan messages, looking for malicious patterns inside them. The use of cryptographic schemes on e-mail clients allows to cypher and sign all messages sent, ensuring message privacy and integrity. Finally, several methods for the authentication of e-mail servers have been recently proposed. All this research activity resulted in important advances in the attempt to solve problems related with authentication, integrity and privacy, but does not provide mechanisms to measure how trustful a server or domain is. In other words, server authentication mechanisms by themselves are not able to minimize the sending of malicious messages (spam).
This work presents a distributed trust model for e-mail servers that uses e-mail classification techniques, a sender authentication model, and social networks, to create an environment able to keep information about legitimate/malicious e-mail servers using a decentralized strategy. Section 2 brings a short review of main threats in e-mail services. Section 3 describes the main techniques used to authenticate e-mail senders; section 4 describes the use of social networks in a distributed environment. Section 5 shows the proposal architecture and details its functional aspects. Section 6 presents the model implementation and discusses some experimental results. Section 7 discusses some related work and compare them with this proposal. Finally, section 8 concludes this work and delineates some perspectives of continuity.

2. T E- S

Current e-mail systems were designed to be simple, as their main target was restricted to a small and trustful environment, constituted basically by the academic community. The SMTP protocol, responsible for e-mail transfers between servers [4, 11] does not provide robust mechanisms for authentication, privacy, and access control. Amongst the problems present in e-mail systems there are *spams*, virus and *scams*, which compromise their performance, robustness, security, and usability.
The most preeminent problem in current e-mail systems is spam, or UCE (*Unsolicited Commercial E-mail*). Recent estimates point that over 90% of all Internet e-mail traffic is spam. Many techniques are being used to control *spam* dissemination. The main techniques are based in trustful and non-trustful servers lists, or in scanning received messages to find suspect content:

- *Black Lists*: distributed *RBL* (*Realtime Blackhole Lists*) servers keep lists of IP address from *spam* spreaders or sources, which can queried through DNS to verify the sender trustworthiness [10];
- *White Lists*: each e-mail server can keep a list of trusted senders; this list is commonly managed through a web-based acknowledging mechanism [8]. A variant of this approach is *greylisting* [9], in which the list of trusted senders is dynamically built and managed;
- *Message Filters*: theses are programs that filter e-mails according to their contents, using statistical techniques, Bayesian classification, neural networks, header analysis, etc. [14, 20].

A second form of threat are virus and *worms*, which are malicious programs that can spread though e-mail messages. An e-mail virus normally is constituted by an e-mail with an executable file attached (or a HTML code able to load an executable file attached or stored in a remote server). This code, which can be unintentionally activated by the receiver or automatically loaded, aims to reproduce the virus and to perform other local actions, like damaging system files, installing spyware applications, and so on.
Finally, the *phishing scams* or simply *scams* are fake messages that use SMTP protocol fragilities to build social engineering attacks. Such attacks aim to deceive e-mail receivers, convincing them to inform personal data like bank account information, credit card numbers, etc. Scam prevention is usually done through e-mail content analysis.

3. S A

The very first step in e-mail security is sender authentication. It aims to guarantee that the source of a message is surely known (i.e. the message source was not forged). Many techniques have been developed with this goal, like *PGP* (*Pretty Good Privacy*), *SPF* (*Sender Policy Framework*), *SenderID* and *DKIM*. *PGP* [19] is a cryptography package aimed at cyphering e-mail messages. Using *PGP*, an e-mail is cyphered using a symmetric session key, and then this key is cyphered using the receiver's public key. This procedure ensures that the e-mail is kept confidential, as only the receiver can decrypt it. The sender's public/private keys are also used, in order to identify her.
SPF [18] makes use of *DNS* servers to provide to a receiver server information about the sender server. Such information is described using an specific language and stored in an extended *DNS* record. When the server receives a message, it performs a DNS query to get information about the origin domain, to verify the sender's authenticity. *SenderID* [17] is an *SPF*'s extension that improves the *SPF* language and extends the *SMTP* protocol, to solve problems related to the authentication of forwarded messages.
DKIM (*Domain Keys Identified e-Mail*) [1, 5] uses a public key system to validate the authenticity of an e-mail server. The sender's private key is stored in a local database, and is used for generating a signature for all e-mails sent, which is included in the e-mail header. The public key is stored in a *DNS* record that can be requested by the receiver server, to validate the signature.
All these techniques provide sufficient information to identify the sender of an e-mail, but they do not allow to judge about his/her trustworthiness. In this paper, our proposal is to combine one of such techniques with a distributed trust model, in order to provide more information about e-mail senders.

4. T R

In human relations, trust is the base for the construction and maintenance of a group of related individuals. Trust relations define how a person acts regarding other persons, known individuals and strangers. Trust relations can be classified as *Hierarchic trust*, *Social groups* and *Social networks* [16]:

- *Hierarchic trust* considers all relationships as an hierarchic model, for instance, the trust of a father on his sons. This trust relation can be represented as a tree, in which all nodes are individuals and the edges define the trust degree between each pair of nodes. Through transitivity, any two participants can define a trust degree between them, event if they have no direct connection.
- *Social groups* define sets of individuals whose activities are systematically related to a well-defined goal. Participants of social groups are able to share common interest information, propagating it to all other participants. Group participants can also make use of *trust groups* to start and keep relationships with other participants. Social groups can be represented as graphs, in which all nodes are participants and the edges are links between individuals.
- The *Relationship* word defines the interaction between individuals, based on reciprocal perceptions. *Social networks* are built from all relationships established by a person. It is possible to "walk" on the social network to find relationship paths between persons.

Computational systems can use the same principles of interpersonal relationships to define trust levels among their distinct elements [12]. In the following, we will present a proposal that applies concepts of social groups to the relationships among e-mail servers. Servers in a group share information about outsider servers, aiming to reduce the quantity of malicious e-mails received by the servers in the group.

5. A T M G E- S

This work defines an approach to create dynamic and decentralized lists of trusted servers, using e-mail classification and e-mail server authentication techniques. Each e-mail server receives messages, filters them and maintains a local opinion about the servers that sent them. Such local opinion is propagated to other servers in its trust group, to help them build a global opinion about the outsider servers. For that, we consider that e-mail servers are organized in trust groups $\mathcal{T}$, manually defined by their administrators[1]. For example, considering the e-mail servers present in figure 1, the trust group of server s_4 is composed by servers s_2, s_3, s_5, and s_9 ($\mathcal{T}_4 = \{s_2, s_3, s_5, s_9\}$).

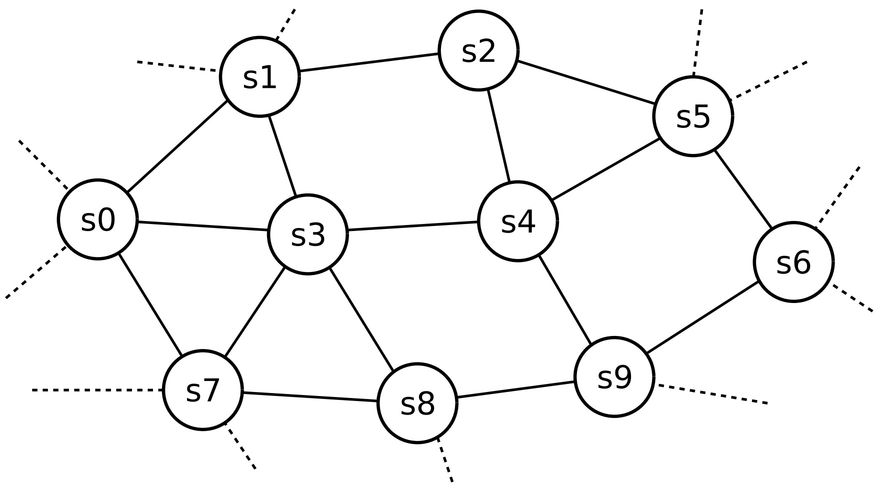

Figure 1: Trust groups

Servers that constitute a trust group cooperate between each other to share their opinions about outside servers known by the group, using trust network techniques. As there can be many distinct and partially overlapped trust groups, such trust information is gradually propagated from a trust group to its neighbor groups.

5.1 Architecture

The system architecture use concepts of trust networks, anti-spam tools, anti-virus tools, and an authentication model to build a trust management system among e-mail servers (*Mail Transport Agents* - MTAs). Figure 2 represents all components that will be implemented/integrated in each MTA participating in a trust group.

- **SMTP Server**: responsible for messages reception; it implements the SMTP protocol.
- **Sender authentication**: implements a domain authentication method, like *SPF* or *DKIM*.
- **Anti-spam and anti-virus**: classifies messages legitimate or malicious. The filter results are used by the trust management system.
- **Trust management system**: maintains trust information about outside servers, according to the messages received from them.

When a SMTP communication is established between two MTAs, the receiver server should execute the following steps for each message received (as indicated in figure 2):

1. Authenticate the server: if the server is correctly authenticated, go to next step, otherwise, reject the message and close the connection;
2. Verify if new messages can be accepted from that server, by querying the trust management system (the rules for this will be defined in section 5.2). If the answer is positive, the system receives the message, otherwise the connection is closed;
3. Submit the received e-mail to anti-spam and anti-virus filters. These filters will inform whether the received e-mail is legitimate or malicious;

[1]The automatic building of a trust group, although being a relevant research subject, is not considered here.

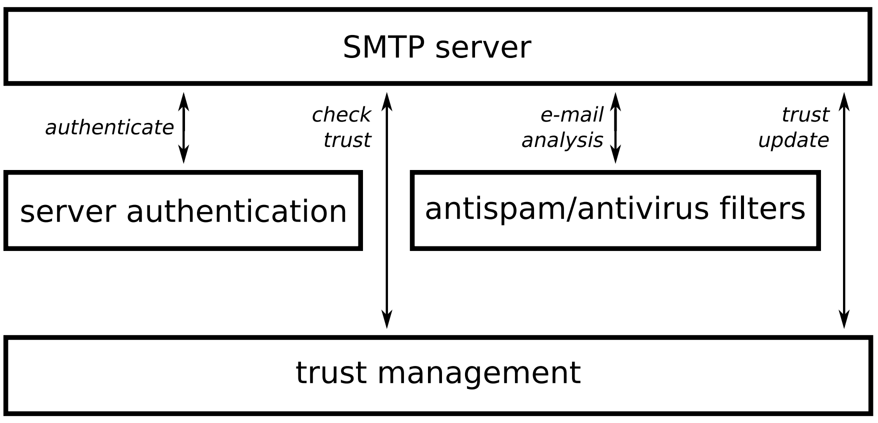

Figure 2: Architecture model

4. Send the filters results to the trust management system, to update the local trust information about that server.

The architecture was structured in three modules: *management*, *storage* and *trust propagation*, discussed in the following sections.

5.2 Trust Management

In a trust group, each server performs actions to build and maintain trust information about e-mail servers external to the group. For each external server, it maintains a *local trust*, defined by its local perception (messages received from that server), and a *global trust*, calculated using information provided by other members of its trust group(s). The combination of these two distinct trusts defines a *final trust*, used to determine the maximum number of messages that it will accept from the foreign server during a given time, defined here as a *cycle*[2]. An external server exceeding this limit is *banned*, i.e. no more messages coming from it will be accepted during the current cycle. The cycle duration is defined by the system administrator (in minutes, hours, days, etc.). Main attributes involved on trust calculation are described in Table 1 (all trust values are reals between 0.0 and 1.0, or 0% and 100%).

Table 1: Trust calculation attributes

Entity	Description
$\mathcal{T}_i$	Trust group of the SMTP server s_i.
$\mathcal{K}_i$	Set of external SMTP servers known by s_i. Servers belonging to its trust group are not considered external: $\mathcal{K}_i \cap \mathcal{T} = \emptyset$.
$tl_i^j(x)$	Local trust information: last value known by s_i of individual opinion of $s_j \in \mathcal{T}_i$ about a server $x \in \mathcal{K}_j$ (each server s_i locally stores values $tl_i^*(x)\ \forall x \in \mathcal{K}_i$.
$tg_i(x)$	Global trust: shows the opinion of the trust group about a server x, calculated by s_i using all local trusts from its trust group members.
δ_t	Trust variation stepping.
t_0	Initial trust to be attributed to new external servers, empirically defined as 50%.
$age_i(x)$	age of data about server x kept by s_i.
age_{max}	maximum value for $age_i(x)$.
$mm_i(x)$	Number of malicious messages received from x server by s_i in the current cycle.
$ml_i(x)$	Number of legitimate messages received from x server by s_i in the current cycle.
mm_{max}	Number of malicious messages to be received during a cycle to reduce by δ_t the local trust on a given server, considering an initial local trust of 100%. This value is defined by the system administrator.
$banned_i(x)$	indicates that server x was banned by s_i; if true, all messages from x will be rejected by s_i until the end of the current cycle.
lim	Number of legitimate/malicious messages necessary to increase/decrease by δ_t the local trust on a server.
tc	Current trust, used on the lim calculation. Defined as the geometric mean between the local trust tl and the global trust tg.

[2]Each server locally controls its cycles, there is no need for global cycle synchronization.

Procedure 1 describes all actions performed by a server $s_i \in \mathcal{T}_i$ when receiving a connection from an external SMTP server $x \notin \mathcal{T}_i$ to deliver an e-mail.

Procedure 1 when s_i receives a connection from server x:

```
1: if $x \notin \mathcal{K}_i$ then
2:     $\mathcal{K}_i \leftarrow \mathcal{K}_i \cup \{x\}$   // $x$ becomes a known server
3:     $banned_i(x) \leftarrow$ FALSE
4:     $tg_i(x) \leftarrow t_0$   // default initial trust
5:     $ll^i_i(x) \leftarrow t_0$
6:     $age_i(x) \leftarrow 0$
7:     $mi_i(x) \leftarrow 0$
8:     $mm_i(x) \leftarrow 0$
9: end if
```

The local trust information is reevaluated each time an e-mail is received. If the number of malicious messages received from a given server exceeds the maximum limit, that server will be banned until the end of the current cycle. Procedure 2 describes the actions performed by a server $s_i \in \mathcal{T}_i$ when it receives a message m from an SMTP server $x \notin \mathcal{T}_i$:

Procedure 2 when s_i receives a message m from server x:

```
1: if $banned_i(x)$ then
2:     rejects the message $m$
3: else
4:     accepts the message $m$
5:     $age_i(x) \leftarrow 0$
6:     $tc \leftarrow \sqrt{tg_i(x) \times tl^i_i(x)}$
7:     analyzes $m$'s contents
8:     if $msg\_legitimate(m)$ then
9:         $mi_i(x) \leftarrow mi_i(x) + 1$
10:        $lim \leftarrow tc \times mm_{max}$
11:        if $(mi_i(x) \geq lim) \wedge (tl^i_i(x) < 1)$ then
12:            $tl^i_i(x) \leftarrow tl^i_i(x) + \delta_t$
13:            $mi_i(x) \leftarrow 0$
14:            send $notify(x, tl^i_i(x))$ to trust group $\mathcal{T}_i$
15:        end if
16:    else
17:        $mm_i(x) \leftarrow mm_i(x) + 1$
18:        $lim \leftarrow (1 - tc) \times mm_{max}$
19:        if $(mm_i(x) \geq lim) \wedge (tl^i_i(x) > 0)$ then
20:            $tl^i_i(x) \leftarrow tl^i_i(x) - \delta_t$
21:            $mm_i(x) \leftarrow 0$
22:            $banned_i(x) \leftarrow$ TRUE
23:            send $notify(x, tl^i_i(x))$ to the trust group $\mathcal{T}_i$
24:        end if
25:    end if
26: end if
```

Procedure 2 shows that, when the local trust on server x decreases, it can be banned until the end of current cycle. The adoption of dynamic thresholds (using the *tc* and *lim* attributes) allows to decrease faster the trust on lowly-trusted servers than increasing it, and to increase faster the trust on highly-trusted servers than decreasing it. The limit *lim* on the number of messages that the server s_i can accept from the server x per cycle is calculated in line 10 (if the message m is legitimate) or in line 18 (if not). In both cases, the limit is calculated as a function of s_i's current trust *tc* on server x.

Finally, when a cycle ends, each server should execute the actions described in procedure 3. Such actions will re-enable banned servers, restart counters and "forget" information about servers that did not sent e-mails to s_i for a while. This forgetting mechanism is important, as it allows to consider only the recent behavior of external servers.

Procedure 3 when s_i finishes a cycle:

```
1: for all $x \in \mathcal{K}_i$ do
2:     $banned_i(x) \leftarrow$ FALSE
3:     $mi_i(x) \leftarrow 0$
4:     $mm_i(x) \leftarrow 0$
5:     $age_i(x) \leftarrow age_i(x) + 1$
6:     if $age_i(x) = age_{max}$ then
7:         $\mathcal{K}_i \leftarrow \mathcal{K}_i - \{x\}$   // "forget" server $x$
8:         remove all local information about $x$
9:         send $notify(x, undef)$ to the trust group $\mathcal{T}_i$
10:    end if
11: end for
```

5.3 Trust Storage

Trust information is locally stored in each e-mail server belonging to a trust group. The following information is maintained by s_i in a local database, for each known server $x \in \mathcal{K}_i$:

- Domain name of x (*FQDN - Fully Qualified Domain Name*);
- IP address(es) of x;
- Local trust $tl_i^*(x)$ and global trust $tg_i(x)$;
- Counters of legitimate messages $mi_i(x)$ and malicious ones $mm_i(x)$ received from x during the current cycle;
- Age $age_i(x)$ of the local information about x.

5.4 Trust Propagation

The trust propagation mechanism spreads out local information about servers known by s_i to all members of its trust group $\mathcal{T}_i$. Trust groups are defined by the e-mail service administrators and indicate which servers should be informed about updates on trust information about external servers (in this proposal, trust groups are manually defined).
As defined in procedure 2, when a server updates its local trust about an external server, it notifies its trust group members, using a *notify* message. When a server s_i receives a message $notify(x, t)$ from s_j, it updates its local information about x, performing the steps defined in procedure 4. Global trust is defined as the mean of local trusts on x for the servers belonging to $\mathcal{T}_i$; it is reevaluated by s_i after each cycle (procedure 5). If any server $s_j \in \mathcal{T}_i$ did not inform its opinion about x ($tl_i^j(x) = undef$), its value is not considered.

Procedure 4 when s_i receives $notify(x, t)$ from s_j:

```
1: if $(s_j \in \mathcal{T}_i) \wedge (x \in \mathcal{K}_i)$ then
2:     $tl_i^j(x) \leftarrow t$
3: end if
```

Procedure 5 when s_i finishes a cycle:

```
1: for all $x \in \mathcal{K}_i$ do
2:     $tg_i(x) \leftarrow mean(tl_i^j(x) \neq undef, \forall s_j \in \mathcal{T}_i)$
3: end for
```

6. I R

We developed a prototype of our proposal (called *TrustMail*) using *Linux*, the *Postfix* e-mail server, the *SpamAssassin* anti-spam filter, and the *Clamav* anti-virus filter. In the following, we will describe our implementation and the results obtained from its evaluation.

6.1 *TrustMail* Prototype

The prototype was implemented using C language and the SQLite database to store local trust information. Communication between *TrustMail* and other parts of the system (e-mail reception and filtering) is made through POSIX message queues. The MTA *Postfix* 2.2 was chosen due to the easiness of its integration with other tools. The authentication system chosen was SPF [18], because its implementation is simple and easy to integrate with *Postfix*. The interaction between the MTA and SPF, and the communication with *TrustMail*, are done through *policyd*, an open source SPF implementation (*Policyd* was modified to support communication with *TrustMail*).
SpamAssassin was chosen as anti-spam filter, because it is the *de facto* standard on Linux. The same applies to the *Clamav* anti-virus, which has a large virus signature list. Both run as *daemons*, easing the integration with other tools. The integration between *Postfix* and the messages filters was done through the *Clamav-Filter* script, modified to support malicious message notifications. The current prototype implementation is illustrated in figure 3. In this figure, is possible to observe that *TrustMail* does not depend on the e-mail server implementation, authentication system, nor messages filters. Thus, the proposed model could be easily integrated with other tools.

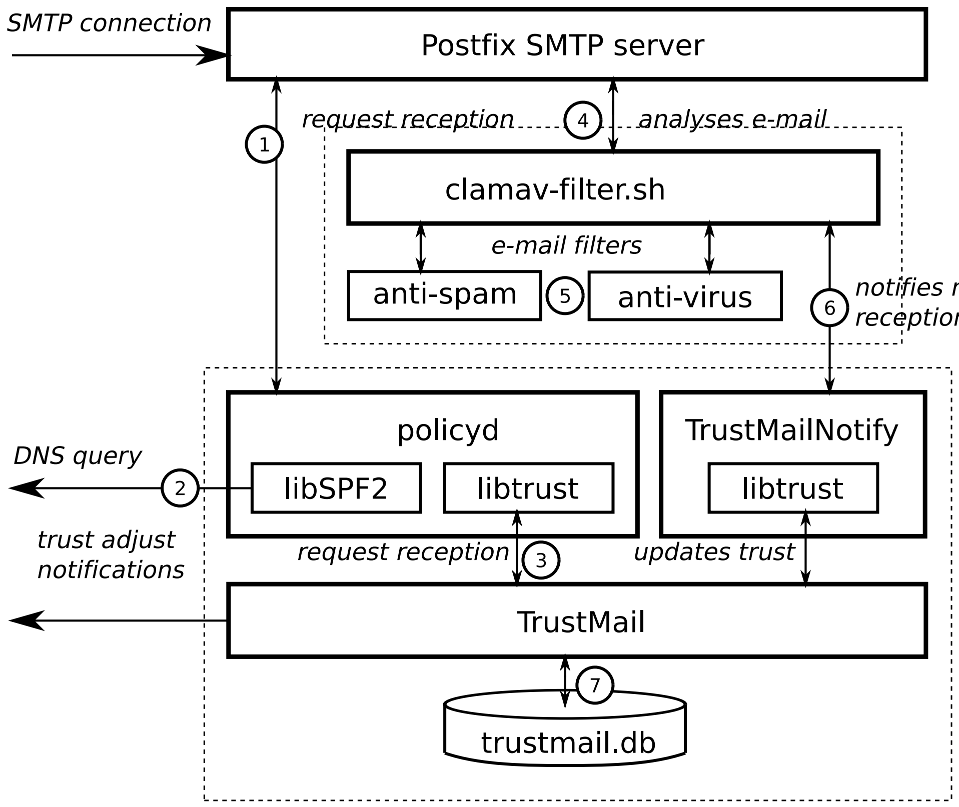

Figure 3: Architecture of the implemented prototype

The following steps are performed by the system components when an e-mail is received:

1. When receiving an SMTP connection, *Postfix* invokes *policyd* with the following parameters: *name*, *domain*, and *IP address* of the server requesting the connection.
2. *Policyd* makes a query to the DNS server responsible for the sender domain, to authenticate it. The query return can be:
   - *Pass*, *Soft Fail*, *Neutral*, *Unknown* or *None*: the procedure can continue;
   - *Error*: Returns the message “450 Temporary failure” to *Postfix*;
   - *Fail*: Return an error and the connection should be closed by *Postfix*.
3. If the server is correctly authenticated, *policyd* calls *TrustMail* to verify if the connection can continue. *TrustMail* calculates the maximum number of e-mails allowed to be received from that server in the current cycle and replies to *policyd*. *Policyd* then sends the response received back to *Postfix*.
4. After the e-mail is received, *Postfix* invokes the *Clamav-Filter* script, which following parameters: *IP address*, *name*, and *domain* of the sender server.

5. *Clamav-Filter* then submits the e-mail through *Clamav* and *SpamAssassin* filters.
6. Next, the *Clamav-Filter* calls the *TrustMailNotify* module, which notifies *TrustMail* that a malicious or legitimate message was received.
7. Finally, *Trustmail* updates the trust information stored in the local database *Trust_Mail.db*.

6.2 Experimental Results

The prototype was implemented and tested in a virtual machine environment using UML (*User-Mode Linux* [6]). The experiment used the topology shown in figure 4, in which four virtual machines implement the trust group ($s_0 \dots s_3$). Another virtual machine was used to simulate an external e-mail server (e_0), which randomly sends e-mails to the servers in the group.

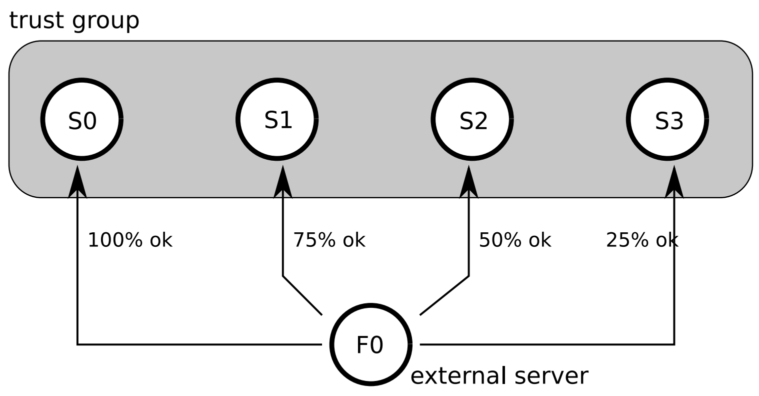

Figure 4: Experimental topology

In the experiment, the external server sends e-mails continuously and randomly (in average one e-mail each 15 seconds), uniformly distributed among the group servers. The external server has a distinct behavior concerning each server in the group: it never sends a malicious e-mail to s_0, but 25% of e-mails sent to s_1, 50% sent to s_2 and 75% sent to s_3 are malicious. The chosen cycle duration was 30 minutes, age_{max} = 10 cycles, t_0 = 50%, δ_t = 10% and mm_{max} = 10 messages. The total execution duration was 24 hours.

To evaluate the group influence on the decisions of each each server, we observed the evolution of local and global trusts about e_0 ($tl_i^i(e_0)$ and $tg_i(e_0)$) for each group member, in two circumstances: without trust update notifications (that is, without group cooperation) and with such notifications. This evaluation is presented in figure 5. It is possible to see that, when there is no cooperation among group members (curves $tl(s_i, x)$), each server will build its own isolated opinion (trust level) about the external server e_0. When notifications are used, cooperation among group members occurs (curves $tg(s_i, x)$), and trust opinions about e_0 converge to similar values.

In the same experiment, we also evaluated the quantity of messages rejected by the banishment of non-trusted servers (line 22 of procedure 2). The results are presented in figure 6. It is possible to see that, excepting s_0 (that did not receive spam), all the other servers reject significant amounts of the e-mails sent to them by e_0. This result shows that the banishment mechanism limits the quantity of messages that an external server can send to each group member, when the trust about it is not 100%.

Finally, we also evaluated the number of notification messages $notify(x, c)$ generated, compared to the number of e-mails received from the external server e_0. During the experiment, e_0 sent 19250 e-mails, which triggered the sending of 2072 notification messages by the trust group members. This amount of notifications means that one notification was produced for each 9.3 e-mails received by the group, which is a reasonably low value.

7. R W

Not much research was proposed using social networks to fight spam. The following works are considered by us the most relevant in this area.

The *MailRank* [3] system is a tool for global *white-lists* construction. Data about user activity are evaluated and grouped in a global social network. Each message sent by an user to another is transformed in a trust vote, used to build the social network. The system is compatible with the current e-mail structure, and is composed by two basic elements: the *MailRank Proxy*, used as a proxy between each e-mail client and its e-mail server, to extract information from e-mails sent and received by each user, and the *MailRank Server*, a central server that collects data from all *MailRank* proxies, to create a global classification of users based on the e-mails they sent and received.

On the other hand, the work presented in [2] shows the construction of an anti-spam tool that extracts relationship information between e-mail users through the analysis of e-mail headers (*From*, *To*, *Cc*, *Bcc*, etc.). This information is

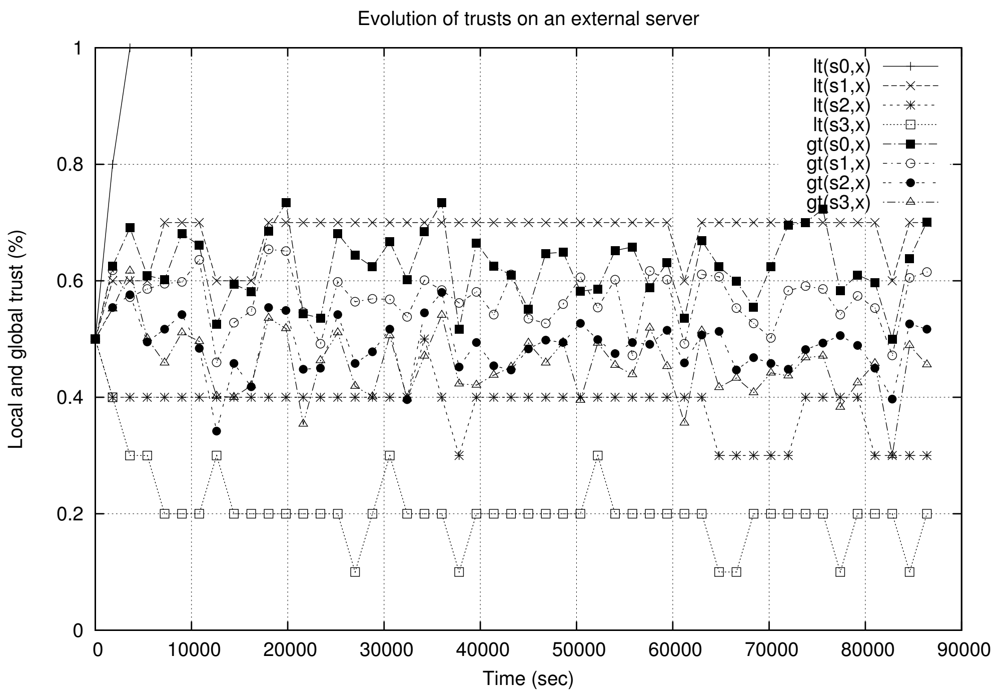

Figure 5: Local and global trust evolution

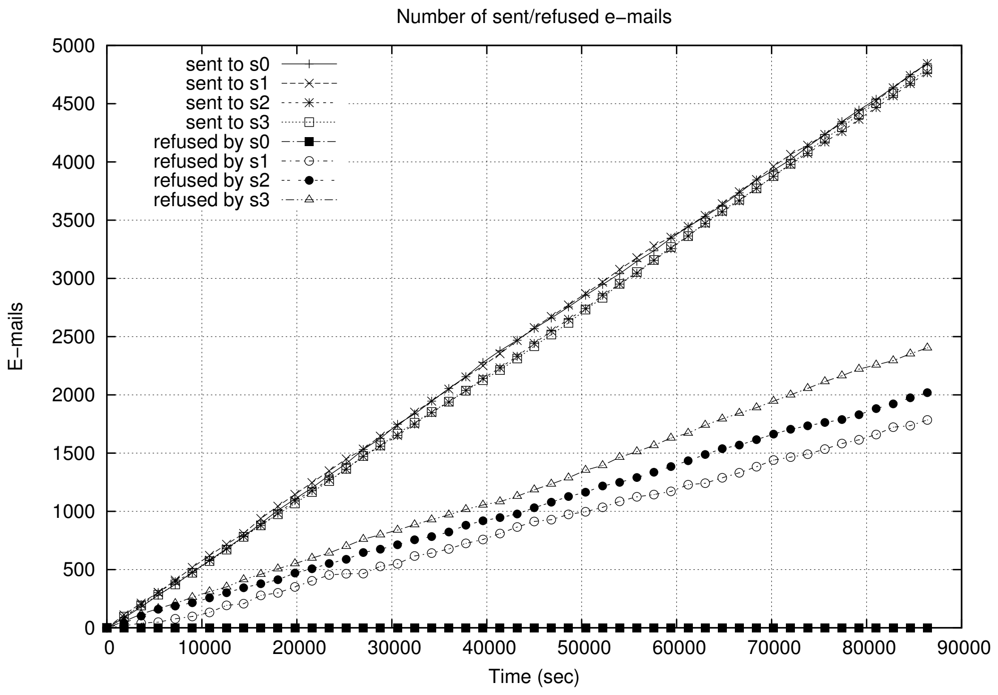

Figure 6: Number of messages sent/rejected by servers

used to build a huge graph of relationships among users. This graph is then used to build *white-lists* for all users, using a social network property called *agglomeration*. Finally, the agglomeration groups are analyzed to find spamming behavior patterns.

The paper [9] proposes a reputation-based greylisting mechanism, in which the time between message delivery attempts induced by the greylisting technique is modulated by a reputation value: higher reputed servers are imposed small waiting times, while suspect servers should wait much more to deliver their messages. However, the reputation is calculated using only local information.

In [15], the author explains the reputation system used in the popular *GMail* web-based e-mail service. Similarly to our approach, they also use the number of legitimate/malicious messages sent by an authenticated sender to calculate

the reputation of her domain. However, all reputations are calculated using only data locally available on the server. The paper [7] proposes a similar approach, in which the reputations are calculated for individuals rather than servers. A distributed architecture is proposed in [13], in which agents collect information from e-mail servers and send it to *nomination* servers to be classified. The nomination server generates signatures for spam messages and feeds them to a trust evaluation system, which can be queried by the e-mail servers. All reputation calculations are centralized in this component.
Our work differs from those in the following aspects: when they classify users as spammers or not, our work concentrates on the behavior of e-mail servers, allowing to a better scalability; moreover, our proposal presents a fully decentralized architecture, in which a group of mutually trusted servers asynchronously cooperate to build global reputations, while the other works need some centralized server for the construction and analysis of the social network. Finally, our proposal is compatible and may coexist with the standard e-mail systems in use today.

8. C

This work proposed and evaluated a trust model for e-mail servers. It defines trust groups whose members interact to exchange "opinions" about external e-mail servers. Each server in a trust group uses the other members' opinions to build a global trust, and uses this information to limit the quantity of e-mail received from an external server. Local trust information is propagated to the trust group using a social network model, decentralizing the maintenance and evolution of trust information.
It is important to observe that the proposed model does not substitute conventional e-mail security mechanisms like sender authentication, anti-spam or anti-virus systems, but complements them. In this sense, it provides information on how other trusted servers (the group members) consider third-party e-mail servers. In our proposal, such information is used to regulate the message flow coming from suspect servers, but it could be used in other more creative ways, like the one proposed in [9].
Other aspects of this work that can be explored in future works are a) the definition of a reputation measure between the members in a trust group, allowing the automatic inclusion or exclusion from members of a group; b) the definition of optimal values for the model's constants; and c) a deeper study of the trust information spreading mechanisms among distinct groups with common servers, in a scalable way.

References

[1] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas. RFC 4871: DomainKeys identified mail (DKIM) signatures, May 2007.

[2] P. O. Boykin and V. P. Roychowdhury. Leveraging social networks to fight spam. *IEEE Computer*, 38(4):61–68, 2005.

[3] P. A. Chirita, J. Diederich, and W. Nejdl. Mailrank: Using ranking for spam detection. *ACM International CIKM Conference*, 2005.

[4] D. Crocker. RFC 822: Standard for the format of ARPA Internet text messages, Aug. 1982.

[5] M. Delany and Yahoo. Domain-based email authentication using public-keys: Advertised in the DNS (DomainKeys). Internet Draft, 2004.

[6] J. Dike. A user-mode port of the Linux kernel. In *Proceedings of the 4th Annual Linux Showcase & Conference*, Atlanta - USA, 2000.

[7] J. Golbeck and J. Hendler. Reputation network analysis for email filtering. In *Conference on Email and Anti-Spam – CEAS*, 2004.

[8] R. J. Hall. How to avoid unwanted email. *Communications of the ACM*, 41(3):88–95, 1998.

[9] A. Janecek, W. Gansterer, and K. Kumar. Multi-level reputation-based greylisting. In *3rd Intl Conference on Availability, Reliability and Security*, 2008.

[10] J. Jung and E. Sit. An empirical study of spam traffic and the use of DNS black lists. In *Internet Measurement Conference*, 2004.

[11] J. Klensin. RFC 2821: Simple mail transfer protocol, Apr. 2001.

[12] V. Krebs. The social life of routers: Applying knowledge of human networks to the design of computer networks. *Internet Protocol*, 3(4), Dec. 2000.

[13] V. V. Prakash and A. O'Donnell. Fighting spam with reputation systems. *ACM Queue*, 3(9):36–41, 2005.

[14] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[15] B. Taylor. Sender reputation in a large webmail service. In *3rd Conference on Email and AntiSpam – CEAS*, 2006.

[16] S. Wasserman and K. Faust. *Social Networks Analysis: Methods and Applications*. Cambridge University Press, 1994.

[17] Wong, Microsoft, and Lentczner. The SenderID record: Format interpretation. http://www.ietf.org/internet-drafts/draft-ietf-marid-protocol-02.txt, 2004.

[18] M. Wong. Sender Policy Framework (SPF): A convention to describe hosts authorized to send SMTP traffic. http://db.org/drafts/internet/mengwong/spf/00/, 2004.

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

[20] C. C. Zou, W. Gong, and D. Towsley. Code Red worm propagation modeling and analysis. In *ACM conference on Computer and Communications Security*, 2002.