A Reputation Framework for Behavioural History

Developing and Sharing Reputations from Behavioural History of Network Clients

Anirban Basu
Software Systems Group
School of Informatics
University of Sussex

A thesis submitted, on January 04, 2010, in partial fulfilment of the requirements for the degree of Doctor of Philosophy (DPhil) in the School of Informatics of the University of Sussex.

To my parents
and
my late grandparents

The water in a vessel is sparkling; the water in the sea is dark. The small truth has words which are clear; the great truth has great silence.
Rabindranath Tagore
Indian poet, artist & philosopher (1861–1941)

Abstract

The open architecture of the Internet has enabled its massive growth and success by facilitating easy connectivity between hosts. At the same time, the Internet has also opened itself up to abuse, e.g. arising out of unsolicited communication, both intentional and unintentional. It remains an open question as to how best servers should protect themselves from malicious clients whilst offering good service to innocent clients. There has been research on behavioural profiling and reputation of clients, mostly at the network level and also for email as an application, to detect malicious clients. However, this area continues to pose open research challenges. This thesis is motivated by the need for a generalised framework capable of aiding efficient detection of malicious clients while being able to reward clients with behaviour profiles conforming to the acceptable use and other relevant policies.

The main contribution of this thesis is a novel, generalised, context-aware, policy independent, privacy preserving framework for developing and sharing client reputation based on behavioural history. The framework, augmenting existing protocols, allows fitting in of policies at various stages, thus keeping itself open and flexible to implementation. Locally recorded behavioural history of clients with known identities are translated to client reputations, which are then shared globally. The reputations enable privacy for clients by not exposing the details of their behaviour during interactions with the servers. The local and globally shared reputations facilitate servers in selecting service levels, including restricting access to malicious clients. We present results and analyses of simulations, with synthetic data and some proposed example policies, of client-server interactions and of attacks on our model. Suggestions presented for possible future extensions are drawn from our experiences with simulation.

Acknowledgments

Special thanks to my supervisors Dr. Ian Wakeman and Dr. Dan Chalmers for their invaluable guidance, constructive criticism and support over the past four-plus years.

This work would have never been completed if it were not for the support of my parents, family and my friends. In particular, I would like to thank my parents for their outstanding support and unparalleled understanding throughout the years.

I would also like to extend my thanks to other members of the Software Systems Group, in particular: Dr. Des Watson, Dr. Jon Robinson, Jian Li, Stephen Naicken, Ryan Worsley, Yasir Malkani, Roya Feizy, Lachhman Dhomeja. Also thanks to members of the Theory Group, in particular: Dr. Berhard Reus and Dr. Ian Mackie. Special thanks to Simon Fleming and James Stanier from the Software Systems group for proof-reading this thesis.

I would also like to thank Dr. Colin Brooks and Mrs. Vivien Brooks for hosting me in their cosy home during the later period of my thesis write-up.

Many thanks to the (erstwhile Department and now) School of Informatics computing service as well as my supervisors for all the computing facilities that I have been provided with.

Table of Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

There's only one corner of the universe you can be certain of improving, and that's your own self.
Aldous Huxley
English critic & novelist (1894 - 1963)

1.1 Motivation

Historically the origins of the Internet (Leiner *et al.*, 2001) can be traced back to the seminal proposal by Leonard Kleinrock (Kleinrock, 1961), then at the Massachusetts Institute of Technology, in which he argued the theoretical feasibility of communication using packet switching instead of circuit switching. Kleinrock's theoretical packet switched network was followed by a key proposal by Lawrence G. Roberts (Roberts, 1967), which led to the eventual development of the ARPANET – the predecessor of the modern Internet. The Internet is built on the concept of open-architecture networking. It was based on the idea that it would comprise multiple independently developed networks of arbitrary design starting with the ARPANET. Robert Kahn first introduced this notion of open-architecture networking, and decided to develop a new version of the Network Control Protocol (Crocker, 1970), which was the original ARPANET host-to-host communication protocol. Kahn's work with Vint Cerf (Cerf & Kahn 1974) was first distributed at the International Network Working Group in 1973 at a conference at the University of Sussex. This laid the foundations of the Transmission Control Protocol (Postel, 1981) and

later the Transmission Control Protocol/Internet Protocol (TCP/IP) (Socolofsky & Kale, 1991).

Various protocols were eventually developed to support the concept of open-architecture inter-networking, which enabled easy and large-scale sharing of resources. This very large-scale open inter-network is what we call the modern Internet. While the Internet makes it convenient for heterogenous hosts to communicate with each other, share resources, provide and consume network services, the open-ness in the design of the Internet created opportunities for abuses and attacks, e.g. the proliferation of unsolicited network communication, both intentional and unintentional.

Back in the days of the ARPANET in 1972, the first "hot" application that led to the massive growth of "people-to-people" network traffic over the modern Internet, was introduced – electronic mail, now commonly known as email. Now, as of November 2009, Senderbase Security network (Cisco Systems, 2009b) observes that email spam or unsolicited bulk email constitutes over 80% of all emails sent out per day worldwide. Besides being a significant cause of annoyance and inconvenience to email users and network administrators, the financial implications of email spam are also very high. In 2001, a study conducted by the European Commission estimated email spam to cost €10 billion per year to Internet users, in connection costs, worldwide (European Union, 2001). Despite various legal and technical anti-spam measures currently in place, email spam has grown steadily over the last decade to a total worldwide volume, according to Senderbase, of occasionally over 300 billion messages per day.

Similar to spam targeting email communications, unsolicited bulk messages affect other services provided over various networks, such as instant messaging, newsgroup and web forums, online gaming, hypertext search engines, blogs and wikis, video sharing, text messaging over cellular networks, amongst others. Some of the research efforts in identifying and proposing solutions to the problems of spam over such services include Cramer 2002; Enck *et al.* 2005; Gyöngyi & Garcia-Molina 2005; Kolan & Dantu 2007; Singhal 2004; Wu & Davison 2005 amongst others.

With the growth of the Internet, especially the World Wide Web (Fielding *et al.*, 1999), there is a growing need to establish trust mechanisms to facilitate reliable communications between individuals and between organisational groups. Trust and reputation systems have been developed to assess the reputations of service providers in order to assist clients in the selection of trustworthy service providers for centralised (e.g. survey of trust: Grandison & Sloman 2000; review of online reputation systems: Zheng & Jin 2009) and de-centralised (e.g. Aberer & Despotovic 2001; Damiani *et al.* 2003; Mengshu *et al.* 2005) network applications.

Above, we have mentioned attacks in the form of unsolicited messages that are possible due to the open architecture of the Internet. A large number of attacks on and abuses of Internet services fall under the following classes:

Abuse of service terms

In this attack, the attacker abuses the terms of a service, e.g. requesting more than an allowed number of TCP connections at a time which could be a result of applications using peer-to-peer protocols in a network where such protocols are disallowed.

Denial of Service (DoS)

In this type of attack, a network resource is exhausted or significantly limited by malicious connections leaving insufficient resources for genuine connections. For example, using a TCP SYN flood (Eddy, 2007) an attacker can consume resources on a server by not sending the ACK response and thus keeping half-open TCP connections. A sufficiently large number of half-open connections may consume enough resources to make the server deny connections to legitimate users. DoS can be distributed (DDoS) when the attack originates from more than one network entity, all operating with the same common malicious objective, i.e. to make the servers deny services to legitimate clients. The set of network entities participating in the attack can be created by the attacker through a number of techniques, for example, an attacker may suborn (e.g. using operating system exploits) a number of unsuspecting network entities.

Unsolicited messages

The attacker sends (an usually large number of) unsolicited messages

(e.g. email spam), which can have financial or legal implications to the victims. Similar to DDoS, the attacker could have multiple Sybil (Douceur, 2002) identities working with the same malicious intent. The intent of an unsolicited message could be to promote products or services, extort sensitive information (e.g. phishing), and so on.

In this thesis, we are mainly concerned with the attacks that either constitute *abuse of service terms* or *unsolicited messages*. There are a number of defences to the aforementioned attacks, some of which we classify as follows:

Network intrusion detection

Audit trails of network packet traces over a network are analysed in retrospect and in real-time to detect anomalies or patterns of known attack signatures. Network intrusion detection can be a collaborative effort aided by participants over an enterprise-wide network or across the entire Internet.

Connection classification

By analysing activity at the end-to-end network layer, a particular network interaction is classified as either malicious or genuine, e.g. an email message, instead of network packets, sent to an SMTP server is subject to content analysis to distinguish whether it is spam or not. If necessary, connection classification may also be applied at lower points in the protocol stack.

Use of behavioural history and reputation

Interactions with a network client identity are decided based on its past local and/or global records of the behaviour and/or any reputation. This leaves room for forgiveness of occasional bad behaviour while identifying repeat offenders. We shall be investigating this type of defence in this thesis.

There are various academic and commercial research efforts to reduce and restrict spam, email spam in particular. Technologies currently in use against email spam include statistical content filtering, blacklisting and whitelisting, firewall port blocking, cost-based rate limiting, reverse DNS checks, SMTP proxy, authentication and challenge-response systems, behaviour profiling and

reputation systems, amongst others. Of particular interest to us is the use of behaviour profiling and reputation (e.g. Golbeck & Hendler 2004; Hershkop & Stolfo 2005; Stolfo *et al.* 2003).

Alongside the use of behaviour profiling as anti-spam measures, researchers in network intrusion detection systems are also looking into the use of behaviour profiles to improve the accuracy of intrusion detections, e.g. Maier *et al.* 2008. In his thesis (Sommer 2005) on network intrusion detection in high-performance network environments, Robin Sommer cited Allman's work as a step in the direction of large-scale sharing of past behavioural patterns of network hosts. Our work is primarily motivated by this key paper (Allman *et al.* 2005b) from Allman, Blanton and Paxon, who presented an architecture of large scale sharing over a Distributed Hash Table of behavioural history of network actors, such as hosts or email addresses in an effort to inform policy decisions about how to treat future interactions. Local observations on the behaviour of actors are submitted to a behavioural history database by *reporters* while evidence of such observations rather than confirmations of the actual observations are also submitted by *witnesses*. This behaviour history database can be queried by service providers to lookup behaviour reports on clients, which can be used to make policy decisions regarding interactions with those clients.

Another key motivational paper behind this thesis, building on Allman's work, is Wei & Mirkovic 2007, which proposed a client reputation system that service providers can use for deciding to accept or decline interactions with a given client. According to the authors, such a system could significantly aid defenses against major security threats (e.g. intrusions, distributed denial-of-service attacks) with a prior knowledge of a client's trustworthiness, provided by reputations. It is also argued that client reputations could be used for traffic prioritisation during periods of network congestion.

1.2 Contribution

Instead of devising independent solutions for the different problems arising out of abuses of service and unsolicited communications, we investigate the

feasibility of a framework-based solution for building and sharing reputations with the following properties:

- *Generalised:* The framework is expected to be independent of the specific implementation of any application or protocol.
- *High-level:* The framework is to allow use of reputations in the higher end-to-end layers.
- *Privacy-preserving:* Reputation instead of behavioural history is to be shared which allows hiding behavioural details behind a rating system. In addition, the framework is to support anonymous sharing of reputation as a further step for privacy protection.
- *Policy-independent:* The framework is expected to be independent of any policy, and it should be possible to fit in policies at various stages of the framework.
- *Context-aware:* Behaviour analysis and reputations are relevant only in particular application scenarios. The framework is expected to be aware of such scenarios (i.e. contexts) while allowing the development and sharing of reputations.

In this thesis, we propose a framework of *client reputation* based on behavioural history. The objective of our framework is to aid service providers in building, and sharing anonymously, the reputations of their clients from their behavioural profiles. We envisage that such reputations can be used not only in deciding to accept or decline interactions but also to vary levels of service within the bounds of the relevant service contracts. Our framework is expected to augment existing detection systems, e.g. intrusion detection, statistical content filtering, and so on. Policies are applied to these observed client behaviour profiles to develop quantised behaviour values at various points in time. Such quantised behaviour data is then subject to further policies by the service provider to build local reputations of clients. These reputations are used by the service providers to determine levels of service. Locally developed reputation data are also shared anonymously so that other service providers can use global reputations at any time to adjust the local reputations about their clients whenever necessary.

Through the process of behaviour profiling and developing client reputation, service providers will have prior knowledge of each client's trustworthiness.

Clients without existing behaviour profiles may be required to prove their worth by developing acceptable behaviour profiles before they are eligible for higher levels of service. On the other hand, clients already catered for with high levels of service are given the benefit of doubt (depending on implemented policy) for occasional unacceptable behaviour, which lowers the probability of false positives.

We demonstrate the suitability of our framework through some example policies that we propose and justify in this thesis. We also evaluate the effects of relevant attacks on the framework.

1.3 Thesis roadmap

Here, we present a brief summary of the thesis.

Chapter 2 – Background: In this chapter, we summarise background work and research challenges in *intrusion detection systems* and *anti-spam systems* as examples of measures against unsolicited messaging and systems that employ client *behavioural history*. We present a survey of approaches through example systems. We also present brief discussions in *identity management* and means of *policy specification*. The chapter ends with the discussion of the research question that we plan to tackle in this thesis.

Chapter 3 – Reputation from behaviour profile: Building on our discussion of the related work and our research question, we summarise our research contributions. The rest of this chapter describes the theoretical model of the open-ended framework to develop, share and use reputations from client behaviour data. We formalise the representations of reputation and server confidence in the context of our work; presenting the functional representation of our model. Thereafter, we describe the different functional stages of the framework with some proposed example policies. We end the chapter with a discussion of a number of adversaries to the model, some of which we later simulate in chapter 5.

Chapter 4 – Experimental setup: In this chapter, we discuss the experimental implementation of our theoretical framework that we described in

chapter 3. We also present a simple discrete event simulator, which we have built to perform high-level simulation of the framework.

Chapter 5 – Evaluation: In this chapter, we utilise the experimental setup described in the previous chapter to simulate our proposed framework. Our simulations illustrate the *reputation responses* to changing client behaviour. We use synthetic data to create experimental scenarios, with client behaviour pertaining to various *actor classes*. From our simulations, we observe the effects of global reputation through various *interpretation policies*. We also discuss and analyse the effects of relevant attacks on the framework.

Chapter 6 – Conclusion and Future work: In the final chapter, we summarise this thesis by re-iterating a summary of our contributions. We conclude with a discourse of future work.

1.4 Published work

Parts of this thesis are published in or are related to:

1. **A. Basu**, I. Wakeman and D. Chalmers. *A Framework for Developing and Sharing Client Reputations*: Submitted to IFIPTM 2010, Morioka, Japan, 2010.

2. **A. Basu**, I. Wakeman, D. Chalmers, and J. Robinson. *A Behavioural Model for Client Reputation*: Proceedings of Trust in Mobile Environments (workshop in IFIPTM), Trondheim, Norway, 2008.

3. J. Robinson, I. Wakeman, D. Chalmers and **A. Basu**. *The North Laine Shopping Guide: A Case Study in Modelling Trust in Applications*: Proceedings of Joint iTrust and PST Conference on Privacy, Trust Management and Security (IFIPTM), Trondheim, Norway. 2008.

4. **A. Basu**, I. Wakeman, and D. Chalmers. *A Behavioural Model for Consumer Reputation*: Poster in the Proceedings of the International Workshop on Self-Organizing Systems, The Lake District, UK, 2007.

5. S. Naicken, B. Livingston, **A. Basu**, S. Rodhetbhai, I. Wakeman, and D. Chalmers. *The State of Peer-to-Peer Simulators and Simulations*: ACM Computer Communications Review, vol. 37(2), pp. 95–98, 2007.

6. S. Naicken, **A. Basu**, B. Livingston, S. Rodhetbhai, and I. Wakeman. *Towards Yet Another Peer-to-Peer Simulator*: Proceedings of The Fourth International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs), Ilkley, UK, 2006.

7. S. Naicken, **A. Basu**, B. Livingston, and S. Rodhetbhai. *A Survey of Peer-to-Peer Network Simulators*: Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK, 2006.

2 Background

If you shut the door to all errors, truth will be shut out.
Rabindranath Tagore
Indian poet, artist & philosopher (1861 - 1941)

2.1 Overview

The any-to-any best-effort service offered by the open architecture of the Internet has been the key behind its success. It has facilitated an enormous growth in "people-to-people" traffic through services such as electronic mail, instant messaging, voice-over-IP, web-based social networks and so on. However, this very openness of the Internet has also invited trouble. The Internet is vulnerable to various types of attacks from malicious users. Various "people-to-people" services are susceptible to abuse, e.g. unsolicited messages, of which email spam is an example. Another common attack on services provided through the Internet is the denial-of-service (DoS) through which an attacker exhausts resources at the victim's (i.e. service provider) end, thereby leaving the service provider unable to respond to genuine service requests. A distributed denial-of-service (DDoS) attack originates from a number of nodes on a network with the same common malicious intent to make a service provider unable to provide service. In many cases, distributed attacks are conducted by an attacker hijacking the identities of otherwise harmless nodes on the network.

In the previous chapter, we have introduced a broad classification of various methods proposed and in use to forestall the attacks on network services. With the evolution and growth of services over the Internet, a number of recent research efforts are focused on the use of behavioural history and reputation of clients to distinguish between genuine and malicious clients. In this chapter, we present existing work in a number of different research areas, which lead us to the vision of a generalised mechanism for precluding attacks on network services.

In this chapter, we discuss the following concepts and present existing work in the relevant areas:

Identity management

To prevent attacks on network services, it is necessary to know the identity of the clients that use such services. We discuss key systems which provide representative notions of identity.

Specification of policies

Agreements between service providers and their clients, such as the Terms of Service (ToS), the Service Level Agreement (SLA), the Acceptable Use Policy (AUP) are legal offline agreements which do not immediately translate into policies that can be interpreted by machines. As more parts of the interaction between the user and the service provider become controllable, there will be increasing direct representations of those agreements within the network. We discuss some existing efforts in this direction.

Measures against unsolicited messages

Unsolicited messages are a type of abuse of the open-ness of the Internet. Taking email spam as an example of unsolicited messaging, we present key existing approaches in anti-spam systems.

Intrusion detection

Intrusion detection is a long-known method of distinguishing malicious behaviour from acceptable behaviour through the analysis of audit trails. We present the various detection techniques and open research challenges in intrusion detection systems.

Behavioural history and reputation

Finally, we present exemplary approaches in the existing research on the use of behavioural history and client reputation. These are our key motivational work and are closely related to this thesis.

Following our survey of the research challenges in the aforementioned areas and a selection of existing research work, we close this chapter with a discussion of our research question and research objectives.

2.2 Identity management

Services and resources made available over computer networks often need to know who the users are, in order to control their access to those services and resources. Identity management comprises the initial stage of allocating identifiers and credentials to users, and the later stage of authenticating those users and controlling their access based on their already known identities. The representation of a user entity in a specific application domain is its identity. Various types of identities include usernames, IP addresses, public keys and so on.

According to Jøsang & Pope 2005, one of the common identity management models is the *isolated user identity model* where the service provider provides the user with its identity and it controls the entire identity namespace. Examples are: a username-password combination generated by and works specifically for a single website; and a "private" class-C network IP address issued to a host by a router with a Network Address Translator (NAT) (Egevang & Francis, 1994; Tsirtsis & Srisuresh, 2000). In this type of identity management, the identifier namespace is isolated and usable only in the context of the specific service provider. Thus, a class-C network IP address `192.168.1.2` behind a NAT is a namespace owned by the NAT and may only be accessible within a "private" network `192.168.1.0/24` (Classless Inter-Domain Routing or CIDR notation, Fuller *et al.* 1993).

The other user identity management model is the *federated user identity model*, in which the same user identifier is recognised by more than one service provider, and belongs to a global namespace. Although the namespace could

still not be universally unique, it can be unique across a number of organisations. The federated model results in a single virtual identity domain for the user, whereby a set of standards and technologies enable a group of service providers to recognise user identifiers within a federated domain. An open-source implementation of this type of identity management model is Shibboleth (Erdos & Cantor, 2002). Other federated identity management systems are Open ID (Recordon & Reed, 2006), a pseudonymous identity proposal, such as Wakeman *et al.* 2007, and so on.

A third set of user identity management types constitute the *centralised user identity model*s. There are various ways centralised models may be implemented. A single authority may act as a user identifier provider for all service providers. An example is a Certificate Authority issued X.509 public key certificate (Housley *et al.*, 1999). A single-sign-on identity domain is very similar to federated identity management because the user identifier necessary for one service provider is also used by other service providers. One example is Kerberos (Kohl & Neuman, 1993) where the Kerberos Authentication Server acts as a centralised identifier and credential provider.

Some identity management schemes use identifiers that are short-lived and re-usable, e.g. dynamically allocated IP addresses, while some other identifiers are long-lived, e.g. a X.509 certificate. In addition, some identifiers are "weak" because of their allocation or verification processes. Those identifiers can be spoofed with little effort, e.g. dynamically allocated IP addresses without extended authentication checks such as 802.1x EAP (Congdon *et al.*, 2003), while some other identifiers are "strong" and cannot be masqueraded, e.g. a Certificate Authority signed X.509 certificate.

2.3 Specification of policies

According to Sloman *et al.* 2001, the main driver behind the research and development of security and management policy based services is to enable dynamic adaptability of behaviour of the system through changes in policies instead of having to change the system itself. The process of refining abstract policies, expressed in business goals, service level agreements or acceptable use policies within or between organisations, gives rise to various challenges.

Furthermore, in large scale systems with a considerably large number of users, it is impractical to specify policies for each user. Instead users are grouped in various ways and policies are applied on these groups. Policies are applied to the users depending on their group memberships. We briefly summarise some exemplary approaches in policy specification, following from the survey in Sloman & Lupu 2002.

In terms of *security policies*, Roles-Based Access Control (RBAC) (Sandhu *et al.*, 1996) allows specification of permissions on users through grouping them into various roles. In essence, it is the "role" of the user rather than the user itself to which a permission is granted. The original RBAC concept faces challenges with modeling inheritance, the complexity of which can be reduced through the use of a capability based system (Sloman & Lupu, 2002). Modifications and enhancements (e.g. Tan 2002) have been suggested to overcome challenges and further the research in the area of access control. Although not a policy specification mechanism as such, RBAC gives the flexibility through which policies for access control are not tied to the system implementation or the owner of each resource; instead they can be derived from organisational policies. IBM's Trust Policy Language (Herzberg *et al.*, 2000) extends RBAC. It describes a public key infrastructure (PKI) certificate based policy language, specified in Extensible Markup Language (XML) (World Wide Web Consortium, 1996), to enable automatic assignment of roles. Logic based solutions for authorisation policy specifications are illustrated in Jajodia *et al.* 1997; Ortalo 1998. There have been Java specifications of security policies from high-level policy descriptions (e.g. Corradi *et al.* 2000; Hashii *et al.* 2000).

In terms of *management policies*, the Policy Definition Language (Lobo *et al.*, 1999) from Lucent Bell Labs is a set of simple `event-condition-action` rules for network management. It can be used to specify complex workflow tasks but it does not support policy composition and re-use of policy specifications. Without the event triggering part, Internet Engineering Task Force (IETF) has defined the Policy Core Information Model (PCIM) (Moore *et al.*, 2001) and its extensions (Moore, 2003). PCIM policies are formed as `condition-event` rules, which support nesting into policy groups. The rules can also have conditions based on time periods and can be either sequential or random in order. Some extensions to PCIM also exist for Quality of Service

(QoS) (Snir *et al.*, 2003). The ISO/IEC reference model for open distributed processing (RM-ODP) (ISO/IEC, 1995), designed to provide a framework for distributed processing specifies five viewpoints. Out of these, the enterprise viewpoint focuses on policies, scope and purpose of a system. There has been work that uses the Unified Modeling Language (UML) (Object Management Group, 1997) to describe the enterprise viewpoint of the RM-ODP, such as Aagedal & Milosevic 1999; Blanc *et al.* 1999; Linington 1999. Ponder (Damianou *et al.*, 2001) is a declarative, object-oriented policy specification language that can be used to specify security policies for access control mechanisms for operating systems, firewalls, databases and so on. It can also be used to specify event triggered `condition-action` rules for management policies of distributed systems.

Apart from security and management policies, technologies such as Web services (Booth *et al.*, 2004) and GRID computing (Foster & Kesselman, 1998) are being increasingly used to develop high-level business applications and services for considerable levels of autonomic operations across organisations. For those services to operate, the contracts between the service providers and their clients must cater for various parameters, such as performance and reliability. Those contracts – Service Level Agreements (SLAs) – need to be represented in specialised languages, which enable autonomic negotiations, adaptation of services and their compositions. SLang (Skene *et al.*, 2004) defines a SLA vocabulary for a number of Internet services, e.g. application service provision, Internet service provision and storage service provision. UML is used to model the language, which contains formal semantics of the behaviour of the services and clients in service provision. Some other similar work on specification of Quality of Service (QoS) and SLAs are Web Service Level Agreements (WSLA) (Keller & Ludwig, 2003) and Frølund & Koistinen 1998; Tosic *et al.* 2002. An example of a business process oriented approach for the choice of Service Level Objectives (SLOs) of an SLA is seen in Sauvé *et al.* 2005.

2.4 Measures against unsolicited messages

In this section, we classify various academic and commercial measures taken against the menace created throughout the Internet by unsolicited messages. We focus on email spam, as an example. The existing literature on the subject

of preventing email spam is very large. We use study a small subset of the existing work categorised according to their functionality under the following taxonomy:

Identity forgery protection
: These systems attempt to stop messages originating from forged identities, e.g. botnets through various sender identity verification mechanisms.

Message filtering
: Such systems use Bayesian message content filtering, sender blacklisting, amongst others.

Message restriction
: These systems impose restrictions on sent messages by quota management, introducing unit cost for sending a message, and so on.

Use of trust and reputation networks
: In these systems, messages are collaboratively filtered by means of trust and reputation networks between senders and recipients.

Before we study existing work, we intend to draw the reader's attention to the fact that false positives in email spam are worse than false negatives because they can have a direct or indirect cost to business (Edwards, 2008; Gaudin, 2003). False positives turn emails into unreliable communication means. When behavioural history is not used, any message from a particular sender is considered separately from others. Hence, if the statistical spam classifier generates a false positive, the message will be marked as spam even if it may not be a spam message. Methods of tuning the spam classifiers (in particular, the text-based Bayesian spam filters) are often suggested but tuning takes a considerable amount of time and often needs to be updated when new words appear in the text of messages sent.

2.4.1 Identity forgery protection

One of the early approaches to differentiate between legitimate and malicious email senders is the use of whitelists, which are lists of legitimate sender addresses. However, in the absence of any authentication mechanism most

whitelists are vulnerable as email addresses and some other user identities can be forged. A long-known procedure of stopping identity forgery is using cryptographic digital signatures, e.g. through PGP (Zimmermann, 1995). Lack of a widely deployed public key infrastructure hinders the use of digital signatures for all emails, which could have solved the forgery problem to a significant extent. Another prevention method uses the concept of trusted senders through which a recipient can determine if an email originated from a mail server in the domain of the sender's address. Sender Policy Framework (Wong & Schlitt, 2006) and Sender ID (Lyon & Wong, 2004, 2006), Yahoo! Domain Keys and its applications (Allman *et al.*, 2005a; Libbie & Ludemann, 2006) are recent proposals using the trusted sender approach. Sender Policy Framework (SPF) can create false positives, e.g. when an email is sent from hosts that are either roaming behind a network address translator or not directly connected to the network under the domain that publishes a SPF record. Including roaming IP addresses in the SPF record could give rise to false negatives when spammers connected to the same roaming networks or re-using the same IP addresses can send emails forging valid email addresses. On the other hand, if a domain does not publish a SPF record then email addresses on that domain cannot be verified. Large-scale use of SPF can be hindered by mistakes in or the absence of the domain SPF records. DomainKeys Identified Mail (Allman *et al.*, 2005a, 2007; Crocker, 2009; Delany, 2007; Hansen *et al.*, 2009) uses a domain-based digital signature and its verification to facilitate end-to-end message integrity, but is however not a means for abuse prevention. SMTP authentication (Siemborski & Melnikov, 2007) is another recent extension to SMTP although it does not guarantee the authenticity of the envelope sender. A malicious sender can still masquerade as a legitimate user. When relaying email the recipient email server ought to trust the relaying mail server, which indicates if the message sender was authenticated. If the relaying mail server itself is malicious then the recipient server cannot verify or trust the authenticity of the sender.

2.4.2 Message filtering

Techniques employing statistical filtering and machine learning to classify text as *spam content* have been in use for quite a number of years (Graham, 2003; Sahami *et al.*, 1998). A number of these use Bayesian spam filtering. In this process, the spam filter calculates the probability that a message is spam if it contains a word, given that the filter has learnt the probability of that

word occurring in messages that are manually classified as spam. A "naive" Bayesian classifier combines such probabilities that the message is spam using the hypothesis that the presence of each such word in the message is an event independent of each other. However, for a natural language, this assumption is often wrong. SpamAssassin (The Apache Software Foundation, 2009) is one of the most widely used system that illustrates the machine learning approach. Bayesian filters are attacked in various ways (e.g. Lowd & Meek 2005; Stern *et al.* 2004; Wittel & Wu 2004), such as increasing the number of legitimate words in spam messages, and using images instead of text amongst others. Classification through statistical filtering often generates false positives and reduces the reliability of emails. There are, however, other discriminative statistical content filters (Goodman & Yih, 2006), such as the CRM114 (Yerazunis, 2009).

Solving the problem of false positives requires human intervention at increased costs. Systems such as Cloudmark (Cloudmark, 2009), Distributed Checksum Clearinghouse (Vixie & Schryver, 2000) and Vipul's Razor (Prakash, 2007) using collaborative human-directed classification based on a centralised database of spam reports. Although fairly accurate, human-directed classification systems could be susceptible to attacks from malicious users, especially the Sybil attack (Douceur, 2002). Message filtering also includes the use of blacklisting, such as Real-time Blackhole List (RBL) (Trend Micro, 2009) and SpamCop (Cisco Systems, 2009c). However, blacklisting an SMTP server for a domain may generate false positives for legitimate users of that domain.

2.4.3 Message restriction

Dwork and Naor (Dwork & Naor, 1995; Dwork *et al.*, 2003) proposed using a CPU time or memory consuming computation that the sender must perform to send every message, which makes it computationally difficult to send unsolicited messages at a very fast rate. Microsoft proposed Penny Black (Abadi *et al.* 2003, website: Birrell *et al.* 2009) which is a centralised and trusted ticket server through which clients are allocated tickets based on some kind of "proof of work". Valid tickets allow clients to use services. Two similar approaches are Camram (Johansson & Dawson, 2003) and Hashcash (Back, 2002). However, Laurie & Clayton 2004 have argued that a "proof-of-work" system is unlikely to filter spam. Analysing "proof of work" systems from both an economic and

a security perspective the authors have proved that a significant number of legitimate users will be unable to send emails, which will make email communication unreliable. This argument, however, has been debated by Liu & Camp 2006. Other research efforts, which suggest that email senders should lose money if a recipient identifies a message as spam include Raymond 2004, Bonded Sender (Return Path, 2009), and so on. Spam-I-Am (Balakrishnan & Karger, 2004) discusses spam control using distributed quota management. The distributed quota management (DQM) infrastructure in Spam-I-Am is somewhat similar to digital payment systems – quota allocation, stamps and quota enforcement correspond to digital cash withdrawal, expenditure of digital cash for each message sent; and forgery prevention and double spending detection respectively.

2.4.4 Use of trust and reputation networks

A number of researchers are working on the use of social networks to either reduce spam and to rank importance of emails. Ebel *et al.* 2002 discusses that the study of a topology of an email social network exhibits small-world behaviour. This result is used by Kong *et al.* 2005 to propose a collaborative content-based email filtering system that rejects spam based on their previous classification in social networks. It, however, presumes that people connected in such a social network have the same definition for *spam content*. Mechanisms for exchanging whitelist information using Bloom filters and SHA-1 (NIST, 1995) hashes have been proposed, e.g. LOAF (Ceglowski & Schachter, 2004) and FOAF (Brickley & Miller, 2004). Although whitelist entries are not exposed in cleartext, both are open to straightforward dictionary attacks. Also it is assumed that the sender address is not forged.

Golbeck & Hendler 2004 present a trust scoring algorithm based on social networks. Although PGP (Zimmermann, 1995) was not originally designed against spam, it uses a web-of-trust model for key distribution which relies on friend-of-friend trust relationships as does Reliable Email (Garriss *et al.*, 2006). In this work, the length of the friend-of-friend relation is an interesting open topic of discussion – the longer it is, the lower the reliability. Other collaborative spam detection approaches include Damiani *et al.* 2004 which uses a peer-to-peer network to filter spam; Chung *et al.* 2002 which discusses a

Chord-based distributed spam detection tool, amongst others (Gray & Haahr 2004; Kong *et al.* 2006; Stolfo *et al.* 2003).

2.5 Intrusion detection systems

Historically, acts of intrusion detection started off as non-automated jobs performed by network administrators monitoring user activities for unusual behaviour, through administration consoles. Such non-scalable means for intrusion detection evolved to the use of printed audit logs, which administrators reviewed for evidence of suspicious activities that could suggest intrusions. This process, too, was not scalable as computer networks started to become larger and faster. Audit logs were often too big to be reviewed manually at real time to identify attackers during intrusions. These logs, therefore, served the purpose of post-attack analysis. The concept of modern intrusion detection system (IDS) is acknowledged to James P. Anderson who first documented the need for automated audit trail reviews in his seminal work (Anderson, 1980). However, due to the computational requirement in analysis of large audit trails, most such automated analyses were performed at times when the user load on the systems were low. This meant that most intrusions were detected after their occurrences.

Since the late 1980s and early 1990s, research has been conducted (e.g. Debar *et al.* 1992; Heberlein *et al.* 1990; Lunt *et al.* 1988; Sebring *et al.* 1988) on the development of real time intrusion detection systems. More recent efforts have concentrated on the deployment of network intrusion detection systems (NIDS) in large and high-performance networks (e.g. Paxson 1999; Sommer 2005). Since the early 1990s, there has been research in the area of distributed intrusion detection systems (DIDS) too, e.g. Gregory *et al.* 1996; Janakiraman *et al.* 2003; Porras & Neumann 1997; Snapp *et al.* 1992. Researchers have also proposed augmenting NIDS with analysis of logs gathered over time (Maier *et al.*, 2008).

2.5.1 Detection techniques

Based on detection techniques, intrusion detection systems can be categorised (Axelsson 2000a; Sommer 2005) into two main types: *anomaly detection* and *signature or misuse detection.*

Anomaly detection This class of IDS uses the assumption that the behaviour of attackers differs from normal behaviour. Such IDS use models of intended behaviour as normal and treat deviations as potential problems. The main advantage of this detection technique is that previously unknown attacks can be detected as they violate what is known to the system as "normal". However, since this technique suspects any deviation from normal behaviour irrespective of cases of actual intrusion, it generates high false positive rates.

Anomaly detection could either be *self-learned* or *programmed* (Axelsson, 2000a). *Self-learning* anomaly detection systems could be further classified into *non-time series* detectors and *time series* ones. The former type builds the model of normal behaviour through the collection of statistical information or through the construction of a rule base; in both cases without considering time series behaviour. The latter type, on the other hand, employs techniques such as hidden Markov models (HMM) or artificial neural networks (ANN) on time series behaviour data. *Programmed* anomaly detection systems require explicit specification of what is considered as abnormal behaviour. This is done through statistical modeling, rule base construction, state series modeling, and so on.

Signature or misuse detection This class of IDS works with the knowledge of preset models – signatures – of intrusive/misuse behaviour. Those signatures are programmed and continually updated so as to keep the knowledge of the IDS up-to-date. No matter what constitutes normal behaviour for a system, signature detection systems flag intrusions based on known patterns. Good models of known intrusion patterns make such IDS more accurate than anomaly detection systems by lowering the false alarm rate. However, this also means that the signature detection systems are unable to detect previously unknown intrusions.

Two other detection techniques which are variations of the aforementioned ones are: *specification-based* and *combined* (Axelsson, 2000b).

Specification-based detection This class of IDS works with explicit specification of allowed behaviour. This is essentially the inverse of signature based detection. Theoretically, it is expected to be as powerful as signature based

detection but it is often less feasible (and not scalable) to explicitly specify the entire range of allowed behaviour.

Combined detection This class of IDS combines the knowledge of intrusion behaviour with the model for normal behaviour. Those systems automatically learn the nature of intrusion alongside normal behaviour through examples of flagged intrusive behaviour intermittently present with normal behaviour. Compound detector systems theoretically have higher correctness in identifying suspicious behaviour since they have knowledge of both normal and intrusive behaviour. This research area is still immature. One such intrusion detection system developed is Lee *et al.* 1999.

2.5.2 Fundamental challenges of (network) intrusion detection

Various literature (e.g. Axelsson 2000a; Kemmerer & Vigna 2002; Sommer 2005) outline the fundamental issues and challenges that intrusion detection systems and network intrusion detection systems face.

Effectiveness and evaluation One of the most important open issues in IDS and NIDS is *effectiveness*. Ideally, efficiency nearing 100% detection rate with minimal false positives is desirable. However, the reality is far from that. Often, the false positive rate (i.e. the ratio of the number of false positives to the number of true positives) is very high. (Axelsson, 2000b) demonstrates that the false alarm rate (i.e. the false positive rate) is a crucial factor in limiting the performance of an intrusion detection system. The author shows that this is due to the importance of the base-rate fallacy phenomenon that a nearly unattainably low false alarm rate is required to achieve relatively high values of Bayesian detection rate (i.e. that an alarm indicates an intrusion). The paper computes the Bayesian detection rate using a realistic true positive rate and an unusually low false negative rate, and shows that the Bayesian detection rate could be as low as 58%, which means nearly half (i.e. 42%) of the alarms are false! Sommer 2005 also discusses that the parameters required for such calculations (e.g. false positive rate and false negative rate) cannot be precisely measured in real world IDS. Further to efficiency itself, it is also very difficult to objectively measure efficiency, quality of alerts, etc. Intrusion detection systems present evaluation difficulties due to the choice of input data.

Performance With the increase in deployment of high-speed networks, the audit logs generated for network intrusion detection systems are also on the increase. Large streams of network traffic data become increasingly difficult to analyse in real-time by any single intrusion detector. To reduce the computational overhead on one intrusion detector, the streams may be split across multiple detectors (although splitting the streams itself poses semantic challenges). Alternatively, the intrusion detectors may be deployed close to the hosts that they ought to protect. This helps leveraging the benefit of natural partitioning of traffic but such distributed deployment is often dependent on type of attack and network topology amongst other factors (Kemmerer & Vigna, 2002).

Another important challenge with stateful NIDS is the expiration of states. In order to stop state data from accumulating and filling up available memory over time, state data needs to be regularly purged. It is hard to decide exactly when state data should be expired. As a result, NIDS tend to depend on a number of heuristics for state expiration (Lee *et al.*, 2002; Levchenko *et al.*, 2004; Paxson, 1999).

Security and user privacy IDS can be the obvious targets of attackers since in the absence of a working detection system no intrusion alerts will be raised. Sommer 2005 classifies attacks on network intrusion detection systems into two categories. One is an *evasion attack* in which a NIDS is misled by the attacker to interpret the semantics of a network stream differently from its involved entities thus making itself unreliable in detecting any hidden intrusion attacks. The other is a *denial-of-service attack* in which the attacker exploits a vulnerability in the NIDS or exhausts the available computational resources (amongst other methods) to take over or crash the detection system.

Intrusion detection systems, as a means of surveillance, record all possible user activity. This raises ethical and legal concerns about user privacy. There has been research to analyse and develop privacy-honouring data collection mechanisms for intrusion detection systems (Pang & Paxson, 2003; Porras & Shmatikov, 2006, 2004).

We have, so far, discussed brief backgrounds on *identity management*, *specification of policies*, *measures against email spam* and *intrusion detection*. We shall now focus on the research efforts related to our work.

2.6 Related work on behavioural history and client reputation

In this section, we will investigate some of the academic and commercial efforts that are closely related to our work presented in this thesis. The systems that we review here are either independent of the applications (e.g. email, web server, etc.) that benefit from the use of behavioural history and client reputation, or they are catered for more than one application.

2.6.1 Allman's work on behavioural history

Robin Sommer in his thesis (Sommer 2005) on network intrusion detection in high-performance network environments describes (in §7.2) that coordinating a distributed detection setup presents challenges; both technical and non-technical (e.g. political and social). He cites Allman's work as a step in the direction of large-scale sharing of past behavioural patterns of network hosts.

Allman, Blanton and Paxon (Allman *et al.* 2005b) state that the information available to a service provider about a service requester are limited in scope and often difficult to obtain for the purpose of stopping unwanted traffic in real time. To overcome these limitations, the paper presents an architecture of large scale sharing of behavioural history of network actors, such as hosts or email addresses in an effort to inform policy decisions about how to treat future interactions.

Cryptographically signed local observations on behaviour of actors are submitted to a behavioural history database by *reporters*. The behavioural history database is maintained on a Distributed Hash Table. The evidence of reported observations rather than confirmations of the actual observations are also submitted by *witnesses*. This behaviour history database can be queried by service providers to lookup behaviour reports on clients, which can be used to make policy decisions regarding interactions with those clients. Although left for

future work, the paper introduces the concept of trust on behaviour information available from the database. Locally developed *reputation* of a reporter is used to determine if behaviour reports from that reporter will be trusted. Further to the behaviour reports and the witness reports, the database also stores annotations signifying whether the behaviour reports have been used or not. The paper mentions that this annotation process, which they call the *signatory mechanism*, plays a part in determining the reputation of the reporters.

The authors discuss a variety of issues associated with the collection, sharing and use of behaviour reports, such as cheating, revocation of reports, etc. One of the issues with linking loosely-bound cryptographic keys to identities of reporters (see §5 in the paper) is that it poses a privacy challenge for the client. The paper does not present any evaluation of the architecture. It is, however, a very important step on the use of behavioural history of network hosts. The paper, according to the authors, focusses "on the big picture, but to bring such a system to fruition will require community efforts on a number of fronts", which include: developing local reputation computation algorithms; obtaining acceptable performance and resilience in reputation lookup; and devising witness message digests.

2.6.2 Wei and Mirkovic's work on client reputation

Similar to Allman's work on behavioural history, Wei and Mirkovic propose a client reputation system (Wei & Mirkovic 2007) that can aid service providers in deciding to accept or decline interaction with a given client. The authors suggest that such a system could significantly aid defenses against major security threats (e.g. intrusions, distributed denial-of-service attacks) with a prior knowledge of a client's trustworthiness, provided by reputations. Client reputations could also be used for traffic prioritisation during congested events. The work presented in this paper largely falls in the class of network intrusion detection systems.

This work provides a thorough overview of the differences between provider and client reputations. The challenges unique to client reputation systems are surveyed. Two architectures for the collection of client behaviour are discussed: a *reporter* model and a *monitor* model. Finally, it is shown how a combination of both these models can help overcome major threats to reputation validity.

The authors propose a pseudonym based anonymised reporting of bad behaviour of clients. Only bad behaviour is reported through something called a bad info-item. Clients are deemed to be good clients if they have never been the object of a bad info-item (see definitions in § 3 of the paper). Clients that are not objects of any bad info-item within a certain recent time are known as *current good clients*, while a client that is an object of at least one bad info item within a certain recent time are *current bad clients*.

In the reporter model, a server submits a report after an interaction with a client. Only bad reports are submitted if the interaction was malicious. Such behaviour reports are submitted to reputation centres, which can be managed as a centralised service or a distributed service. One of the advantages of this approach is, as the authors claim, that the reputation system can be designed in a fully decentralized manner using approaches from peer-to-peer reputation systems. Thus, no participating server needs to be trusted by all others. Another advantage is that a server has a very reliable knowledge of its full interaction with a client.

In the monitor model, monitoring nodes are present on routers that collect observations about the clients from the traffic they relay. Each monitor uses the traffic it relays to summarise and build the client's profile. Due to the distributed nature of monitor nodes, different client profiles may be built of the same client by different monitors. These are periodically synchronised. Monitors can report, a suspicious but not bad, info-item if anomalous behaviour is detected for a particular client.

The paper shows that both the reporter and the monitor model suffer from serious pitfalls arising from the reporter lying, false positives and spoofing. The authors combine the two models to address such challenges. The problem of lying reporters is handled by introducing a concept of a verifier, which can be either be a monitor node that places a secret mark on each packet in the stream, or be a trusted third-party, which knows the monitor's secret. The mark is bound to the packet to prevent it from stealing and re-use. Each report is verified for accuracy using the knowledge of the secret mark. However, this verification process only makes sure that the header values in a sample of packets fits the context field of a report but it does not and cannot verify if

the traffic was malicious. The problem of spoofing is handled by proposing that monitors perform core-filtering approaches (e.g. Duan *et al.* 2006; Perrif *et al.* 2003).

Finally, behaviour reports are aggregated into a reputation score either by the reputation system or the reputation user. An aggregation method based on the combined reporter-monitor model and assuming that only bad and suspicious reports are generated. The method, according to the authors, "yields positive reputations for good clients, large negative reputations for bad clients whose maliciousness and identity can be verified, and small negative reputations for bad clients whose bad reports contain no-spoofing flag reset".

This work presents neither any analytical results nor any results from simulation, emulation or practical deployment. In the absence of such results, it is difficult to assess how the system will perform in the real world. We note that in this paper the nature of behaviour is observed as either good or bad (with some levels of badness) thus leaving not enough room for variable degrees of badness or goodness that can be a result of interpretation of acceptable use policies. In addition, reporters and monitors report behavioural history to reputation centres which, as well as reputation users, can then form reputations of particular clients. The calculation of reputation from a behavioural history is specified in the model, which essentially means that the model specifies a reputation generation policy. However, such policies could differ between reputation users and therefore a reputation is more likely to be open to interpretation.

2.6.2.1 Wei and Mirkovic's work on host profiling and clustering

Related to the paper on the client reputation system, another research paper from the same authors is Wei *et al.* 2006. It describes means of profiling the behaviour of Internet hosts and then applying clustering techniques to categorize those hosts. This work is meant to be the first step in the research on developing an Internet-wide host reputation system, which the authors call the *Internet Credit Report*.

The paper is motivated by the hypothesis that users (representing hosts) over the Internet exhibit slow-changing patterns of their behaviours over long

periods of time. Authors argue that if this hypothesis holds then it could be utilised to detect Internet-wide behaviour anomalies. To achieve this, the authors set out to identify behaviour profiles of hosts. The paper described extraction of a number of features of a host, as observed from network traffic sent and received by the host, to characterise the host's behaviour and to keep it up-to-date when the host's behaviour patterns show small variations over time. From such characterisations of host behaviour, the paper presents a scheme to generate clusters which represent hosts with similar behaviour.

The paper presents experiments on a number of datasets. It claims that the results of those experiments validate their hypothesis and that a large majority of the sampled Internet hosts can be categorised into large behaviour clusters, which represent the routine usage of most Internet hosts. The paper also presents the application of clustering in anomaly detection from observing traces under a worm attack. Even though the paper cites reasons, the experiment on anomaly detection demonstrates a relatively noticeable false positive count. However, real time analysis of Internet-scale backbone traces is very difficult. The authors leave their investigations to use sampled backbone traces for future work.

2.6.3 Natu and Mirkovic's work on capabilities using client reputation

The use of tokens representing single-use capabilities as a solution to Denial-of-Service attacks on the Internet was first proposed by Anderson *et al.* 2004. There are a number of research papers related to the subject of capabilities (e.g. Yaar *et al.* 2004; Yang *et al.* 2005). We observe that Natu & Mirkovic 2007 extend the concepts of capabilities by adding reputation-based granting of capabilities tickets. The authors identify and provide solutions to three key deficiencies in prior capabilities research.

First, they tackle the issue of distinguishing between legitimate and malicious clients to which capabilities are granted. Means of using the long-term knowledge of clients' behaviour profiles in the capabilities (or tickets) granting mechanism is proposed. Degrees of trust are associated with the clients through assignment of *credits* and *penalties* based on their behaviour profiles.

The second identified deficiency is the use of binary capabilities. The possession of valid capability tickets grants full access to the DoS victim, which enables malicious clients to first obtain tickets and then co-ordinate attacks. The authors propose to solve this issue through fine-grained capabilities that imply priorities depending on clients' long-term behaviour profiles. This allows suspicious clients to be penalised while providing high quality service to clients that consistently behave well.

Finally, the authors argue the difficulty posed by route-dependent capabilities generation. If routers in a path are allowed to participate in ticket generation then such route-dependent capability tickets will make legitimate clients suffer in case of change of route or multi-path routing. The proposed ticket generation system takes out upstream routers, unless specifically authorised by the DoS attack victim, from the process of ticket generation. Thus, the tickets are generated only by the traffic destinations. It is argued that the operational cost of DoS defense is also reduced through this process.

As solutions to the first two of the aforementioned deficiencies, the paper postulates mathematical schemes for the calculation of credits, penalities and client class from behavioural profiles where each client class is assigned a share of the resource. We note that such schemes for calculation of credits, penalties and client class essentially specify policies through which servers should relate behavioural profiles to notions of credits and penalties. IP addresses are used as a means of identity while considerations for IP network address translation (Egevang & Francis, 1994) are left for future work. The paper presents results of experiments on the Emulab testbed (Flux Group, 2000) of a relatively small network topology and synthetically generated traffic data. The paper claims that the results demonstrate that their proposed defense handles sophisticated attacks, in which legitimate traffic is guaranteed consistent access to resources while malicious attack traffic is penalised.

2.6.4 Other commercial work

There are various commercial works which take into account the behavioural history of Internet hosts and users. Although detailed technical information about their models are not available, we present information on some of the well-known commercial products and services related to behavioural history.

2.6.4.1 Arbor Peakflow X

Arbor Peakflow X (Arbor Networks, 2009) is a solution built for deployment in enterprise networks. It provides a visualisation and management of the entire enterprise-wide network allowing the administrators to gain a clear understanding of multi-protocol label switching (MPLS) virtual private network (VPN) traffic. This visibility enables the administrator to respond to attacks as they happen when any endangered routers can be quarantined. Peakflow X also uses IP flow statistics from network devices and raw packet data to create the baseline definition of quantified normal network behaviour. Thereafter, *network behaviour analysis* is done in real-time by comparing traffic with baseline behaviour information to detect anomalies. This enables detection of attacks for which signatures do not yet exist. The solution also provides what they call *Application Intelligence* through which behaviour analysis is done on network and application traffic. The administrator is presented with an integrated solution to optimise network and application performance. The network behaviour analysis in Peakflow X is complemented by *Active Threat Feed*, which provides contextualised information on attacks from a global and local perspective. Peakflow X also allows *real-time risk assessment* of potential threats in the network. IP based identities are flexibly tracked so that information on both their current use and historic use is visible to the network administrators. Network analysis data is stored in storage area networks (SAN), which can be used for forensic analysis and compliance monitoring. The Peakflow X solution provides extensive reporting features, which provide administrators with graphical views of complex statistical data.

2.6.4.2 Riverbed Cascade

Riverbed Cascade (Riverbed, 2009) provides means of application performance management through the analysis of interactions of users, applications, systems, and network interfaces. It is essentially a network analysis and reporting system. Similar to Arbor Peakflow X, Riverbed Cascade uses network flow statistics along with higher level application and user identification to create a baseline, which is then compared with behavioural analysis to signal performance or security issues. Cascade also claims to present the administrators with easy-to-read reports and views of network analysis.

2.6.4.3 DShield

DShield (DShield, 2009) is a free distributed intrusion detection system, sponsored by the SANS Institute (The SANS Institute, 2009) and supported by the community. DShield collects and processes outputs of simple packet filters about malicious activity across the Internet. Any user can report to DShield. The database of such collected reports can be used to discover and confirm widespread attacks on the Internet, and also to prepare better firewall rules.

2.6.5 Senderbase

Senderbase (Cisco Systems, 2009b) is the world's largest Web and email traffic monitoring service. It collects data from more than 25% of the world's Web and email traffic from a diverse group of more than 100,000 contributor organisations. It provides real-time view of email and Web security threats around the world. Such large volume of data provides a statistically significant sample size. Senderbase examines more than 90 different parameters for email and 20 different parameters for Web traffic. Senderbase Reputation Score ranks IP addresses or domains in a numeric range but is often grouped together as either "good", "neutral" or "poor". Senderbase Reputation Score is used by Cisco's IronPort Reputation Filter (Cisco Systems, 2009a) technology.

2.7 Research question and objectives

The related academic work on behavioural history and client reputation suggests that there is as increasing rationale for using and sharing client behavioural information in order to protect network services from attacks. We have also seen that this is a relatively new research area and there are several open research issues.

We set out to explore the research question: *can a generalised, high-level, privacy-preserving, policy-independent, context-aware framework for building local and sharing global reputations based on behavioural history of network client identities be useful to servers in determining levels of service to those clients in current and future service offerings?*

Let us now break down our research question and analyse the various objectives we are aiming to achieve.

Policy-independent and high-level framework: One of the fundamental objectives is to develop an open framework that will integrate with existing network protocols and applications. It is essential that policies can be applied at various stages of the framework without having the framework to undergo design changes. Even though we propose example policies in this thesis to evaluate our framework, the purpose of this work is not to specify policies. The framework is expected to enable behaviour profiling of clients not just at the packet level. Unlike related work of behaviour profiling in network intrusion detection through the inspection of packets, our framework is expected to "observe" client behaviour from a variety of sources. It is often easier to interpret application level semantics to determine the nature of a certain client behaviour than to deduce similar conclusions by observing packet traces.

Fine-grained, context-aware reputation corresponding to behaviour: Existing research in the area of behaviour profiling of clients tend to develop binary and rather discrete levels of client reputation. This restricts the flexibility of associating service levels with reputations. Our framework is intended to offer a continuous range between -1 and $+1$ for client reputation values. Service levels can then be associated with (and varied within) certain bands in this range. This allows fine-grained control of service levels with respect to reputation. The level of detail in recording reputation is also increased by the use of application contexts. This means that the reputations developed for clients for a certain type of application do not impact on reputations developed for a different type of application.

Levels and quality of service: In our framework, service levels are associated with reputation values. It is important to note that the association does need to take other factors into consideration, e.g. client class or identity. For example, a client belonging to a "premium" class of clients will have a different set of service levels than a client belonging to a "standard" class. However, given the relation between service levels and reputation, we argue that our reputation framework could also act as means of assuring quality of service.

In situations of service or resource overload, servers may choose to prioritise service to clients with higher reputations.

Privacy-preserving global sharing of reputation: Having formed local reputations of clients, it is useful to share this information globally such that other servers who have not interacted with certain clients can form initial opinions. Global reputations can also be used to form on-going opinions in recurring service consumptions. The framework is expected to provide a privacy-preserving way of sharing global reputations, such that a server querying the global reputation of a client cannot find out which other servers it has interacted with. Global sharing of reputation instead of sharing behavioural history aids preserving the privacy of client's local behaviour.

Similarity of opinions: As local reputations are shared globally, some policies for interpreting global reputations depend on a kind of "trust" between the user (i.e. querying server) and the reporters (i.e. other servers) of the shared data. We investigate how automated statistical measures of similarity provide useful measurements of trust (or the lack of it).

Combining the aforementioned research objectives, we introduce (in chapter 3) a formal description of our framework. We describe our framework in terms of the various components that it is comprised of, and the policy functions, examples of some of which we propose. The policy functions keep the design of our framework flexible and open-ended, allowing any relevant policy to be implemented depending on the needs of the application and the policy requirements.

2.8 Summary

The open architecture of the Internet has been the key driver behind its success and growth. However, this open-ness has also made services on the Internet vulnerable to various forms of abuse. In the previous chapter, we have categorised a number of attacks and also their defences. In this chapter, we organise the existing literature in a number of areas.

First, as background to this thesis, we have introduced some existing approaches in identity management infrastructures, which is a necessity to record

any behavioural history or to develop reputation. Organisational policies, which can help in behaviour analysis, are often abstract and offline. These policies cannot be understood by machines and automatically negotiated. Therefore, following our introduction to identity infrastructures, we have discussed a subset of the various approaches in policy specification.

Having classified (in the previous chapter) unsolicited messages as one of the types of abuse of services on the Internet, we have presented a selection of existing approaches to measures against unsolicited messages, taking email spam as our example. Since the existing work, both academic and commercial, in the area of anti-spam systems is very large, we have presented a selection of those approaches under the following taxonomy:

- Identity forgery protection
- Message filtering
- Message restriction
- Use of trust and reputation networks

Following our discussions on anti-spam, we have introduced a long-known area of research against (network) attacks: intrusion detection. We have discussed the various approaches in intrusion detection and present some of the open research challenges. We have, then, surveyed some of the key motivational work on behavioural history and client reputation. In this survey, we have also presented some existing commercial approaches. The related academic work on behavioural history and client reputation suggests that there is as increasing rationale for using and sharing client behavioural information in order to protect network services from attacks. We have seen that this is a relatively new research area and there are several open research issues.

We have explored the research question: *can a generalised, high-level, privacy-preserving, policy-independent, context-aware framework for building local and sharing global reputations based on behavioural history of network client identities be useful to servers in determining levels of service to those clients in current and future service offerings?* Given this research question, we have discussed the various research objectives and areas we are aiming to tackle in the remainder of the thesis:

- Policy-independent and high-level framework

- Fine-grained, context-aware reputation corresponding to behaviour
- Levels and quality of service
- Privacy-preserving global sharing of reputation
- Similarity of opinions of client reputations between servers

Taking these objectives into account, we propose a framework for developing and sharing client reputations in the following chapter.

3 Reputation from behaviour profile

A life spent making mistakes is not only more honorable, but more useful than a life spent doing nothing.
George Bernard Shaw
Irish dramatist & socialist (1856 - 1950)

3.1 Proposition of a reputation framework

In the previous chapter, we have seen that Allman *et al.* 2005b laid out the architectural overview of a system for storing and retrieving activity reports of network "actors", which are clients requesting services. We have also seen that Wei & Mirkovic 2007 built on that idea to discuss an Internet-scale client reputation system to thwart malicious clients.

Instead of devising a new mechanism of defence against attacks on each type of network service, we ask in § 2.7 if a generalised framework, with configurable policies, can be useful to reduce abuse and unsolicited communication originating from malicious clients. In this chapter, we propose this generalised framework and describe it in detail along with some example policies.

3.1.1 Research contributions

The main contribution of this thesis is a novel *high-level, privacy-preserving, policy-independent, context-aware framework* to which policies can be fitted at

various functional stages. The framework is intended to augment existing network protocols and applications. Servers are expected to utilise this framework to develop and share reputations of the clients they provide service to. The reputations developed and shared globally are sensitive to the context of specific applications or services, thus allowing flexibility to form application-specific (i.e. context-aware) opinions about the clients. Another contribution of this thesis is a notion of "trust" between servers, which forms the basis of trustworthiness of globally shared client reputations.

3.2 Definition of frequently used terminology

Before we set out to describing our framework, we present a list of terminology that we shall be using for the remainder of this thesis.

Client A software program, possibly controlled by a human actor, that consumes services provided by servers over a network.

Server
A software program, which may or may not be managed by a human actor, that provides services to clients over a network.
Alternative terms: service provider, provider

Global Reputation Analyser (GRA)
A distributed software system hosted on trusted nodes over a network that is responsible for storing client reputations reported by servers and for responding to client reputation queries performed by servers.

Identity infrastructure
The mechanism that dictates the identification of "actors" over a network. For example, clients can be identified by IP addresses within a specific address space; they could also be identified by their cryptographic public keys. In these cases, the IP address and the public key, respectively, form the basis of the identity infrastructure.
Alternative terms: Identity management system, identity mechanism

Context
The type of application or service that is the circumstantial basis for forming client reputations. Different reputations can be developed for

the same client under different contexts.
Alternative terms: application context

Behavioural history (of a client)

A history of quantised behaviour of a client as recorded by a server. Client behaviour is observed through a variety of monitoring mechanisms, and quantised in relation to application policies (e.g. terms of use). Observations of behaviour from multiple monitoring sources can be grouped into one context through behaviour analysis. For example, the number of simultaneously opened TCP connections to an SMTP server and the Bayesian spam score of an email message could both be used in the context "email".
Alternative terms: Behaviour profile

"Good" and "bad" behaviour

The types of behaviour associated with a client which can either be the result of behaviour analysis derived from observations from various monitoring mechanisms, or it can be the result of the accumulation of the results of multiple behaviour analyses. "Good" or "bad" does not indicate a binary decision. Instead, there can be different degrees of "goodness" and of "badness" corresponding to various quantised values of behaviour. However, "good" behaviour will signify positive quantised behaviour values while "bad" behaviour will signify negative values.

Reputation (of a client)

A context-aware quantitative perception that a server forms of a client, expressed in the continuous range $[-1 \quad 1]$ (see § 3.4.1), based on the cumulative behaviour of that client which is recorded by the server over time. This perception could relate to the *risk* of providing a service.

Reputation-response policy

A policy specifying how reputation is developed given a particular behaviour.

Confidence (between two servers)

A context-aware quantitative measure of belief between two servers, expressed in the continuous range $[-1 \quad 1]$ (see § 3.4.1), which is used

to evaluate the extent by which one server may believe another regarding an opinion of a particular client for a particular application context.

Authorisation token

A certificate of permission to query and report reputations issued to a server by a client that requests service from it. This token is the supportive evidence of interaction between the client and the server.

3.3 A note on identity infrastructure

In order to record a sustainable behavioural history of network clients, the notion of identities is necessary. A brief study of identities is mentioned in § 2.2. Unless otherwise stated, we assume the use of Public Key Infrastructure (PKI) as the identity framework throughout the rest of this thesis. Using PKI, all network entities are identifiable by their public keys.

It is, however, worth noting that our framework is not dependent on a strong long-lived client identity such as PKI. Our framework can cater for a short-lived re-usable semi-static client identity, e.g. IPv4 address, so long as reputations associated with such short-lived identities are re-calculated when the identities are re-used. IPv4 addresses are semi-static, attached to Media Access Control (MAC) addresses (IEEEProject802, 1986), or are bound by Dynamic Host Configuration Protocol (DHCP) (Droms, 1997) leases. We shall see how DHCP lease periods can be useful to reputation calculation during IP address re-use. Another way of catering for IP-based short-lived identities is using a sub-net of IPs instead of one single IP as a client identity. The identities of the servers are expected to be long-lived.

We discuss, as an extension, the possible use of pseudonymous long-lived identities in the future work section (§ 6.2).

3.4 Overview

In the model of our framework, we identify three types of network entities – *clients*, *servers*, and the *Global Reputation Analyser (GRA)*. Servers provide

services, which clients consume. Servers maintain local observations of behavioural histories of clients that they provide service to and form reputations of those clients. Servers also report the local reputations of their clients to the GRA. The GRA, in turn, can be queried by servers to obtain records of reputations of clients reported by other servers. Global and local reputations are useful to servers for determining levels of service that can be provided to clients. Variations in service levels help control the proliferation of unsolicited network transactions. In addition, variations in service levels can also be used as measures for quality of service.

While the GRA appears to be a single entity to servers it is likely to be implemented, in a real world scenario, as a cluster or a peer-to-peer overlay of closely administered and trusted inter-institutional or intra-institutional nodes. For example, an overlay could be formed out of geographically dispersed servers colocated with core routers of the Internet. Alternatively, an overlay could be formed of certain dedicated nodes in each department of a University. A conceptual diagram of a system with four servers, some common clients and a distributed GRA is illustrated in figure 3.1.

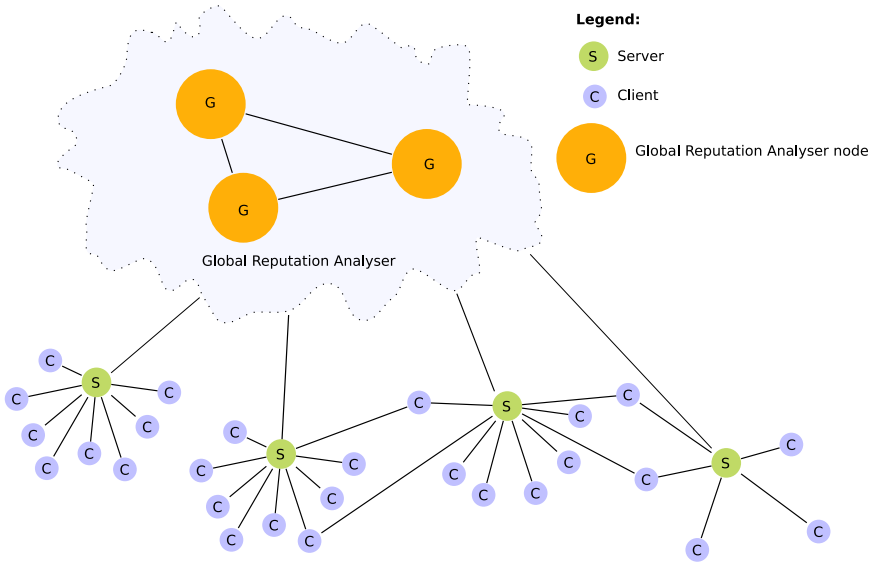

Figure 3.1: Conceptual overview of a system of four servers with some common clients and a Global Network Analyser cloud.

The framework discussed in this chapter is different from similar research (see Allman *et al.* 2005b; Wei & Mirkovic 2007) in that our framework does not share observations of behavioural history of network clients. Instead, it shares reputations of clients. This adds privacy protection to the details of client behaviour as observed by servers. In addition, observations of behavioural history are not limited only to packet traces. Our framework enables the collection of behavioural history from a variety of monitoring systems, including those at the application layer. For example, observations from stateful packet inspection can be taken into account along with those from an application layer monitor such as an email content filter, which could flag potentially malicious behaviour. The framework helps aggregate such observations to enable a server to generate quantised behaviour changes for any given client. Those quantised behaviour changes are then used by the server to develop a local reputation for a given client as service is being provided. This locally developed reputation is also reported by the server to the Global Reputation Analyser. The server can use the locally developed reputation to alter service levels as deemed appropriate within the bounds of the service contract that applies to the class of the client. This means, for example, that given the same reputation, a "premium" client may be eligible for a different service level than a "standard" client. The globally shared reputations can be obtained anonymously by other servers if the same client asks for services from them. The globally shared reputations assist servers that have no prior knowledge of a particular client to form an initial knowledge (i.e. reputation) for that client; or to re-adjust their previous knowledge to a more up-to-date global view of the same client. This can help servers provide high levels of service to known well-behaving clients and to be cautious with clients that have questionable reputations.

The framework consists of a number of functional parts, which are described in § 3.5:

- *Analysis of behaviour:* Client behaviour is observed through various monitoring systems, analysed and quantised through the application of policies.
- *Building of local reputation:* Quantised behaviour obtained through analysis of client behaviour is further developed into context-aware reputation values through application of reputation-response policies.
- *Global reputation reporting:* At the end of provision of services or during

ongoing services, servers report their local reputations of clients to the GRA.

- *Global reputation query:* At the start of service provisions with new clients or during on-going services, servers can query the GRA for reputations of clients to initialise their local reputation values. Alternatively, servers can adjust their local reputation values if necessary with more up-to-date global values. This step also involves the selection of globally available reputation values through variety of policies. One such policy involves the use of confidence between servers.
- *Decision making for service level:* During ongoing services, servers can quickly respond to changes in client reputations by adjusting relevant service levels to prioritise or restrict services depending on client reputations.

Before describing the aforementioned functional parts in greater detail in § 3.5, we introduce the representations of reputation and confidence in our model in § 3.4.1.

3.4.1 Representation of reputation and confidence

Our framework develops a *reputation* of a client in the local perspective of a server as well as in a global perspective, which is shared by many servers. Servers have *confidence* in each other depending on the similarity of their opinions. For both reputation and confidence, we use a continuum symmetric around zero from -1 to $+1$, similar to the trust continuum expressed in Marsh & Briggs 2008 as shown in figure 3.2. In case of reputation, a value of -1 signifies maximum risk to the server for service provision while a value of $+1$ signifies minimum risk. In case of confidence (between servers), a value of -1 signifies maximum disagreement in opinions of clients while a value of $+1$ signifies maximum agreement in opinions.

In the case of reputation, a value of 0 could signify either ignorance or indifference on the part of the server. Ignorance follows from a server interacting with a client for the first time while the server exhibits indifference about a zero reputation value despite the fact that this value may have been the result of a series of good and bad client behaviour from the past. However, a zero reputation resulting from previous activity may deserve to be treated differ-

ently than a zero reputation with no activity. The server can maintain a list of the clients it interacts with in order to determine if a client is new, hence starting at zero reputation or a zero reputation is a result of earlier activities of a known client. In the case of confidence, a value of 0 signifies neither similarity nor dissimilarity in opinion.

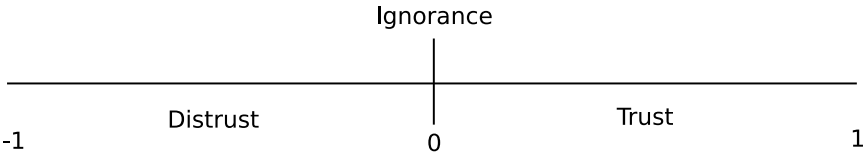

Figure 3.2: Trust continuum: distrust, ignorance and trust

3.4.1.1 Local reputation

We begin our formalism of reputation and confidence by denoting reputation by r. Similar to the concept of *situational trust* in Marsh & Briggs 2008, reputation in our model depends on the application context for which the reputation is being calculated. The application context could be "email" or "Web service", for example. We will re-visit application contexts again later in this chapter. The locally developed reputation for client y observed by server x in the application context α, can be expressed as:

$$r_x(y, \alpha) = F_{lr}(b_x(y, \alpha), t) \tag{3.1}$$

and

$$r_x(y, \alpha) = F_{lt}(t) \tag{3.2}$$

where $b_x(y, \alpha)$ is a quantised cumulative behaviour obtained from behaviour analysis by server x corresponding to the recorded changes in behaviour of client y for the application context α; t signifies time; F_{lr} is a function, defined through policies that govern the server's response to time series data of client behaviour; F_{lt} is the function that governs the server's response with respect to time (e.g. time decay with no activity). F_{lr} and F_{lt} functions constitute the *reputation-response* terminology described in § 3.2.

Although the policies that represent the implementations of function F_{lr} could vary widely between different servers, all servers should be in agreement

regarding the meaning of any particular value of $r_x(y, \alpha)$ even if the policies resulting in that value may be different.

3.4.1.2 Confidence between servers

The *global reputation* of a client is obtained through a reputation lookup query as an anonymised vector of local reputation values for the particular client reported by various servers for a specific application context. Each such global reputation record in the vector contains a reputation value associated with a quantitative measure of how much a (querying) server can be expected to agree with that particular reputation value. This measure is referred to as the *confidence* the querying server has in the server which reported that reputation. Note that the confidence measure does not enforce that the querying server must agree with a particular global reputation value. We denote this confidence by c. We express the confidence a server i has on another server j in the application context α as:

$$c_i(j, \alpha) = F_c(\mathbf{r_j}, \mathbf{r_i}) \tag{3.3}$$

where $\mathbf{r_j} = (r_j(y, \alpha))$ with $y = [0 \dots n]$ is the vector of global reputation records (derived locally according to equation 3.1) that server j has shared, in the application context α, of clients $y = [0 \dots n]$, which are common clients of server i for the same application context. Similarly, $\mathbf{r_i} = (r_i(y, \alpha))$ where $y = [0 \dots n]$. The function F_c is a policy-driven function for determining similarity between the vectors $\mathbf{r_j}$ and $\mathbf{r_i}$. Depending on the similarity function, $c_i(j, \alpha)$ could be the same as $c_j(i, \alpha)$ in which case F_c is symmetric.

Confidence between servers is further discussed in the context of our example policies in § 3.5.4.1.

3.4.1.3 Globally shared reputation

Over time as a client interacts with many servers, the servers share their local reputation values for the client. The global reputation of a particular client y as interpreted by a server x for the application context α can be expressed as:

$$r_x(y, \alpha) = F_g(\mathbf{rc_y}) \tag{3.4}$$

where F_g is a policy driven interpretation function applied on the vector $(\mathbf{rc_y})$ of reputation-confidence tuples (i.e. $\mathbf{rc_y} = (\{r_k(y, \alpha),\ c_x(k, \alpha)\})$ where $k =$

$[0 \dots m]$) in which the client y is expected to have interacted with servers $k = [0 \dots m]$ that also have other common clients which have interacted with the querying server x. Server confidence is defined in equation 3.3.

It is important to note that the Global Reputation Analyser receives more than just the client reputation from the server when the reputation is reported. The information for the application context α reported by server x about client y to the GRA can be represented by a tuple $\gamma_x(y, \alpha)$, given as:

$$\gamma_x(y, \alpha) = \{\alpha, r_x(y, \alpha), \psi_x(y, \alpha), t_{report}\} \tag{3.5}$$

where $\psi_x(y, \alpha)$ is a set of reputation-response policy parameters used by the server x for client y in the application context α, and t_{report} is the timestamp specifying the time when this reputation report tuple has been sent to the GRA. The reputation value in each global reputation record tuple (i.e. $r_k(y, \alpha)$) is subject to a time-based decay function F_{gt} in the GRA, expressed as:

$$r_x(y, \alpha) = F_{gt}(t) \tag{3.6}$$

3.4.1.4 Levels of service

If the service provided by a server to a client can be segmented into different levels, denoted by the vector $\mathbf{SL} = (SL_i)$ where $i = [0 \dots n]$ with $SL_n \rhd SL_{n-1} \rhd \dots \rhd SL_0$ then we also denote reputations that are required for such levels as the vector of reputation levels (or bands) $\mathbf{rl} = (rl_i)$ where $i = [0 \dots n]$ and $rl_n > rl_{n-1} > \dots > rl_0$. The notation $SL_n \rhd SL_{n-1}$ implies that SL_n is a more privileged service level than SL_{n-1}. Thus, we can express the condition for a service level k of a server x for client y in the application context α as:

$$rl_k \leq r_x(y, \alpha) < rl_{k+1} \implies SL_k \tag{3.7}$$

The reputation $r_x(y, \alpha)$ in equation 3.7 is either the locally observed reputation or the interpreted global reputation. If the vector $\mathbf{SL} = (SL_i)$ and $i \in \{0, 1\}$ then it implies a binary state for service provision, i.e. $SL_0 \implies$ *no service* and $SL_1 \implies$ *full service.*

This formalism of levels of service only shows the relation between a level of service and corresponding reputation value. It is very important to note that in addition, there can be other external factors affecting levels of service which are beyond the scope of consideration in our model. One such factor could be the consideration of the class of the client. Two different clients having the same reputation but belonging to different classes may not be entitled to the same service level.

3.5 The framework with example policies

Continuing from the functional parts of our framework described in § 3.4, we now present our proposed framework in detail. We also propose some example policies to work with our framework.

3.5.1 Analysis of behaviour

A server develops the reputation of a client by recording and analysing the history of its behaviour. However, the interpretation of "good" or "bad" behaviour is often relative; hence dependent on specific policies implemented by servers. Behaviour observation can be achieved through a variety of monitoring mechanisms, e.g. a Bayesian spam filter could scan text in an email message to determine its rank as spam. On the other hand, a router traffic monitor could detect how many simultaneous connections are opened by a particular client at any particular period in time. More than one such monitoring system may be used at a time to gather information about a client. Data obtained from each of these monitoring systems can be generalised and expressed as a tuple, given in equation 3.8.

$$\tau = \{client\_id, observed\_values, observed\_type, timestamp\} \tag{3.8}$$

We define $client\_id$ as the long-lived identity of the client; $observed\_values$ as the set of output values of the monitor (e.g. $\{0.9\}$ on a scale of 0 to 1 from a spam filter); $observed\_type$ identifies the type of behaviour observation, which could be, for example, email spam analysis, or low-level TCP packet analysis and $timestamp$ as the time at which the monitor observed a client behaviour. This $timestamp$ is used to detect multiple occurrences of similar observations,

i.e. repeated behaviour patterns. It is to be noted that *observed_type* and the application context α are not the same thing. In fact, same observations from various behaviour monitoring systems can be useful for behaviour quantisation for multiple application contexts. The behaviour analyser maintains a history of previously observed behaviour as a vector of τ observations $\boldsymbol{\Xi}$, where $\boldsymbol{\Xi} = (\tau_i)$ where $i = [0 \dots n]$. The implementation and policies determine how large this recorded history can be.

Using this input vector $\boldsymbol{\Xi}$, a policy-specific *behaviour analyser* can be implemented. Such an analyser evaluates all or part of $\boldsymbol{\Xi}$ to output the interpretation of behaviour in quantised form (both positive and negative), which is fed into the reputation building stage. Such a policy-specific behaviour analyser is mathematically generalised in equation 3.9, where $\Delta b_x(y, \alpha)$ is the quantised behaviour recorded by server x for the client y in the application context α, F_b is the policy-dependent function applied on $\boldsymbol{\Xi}$.

$$\Delta b_x(y, \alpha) = F_b(\boldsymbol{\Xi}) \tag{3.9}$$

This formalism keeps the behaviour analyser open-ended and it can augment existing network activity monitoring systems. The description of the function F_b and the specifics of implementation of the behaviour analyser fall outside the scope of this thesis because these are strongly tied with offline policy descriptions. We expect that organisational policies can be represented using any existing means of policy specification (see § 2.3), and thereby define the function F_b. However, we further discuss policy specification for the behaviour analyser in future work.

Note that $\Delta b_x(y, \alpha)$ and not $b_x(y, \alpha)$ is used to denote the output of the behaviour analyser. This is because behaviour is represented by $b_x(y, \alpha)$, which is accumulated over time and each output from the behaviour analyser signifies a *new instance* of behaviour, hence $\Delta b_x(y, \alpha)$. Although $b_x(y, \alpha) = \sum \Delta b_x(y, \alpha)$ in general, the cumulative behaviour $b_x(y, \alpha)$ needs to be calculated with respect to the reputation when the reputation is re-adjusted due to a time decay or adjustment from global reputation. We shall examine this in the following sections.

3.5.2 Building of local reputation

While providing service, a server should evaluate the reputation of a particular client for every (behaviour) input fed in by the behaviour analyser. This process of evaluation of behaviour data is called the local reputation response to client behaviour. The exact nature of the reputation response is governed by policies applied by the server.

To describe our model further, we propose a *logarithmic reputation response policy* as an illustrative example. In essence, we define the function F_{lr} from equation 3.1. Note that in the representation of a policy (e.g. the reputation response policy), the references to the client, the server and the application context are implicit. Thus, cumulative behaviour $b_x(y, \alpha)$ is represented as b as we are implicitly working with one particular client, server and an application context.

We chose a reputation response which enables fast reaction to bad behaviour and relatively slow response to good behaviour, with the following characteristics:

- Good reputation gets better with good behaviour until it reaches a positive saturation. The gradient of improvement slows as reputation rises.
- Good reputation will decrease more rapidly with bad behaviour than it will improve with good behaviour.
- Bad reputation gets worse with bad behaviour until it reaches a negative saturation. The gradient of worsening reputation becomes less steep as the reputation falls.
- Bad reputation increases with good behaviour at a slower rate than it worsens with bad behaviour (for the same magnitude of change in behaviour).

Any alternative reputation response with different characteristics could be used depending on the policy requirements. We use the reputation response described above for our simulation (see Chapter 5).

The following mathematical model has been found to fit the aforementioned reputation response. Let us assume that client reputation is denoted with r (see equation 3.1); cumulative behaviour variable with b (i.e. Δb denotes

behaviour outputs fed in by the behaviour analyser, see equation 3.9); positive reputation saturation level with R_{psat}; negative reputation saturation level with R_{nsat}; and two adjustable response parameters λ and μ. Note that the constants $R_{psat} = 1$ and $R_{nsat} = -1$ according to our continuum range shown in figure 3.2. For an event at any point in time, t, for which a change of behaviour is recorded, the corresponding cumulative behaviour is $b[t]$ and the corresponding reputation is $r[t]$; while $b[t']$ and $r[t']$ correspond to the previous event.

The equations are presented below.

The equation for good reputation that improves with good behaviour is:

$$r[t] = \left(1 - e^{-\lambda b[t]}\right) \quad \text{for} \quad \Delta b > 0,\, b[t] > 0,\, r[t'] \geq 0 \tag{3.10}$$

and the equation for bad reputation that worsens with bad behaviour is:

$$r[t] = \left(e^{\lambda b[t]} - 1\right) \quad \text{for} \quad \Delta b < 0,\, b[t] < 0,\, r[t'] \leq 0 \tag{3.11}$$

and the equation for arbitrary good reputation ($r[t']$) that worsens with bad behaviour is:

$$r[t] = \frac{r[t']}{b[t']} b[t] \quad \text{for} \quad \Delta b < 0,\, b[t] > 0,\, r[t'] > r[t] \geq 0 \quad \text{and} \quad r[t'] = \left(1 - e^{-\lambda b[t']}\right) \tag{3.12}$$

and the equation for arbitrary bad reputation ($r[t']$) that improves with good behaviour is:

$$r[t] = \frac{r[t']}{\left(1 - e^{\mu b[t']}\right)} \left(1 - e^{\mu b[t]}\right) \quad \text{for} \quad \Delta b > 0,\, b[t] < 0,\, r[t'] < r[t] \leq 0 \quad \text{and} \quad r[t'] = \left(e^{\lambda b[t']} - 1\right) \tag{3.13}$$

Figure 3.3 combines equations 3.10, 3.11, 3.12 and 3.13 to illustrate the nature of reputation response to behaviour. Calculation of reputation is stopped when the reputation value is close enough to either the positive or the negative saturation (e.g. within 0.1%) and the change in behaviour tends to saturate the reputation further. By stopping this saturation, it is possible for the reponse to be fast when the change in behaviour is of a different sign from the

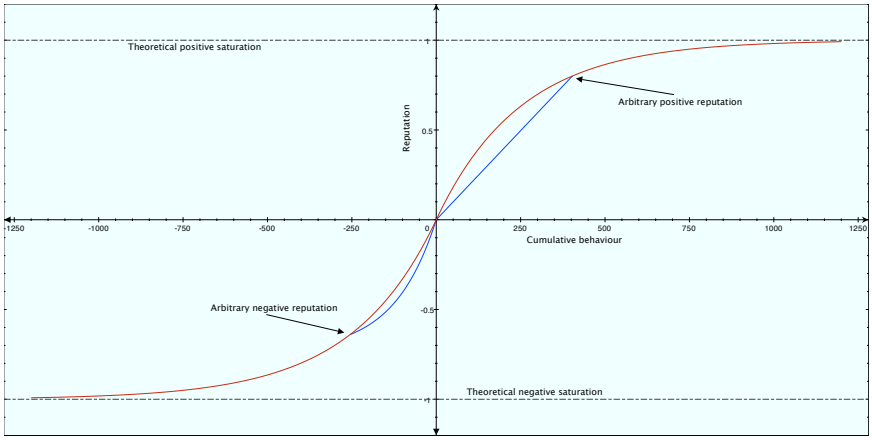

Figure 3.3: Graph of reputation versus behaviour

cumulative behaviour. It is evident from the graph that an identity with a high reputation (i.e. near positive saturation) will not be able to exploit it because poor behaviour will result in its reputation being reduced along the linear curve in the aforementioned figure (first quadrant). Similarly, if an identity has a bad reputation, it would require a demonstrable amount of good behaviour to improve its standing (third quadrant).

3.5.2.1 Time decay of local reptuation

A positively or negatively saturated reputation is considered "too good" or "too bad" respectively. Over time, reputation often needs to decay when there is no client-server activity. This helps a saturated bad reputation to recover slowly with time. It also questions a saturated good reputation if there has been no activity over time. Such time decays can be implemented through various policies. As an example, we choose a reasonable time decay policy in conjunction with the reputation response policy described above. This policy defines function F_{lt} from equation 3.2.

A neutral zone (between default values) $[R_{ndef} \quad R_{pdef}]$ such that $R_{nsat} < R_{ndef} < 0$ and $0 < R_{pdef} < R_{psat}$ where $R_{psat} = 1$ and $R_{nsat} = -1$ is defined for this purpose. Positive reputation higher than R_{pdef} decays to the positive default, while negative reputation lower than R_{ndef} 'decays' (in essence, increases) to the negative default. An adjustable decay rate parameter ϵ is introduced for this purpose. The equation for reputation decaying over

time from an arbitrary reputation value $r[t']$ in the past is denoted as $r[t]$ (i.e. reputation after time $t - t'$) and is given as:

$$r[t] = \begin{cases} r[t'] \left(1 - \epsilon(t - t')^2\right) & \\ R_{pdef} & \text{for } 0 < r[t] < R_{pdef} \\ R_{ndef} & \text{for } 0 > r[t] > R_{ndef} \end{cases} \tag{3.14}$$

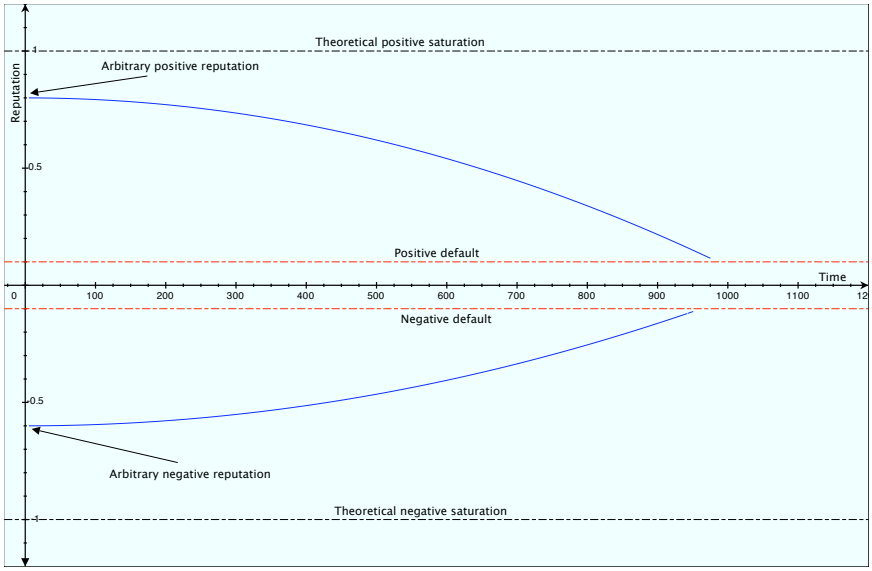

Figure 3.4: Graph of reputation decay with time

Figure 3.4 illustrates equation 3.14. Once the decayed reputation reaches the boundaries of the neutral zone, the decay stops. When time decay happens to reputation, it is necessary to adjust the cumulative behaviour value before calculating reputation response to behaviour. Through this adjustment, the cumulative behaviour assumes such a value that would have generated the time decayed reputation through either equation 3.10 or 3.11 depending on whether the reputation is positive or negative. Thus, this adjustment to cumulative behaviour is achieved by the inverse functions of those defined in equations 3.10 and 3.11:

$$b[t] = -\frac{log_e(1 - r[t])}{\lambda} \quad \text{for} \quad r[t] \geq 0,\, r[t] \neq 1 \tag{3.15}$$

and

$$b[t] = \frac{log_e(1 + r[t])}{\lambda} \quad \text{for} \quad r[t] \leq 0,\, r[t] \neq -1 \tag{3.16}$$

If using weak identities, note that the time decay policy can be used to accommodate IP-based identities. When IP addresses are frequently re-used (thus making them weak and short-lived identities), the time decay policy can be made to ensure that the the reputations are decayed to default values within the DHCP minimum lease period.

3.5.3 Global client reputation reporting

Observed local reputations are submitted by servers to the GRA. The process of submission can either happen at the end of a sporadic service or during an ongoing service. Submissions can be made several times as long as the evidence of service interaction between a server and a client is valid. This evidence is an *authorisation token*, which the client provides to the server. This token may contain information relevant to the implementation. However, it will at least contain a timestamp signifying the expiry of the token, the identity of the server and the application context for which the token is intended. At any point in time, a server cannot hold more than one authorisation token (not even expired) from the same client for the same application context. The application context identifies the type of application for which the reputation is developed. This is different from the *observed_type* parameter in the behaviour analyser because *observed_type* offers more granularity in behaviour analysis whereas the application context describes the type of application for which reputation is developed. For example, the *observed_type* parameter could represent analysis of the number of concurrent connections open by a client to a router or the spam score of a particular email message, whereas this information could be useful for developing reputation for an application context such as "email". We leave the semantics of the application context parameter for future work. It is essential that prior to this reporting mechanism, the identities of the client and the server are known to the GRA. We assume that any communication between a client and the GRA or between a server and the GRA is safe against a man-in-the-middle attack. This can be ensured through the use of encryption or digital signatures. The algorithm showing the steps from reputation query to global reputation reporting is illustrated in algorithm 3.1. Prior to the steps mentioned in this algorithm, both the client and the server are expected to have notified the GRA of their identities. Note that the following steps in reporting global reputation are based on the assumption that a public-key identity infrastructure is in place. For other

identity infrastructures, the algorithm will need some adaptation, which we leave to future work.

Algorithm 3.1 Global reputation reporting

Step 1: The client provides the GRA with a signed authorisation token, in an application context, for a server that it will request service from.

Step 2: If the GRA accepts the token, the client also sends the same signed authorisation token to the target server. The GRA may not accept the token if it already has another (possibly unused) token for the same application context and the same server.

Step 3: The server signs the authorisation token again to query the global reputation of the client from the GRA.

Step 4: The GRA performs a token authenticity check confirming that: the two copies of token are equal; the copy from the server comes from the server identified in the token by the client; and that the timestamp is valid.

Step 5: On a successful authenticity check, the GRA scavenges out-of-date reputation reports (see equation 3.6), and returns the list of global reputations of the client with respect to server's confidence in other servers that have reported the reputations of the client.

Step 6: The server makes inferences from the global reputations of the client obtained from the GRA.

Step 7: The server provides service to the client and uses a reputation-response policy to form its local reputation of the client which, if required, is used to change service levels accordingly.

Step 8: Either at the end of a sporadic service or during an ongoing service, the server signs the authentication token, and sends it with its local reputation of the client and other necessary parameters (see equation 3.5) to the GRA.

Step 9: The GRA performs a token authenticity check. Unlike the reputation query, the request to report reputation is granted even if the authorisation token has expired as long as the token was once valid. This caters for network delays and failed connections between the server and the GRA. On a successful token authenticity check, the GRA records the reported reputation and other associated parameters and invalidates the authorisation token. If the server had reported a reputation for the client in the past then the former is overwritten with the latter if both reports are in the same application context.

Step 10: If the server wants to report more than once during an on-going service, it has to request a new authorisation token from the client every time. For an ongoing service, a client could be required to re-issue another authorisation token as soon as the server reports its reputation to the GRA.

Figure 3.5 expresses algorithm 3.1 as a UML sequence diagram. Note that the step numbering in the algorithm should not be confused with that in the UML sequence diagram.

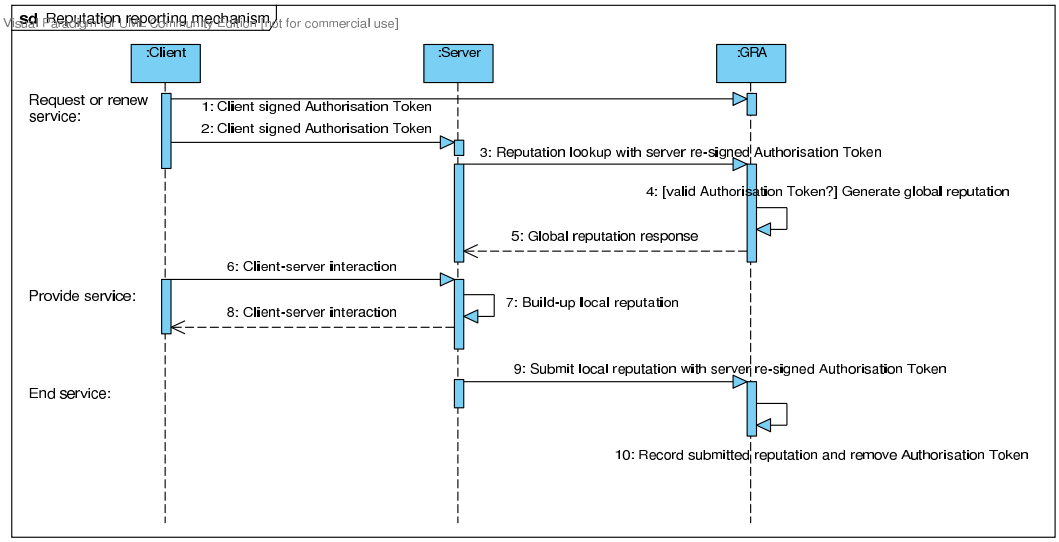

Figure 3.5: UML sequence diagram for reporting reputation

3.5.4 Global reputation query and interpretation

Global reputations of clients act as opinions shared between servers. The precise interpretation of the global reputation depends on the perspective of a querying server. Through the process of reputation reporting, a set of reported reputations for a particular client are stored by the GRA. The size of this set may grow in proportion (e.g. logarithmically, linearly) to the total number of servers the client has interacted with and is interacting with. There is an age-based method of scavenging which prunes off older reputation values ensuring that the reputations stored by the GRA are not out-of-date, as formalised in equation 3.6.

Using the example policy for local reputation response described above, for a particular client, server and application context, the tuple of reputation and associated parameters submitted by the server is γ from equation 3.5 represented with the appropriate parameters substituted for $\psi_x(y, \alpha)$ as:

$$\gamma = \{\alpha, r, \{\lambda, \mu\}, t_{report}\} \tag{3.17}$$

where $\psi = \{\lambda, \mu\}$; λ and μ are reputation response parameters defined in equations, in § 3.5.2, for the local reputation response policy for a specific client y, server x and application context α. It follows from equation 3.6 that

over time when several servers submit such reputations for the client, a set of recorded reputations is defined as:

$$\Gamma = \{\gamma : \text{where } r \text{ component in } \gamma \text{ is not too old}\} \tag{3.18}$$

To determine whether r in γ is too old or not, we use the following age-based scavenging policy (i.e. function F_{gt} from equation 3.6). In essence, any other scavenging policy can be specified. We use the following example policy that is consistent with our chosen local reputation response described earlier.

In a particular γ-tuple at any time t:

- if $0 < r \leq 1$ and $\lambda(t - t_{report})^2 \geq 1$ then the tuple is scavenged;
- if $-1 \leq r < 0$ and $\mu(t - t_{report})^2 \geq 1$ then the tuple is discarded;
- if, however, $r = 0$ the tuple is discarded when both conditions are met.

This scavenging method ensures that reputations that have been generated by servers with tougher reputation response conditions (i.e. lower λ and μ) are decayed slower than the ones with more lenient conditions. The characteristic of the decay is similar to that defined in equation 3.14 for time decay of local reputations. In general, it is better to have a slower decay of global reputation than local. Therefore, the time values can be scaled down by a factor such that the decay is slower than the local reputation time decay.

When a server x queries the global score of the client y for a particular context α, the query returns a set of client reputations for y formed from each $r_i(y, \alpha)$ from the i^{th} not age-scavenged γ-tuple with its egress confidence $w_{x,i}$ for the context α. The result of the query also contains each $r_j(y, \alpha)$ for which $w_{x,j}$ cannot be calculated.

3.5.4.1 Confidence between servers

When the reputations of clients are shared globally between servers, it is necessary that there is a notion of trust in the shared information. In an earlier version of our model (Basu *et al.* 2007), there was a concept of developing a ranking of servers by clients in order to infuse some kind of feedback on the reputation of clients reported by servers. However, this opened up the model to possibilities of attacks where clients could give enough negative feedback on

a particular server to render its opinion (i.e. reported reputation) of any other client useless.

To facilitate interpretation of globally reported client reputations based on similarity of opinions, we introduce a concept of *confidence* between servers. It is denoted by w with range $[-1 \quad 1]$. Values above zero (i.e. positive confidence) indicate agreement of opinion whereas values below zero (i.e. negative confidence) indicate disagreement in opinion. If the value is zero (i.e. non-existent confidence) then it signifies indifference between the opinions of the servers.

The model evolved into a matrix of confidence ratings formed between servers from their social network asymmetric weighted digraph (Basu *et al.* 2008). Intially, the allowed degree of separation was more than one but it soon posed problems of fast recalculation of the matrix (depending on how large the allowed degree of separation was), and also the strategy of edge traversal (i.e. min-flow, max-flow, etc.) taken in recalculating the confidence matrix. Therefore, a single degree of separation was adopted and transitivity of confidence was dropped. An asymmetric confidence could be explicitly specified ($w_{x,y}$) by one server (e.g. x) on another (e.g. y) but this confidence value is not aware of the application context.

As the model evolved further, the confidence ratings were made context-aware, and the calculation of confidence derived automatically from statistical similarity functions. This confidence rating can be used to interpret results of a global reputation query. It is context-aware, which means the confidence between two servers is dependent on the application context specified in the global reputation query. The confidence is calculated at the time of the global reputation query and depending on the similarity function (see equation 3.3), it can be symmetric (i.e. if server x has a confidence $w_{x,y}$ on server y then $w_{y,x} = w_{x,y}$).

Researchers have analysed several methods of estimating such trust based on similarity of opinion (e.g. Herlocker *et al.* 1999, Breese *et al.* 1998), while others have argued the efficacy of statistical correlation coefficients (e.g. Lathia *et al.* 2008). We use, as an example policy, a statistical correlation coefficient

to determine similarity of opinion and thus define F_c from equation 3.3. Any other similarity measure including any correlation coefficient other than what we have chosen can be used. To calculate the confidence, the set of reputations submitted by a particular server for an application context for clients that are in common with another server is analysed. If this reputation data fits a Normal distribution then we use the parametric Pearson's correlation coefficient (Rodgers & Nicewander, 1988; Stigler, 1989). If the data does not fit a Normal distribution then the non-parametric Spearman's rank correlation coefficient (Spearman, 1987) is applied by converting those reputation values to ranks.

Using the local reputation response described above, the reputation data submitted by servers is put through a test of Normality. We prefer an extended Shapiro-Wilk W statistic (Royston 1982) over Lilliefors tests (Lilliefors 1967) because of the possible presence of ties in the data. The extended Shapiro-Wilk W test is seen as one of the most accurate tests for Normality. If the null hypothesis that the data is Normal cannot be rejected then we apply the Pearson's product-moment correlation coefficient to the reputation data vectors. If, on the other hand, the null hypothesis that the data is Normal can be rejected, we apply the Spearman's rank correlation coefficient. Therefore, if we denote any i^{th} element of the reputations reported by server x as r_{x_i} and similarly any i^{th} element of the reputations reported by server y as r_{y_i} (or if r_{x_i} and r_{y_i} denote the equivalent ranks if the reputation vectors are not Normal) then we can define the Pearson's product-moment correlation coefficient $w_{x,y} = w_{y,x}$ as:

$$w_{x,y} = w_{y,x} = \frac{n\Sigma r_{x_i} r_{y_i} - \Sigma r_{x_i} \Sigma r_{y_i}}{\sqrt{n\Sigma r_{x_i}^2 - (\Sigma r_{x_i})^2}\sqrt{n\Sigma r_{y_i}^2 - (\Sigma r_{y_i})^2}} \tag{3.19}$$

A positive correlation between reputations submitted by servers of clients that they have in common defines a measure for similarity of opinion. Similarly, a negative correlation defines a measure for dissimilarity of opinion.

It is to be noted that even if we use, as an example, Pearson's correlation coefficient and Spearman's rank correlation coefficient as the similarity measure for Normal and not Normal distributions respectively, it is possible to use

a non-parametric similarity measure (e.g. Kendall 1938; Siegel & Castellan 1956) throughout.

3.5.4.2 Interpretation of global reputation

What the querying server does with the reputations and the confidences is implementation and policy specific. Also, the interpretation can happen at either of the following two stages:

- when the querying server has not interacted with the client for which it is querying a global reputation; and
- when the querying server has previous history of the client but is using the client's global reputation to re-adjust its local interpretation.

For example, the querying server may do any of the following with a global reputation result:

- Alter parameters in the reputation response such that the response is either made more strict or more lenient. In the example logarithmic response policy, this will mean altering the reputation response parameters (i.e. λ and μ).
- Initialise the local reputation for the client from a statistical measure of the data obtained from the global reputation query. This can effectively serve as using the global reputation as a recommendation. Such an initialisation could mean that the querying server may re-initialise its local reputation observation of a client with the one with least deviation that it obtains from the global reputation query (i.e. deviation test). Alternatively, the querying server could initialise the local reputation of a client (for which it has no past observation) to the global reputation value that has the highest positive confidence.

Note that if re-initialisation happens to reputation, it is necessary to adjust the cumulative behaviour value before calculating the reputation response to behaviour. Through this adjustment, the cumulative behaviour assumes such a value that would have generated the global reputation through the local reputation response function, i.e. either equation 3.10 or 3.11 depending on whether the reputation is positive or negative. Thus, this adjustment to cumulative behaviour is achieved by the equations 3.15 and 3.16 for our example reputation response.

In our simulations, we compare the following interpretation policies (i.e. function F_g from equation 3.4) as examples. Many other interpretation policies could be used, which may constitute using the highest positive reputation; or the lowest negative reputation; or the the positive reputation with the highest confidence, etc.

- *Least deviation:* use the global reputation component that is the least deviated from the last local reputation observation (before applying time decay, if relevant).
- *Highest confidence:* use the reputation component that has originated from the server with the highest confidence; or use the geometric mean of the reputations that have the same highest confidence.
- *Highest:* use the highest available reputation component, which signifies a highly optimistic interpretation.
- *Lowest:* use the lowest available reputation component, which signifies a highly pessimistic interpretation.

3.5.5 Decisions on levels of service

The server may wish to divide the entire range of reputation $[-1 \quad 1]$ into various bands; each band signifying a service level. Once reputation data about a client is formed either locally or obtained from the GRA, the server can choose to allocate a service level corresponding to the reputation. This allocation is policy specific and it is to be kept in mind that there are other factors, beyond the scope of this thesis, that could affect service level allocation. For example, two clients having the same reputation with the same server for the same application context may still be allocated different levels of service because the two clients belong to two different classes of service according to their service level agreements. The allocation of service levels also imply that the client could be allocated a service level which denies service to it altogether. On the other hand, generally well-behaving clients could be given the benefit of doubt for occasional bad behaviour by relegating them a few service levels but still providing them some service, which is a step to reduce false positives.

As an extension, the server can also use the allocation of service levels to manage quality of service to the clients. For example, in times of server overload, the clients eligible for higher service levels can be prioritised with higher levels of quality of service.

3.6 Adversary model

The European Network and Information Security Agency (ENISA) position paper 2 (Carrara & Hogben, 2007) presents a number of threats and attacks (and recommendations against them) on reputation systems, derived from four main use cases: online markets, peer-to-peer networks, anti-spam techniques and public key authentication (web-of-trust). We develop from their taxonomy to study the threats that are relevant to our model and discuss possible measures against such attacks. For each attack, we define the target of the attack, describe the attack and discuss how, if at all, our framework can cope with it.

3.6.1 Identity threats

3.6.1.1 Whitewashing

Attack target: Server

Attack: A malicious client (the attacker) rids itself of its bad reputation by rejoining the system as a new identity.

Defence: This is primarily an attack on the identity infrastructure and hence outside of the scope of this thesis. The stronger the identity infrastructure and the less often it allows effortless change of identity, the less likely the threats from this attack. In addition, depending on the policy of initialising reputation for new clients, this attack may not be able to cause substantial harm to the server. For example, if any new client identity is assigned a minimal service level and the new client is required to prove itself through good behaviour then the attacker is unlikely to be able to perform malicious activities without earning a bad reputation quickly. Also if the attacker behaves well for a while, thus earning good reputation, and misbehaves after that then the penalty, depending on policy, could mean a fast fall of reputation and hence denial of higher service levels.

If a weak identity scheme, such as IP addresses is used then whitewashing is obvious to accommodate for real identities reusing IP addresses. As we have seen in § 3.5.2.1, adjusting the time-decay parameter to make the reputation of a particular IP (identity) fall to a default level after the DHCP minimum lease

period (of inactivity) is essentially a server-induced whitewashing technique to cater for re-use of the IP address.

3.6.1.2 Sybil

Attack target: Server

Attack: A client can have multiple Sybil identities, e.g. see the Sybil attack described by Douceur 2002. The same malicious client with multiple identities can disguise itself to a server, thus developing separate behaviour profiles with the same malicious intent.

Defence: Due to the centralised nature of the GRA, the one-to-one correspondence between a real entity and an identity may be ensured through a strong identity infrastructure. However, there is no built-in defence against the Sybil attack. This attack is similar to the use of botnets to generate email spam. Even if a different behaviour profile is developed for each Sybil identity, instances of bad behaviour will be penalised likewise. It would be interesting to group identities using behaviour profiling to eliminate the threats of a Sybil attack (e.g. Wei *et al.* 2006). However, we will leave this for future work.

3.6.1.3 Impersonation and reputation theft

Attack target: Client

Attack: A malicious client (the attacker) acquires the identity of another client to use its good reputation.

Defence: This is an attack on the identity infrastructure. There is no built-in defence against this attack in our reputation model. It is up to the identity infrastructure to ensure that identities cannot be masqueraded.

3.6.2 Vulnerabilities in and threats to reputation systems

3.6.2.1 Reputation bootstrap issue

Attack target: Server

Attack: The choice of the initial reputation value for a new client could potentially pave the way for Sybil or whitewashing attacks.

Defence: The choice of the initial reputation (i.e. "bootstrap" value) is not trivial. Although it is policy dependent, the initial reputation value can be set at zero unless there is enough confidence on a particular global reputation to set it to a different value. In addition, the option of varying the reputation response parameters offers some defence mechanism to the servers to take action quickly if a client misbehaves after its initial reputation has been set to a relatively high value.

3.6.2.2 Extortion, denial of reputation and ballot stuffing

Attack target: Client and server

Attack: Some "bad" servers can collaboratively damage the global reputation of a "good" client by reporting false bad reputation. Alternatively, some "bad" servers can collaboratively improve the global reputation of a "bad" client by reporting false good reputation. Alternatively, an otherwise "good" client can behave badly with one or few target servers so that the global reputation of the client is good enough to misguide the target servers.

Defence: The presence of a correlation between reputations reported by servers acts as a defence against this attack. While a group of "bad" servers can still report false reputation of a client, a "good" server will not receive positive confidence or will receive no confidence on such reports due to the inherent likeliness of dissimilarity of opinions between the "good" server and the "bad" servers. However, further enhancements to the defence against this attack falls in the remit of future work.

3.6.2.3 Repudiation of data or repudiation of transaction

Attack target: Server

Attack: In this attack, a malicious server can deny that a network transaction (i.e. an interaction between a server and a client) has happened or it can choose to not report its local observations to the GRA.

Defence: Every look-up of reputation by a server requires an authorisation token from the client. At any point in time, a server cannot hold more than one authorisation token (not even expired) from the same client for the same application context. Therefore, if the server has not submitted its local observation of reputation, the GRA will still maintain an authorisation token (even after it expires). Therefore, the server will be unable to query the reputation of a particular client for the same application context as long as such an unused authorisation token exists. This means that, for any particular client for a particular application context, if the server does not contribute to reporting reputations then it will not be able to query the global reputations either. This is a discouragement against repudiation of reputation data of transaction. Further encouragement to report such reputation data could be enforced through offline policies, which are beyond the scope of this thesis.

3.6.3 Reputation infrastructure threats

3.6.3.1 Attacks on the underlying network

Attack target: Client, server and GRA

Attack: This could be a denial of service (DoS) or distributed DoS attack on the client, the server or the GRA. It could also be a man-in-the-middle attack on any of the three entities.

Defence: There is partial defence against a DoS attack on the client, the server or the GRA. However, a DoS attack does not affect the reputation system to a large extent. If the network entity is completely inaccessible from the network due to a DoS then there is no defence against it in our framework. If the service is still provided and behaviour recorded, then the server can still maintain its local observation of reputation. The attacker will, in that way, develop bad reputation due to the attack and service (or connection) to the attacker will be eventually cut off. In the event of a DoS attack on the GRA which is capable of making the GRA completely unreachable on the network, it becomes impossible to make global reputation queries and reports but servers can still cope with the temporary absence of the GRA by using their local observation of reputation. If the design of the GRA is distributed,

it is unlikely to have a single point of failure even under a distributed DoS attack.

The defence against man-in-the-middle attack is achieved through encrypted or digitally signed communications between the client and the server, the client and the GRA and the server and the GRA.

3.7 Summary

Following from our research question described in § 2.7, in this chapter our main research contribution is a high-level, policy-independent, privacy-preserving, context-aware framework to develop and share client reputation based on behavioural history. The reader has been introduced to definition of frequently used terminology in § 3.2. We have, then, described the mathematical representations for *reputation* and *confidence* in the context of our framework, with notations similar to Marsh & Briggs 2008.

This has been followed by the description of the framework in the following functional stages.

- Analysis of behaviour
- Building of local reputation
- Global reputation reporting
- Global reputation query
- Decision making for service level

Alongside the description of the framework, we have also proposed examples of some policies for developing and sharing client reputation. Having described the functional stages of the framework, we have listed the various possible attacks on our framework based on the taxonomy from Carrara & Hogben 2007. We have described the defences against those attacks wherever appropriate. In the following chapters, we illustrate the simulation strategies of our framework and examine simulation results.

4 Experimental setup

Do not worry about your difficulties in Mathematics. I can assure you mine are still greater.
Albert Einstein
German-born US physicist (1879 - 1955)

4.1 Overview

In this chapter, we describe the experimental setup required to evaluate the framework described in chapter 3. We briefly discuss some implementation aspects required for simulation and also describe our home-grown discrete event simulator that is used in chapter 5 to simulate the model. The simulator will be released as open source, so that our simulation results can be verified later. The simulator is written entirely in Java (Arnold *et al.*, 2005; Gosling *et al.*, 2005), while graphs are generated by MATLAB (Mathworks, 1984) scripts parsing the simulator log files.

The complexity of the design of network applications on and the learning curve associated with certain simulators (e.g. such as ns-2 and ns-3 USC Information Sciences Institute 2009a,b) have led us to design our own simple discrete event simulator sufficient to simulate our framework at a high level without the unnecessary protocol specific and network level details.

4.2 Simulated application

The *simulated application* is an application built atop our *discrete event simulator*. The application is an implementation on the simulator of the relevant stages in our framework proposed in chapter 3. Before we discuss our discrete event simulator, we describe how various entities, events and policies are implemented in this application. The simulator simulates the simultaneous interaction between a number of clients and servers. Some of these servers have common clients between them. The purpose of the simulation is to illustrate graphically the reputation response corresponding to changes in behaviour. We also show the effect of initialisation and re-adjustment of local reputation using various policies for global reputation interpretation.

4.2.1 Network entities

Similar to the theoretical model of our framework, the simulated application consists of three types of network entities – the client, the server and the Global Reputation Analyser. Each of them has an incoming and an outgoing network link, which can be used to simulate network failures.

4.2.1.1 Client

A client is a network entity associated with its public and private key pair, which are generated prior to a simulation run. A client is able to generate (and sign) authorisation tokens and maintain a list of authorisation tokens in-use.

4.2.1.2 Server

A server is a network entity associated with its public and private key pair, which are generated prior to a simulation run. A client can request a service and consume a service from a server. A server can query global reputation of a client and report the locally observed reputation of a client to the Global Reputation Analyser. Servers maintain persistent data, such as local reputations and associated parameters, about the clients they interact with. In the implemented application, such persistent data is maintained in an embedded database (Oracle Berkeley DB Olson *et al.* 1999; Oracle 2006) shared by all servers. The database is transactional for consistency reasons of semantic ordering of events (which we will discuss later in this chapter), but the Durability

criterion of Atomicity Consistency Isolation and Durability (ACID) criteria of transaction processing (Haerder & Reuter, 1983) is dropped in favour of performance by reducing disk writes. Thus, completed transactions are held in memory and written to disk only at the end of simulation run.

4.2.1.3 Global Reputation Analyser

A Global Reputation Analyser is a network entity, which has only one instance available to the entire application. It maintains a range of persistent data, such as client profiles, server profiles, authorisation tokens, client reputations as reported by servers, and so on. Similar to the implementation of servers, such persistent data is maintained in a single embedded transactional database with no disk writing per transaction.

4.2.2 Implemented events

The simulated application implements the events presented in table 4.1.

Table 4.1: Implemented events

Event name	**Event specification**[a]
RegisterClientEvent	`<event-time> regcli <client-id>`
RegisterServerEvent	`<event-time> regsrv <server-id>`
CreateAuthorisationTokenEvent	`<event-time> mkatok <application-context> <client-id> <server-id> <token-expiry-time>`
RequestServiceEvent	`<event-time> reqsvc <application-context> <client-id> <server-id>`
ConsumeServiceEvent	`<event-time> eatsvc <application-context> <client-id> <server-id> <quantised-behaviour>`
ReportGlobalEvent	`<event-time> putglo <application-context> <client-id> <server-id>`
NetworkDownEvent	`<event-time> netdn <target-type> [<target-id>] <direction>`
NetworkUpEvent	`<event-time> netup <target-type> [<target-id>] <direction>`

[a]Each event specification is a single line. Line breaks in this table are due to text wrapping.

RegisterClientEvent This event represents the registration of a client identifier with the Global Reputation Analyser.

RegisterServerEvent This event represents the registration of a server identifier with the Global Reputation Analyser.

CreateAuthorisationTokenEvent This event represents creation of an authorisation token by a particular client for a particular server for a specific application context.

RequestServiceEvent This event represents the request of service by a particular client from a particular server for a specific application context for which an authorisation token has already been created. This request makes the server query the Global Reputation Analyser for the client's global reputation.

ConsumeServiceEvent This event represents the consumption of service by a particular client from a particular server for a specific application context for which a service request has been made. Even if the result of a global reputation query may indicate that the server should not provide service to the client, our simulator does not stop the client from getting service. In this way, we can generate behaviour and reputation graphs with respect to time, which we further analyse to point out how a particular client may not have received any service if the server was using a particular entry-level policy for access to service.

ReportGlobalEvent This event represents the reporting of global reputation by a particular server of a particular client for a specific application context. This event also invalidates the authorisation token generated for the particular client, server and application context. If the client is supposed to continue with service consumption, the client is required to generate another authorisation token and the server is required to make a global reputation query. This means that a server providing service to a client for a long period of time can keep reporting its local reputation of the client to the Global Reputation Analyser from time to time and also make global reputation queries so as to make decisions in continuing service provision.

NetworkDownEvent This event represents the failure of either the incoming or the outgoing network link or both for a particular entity – a client, a server or the GRA.

NetworkUpEvent This event represents resumption (after a failure) of either the incoming or the outgoing network link or both for a particular entity – a client, a server or the GRA.

For both *NetworkDownEvent* and *NetworkUpEvent*, `<target-id>` is optional if `<target-type>` is the GRA. Otherwise, it is either the client identifier or the server identifier depending on whether the `<target-type>` is either client or server respectively.

4.2.3 Event ordering and interaction cycle

In our simulation as well as in the real-world implementation of our framework, certain events must occur before certain others. For example, we have seen in the previous chapter that the identities of clients and of servers must be known to the GRA before the servers can provide services to the clients. Similarly, the first event of consumption of service (between any particular server and a client for a particular application context) must be preceded by an event in which the client requests service from the server, triggering a global reputation look-up. The following list illustrates the semantic order of precedence (with lower number signifying higher position in the order) of the implemented events.

1. *RegisterClientEvent* and *RegisterServerEvent* (only once for each client identifier or server identifier)
2. *CreateAuthorisationTokenEvent*
3. *RequestServiceEvent*
4. *ConsumeServiceEvent*
5. *ReportGlobalEvent*

A *CreateAuthorisationTokenEvent* can follow a *ReportGlobalEvent* for the same client and server pair for the same application context. Events such as *NetworkDownEvent* and *NetworkDownEvent* do not need any order of preference in relation to the aforementioned events. However, a *NetworkUpEvent* must be preceded by a NetworkDownEvent for the same network entity (i.e. a client or a server or the Global Reputation Analyser).

For a particular client, a server and an application context, a set of events starting with *CreateAuthorisationTokenEvent*, followed by a *RequestServiceEvent*

and a number of *ConsumeServiceEvent*s, and ending with *ReportGlobalEvent* constitutes what we call an *interaction cycle*. We shall re-visit this concept in section 4.3.3.

4.2.4 Implemented policies

In chapter 3, we emphasized that many aspects of our theoretical model are governed by policies. Here we describe our choices for implementing examples of such policies for the purpose of simulation.

4.2.4.1 Behaviour analyser and behaviour quantisation

The behaviour analysis stage in our framework is not simulated. This is because the behaviour analyser involves application of security and management policies on network actors, thus forming quantised behaviour. However, the specification of such policies is outside the scope of this thesis. Instead, we use population statistics of real world behaviour data (e.g. email spam statistics from Senderbase, Cisco Systems 2009b) to generate synthetic quantised behaviour values pertaining to certain actor classes (e.g. "very good client", "spammer", "client that inadvertently causes bad behaviour from time-to-time" for email clients), which we define for the particular application context that is simulated.

4.2.4.2 Local reputation response policy

In our simulator, we implement two local reputation response policies – the logarithmic response described in the previous chapter and a scaled linear no-history response. With the latter reputation is linearly proportional to behaviour: the characteristics of the reputation versus time graph will be the same as that of the graph of cumulative behaviour versus time. Thus, the cumulative behaviour itself is used as a base reference to show how the logarithmic response is useful in comparison with a response that does not preserve history. The no-history base reference response is in direct linear relationship with cumulative behaviour.

4.2.4.3 Local reputation saturation policy

In the previous chapter we described that the calculation of reputation is stopped when reputation is close enough to a saturation level and a behaviour

change leads the reputation closer to the saturation. For most of our simulations, we use an example policy with an arbitrary allowed closeness of saturation at 99%. This means the calculation of reputation will be stopped when the reputation reaches 99% of a saturation level, and only if any further change in behaviour will make the reputation value reach closer to the already approaching saturation level. If the age-scavenging policy is enabled, note that the age scavenging policy is applied first. Therefore, a change in reputation due to age scavenging may mean that the reputation is no longer close to saturation.

4.2.4.4 Global reputation interpretation policy

We implement a number of example global reputation interpretation policies as mentioned in § 3.5.4.2. In our simulation results, we compare those policies alongside one another.

4.2.4.5 Age-based scavenging policy

Both the local and the global reputations are subject to age-based scavenging policies described in the earlier chapter. The scale-down factor for time in global reputation age-based scavenging is set to 1000.

4.2.4.6 Server confidence policy

Described in § 3.5.4.1, we use Pearson's product-moment correlation coefficient or Spearman's rank correlation coefficient depending on the results from Shapiro-Wilk W test for Normality. Implementation and interpretation of the extended Shapiro-Wilk W test is derived from Limewire (Limewire, 2008) and R94 (Royal Statistical Society, 1995).

4.2.5 Modeling attacks

While most of the attacks described in § 3.6 are modeled through varying the behaviour of clients, the Denial of Service attack is not. In our simulation, we simulate the extreme, and most unlikely, cases where a Denial of Service attack is able to completely cut off either a server or the GRA from the network.

4.3 A discrete event simulator

Having described certain aspects of the simulated application, we now describe some features of our discrete event simulator. Our simulator is built on the basic concepts of discrete event simulation, described in Law & Kelton 1997.

4.3.1 Event model

In our discrete event simulator, events are generated by an event generator before the simulation is started. During simulation, no further causal events may be generated and added to the event queue. All pre-generated events are parsed from an events specification file and loaded onto a priority queue where the priority is given to time. Each event has an associated time, which is the simulation timeslot it belongs to. This is the time at which the event is handled. There may be more than one event in one timeslot. Although the ordering of events in one timeslot is the order in which they are parsed from the events specification file, there is no guarantee in which order they will be executed if the simulator is running in multi-threaded mode. However, there is a guarantee that all events in a particular timeslot will always be executed before any other event in a subsequent timeslot. This execution order is used by the event generator to generate events in an order that is semantically correct (see § 4.2.3) for simulation.

4.3.1.1 Event parser

The event parser is obtained from a parser generator and a grammar specification. The grammar specification describes the acceptable event formats. The parser checks the syntax of each line in the events specification file. If correct, the event specified in that line is loaded into the event queue. If any line in the file is incorrect according to the grammar, the parser aborts parsing the entire file. Each line is treated separately and the parser does not check semantic ordering of events. In addition, the parser does no integrity check on the event specifications. For example, if there have been some consume service events between a client and a server, the parser does not check if there is also an event to report global reputation at some point after the consume service events. Such integrity is maintained by the event generator along with the events macro file, described later on in this chapter.

4.3.2 Event dispatcher and event handlers

When the simulation clock reaches a particular timeslot, all events in the timeslot are dispatched by a event dispatcher to separate event handlers. However, the simulation clock is not advanced further until all events in that timeslot have been simulated. The event dispatcher may run in a single-threaded or a multi-threaded mode. In the multi-threaded mode, the dispatcher uses a fixed thread pool with a number of worker threads in proportion to the logical CPU cores available to the Java Virtual Machine. Whether handled by a single thread (i.e. single-threaded mode) or any thread from a thread pool (i.e. multi-threaded mode), each event handler notifies the event queue when it finishes execution. Thus when all events from a particular timeslot have been simulated, the event queue can proceed to the next timeslot.

The multi-threaded execution of event handlers means that any event in a particular timeslot may execute before any other event in the same timeslot. This is an essential consideration that the event generator makes to ensure that there is no dependency between such events in one timeslot. In other words, the event generator does not generate any event that may depend on another in the same timeslot; e.g. an event to request service from a server depends on an event to create an authorisation token for the same client and server pair for the same application context. If such dependent events are present in one timeslot and the execution order by the thread executor is different from their intended semantic order then the simulator will generate semantic errors. In this example, the semantic order states that a client cannot request a service from a server before it has provided the server with an authorisation token. To be on the safe side, our event generator does not generate any two events for any particular client and server pair in the same timeslot. It is important to note, however, that certain other subtle dependencies will not pose a problem. For example, an event to report global reputation involving one client and a server can implicitly affect the event for requesting service involving the same client and another server. This is because of the impact on global reputation and confidences. However, this does not lead the simulator to a execution state that is semantically wrong. Those dependencies will only introduce certain acceptable degrees of unpredictability of results due to execution order, which is similar to real world scenarios.

4.3.3 Event generator

The event generator runs either in an user-interactive mode or in macro-mode. When in user-interactive mode, the event generator asks the user a number of questions to generate a list of *interaction cycles* as well as other associated events. After a user-guided event generation is finished, the user inputs are written to a macro file. The event generator can be run in macro-mode using such a macro file. In macro-mode, the event generator does not ask the user for any answers but retrieves those answers from the macro file. The macro mode is useful to generate events with large numbers of interaction cycles without having to prompt the user for the specification of each interaction cycle. It is worthwhile to note at this point that the input provided by the user must conform to the semantic ordering of events.

4.3.3.1 Event macro

The event macro, in particular, maintains the specification for each interaction cycle (and its repeat occurrences) as follows:

- client identifier
- server identifier
- application context
- simulation clock time when the first occurrence of this interaction starts
- the length of the interaction, in simulation clock ticks
- number of repeats of such interaction cycles
- the minimum length of any repeated interaction; the maximum is the length of the first occurrence
- the minimum gap, in simulation clock ticks, between repeats
- the maximum gap between repeats
- a probability of interaction (between 0 and 1), which determines how frequently (over the length of an interaction) will consume service events be present
- a behaviour specification for these interaction cycles, which points to a class of behaviour generated by a behaviour generator

It is evident that the behaviour class of the client associated with one interaction cycle specification and its repeated occurrences remains the same. However, there can be situations where the client changes from one behaviour

class to another for one or more interaction cycles. Such a change cannot be specified in one interaction cycle specification. It is, however, achievable by specifying more than one interaction cycles between the same client and the same server for the same application context but for different simulation times keeping in mind the semantic ordering of events.

Note that the event macro does not support generation of network failure events. We generate such events manually, prior to a simulation run.

4.3.4 Logging and statistics

Throughout the simulation run, all outputs from the simulator and the simulated application are written to one single log file. The level of detail in the log can be controlled, so that debug messages can be turned off. At the end of simulation run, a log analyser reads the log file and extracts relevant lines that are useful to generate statistical data about the simulation run. These extracted lines are written to another file, which is again parsed to generate a number of statistics files. These files are used by MATLAB scripts to draw graphs, which help visually present and conclude results about the simulation scenarios.

During simulation, there is also a miminal output on the console containing messages from the simulator, the event dispatcher and any exception from the simulated application along with a progress indicator. This output is not written to a file and is purely included for monitoring a simulation run.

4.4 Summary

In this chapter, we have discussed our own multi-threaded discrete event simulator written in Java, which is used to simulate the framework that we have proposed in chapter 3. The simulator generates various log files after a simulation run, some of which are directly readable by MATLAB. We use MATLAB scripts to visualise the simulation output data in the form of graphs. However, before describing our simulator in this chapter, we have discussed the relevant details of implementation of the entities, events and policies that constitute the framework.

We shall now investigate the simulation results in the following chapter.

5 Evaluation

Reason has always existed, but not always in a reasonable form.
Karl Marx
German economist & political philosopher (1818 - 1883)

5.1 Simulation objectives

The purpose of our simulations is to test the impact of changing client behaviour on client reputation within the proposed framework. We aim to illustrate, through our simulations, that across a range of client behaviours, the measured reputation tracks the values expected from the reputation-response policy proposed in chapter 3.

Due to the open-ended nature of our framework it is very difficult, if not impossible, to present simulation results without relying on the effects of example policies. This is evident from chapter 3 in which our framework specifies functional components, but the functions themselves are defined through policies. In our experiments, we choose not to simulate a behaviour analyser because behaviour analysis largely depends on an identity management infrastructure, which is beyond the scope of this thesis. It also depends on offline policies, such as the Acceptable Use Policy or the Terms of Service that are interpreted by a variety of *detectors*, such as email spam content filters, intrusion detection systems, amongst others. The evaluation of our framework does not necessitate simulating such detection systems, which act only as providers of behaviour input data to the rest of the framework.

Several offline privacy and security reasons have prevented us from obtaining exhaustive email server logs. Therefore, we simulate the framework with quantised behaviour values representing various classes of behaviour, which is the expected output from the behaviour analysis stage of our framework. In the absence of a behaviour analyser, the simulations can be exhaustive with synthetic data alone. This is because we are concerned with the quantised behaviour inputs, or with how those inputs are generated through policies applied to actors. Real data, such as email spam traces (if available), is not helpful with the generation of our simulation scenarios because without a pre-determined identity management infrastructure, we cannot translate real server logs to quantised behaviour. In addition, we also observe that in the absence of a known long-lived identity management infrastructure, it is impossible to infer per-actor behaviour.

Therefore, we choose to carry out our experiments using synthetic data generated to match population statistics of real data (e.g. total number of spam messages versus total number of messages containing virus). Use of synthetic data easily allows setting up scenarios that are possible in reality but rare to find.

The results that we present in this chapter follow from simulations conducted with 50 clients and 10 servers, where a number of (but not all) clients interact with the various servers. There are some common clients between the different servers. The simulation results graphically illustrate the reputation response to change of behaviour. Each such graph is the result of interaction between one server and one client. We also show the effect of the various global reputation interpretation policies, which are used as the starting points as well as re-adjustment points in the generation of global reputation. To clarify this, for example, when the policy to use the global reputation with highest confidence is used, the simulation shows the formation of local reputation with the highest confidence global reputation used for the starting values and re-adjustment values.

5.2 Simulation scenario: email delivery

Using various synthetic input data, a number of simulations have been performed on our model. With different variations of statistical parameters of the input data in the different simulations, we have reached statistically similar conclusions. Therefore, in this chapter, we choose to present only one simulation scenario which is closest to real world data in terms of available population statistics. In these sets of experiments, we will simulate the effect of our model in email delivery. For those experiments, we assume identities to be strong and long-lived.

There are two ways of identifying the "client" – a sender in email delivery. One is to use the sender's identity and the other is to use the sending email server's identity. In the latter procedure, it is possible to use the hop immediately before the receiving email server as a "client" identity, which means either the actual sending server or the last (in a chain, if present) store-forward mail server will be identified as a sender. For example, any store-forward message transmission agent (MTA) that would let emails pass through it will want to make sure that forwarding such emails will not damage its reputation and consistency of behaviour. In this way, the sending server or the store-forward MTA will act as a group identity for any sender that sends emails through it. The sending server's identity could be a single static IP address, a subnet or some other means that will ensure that it is long-lived. There are various other existing MTA identity checks ranging from reverse DNS verifications (e.g. see Barr 1996; Eidnes *et al.* 1998; Lottor 1987) to SMTP authentication (Siemborski & Melnikov, 2007).

On the other hand, a strong identity can be used to identify a sender, such as the public key or a sender (user) authentication on the network or the email server. It is important to note that depending on the choice of "client" identification, the "server" recording reputation will differ. With "client" being the sender (i.e. the user) itself, the "server" could either be the sending server or the receiving server. If the sending server's identification (or that of the last store-forward MTA) is used as the "client" then the "server" refers to the receiving server only. In the simulations presented in this chapter, we identify the "client" as the sender (i.e. user) instead of the sending server, and the "server" refers to the recipient server.

5.2.1 Actor (sender) classification

The plethora of available email server logs do not offer much in the way of developing per-actor (i.e. per-sender) behaviour statistics because of absence of a strong identity system in most SMTP servers. The same actor (i.e. actual person) could have multiple email addresses and could send emails from multiple hosts. A strong identity mechanism, similar to Verisign PIP (Verisign Labs, 2009) or 802.1x EAP (Congdon *et al.*, 2003) on a particular network could help identify an actor irrespective of the email addresses or hosts they are using. In the absence of such facilities, we will run our simulations based on per-class behaviour statistics instead of per-actor equivalent. We believe that the real users (i.e. senders) can be represented by a number of specific actor classes. We choose to define the following classes of actors for our experiments.

- Type 1: Generally good behaviour, occasionally bad (i.e. usual email sender profile)
- Type 2: Generally malicious (i.e. spammer)
- Type 3: Very good behaviour, this is essentially Type 1 with a very low proportion of bad behaviour (i.e. cautious email sender)
- Type 4: An equal mix of good and bad behaviour (i.e. malicious email sender, including spammers)

Having defined the actor classes, we define quantised values associated with behaviour to be able to generate synthetic input data. These values represent typical outputs from behaviour analysis, which serve as input behaviour data to build reputations on. There is no set scale for such quantised values, and are completely dependent on the policy-specific implementation of behaviour analysis. At this point it is worth noting that good behaviour is "produced" when, for a particular observed event (e.g. sending an email), a client is in agreement with the acceptable use policy. Depending on the type of agreement, good behaviour could be graded. Similarly, bad behaviour is produced when for such an observation, the client violates the policy. Depending on the type of violation, bad behaviour could be graded. We grade bad behaviour, in terms of degree of badness, in the range between neutral and worst behaviour. In this simulation scenario, we do not grade good behaviour in terms of varying degrees of goodness for simplicity, although we have observed that graded good behaviour would not have produced statistically dissimilar results for the purpose of evaluating our framework. We choose a set of values based on the

general bias that the worst behaviour is higher in its absolute value than the best behaviour. We set good behaviour to generate a value of 4.0, virus or malicious content to generate a value of −10.0, spam from anywhere between 0 and −5 with −5 being equivalent to a spam score of 1 in a scale of 0 to 1. In accordance with Senderbase (Cisco Systems, 2009b) classification, we also grade bad behaviour that is suspected to forge identities (e.g. spoofed sender IP address) at −2.

The actor classes are not quantised in terms of quantities of different types of behaviour but as we generate synthetic behaviour data, we ensure that the population statistics conform with global email behaviour spam statistics. We generate synthetic behaviour data, conforming with Senderbase (Cisco Systems, 2009b) statistics, and setup the experiments by selecting *interaction cycles* (see 4.2.3) between a list of clients and a list of servers. Per-actor behaviour is generated using a uniform distribution which conforms with the population statistics. To enable the effect of global reputation, the interaction cycles are generated in ways such that there are common sets of clients between any two participating servers. Apart from quantisation of typical behaviours, the actor classes also reflect their frequency of activity, i.e. *Type 1* and *Type 3* email senders, for example, have less activity per unit time while spammers (e.g. *Type 2*) have substantially more activity per unit time.

5.2.2 Impact of reputation on unimplemented service levels

In our experiments, we record the reputations developed per actor with various global reputation interpretation policies. We, however, do not implement service level cut-offs based on varying reputation. It is straightforward to interpret the reputations to understand how service levels could be impacted by varying reputation values. In a real world situation, if a client is denied a particular service level or denied service altogether, behaviour may still be recorded (although this depends on policy) as attempts so that local reputation can be formed. For example, a spam message may not be sent (i.e. service denied) but the fact that the sender attempted to send spam can be recorded. Additionally, in our simulations we do not simulate dynamically varying reputation response parameters. When repeating cycles of behaviour (e.g. from good to bad and to good again) are observed, which can be detected in behaviour analysis by comparing with an a short history of earlier

behaviour, it may be appropriate (depending on offline policies) to change reputation response parameters to somewhat neutralise the effects of the cycles. This can be done with the anticipation that repeating cycles of similar behaviour may actually have malicious intent. Instead of varying the reputation response parameters in our simulations, we analyse with fixed reputation response parameters and discuss at which point in (discrete) time a change of the parameters could yield optimal results.

From our experiment runs, we develop reputation-time graphs for the global reputation interpretation policies discussed in section 3.5.4. The results of ignoring global reputation are also compared. In addition, we demonstrate the effect of varying the time decay parameter. As the time decay parameter depends on the time scale chosen, we are not so concerned about its absolute numerical values. Further to that, we simulate various attacks against our model. Based on those results, we evaluate our model critically and make some recommendations.

5.2.3 Interaction with clients pertaining to various actor classes

To start with, we shall analyse the effects of using various global reputation interpretation policies for a number of interactions between various pairs of clients and servers. Each MATLAB-generated graph that we present for this purpose contains a sub-graph of behaviour versus discrete time; and a sub-graph of the recorded local reputations versus the same discrete time scale. In the reputation-versus-time sub-graphs, when ignoring global reputation altogether, reputation is plotted in red. Similarly, when selecting only the highest possible global reputation, the plot is in pink. The reputation value that corresponds to the reporting server in which the querying server has the highest confidence is plotted in green. When selecting the lowest possible reputation, the plot is in yellow. Finally, the plot is in blue for the selected reputation value that has the least deviation from the local value before re-adjustment. Depending on various parameters and event characteristics, many of these global reputation interpretation policies will result in almost the same reputation. By observing more plots presented later on, we will be able to draw some general characteristics of those reputation interpretation policies.

5.2.3.1 Type 1 client – usual email sender

In figure 5.1, we present a reputation comparison graph for a *Type 1* client interacting with a server. The parameters set for this experiment are: $\lambda = 0.01$, $\mu = 0.004$, $\epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on.

It is seen from the upper sub-plot (behaviour versus time) that the client exhibited generally good behaviour with intermittent slips into bad behaviour. The lower sub-plot illustrates the change of reputation over the same discrete time recorded for the upper sub-plot. The reputation-versus-time plot displays the change of reputation under the effect of various global reputation interpretation policies. These policies control how the local reputation value is re-adjusted after a global reputation query. The effect of time decay is particularly visible in the policy where the global reputation is ignored (i.e. the red plot). It is evident that despite having consistent good behaviour over certain periods of time, the reputation seems to dip over times of inactivity. Although some other policies exhibit such behaviour but with a relatively small time decay parameter, the decay is often eclipsed by the re-adjustment of the local reputation value to an usually higher value.

With faster time decay: A faster time decay can be observed for the same client under the same conditions by increasing the time decay parameter tenfold: $\epsilon = 0.0001$. This is illustrated in figure 5.2. Depending on the input behaviour, certain global reputation interpretation policies will look more conservative than others. After we present a number of further comparisons across clients of different classes, it will be clear that a combination of a number of interpretation policies, when varied dynamically at runtime, can yield optimal reputation responses to behaviour. With a faster time decay, we observe that the effect of the time decay is most felt when the global reputation is ignored (i.e. red plot).

Reputation-response parameters: In both cases, we observe that the client exhibits repeating cycles of generally good behaviour separated by periods of inactivity. Accordingly, figure 5.2 reveals that the time decay of reputation being rather high produces corresponding cycles in the reputation if the global reputation is ignored. Observing this trend that the client's inactivity

leads to substantial decay in reputation followed by an almost equal increase when the client interacts again, the server can reduce the time decay parameter so that the client is not deprived of any service level when it keeps repeating cycles of inactivity with no malicious intent.

Global reputation interpretation policies: The client in the above two experiments is consistent in its type of behaviour with all the servers that it interacts with. This is the reason why all the global reputation interpretation policies yield similar results, although those differences between the interpretation policies become amplified with faster time decay. In addition, global reputations, especially the ones with highest confidence and with least deviation, suggest fairly accurate picture for the client's behaviour.

5.2.3.2 Type 2 client – spammer

Let us now examine the reputation response for a *Type 2* client as presented in figure 5.3. The parameters set for this experiment are: $\lambda = 0.01$, $\mu = 0.004$, $\epsilon = 0.00001$ (i.e. slower time decay). Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. It is evident from the upper sub-plot (behaviour versus time) that the client exhibited generally bad behaviour with intermittent slips into good behaviour. The reputation response takes account of this behavioural history and the reputation steadily deteriorates inspite of intermittent good behaviour. However, it is clear that the local reputation response with the global reputation ignored (i.e. red plot) is more forgiving towards bad behaviour by periodically recovering during periods of no activity. A slower time decay would have made the reputation response less forgiving during periods of inactivity. The client, in this example, is consistently identified as malicious by other servers it interacts with. Hence, the global reputations interpreted with various policies are very close to each other.

Reputation-response parameters: In this experiment, we observe that the client exhibits repeating cycles of behaviour consisting of varying bad behaviour followed by short periods of inactivity and occasional slips of good behaviour. This could be driven by malicious intent, in order to regain reputation after exhibiting bad behaviour. Figure 5.3 reveals that the time decay of reputation and the varying bad behaviour produce cycles in the reputa-

tion when the global reputation is ignored. Observing the trend in which the client's inactivity leads to substantial decay (i.e. improvement, in this case) in reputation followed by reputation decrease when the client interacts again, the server can reduce the time decay parameter so that the client is not given the benefit of doubt every time there is a period of inactivity. If such a measure is taken by adjusting the time decay parameter then as the client continues to repeat this cycle, it will require even longer periods of inactivity for the client in order to gain any reputation. This will thwart the client from its malicious activity.

Global reputation interpretation policies: The client in the above experiment is consistent in its type of behaviour with all the servers that it interacts with. This is the reason why all the global reputation interpretation policies yield very similar results, which are fairly accurate representations (of reputation) corresponding to the client's behaviour profile.

5.2.3.3 Type 3 client – cautious email sender

Let us now examine the reputation response for a *Type 3* client as presented in figure 5.4. The parameters set for this experiment are: $\lambda = 0.01$, $\mu = 0.004$, $\epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. It is observed from the upper sub-plot (behaviour versus time) that the client exhibited no bad behaviour. The reputation response takes account of this behavioural history and the reputation stays positive all the time. However, due to bursts of activities mixed with longer periods of no activity, the local reputation tends to fall towards the positive default when the global reputation is ignored (i.e. red plot). On the other hand, the client proves to be quite consistent in the reputations it develops with other servers, which is why the various interpretations of global reputation tend to be in agreement.

More active client: In contrast, we now examine the reputation response for a more busy *Type 3* client as presented in figure 5.5. The parameters set for this experiment are: $\lambda = 0.01$, $\mu = 0.004$, $\epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. It is evident from the upper sub-plot (behaviour versus time) that the client exhibited very frequent good behaviour with very occasional slip into bad

behaviour. The reputation response takes account of this behavioural history and the reputation stays positive all the time, ignoring the very occasional bad behaviour. Due to the more frequent nature of good behaviour, time decay is less prominent compared to figure 5.4. However, time decay is still visible when the global reputation is ignored (i.e. red plot).

Reputation-response parameters: In these two experiments, we observe that the clients exhibit repeating cycles of very good behaviour with intermittent periods of inactivity. Figure 5.4 and figure 5.5 reveal that the time decay of reputation produces cycles in the reputation when the global reputation is ignored. This is particularly visible in figure 5.4 where the client is less active allowing the reputation to decay considerably during periods of inactivity. The server can compensate for such cycles by reducing the time decay parameter, particularly in the first case such that the client continues to enjoy similar service levels even after periods of inactivity followed by good behaviour, similar to that before the periods of inactivity.

Global reputation interpretation policies: The clients in the above experiments is consistent in their types of behaviour with all the servers that they interact with. This is the reason why all the global reputation interpretation policies yield similar results, although some differences are visible with a less active client in figure 5.4. The global reputations interpreted through the policy of highest confidence value selection and with the least deviation value selection act as reasonable suggestions for the client's behaviour profile.

5.2.3.4 Type 4 client – malicious email sender

Now, we shall examine the reputation response for a *Type 4* client as presented in figure 5.6. The parameters set for this experiment are: $\lambda = 0.01$, $\mu = 0.004$, $\epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. The upper sub-plot (behaviour versus time) illustrates that the client exhibited almost equal good and bad behaviour. The reputation response takes account of this behavioural history and the reputation fluctuates around values close to zero (neutral). The red plot and the blue plot (i.e. least deviation interpretation of global reputation) present the optimum reputation responses to the fluctuating behaviour.

Very active client: Here, we present the reputation response for a more busy *Type 4* client as presented in figure 5.7. The parameters set for this experiment are: $\lambda = 0.01, \quad \mu = 0.004, \quad \epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. It is observed from the upper sub-plot (behaviour versus time) that the client exhibited very frequent fluctuations between good and bad behaviour. The reputation response takes account of this behavioural history and the reputation fluctuates around values close to zero (neutral) although it is generally positive. The red plot provides the most optimum response to the fluctuating behaviour, similar to figure 5.6.

Reputation-response parameters: In these two experiments, the clients exhibit alternating good and bad behaviour. Figure 5.6 and figure 5.7 reveal that when the global reputations are ignored, the clients do not gain much from this cyclic behaviour alternating between good and bad in figure 5.4 where the client is less active allowing the reputation to decay during periods of inactivity. However, in the second case, the reputation response is not optimal given the frequent changes in behaviour from good to bad. Having detected the alternating behaviour, the server can adjust the reputation response parameters such that the build up of reputation due to good behaviour is slower than the fall of reputation due to bad behaviour. This means the parameter λ should be different in the two equations 3.10 and 3.11, say λ_p and λ_n respectively with $\lambda_p < \lambda_n$. This may call for averaging, e.g. a non-weighted mean: $\lambda = (\lambda_p + \lambda_n)/2$ or some other adjustments (e.g. different policies in global reputation age scavenging) when it comes to reporting reputations to the GRA with such parameters. In addition, the parameter μ in equation 3.13 should also be reduced to ensure that the client with negative reputation finds it harder to gain its reputation by attempting good behaviour. Alternatively, a different reputation response policy can be used to achieve similar objectives. If the client continues to alternate in its behaviour and the server continues to adjust the reputation response accordingly, the effect of the alternating behaviour will be eventually ironed out.

Global reputation interpretation policies: The clients in the above experiments are not entirely consistent in their type of behaviour with all the servers that it interacts with, especially in the first case. For this reason, the

global reputation interpretation policies yield disagreeing results, particularly in figure 5.6. While the global reputations with least deviation suggest fairly similar results as the one generated by ignoring global reputation, the highest confidence values could be rather misleading. This highlights the difficulties with the policy used in calculating confidence between servers.

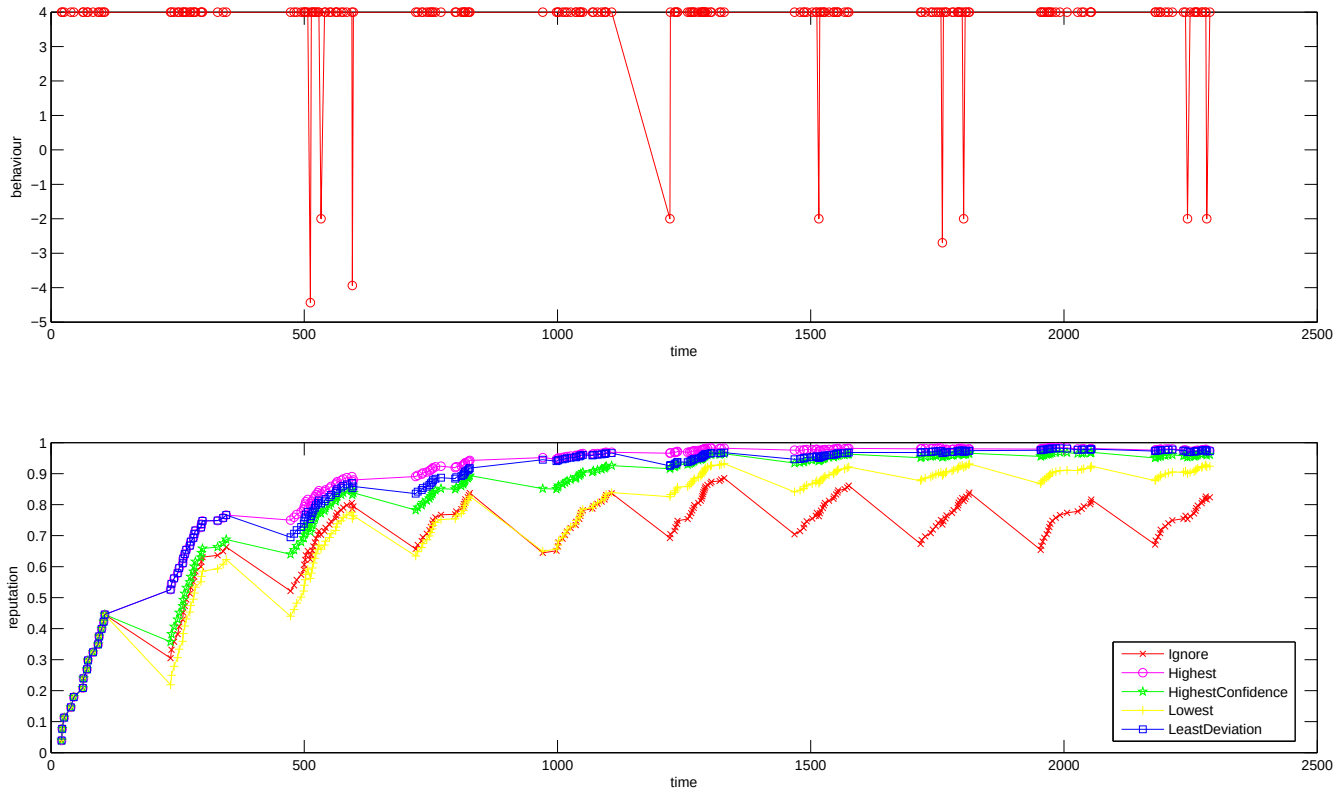

Figure 5.1: Reputation records for a Type 1 actor

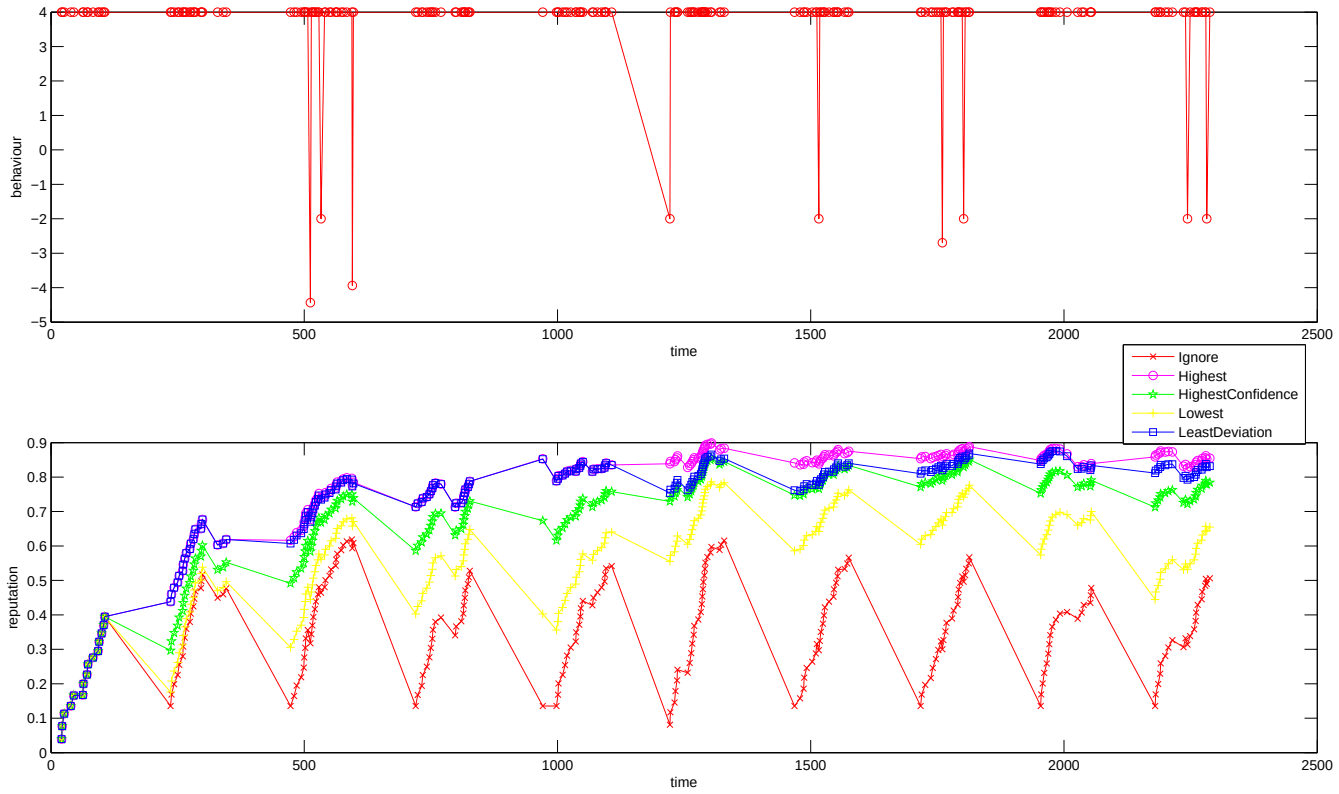

Figure 5.2: Reputation records for the same Type 1 actor as figure 5.1 with faster time decay

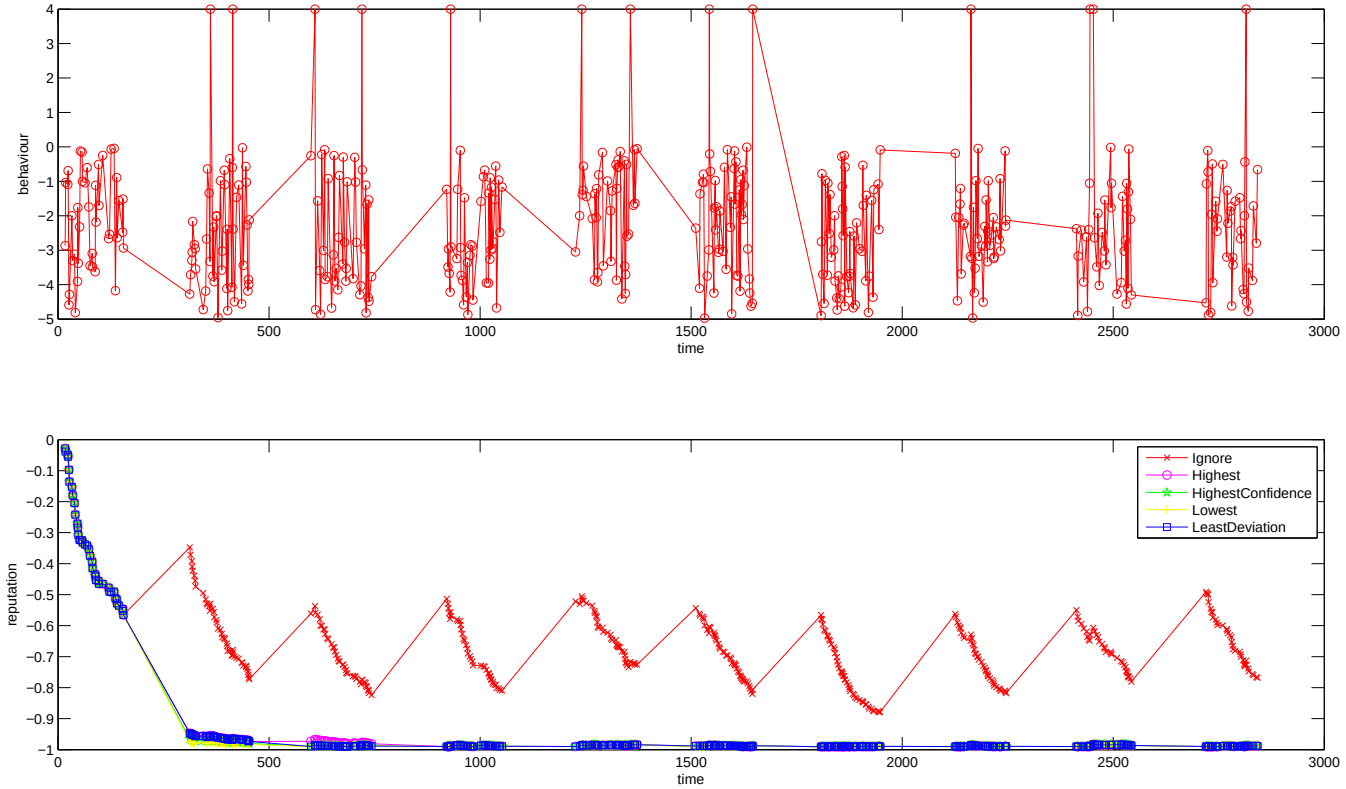

Figure 5.3: Reputation records for a Type 2 actor

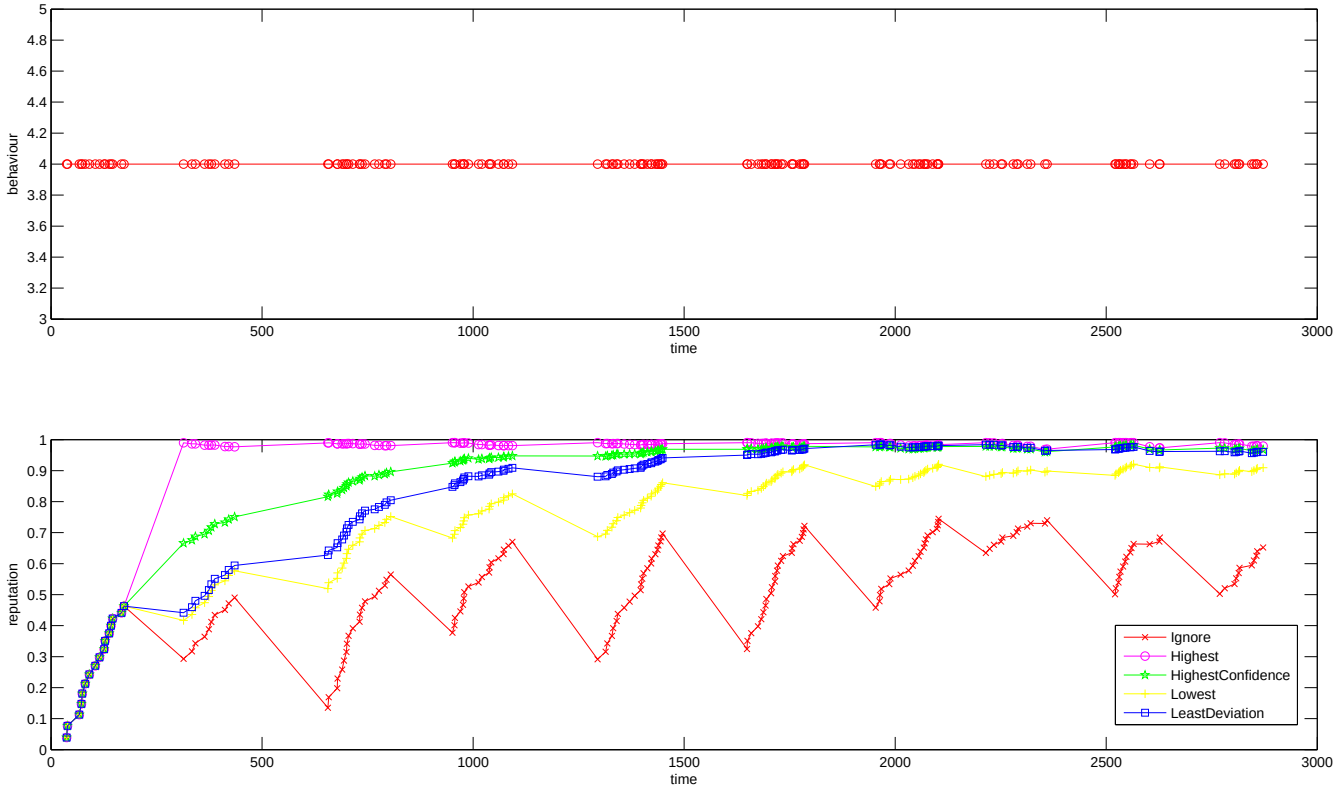

Figure 5.4: Reputation records for a Type 3 actor

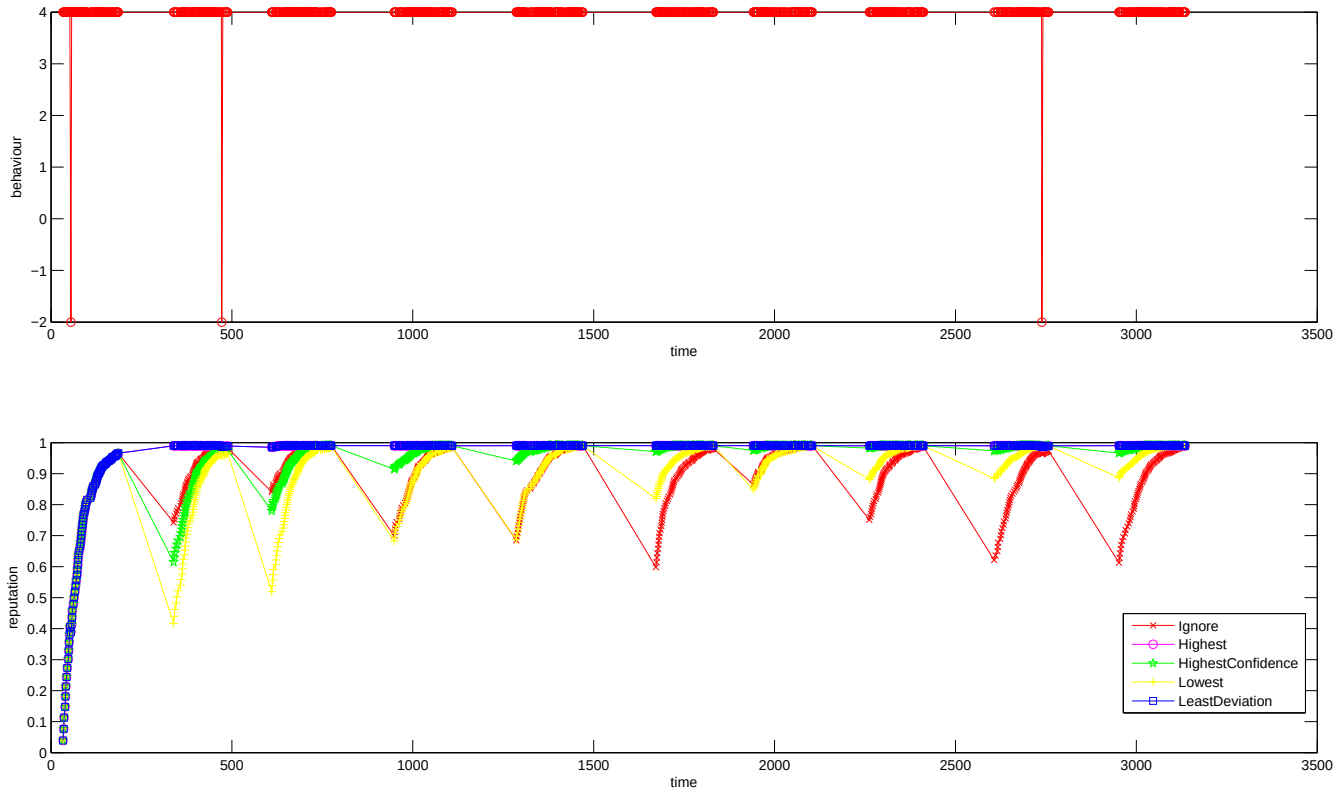

Figure 5.5: Reputation records for a more active Type 3 actor

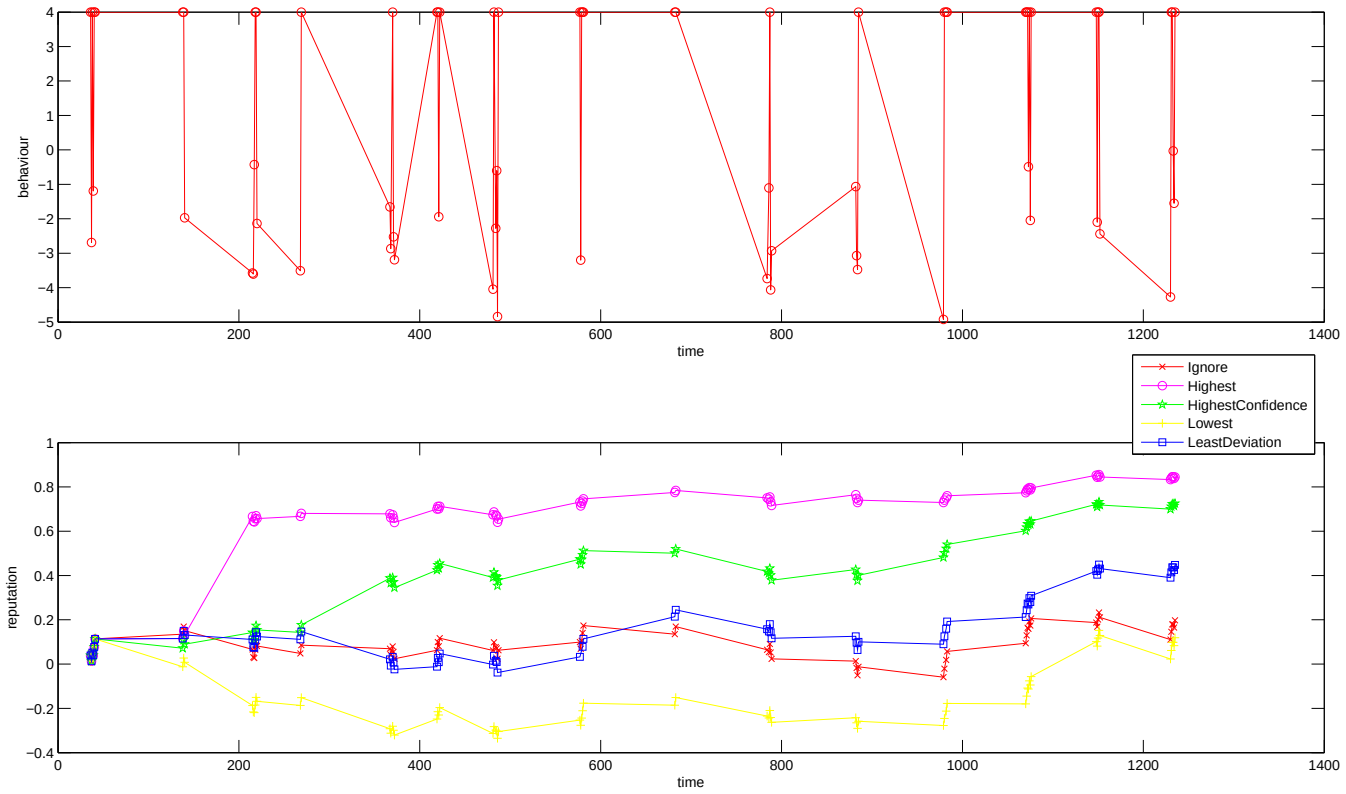

Figure 5.6: Reputation records for a Type 4 actor

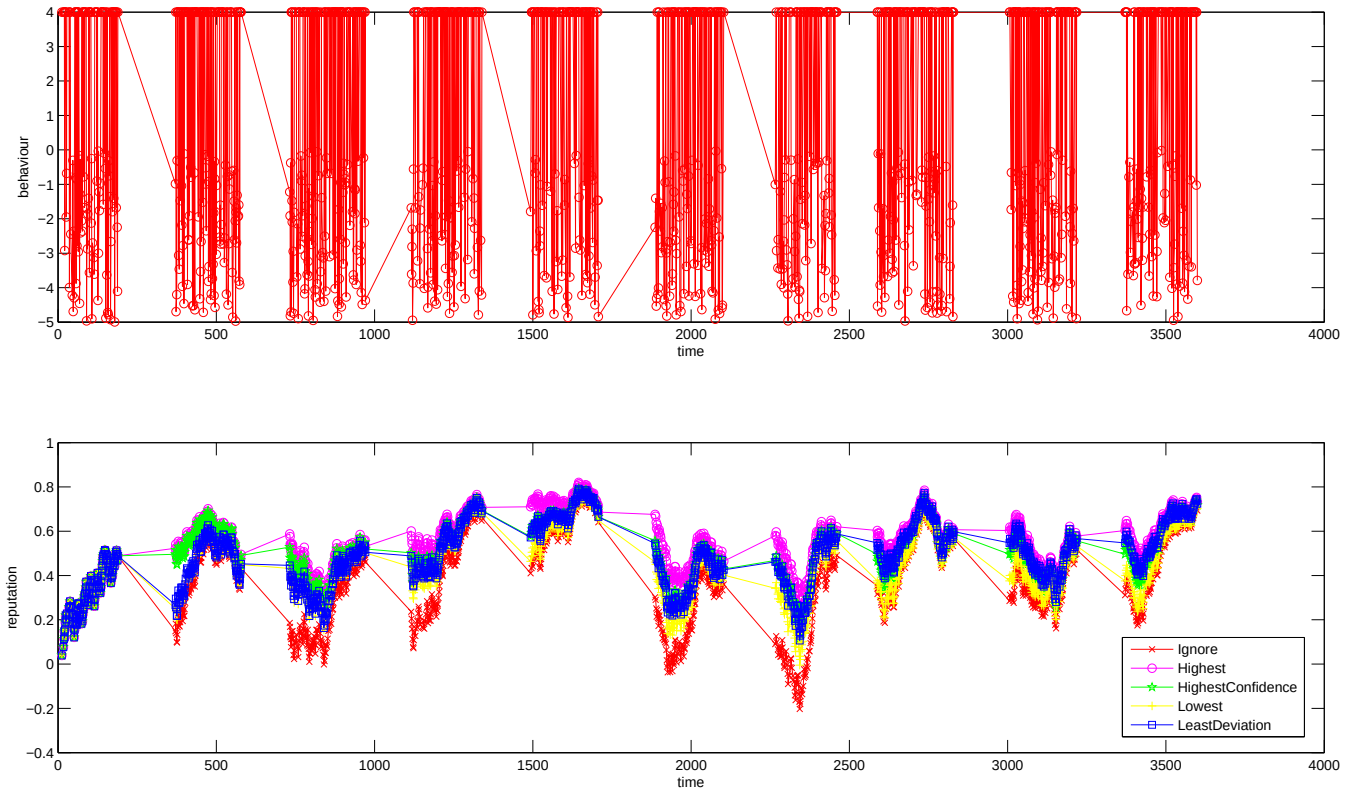

Figure 5.7: Reputation records for a more active Type 4 actor

5.2.4 Attacks on the model

In this section, we discuss and evaluate the system under a collection of attacks relevant to our model, following from our earlier discussion in section 3.6. We do not consider attacks on the identity infrastructure because it is beyond the scope of this thesis.

5.2.4.1 Reputation bootstrapping issue

When a particular server has not interacted with a certain client, the choice of initial reputation (i.e. "bootstrap" value) can have significant effects on the initial service level available to the client as well as its subsequent reputations, which can pave the way to attacks if the malicious client is unduly provided a high service level. In all the simulation results presented in the previous sections, we have used zero (i.e. neutral) as the initial reputation.

Amongst the policies simulated for interpretation of global reputation, the *least deviation* is irrelevant as at the bootstrapping stage there is no previous reputation to compare with in order to select a global reputation that deviates the least from the previously recorded local reputation. Selecting the highest or the lowest value of global reputation risks high positive or negative bias in the bootstrapping value. While the negative bias may make it unnecessarily difficult for an otherwise harmless client to get any good service level, a positive bias can open up the high reputation for misuse by a potentially malicious client. If the global reputation with highest confidence is selected, it is possible to initialise the reputation at a reasonable starting value depending on the level of confidence. However, in the potential absence of sufficient data to calculate confidence (e.g. if the server does not have comparable "common clients" list with other servers), the global reputation with highest confidence may not be usable. Therefore, given our example policies it is safest to initialise the bootstrapping value at zero because from that point, the simulated reputation response policy is fast enough to respond to any good or bad behaviour. However, this does mean that at least some level of service should be available for a zero reputation client so that the client can prove itself worthy of (or not) higher service levels with its behaviour.

5.2.4.2 Extortion, denial of reputation and ballot stuffing

In this attack, some "bad" servers can collaborate to damage the global reputation of a "good" client by reporting false bad reputation. Alternatively, some "bad" servers can collaborate to improve the reputation of a "bad" client by reporting false good reputation values. When a "good" server has built up a reasonably large "common clients" list with other servers, this attack is unlikely to affect the global reputation because of the confidence values between the "good" server and other servers with which it has similarity of opinion. However, in the extreme case, the "good" server could be faced with a situation where the majority of other servers with which it has a list of common clients collaborate to report false reputation. Another form of this attack is that an otherwise "good" client decides to misbehave with a particular server, thus making it unpredictable from its global reputation records.

In the afore-mentioned cases, this attack will be the same as only the "good" server experiencing behaviour from a client, which is completely different from the corresponding reputations reported by other servers. We simulate the worst case scenario: all servers with common clients list with the particular server differ in opinion about a particular client.

Let us analyse the reputation response for a *Type 2* client as presented in figure 5.8. The parameters set for this experiment are: $\lambda = 0.01$, $\mu = 0.004$, $\epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. This particular client is reported by most other servers as being good but it is evident from the upper sub-plot (behaviour versus time) that the client exhibited generally bad behaviour with intermittent slips into good behaviour. While the local reputation response is correct if the global reputation is ignored (i.e. red plot), it is evident that the server is misinformed if it takes into account the global reputation using any of the *least deviation*, the *highest* and the *highest confidence* global reputation values. We explained earlier that this is a highly unlikely case where either all servers have collaborated to report false reputation; or an otherwise "good" client has decided to act as a "bad" client with only one server. In most usual cases, there will be some other servers reporting similar reputation, which will lead to better uses of the confidence values for global reputations.

However, in this particularly unusual case, we observe a specific pattern in the reputation response when the server is misinformed by the global reputation it chooses to use. The pattern shows that every time the global reputation value is used to re-adjust the previously known local value by a substantial amount. This, we propose, could be used as an indicator of an attack. One of the ways (particularly applicable to our example policies) the server could avoid this attack is by choosing not to re-adjust its previously known local reputation value if the adjustment is relatively large (determined by policies on the server), and if such large re-adjustment pattern repeats itself every time the global reputation is looked up.

Let us analyse the reputation response for a *Type 3* client as presented in figure 5.9. The parameters set for this experiment are: $\lambda = 0.01$, $\mu = 0.004$, $\epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. This particular client is reported by all other servers as being bad but it is evident from the upper sub-plot (behaviour versus time) that the client exhibited very good behaviour with no slips into bad behaviour. It is also important to note that the frequency of interaction with a *Type 3* client is much lower than a *Type 2* client, which makes the effect of global reputation more felt than that observed in the previous example.

Although it is not so visible in the yellow plot, a distinct pattern in this example is that every time after the global reputation is used to re-adjust the local reputation to a negative value, the client behaviour is such that the newly re-adjusted reputation decreases towards less negative values. This is particularly visible in the blue and the green plots in the reputation-time graph. Although this pattern in itself is not conclusive enough (since it depends on the fact that the behaviour does not slip to negative at all), it still indicates that the trend indicated by global reputation is in sharp disagreement with the locally observed phenomena. As with the previous example, such a pattern could be used by the server to stop using the global reputation and in that case, the client will soon follow the trend set by the red plot in the reputation-time graph.

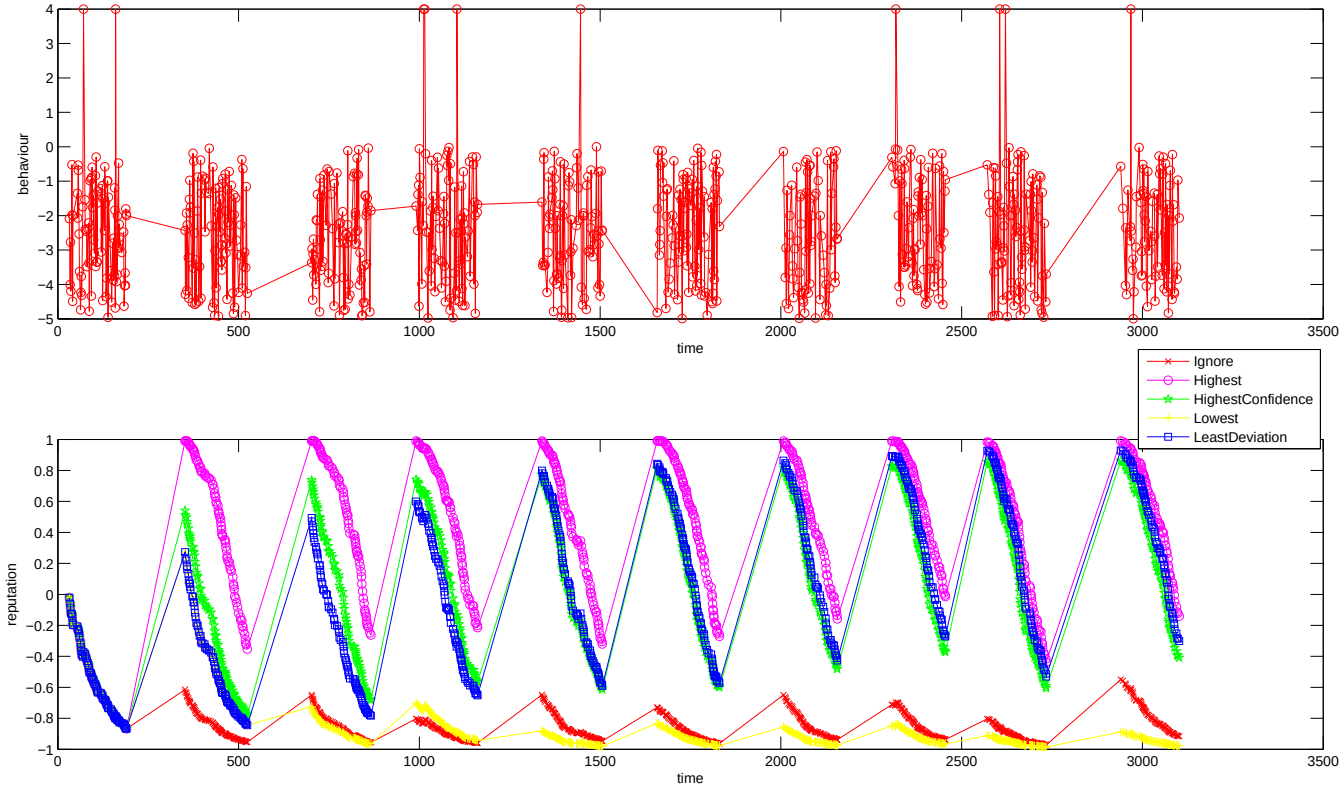

Figure 5.8: Reputation records for a misinformed Type 2 actor

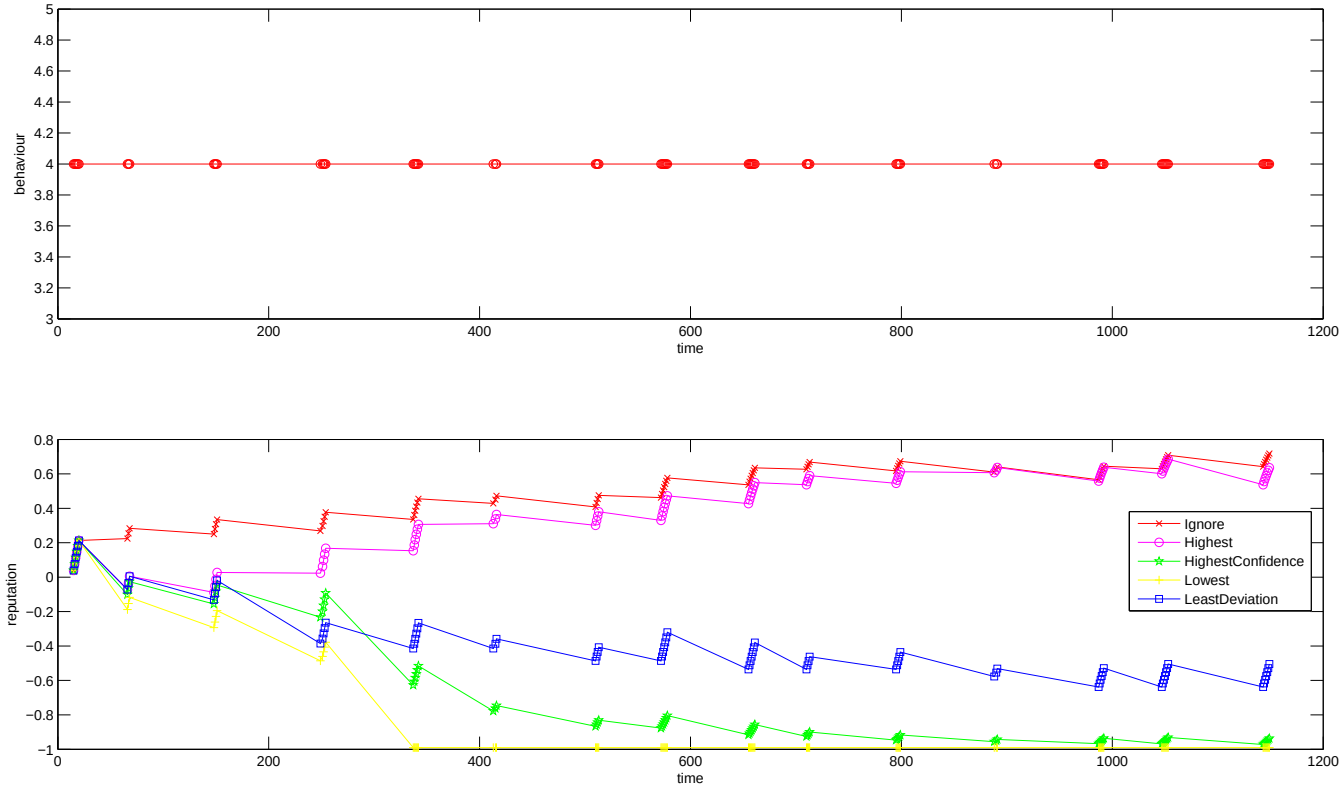

Figure 5.9: Reputation records for a misinformed Type 3 actor

5.2.4.3 Denial of Service affecting a server

Although, as mentioned earlier in chapter 3 that low-level Denial of Service (DoS) attacks are outside the remit of our model, we still present the side-effects on the reputation system caused by such DoS attacks. It is important to note that the application-level DoS attacks will be dealt with by our framework as if these were any other malicious client behaviour. The result of a DoS (or Distributed DoS) can either leave the server resources severely over utilised or completely exhausted. In the former case, it is possible for genuine clients to still continue to consume service although the quality of service is expected to suffer due to an on-going DoS attack. In the latter case, the server is unable to provide services to any legitimate (or even malicious) client and appears to be unavailable on the network. If the server's resources are severely over utilised and the DoS attack is at the application level, the server can promptly respond to the attack by developing local reputations based on behavioural history of the attacker (or multiple identities of the attacker in case of a DDoS). Once the client is identified as malicious, the server can deny service to it which will eventually put an end to the DoS attack.

We examine simulation results of a DoS attack on a server during service provision. In this simulation, the DoS attack results in complete failure of incoming and outgoing communication of the target server. The purpose of our simulation is to observe how the server copes with clients (both legitimate and malicious) after the DoS attack has ended.

First, we examine the same *Type 2* client and the server (presented in figure 5.3) while the server is under a DoS attack between times $t = 1944$ and $t = 2537$ as presented in figure 5.10. During this period, the server is unreachable and it cannot reach any external network resources either, including the GRA. The parameters set for this experiment are: $\lambda = 0.01$, $\mu = 0.004$, $\epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. It is evident from the upper sub-plot (behaviour versus time) that the client exhibited generally bad behaviour with intermittent slips into good behaviour. Furthermore, there is no behaviour recorded during the time when the server is under a DoS attack.

One of the main differences with the same client-server pair presented in figure 5.3 is the sharp rise of reputation at the end of the DoS attack. The second, rather implicit, observation is that since the end of the DoS attack, the result of the various global reputation interpretation policies are nearly the same. Examining the simulation logs, we observe that this is caused by a different problem: the server has not been able to query global reputations after the DoS attack. It has not been able to report global reputations either. Therefore, the reputation response graphs for various policies match closely as those policies were never applied because the reputation lookup from the GRA failed. Effectively, the server has relied on its local reputation responses without using the global reputation altogether. The reason why the graphs for the various interpretation policies are not exactly the same after the DoS is because of the slightly different reputation values that they have recorded before the DoS.

Now we examine a *Type 3* client with the same server under a denial of service attack between times $t = 1944$ and $t = 2537$ as presented in figure 5.11. The parameters set for this experiment are: $\lambda = 0.01$, $\mu = 0.004$, $\epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. It is observed from the upper sub-plot (behaviour versus time) that the client exhibited very good behaviour with very rare slips into bad behaviour. In addition, there is no behaviour recorded during the time when the server is under a DoS attack. The effect of the DoS attack is similar to that with the *Type 2* client.

Investigation into why, in both cases, the server was unable to request and report global reputation from and to the GRA concludes that this is because of the presence of:

1. an expired authorisation token which was not used due to the DoS attack to report global reputation; and
2. a request for global reputation without acknowledging a pending report of global reputation because of the DoS attack.

We also note that unusual decay of reputation during the DoS period, which can be used by a client with bad reputation to gain better levels of service after the DoS period. It is worthwhile to note that the DoS attack on one server

affects other servers only to the extent that the queried global reputation may not contain successful reports by the server under attack.

5.2.4.4 Denial of Service affecting the GRA

Now, we present the side-effects of a simulated DoS attack on the GRA. In this simulation, the DoS attack results in complete failure of incoming and outgoing communication of the GRA, which means all servers and all clients trying to reach the GRA will be affected. Unlike the server, which monitors client behaviour, the GRA does not have any built-in defense for any DoS attack, even if it is at the application-level. However, note that for a large-scale distributed implementation of the GRA (e.g. a Distributed Hash Table based lookup of global reputation), a complete knock-out of the GRA through DoS or Distributed DoS (DDoS) is highly unlikely. With an optimised level of redundancy in the global reputation data over a distributed implementation of the GRA, it is also very unlikely that an application level DoS (e.g. excessive queries or reports) will be able to render the GRA unusable.

We set up the DoS attack simulation for the GRA between the same times as we did in the DoS on the server example, i.e. between times $t = 1944$ and $t = 2537$. The DoS attack on the GRA affects all clients and servers. We choose the client-server pairs we have used in the simulation of the DoS attack on the server.

Let us first examine the reputation response to a *Type 2* client as presented in figure 5.12. The parameters set for this experiment are: $\lambda = 0.01, \quad \mu = 0.004, \quad \epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. It is clear from the upper sub-plot (behaviour versus time) that the client exhibited generally bad behaviour with intermittent slips into good behaviour. The reputation response graphs look fine until the time the DoS attack begins. After that, all the global reputation interpretation policies seem to yield similar results.

We notice a difference in this pattern by examining the case for the *Type 3* client as presented in figure 5.13. The parameters set for this experiment are: $\lambda = 0.01, \quad \mu = 0.004, \quad \epsilon = 0.00001$. Saturation closeness is 99%. Age scavenging (i.e. time decay) has been turned on. It can be observed from the upper

sub-plot (behaviour versus time) that the client exhibited generally very good behaviour with very rare slips into bad behaviour. The reputation response graphs look fine until the time the DoS attack begins. During the DoS attack, all the global reputation interpretation policies seem to yield similar results. However, after the DoS attack the results of various global reputation interpretations are not the same although their correctness could be questionable. This is different from what we see in the case of the *Type 2* client.

A closer inspection reveals that the reason why all the global reputation interpretation policies seemed to yield similar results in the case of the *Type 2* client is because the server failed to query global reputations after the DoS attack. This is because of unused authorisation tokens as we have seen in the case of DoS attack on a server. However, for the *Type 3* client, this is not the case. This is explained by investigating the events close to and during the DoS attack. The events corresponding to creating new authorisation tokens for this client and server pair are[1]:

```
1678    mkatok  email   CLI-29  SRV-4   1852
2021    mkatok  email   CLI-29  SRV-4   2196
2320    mkatok  email   CLI-29  SRV-4   2504
2637    mkatok  email   CLI-29  SRV-4   2813
```

The events corresponding to the global reputation lookup queries for the same client-server pair are:

```
1679    reqsvc  email   CLI-29  SRV-4
2022    reqsvc  email   CLI-29  SRV-4
2321    reqsvc  email   CLI-29  SRV-4
2638    reqsvc  email   CLI-29  SRV-4
```

Finally, the events corresponding to reporting global reputation for the same client-server pair are:

```
1852    putglo  email   CLI-29  SRV-4
2196    putglo  email   CLI-29  SRV-4
```

[1]See section 4.2.2 and table 4.1 for event specifications

```
2504    putglo  email   CLI-29  SRV-4
2813    putglo  email   CLI-29  SRV-4
```

It can be seen that during the DoS attack (i.e. $t = 1944$ to $t = 2537$), two authorisation token creation requests were made (i.e. at $t = 2021$ and $t = 2320$) which were followed by two global reputation lookup queries at $t = 2022$ and $t = 2321$. These were followed by the end of each interaction cycle and a corresponding global reputation report (i.e. at $t = 2196$ and $t = 2504$). Simulation log files reveal that all these requests were not honoured by the GRA, which was under a DoS attack. However, due to the spacing of events with relation to the DoS on the GRA, it is clear that the consecutive reputation lookup and reputation reports were not affected after the DoS attack has ended. This is because there were no unused authorisation tokens left.

For comparison, we present the reputation response graph in figure 5.14 for the aforementioned *Type 3* client and the same target server with the same experiment parameters with neither the server nor the GRA under any DoS attack. A close inspection will reveal that there are differences in the global reputation interpretation between the case without DoS and that (see figure 5.13) after the DoS on the GRA. This is due to global reputation values unreported by other servers, which caused a synchronisation problem although in this particular case, the synchronisation problems are not particularly prominent.

5.2.4.5 Improvements against DoS

Although various policy changes can be made to overcome the side-effects of a DoS attack on the server and on the GRA, we suggest the following framework level improvements and leave them for future work.

1. Servers should turn on time-decay (applicable according to policy) only after a service contract has ended cleanly, i.e. with a successful reporting of global reputation. Until such a clean end of service contract, servers can interpret inactivity from the client as temporary failure of communication including a DoS affecting the server itself. This means if a client wishes to improve (i.e. decay from negative) its bad reputation with no activity, it will have to tell the server that it is willing to end its service contract, which will trigger a global reputation report.

2. If the GRA receives a request from the client to create a new authorisation token for a particular server and a particular application context while one such token is already in possession with the server from an earlier request then the GRA should queue the creation of the new token. Following that, when the GRA receives a global reputation lookup request from the server using the new token, the GRA should ask the server to report its previously observed reputation about the client using the previous token before the reputation lookup request is honoured. This will ensure synchronisation of previously unreported global reputations.

The first recommendation applies only to the side-effect of a server under DoS while the second recommendation applies to either the server or the GRA or both under DoS.

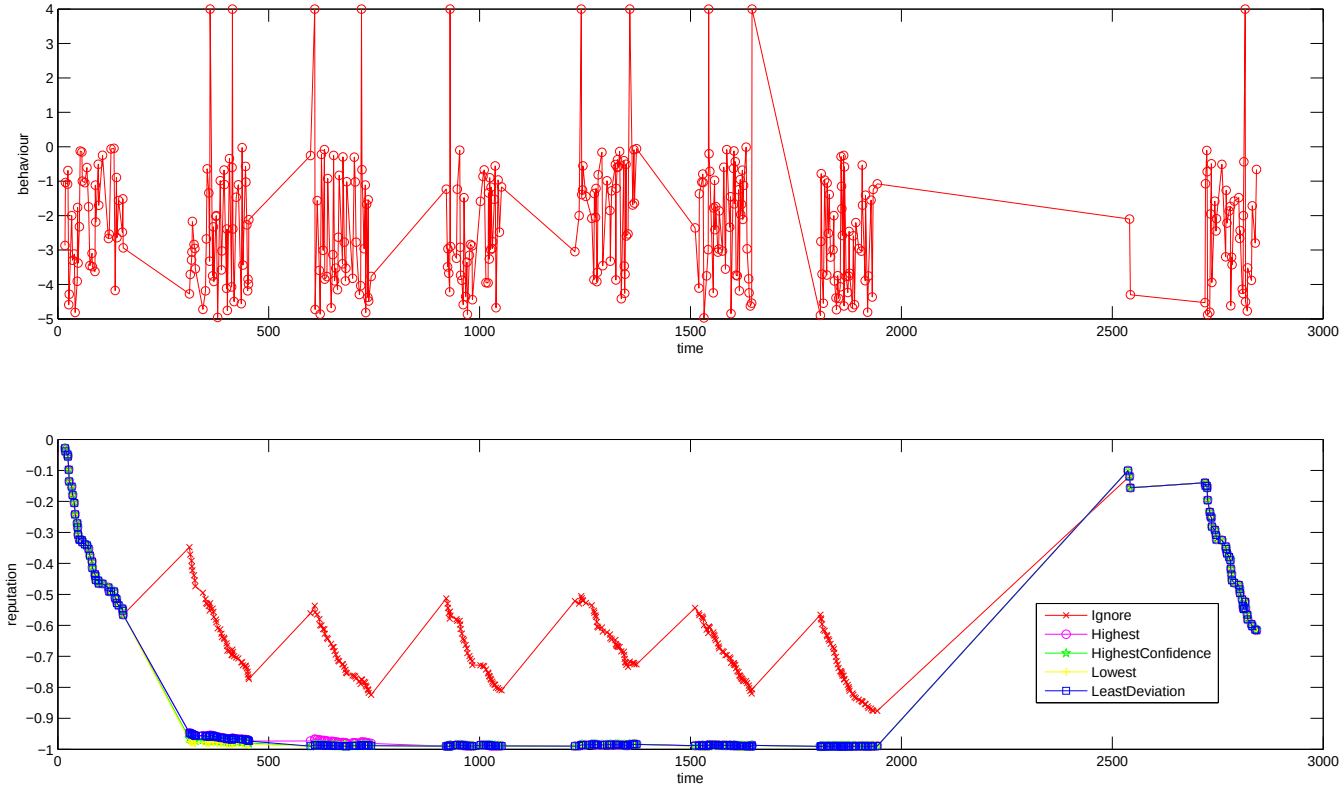

Figure 5.10: Reputation records for a Type 2 client with server under DoS

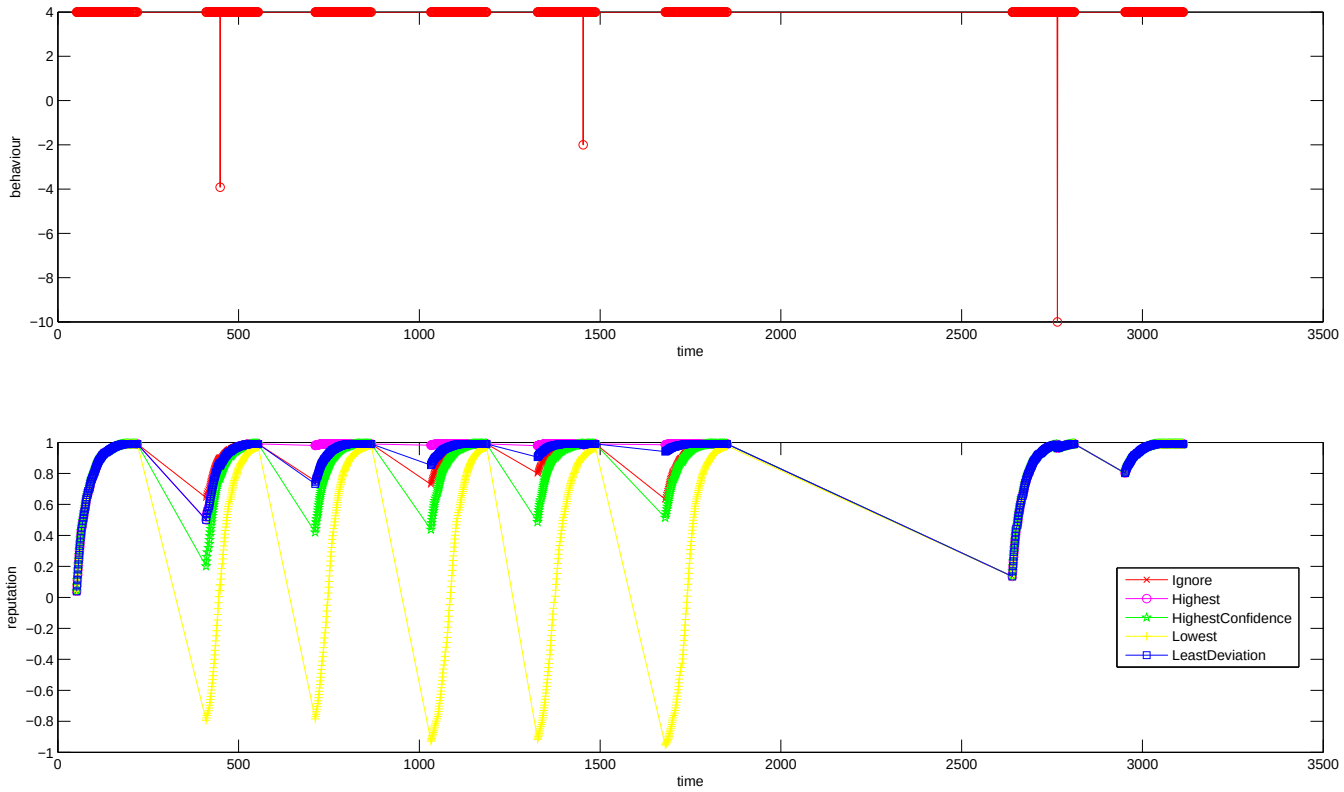

Figure 5.11: Reputation records for a Type 3 client with server under DoS

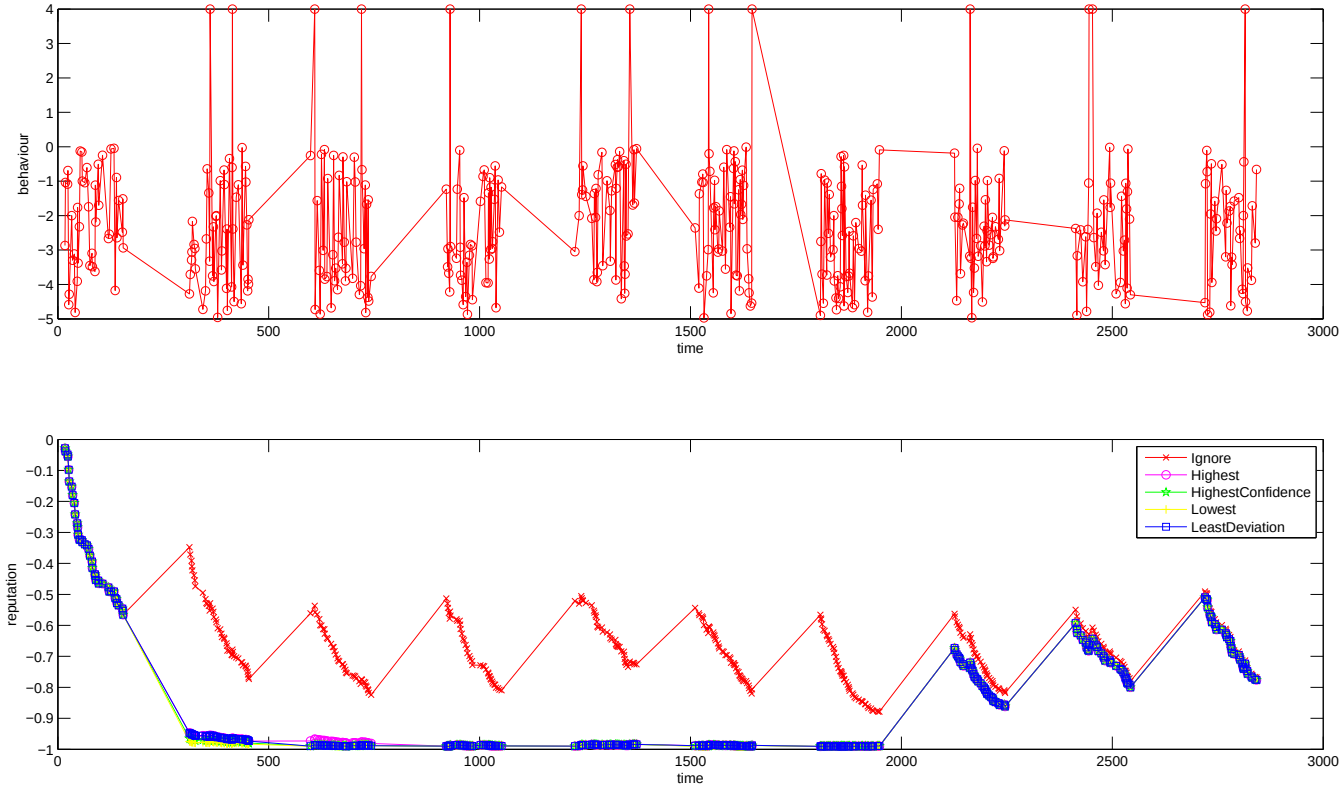

Figure 5.12: Reputation records for a Type 2 client with GRA under DoS

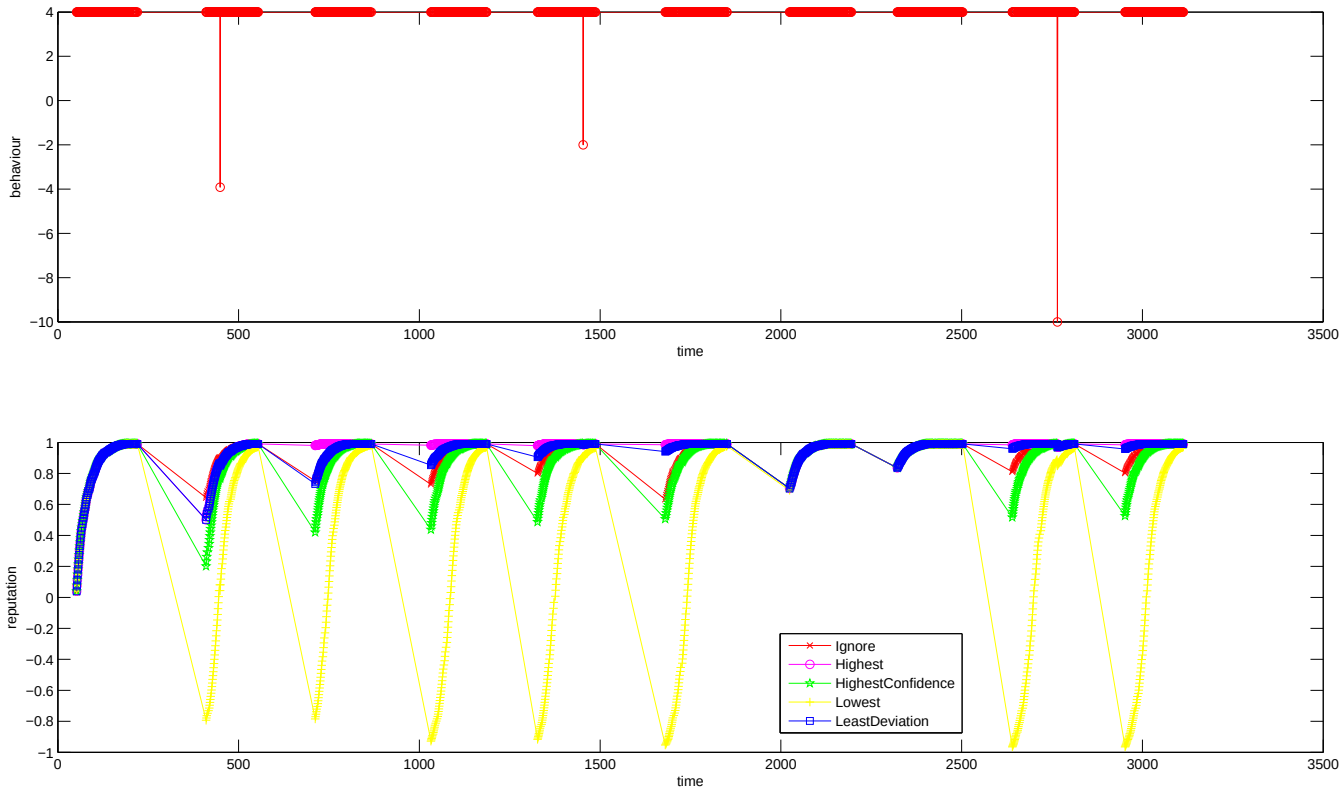

Figure 5.13: Reputation records for a Type 3 client with GRA under DoS

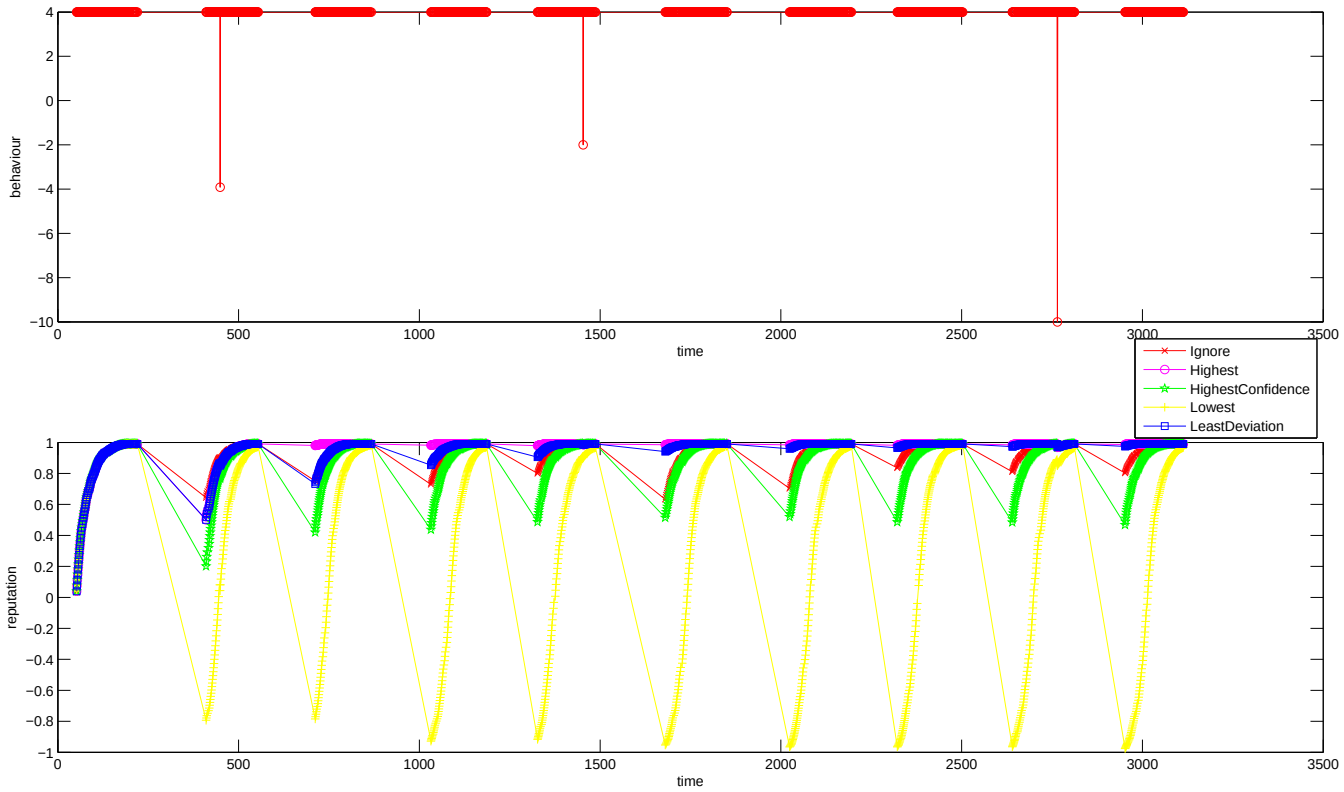

Figure 5.14: Reputation records for a Type 3 client without the effect of DoS

5.2.5 Effects of global reputation, improvements

By analysing the results of various global reputation interpretation policies presented in the earlier sections, we observe that the *HighestConfidenceReputation* policy can, in certain situations, reflect the ineffectiveness of using a statistical similarity measure (i.e. correlation coefficient, as a policy) to calculate the confidence between servers. The nature of the similarity measures ensure that slight differences in opinion from servers will be disregarded. This can be misleading however if the server depends only on global reputation to re-adjust its local reputation values.

One of the ways to counter-balance the adverse effects of using global reputation is to not use it when the difference between local observation and the suggested global reputation value is significantly large, e.g. figure 5.8. Another possible improvement is that the server calculates two local reputation values for the same client, i.e. with and without the use of global reputation. When the vectors representing these two values become significantly dissimilar, it acts as an alert to the server that the global reputation is likely to be misleading.

5.2.6 Levels of service

From the experiments in the previous sections, we have observed the varying reputations of clients corresponding to their behaviour profiles. Following from § 3.4.1.4 (where we presented the representation of service levels), a server can associate a particular level of service with a certain band of observed reputation. When the client achieves a reputation that falls in the specified band, it will be provided with the particular service level. Depending on security measures, policies and other requirements, such bands could be quite narrow to ensure finer control over any change of reputation of the client. Beyond the scope of the reputation responses, levels of service may also be dictated by specific service contracts that the client is bound by, such as the class of the client whereby a "premium" client is entitled to a higher level of service than a "standard" client. If the highest level of service that a client A is entitled to is higher than that of a client B, reputations will need to be capped accordingly such that no matter how well client B behaves, its corresponding reputation will not imply a level of service higher than that of client A when client A exhibits equally good behaviour. Such provisions can be made by

modifying reputation response policies without having to make any changes to the framework itself.

In the context of our example scenario (i.e. email delivery), such levels of service could correspond to various limits on the number of emails a sender is allowed to send per unit time, the size of the email, the number of recipients, etc. Senders with higher (and positive) reputations will be subject to fewer restrictions. Senders with negative reputations may be able to send emails only if such actions constitute good behaviour; and if their reputations are lower than required for service levels then they will have to wait for periods of no activity to regain their reputations.

5.3 Summary

Using the discrete event simulator and policy implementations described in chapter 4, in this chapter we have evaluated our framework. Email delivery is chosen as our simulation scenario. We have generated synthetic input behaviour data conforming with the population statistics available from Cisco Systems 2009b. We have defined a number of actor classes representing the different email sender profiles, such as:

- Type 1: Generally good behaviour, occasionally bad (i.e. usual email sender profile)
- Type 2: Generally malicious (i.e. spammer)
- Type 3: Very good behaviour, this is essentially Type 1 with a very low proportion of bad behaviour (i.e. cautious email sender)
- Type 4: An equal mix of good and bad behaviour (i.e. malicious email sender, including spammers)

Comparing the effect of the various global reputation interpretation policies, we have simulated the reputation response of the framework to changing behaviour from actors with the profiles listed above. We have also presented simulation results illustrating the effect of time decay of reputation. Further to that, we have simulated the effects and the side-effects of the relevant attacks described in § 3.6. Finally, through analysis of the simulation results, we have also proposed some improvements to the framework and policies.

6 Conclusion and Future work

The only good is knowledge and the only evil is ignorance.
Socrates
Greek philosopher in Athens (469 BC - 399 BC)

The main aim of this work has been to build a generalised privacy-preserving framework for developing and sharing the reputation of network clients based on their behaviour profiles.

In this thesis, our study started with looking at how the open architecture of the Internet which has been the key factor behind its success also made it vulnerable to abuses of services provided on the Internet. We have categorised some of the types of attacks on services offered over a network, and their defences. Then we started focusing on the use of behavioural history of clients, as one of the possible defences against various types of attacks.

We then surveyed a number of different approaches to defences against abuse of network services, followed by a detailed analysis of the motivational and related work (e.g. Allman *et al.* 2005b; Wei & Mirkovic 2007) in the use of behavioural history and client reputation. We observed that instead of building defences against every type of attack, a generalised framework augmenting existing defences and using behavioural history and client reputation can be useful to protect Internet services.

6.1 Summary of contributions

The main contributions of this thesis are:

- A generalised privacy-preserving policy-independent framework is presented for developing and sharing client reputation based on behavioural history.
- Although the general theme in this thesis has been in favour of long-lived strong identities, we have also discussed how weak and re-usable identities such as IP addresses can also be used with our framework.
- The framework allows collecting behaviour data from various monitoring mechanisms, and thus is not restricted only to packet-level analysis.
- The framework empowers servers with the means that enable them to vary levels of service depending on the reputation of clients, thus restricting service to malicious clients while prioritising service to clients with higher reputations.
- In addition to the locally developed reputations of clients, servers are able to obtain global "views" about the reputations of their clients.
- The global client reputations can be interpreted in various ways by the servers; in particular through means of their confidence between servers, which can be a measure of similarity of their opinions of the clients.
- Various policies can be fit into the framework, thus keeping the framework open-ended and flexible.
- Sharing of reputations, instead of behavioural history (e.g. Allman *et al.* 2005b), enables privacy for clients.

6.2 Future work

Further to the improvements discussed following our evaluation in chapter 5, we propose the following directions of future work.

6.2.1 Extensions to identity infrastructure

Identity clustering: We have seen in (Wei *et al.*, 2006) that methods of clustering are used based on behaviour profiling of hosts. We envisage that such clustering could be used in the behaviour analysis stage to group identities with similar behaviour patterns. Those groups could be used as virtual identities. The servers develop and share reputations of such virtual groups of

identities instead of individual ones. The algorithm 3.1 for reporting of global reputation will need to be adapted to cater for virtual identities. The use of identity clustering is useful in restricting email botnets or similar malicious Sybil identities.

Pseudonymous identities: The heterogeneity of identity schemes (e.g. IPv4, IPv6, PKI) leads to the problem that the implementation of certain policies in our framework will need to be adjusted based on the identity scheme in use. Also, the extremely short-lived nature of identities in some applications (e.g. mobile ad-hoc networking) or high rates of churn (e.g. peer-to-peer networks) also necessitate a unified, re-usable but long-lived identity mechanism. The development of a pseudonymous generalised identity infrastructure that will work with the framework is a possible future step. A proposal for a pseudonymous identity infrastructure exists in (Wakeman *et al.*, 2007).

6.2.2 Policy-specific and implementation-specific extensions

Capping of reputations: Very often, clients belong to classes of users which are bound by different service contracts and hence are eligible for different service levels even if the reputations developed from their behaviour profiles may be the same. Our example policy described in § 3.5 does not cater for such a requirement. It needs to be factored into policy specifications for local reputation response.

Policy specification language: If the existing policy specification languages prove to be inadequate in specification of policies in the behaviour analysis stage, it will be the avenue of future work for us to produce extensions to a existing policy specification language, e.g. Ponder (Damianou *et al.*, 2001) to work with the requirements of our framework.

6.2.3 Framework specific extensions

Application context: In chapter 3, we have specified that our framework is context-aware. Future work may be done on the use of the application context. For example, if there will be a globally available list of integer values specifying various application contexts; or if servers are allowed to use free text to specify application contexts while global reputation queries may use

natural language techniques to identify similarities between the meanings of the application context. Although in this thesis, we show that application contexts are independent of each other, an interesting question to explore is whether there can be any dependence between certain application contexts and how can this be addressed.

Reputation feedback mechanism: We have seen in chapter 3 that a policy-specific behaviour analyser may be designed to detect repetitions of behaviour, which may be indications of malicious activity. A feedback mechanism of the locally developed reputation or interpreted global reputation may also provide sufficient information to the behaviour analysis stage, which can further quantify accurately a particular behaviour observation. In addition, such a feedback could also be useful to notify the various behaviour monitoring sources to use means in their capacity to thwart malicious activities. For example, upstream routers may restrict certain clients if bad reputation feedback is received for them. This is particularly useful for stopping malicious clients that intend to mount DoS attacks.

Size of global reputation query response: The size of the result of the global reputation query in the current framework is relatively large, and is related to the number of active clients and servers in the network. To help scaling up the framework to cater for very large networks and to optimise the network footprint of the reputation query response, it is necessary to select a subset of the set of available global reputation responses. The means of this subset selection can be added to the framework as a policy.

6.2.4 Further simulation

Simulation of policies for service levels: We have simulated the reputation response to changing behaviour in chapter 5, and described the impact on service levels. In future work, it would be interesting to evaluate the effect of policies for applying various service levels to the reputation responses.

6.3 Closing remarks

The objective of this research was to investigate the use of behavioural history and client reputation in a generalised framework, which could be used to

restrict malicious clients from abusing network services. Our simulation results prove the feasibility of this generalised framework, which is not necessarily tied to low-level packet analysis. In the time taken to complete this thesis, there have been a growing number of systems that have started to use application-level behaviour analysis along with packet analysis to form behavioural history of clients, and hence aid better network intrusion detections, e.g. (Arbor Networks, 2009; Cisco Systems, 2009b; Riverbed, 2009). In particular, (Cisco Systems, 2009b) attempts to develop reputations of Internet hosts from a very large sample of Web and email traffic that is analysed everyday. This trend of commercial work using behavioural history and client reputation in this way justifies our original research view that a generalised framework for developing and sharing client reputation is a bona fide way to tackle problems of abuse of network services.

References

Aagedal, J. Ø., & Milosevic, Z. 1999. ODP enterprise language: UML perspective. *Pages 60–71 of: Third International Enterprise Distributed Object Computing Conference.*

Abadi, M., Birrell, A., Burrows, M., Dabek, F., & Wobber, T. 2003. Bankable Postage for Network Services. *Proceedings of the 8th Asian Computing Science Conference, Mumbai, India, December.*

Aberer, K., & Despotovic, Z. 2001. Managing Trust in a Peer-2-Peer Information System. *Pages 310–317 of: Proceedings of the 10th International Conference on Information and Knowledge Management.* ACM Press.

Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., & Thomas, M. 2005a. Domainkeys identified mail (DKIM). *Internet Engineering Task Force (IETF) Draft, July.*

Allman, E., Callas, J., Delany, M., Libbey, M., Fenton, J., & Thomas, M. 2007 (May). *DomainKeys Identified Mail (DKIM) Signatures.* RFC 4871 (Proposed Standard). Updated by RFC 5672.

Allman, M., Blanton, E., & Paxson, V. 2005b. An Architecture for Developing Behavioral History. *In: Proceedings of the Workshop on Steps to Reducing Unwanted Traffic on the Internet.*

Anderson, J. P. 1980. *Computer Security Threat Monitoring and Surveillance.* Tech. rept. James P. Anderson Company, Fort Washington, Pennsylvania.

Anderson, T., Roscoe, T., & Wetherall, D. 2004. Preventing Internet denial-of-service with capabilities. *ACM SIGCOMM Computer Communication Review*, **3**4(1), 44.

Arbor Networks. 2009. *Arbor Peakflow X: Enterprise Network Monitoring, Protection and Visibility.* `http://www.arbornetworks.com/en/peakflow-x.html`.

Arnold, K., Gosling, J., & Holmes, D. 2005. *The Java (TM) Programming Language.* Addison-Wesley Professional.

Axelsson, S. 2000a. Intrusion detection systems: A survey and taxonomy. *Chalmers University of Technology, Dept. of Computer Engineering, Göteborg, Sweden, Technical Report*, 99–15.

Axelsson, S. 2000b. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security*, **3**(3), 186–205.

Back, A. 2002. *Hashcash.* `http://www.hashcash.org/`.

Balakrishnan, H., & Karger, D. R. 2004. Spam-I-am: A Proposal for Spam Control using Distributed Quota Management. *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets), San Diego, CA, November.*

Barr, D. 1996 (Feb.). *Common DNS Operational and Configuration Errors.* RFC 1912 (Informational).

Basu, A., Wakeman, I., & Chalmers, D. 2007. A Behavioural Model for Consumer Reputation. *(Poster) Proceedings of the 2nd Intl. Workshop on Self-Organising Systems, The Lake District, UK.*

Basu, A., Wakeman, I., Chalmers, D., & Robinson, J. 2008. A Behavioural Model for Client Reputation. *Proceedings of Trust in Mobile Environments (workshop in IFIPTM 2008), Trondheim, Norway.*

Birrell, A., Goldberg, A., Manasse, M., Mironov, I., & Wobber, T. 2009. *Penny Black – Microsoft Research.* `http://research.microsoft.com/en-us/projects/PennyBlack/`.

Blanc, X., Gervais, M. P., & Le-Delliou, R. 1999. Using the UML language to express the ODP enterprise concepts. *Pages 50–59 of: Third International Enterprise Distributed Object Computing Conference.*

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., & Orchard, D. 2004. *Web Services Architecture.* `http://www.w3.org/TR/ws-arch/`.

Breese, J. S., Heckerman, D., & Kadie, C. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Learning*, **9**, 309–347.

Brickley, D., & Miller, L. 2004. *FOAF Vocabulary Specification*. `http://xmlns.com/foaf/spec/`.

Carrara, E., & Hogben, G. 2007. Reputation-based Systems: a security analysis. *Position paper 2 in European Network and Information Security Agency*.

Ceglowski, J. S. M., & Schachter, J. 2004. *List-of-all-Friends (LOAF)*. `http://loaf.cantbedone.org`.

Cerf, V. G., & Kahn, R. E. 1974. A Protocol for Packet Network Interconnection. *IEEE Transactions on Communications*, **2**2(5), 637–648.

Chung, A., Tarashansky, I., Vajapeyam, M., & Wagner, R. 2002. SpamStrangler: A Chord-Based Distributed Spam Detection Tool. *MIT 6.824 Fall 2002 Project Reports*.

Cisco Systems. 2009a. *Cisco IronPort Reputation Filters*. `http://www.ironport.com/technology/reputation_filters.html`.

Cisco Systems. 2009b. *Cisco IronPort Senderbase Security Network*. `http://www.senderbase.org/`.

Cisco Systems. 2009c. *SpamCop.net*. `http://www.spamcop.net/`.

Cloudmark. 2009. *Cloudmark Message Security*. `http://www.cloudmark.com`.

Congdon, P., Aboba, B., Smith, A., Zorn, G., & Roese, J. 2003 (Sept.). *IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines*. RFC 3580 (Informational).

Corradi, A., Montanari, R., Lupu, E., Sloman, M., & Stefanelli, C. 2000. A flexible access control service for Java mobile code. *Pages 356–365 of: The 16th Annual Conference on Computer Security Applications*.

Cramer, E. 2002. The Future of Wireless Spam. *Duke L. & Tech. Rev.*, **2**002, 21–28.

Crocker, D. 2009 (Aug.). *RFC 4871 DomainKeys Identified Mail (DKIM) Signatures – Update*. RFC 5672 (Proposed Standard).

Crocker, S. D. 1970 (Mar.). *Protocol Notes.* RFC 36. Updated by RFCs 39, 44.

Damiani, E., De Capitani Di Vimercati, S., Paraboschi, S., & Samarati, P. 2003. Managing and Sharing Servants' Reputations in P2P systems. *IEEE Transactions on Knowledge and Data Engineering*, **1**5(4), 840–854.

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P., di Tecnologie dell'Informazione, D., & Crema, I. 2004. P2P-based collaborative spam detection and filtering. *Pages 176–183 of: Proceedings of Fourth Intl. Conference on Peer-to-Peer Computing, 2004.*

Damianou, N., Dulay, N., Lupu, E., & Sloman, M. 2001. The ponder policy specification language. *Lecture Notes in Computer Science*, 18–38.

Debar, H., Becker, M., & Siboni, D. 1992. A neural network component for an intrusion detection system. *Pages 240–250 of: 1992 IEEE Computer Society Symposium on Research in Security and Privacy, 1992. Proceedings.*

Delany, M. 2007 (May). *Domain-Based Email Authentication Using Public Keys Advertised in the DNS (DomainKeys).* RFC 4870 (Historic). Obsoleted by RFC 4871.

Douceur, J. R. 2002. The Sybil Attack. *In: Proceedings for the 1st Intl. Workshop on Peer-to-Peer Systems (IPTPS.02).* Springer.

Droms, R. 1997 (Mar.). *Dynamic Host Configuration Protocol.* RFC 2131 (Draft Standard). Updated by RFCs 3396, 4361, 5494.

DShield. 2009. *DShield – Cooperative Network Security Community.* `http://www.dshield.org/`.

Duan, Z., Yuan, X., & Chandrashekar, J. 2006. Constructing Inter-Domain Packet Filters to Control IP Spoofing Based on BGP Updates. *In: IEEE Infocom.*

Dwork, C., & Naor, M. 1995. *Pricing Via Processing Or Combatting Junk Mail.* Weizmann Institute of Science, Dept. of Applied Mathematics and Computer Science.

Dwork, C., Goldberg, A., & Naor, M. 2003. On Memory-Bound Functions for Fighting Spam. *In: Proceedings of Advances on Cryptology (CRYPTO 2003), Santa Barbara, CA, USA, August*. Springer.

Ebel, H., Mielsch, L. I., & Bornholdt, S. 2002. Scale-free topology of e-mail networks. *Physical Review E*, **6**6(3), 35103.

Eddy, W. 2007 (Aug.). *TCP SYN Flooding Attacks and Common Mitigations*. RFC 4987 (Informational).

Edwards, J. 2008. *False Positives Equal Lost Business*. `http://www.itsecurity.com/features/false-positives-022808/`.

Egevang, K., & Francis, P. 1994 (May). *The IP Network Address Translator (NAT)*. RFC 1631 (Informational). Obsoleted by RFC 3022.

Eidnes, H., de Groot, G., & Vixie, P. 1998 (Mar.). *Classless IN-ADDR.ARPA delegation*. RFC 2317 (Best Current Practice).

Enck, W., Traynor, P., McDaniel, P., & La Porta, T. 2005. Exploiting open functionality in SMS-capable cellular networks. *Page 404 of: Proceedings of the 12th ACM Conference on Computer and Communications Security*. ACM.

Erdos, M., & Cantor, S. 2002. Shibboleth Architecture Draft v05. *Internet2/MACE*, **2**.

European Union. 2001. *Data protection: "Junk" e-mail Costs Internet Users EUR 10 billion a year worldwide – Commission study*. `http://europa.eu/rapid/pressReleasesAction.do?reference=IP/01/154`.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. 1999 (June). *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). Updated by RFC 2817.

Flux Group. 2000. *Emulab – total network testbed*. `http://www.emulab.net/`.

Foster, I., & Kesselman, C. 1998. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Frølund, S., & Koistinen, J. 1998. *Qml: A language for quality of service specification*. Tech. rept. HPL-98-10. Hewlett-Packard Laboratories.

Fuller, V., Li, T., Yu, J., & Varadhan, K. 1993 (Sept.). *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy.* RFC 1519 (Proposed Standard). Obsoleted by RFC 4632.

Garriss, S., Kaminsky, M., Freedman, M. J., Karp, B., Mazières, D., & Yu, H. 2006. Re: Reliable Email. *In: Proceedings of the 3rd Symposium of Networked Systems Design and Implementation (NSDI '06).*

Gaudin, S. 2003. *False Positives: Spam's Casualty of War Costing Billions.* `http://itmanagement.earthweb.com/secu/article.php/2245991`.

Golbeck, J., & Hendler, J. 2004. Reputation Network Analysis for Email Filtering. *In: Proceedings of Conference on Email and Anti-Spam (CEAS).*

Goodman, J., & Yih, W. T. 2006. Online Discriminative Spam Filter Training. *In: Proceedings of the 3rd Conference on Email and Anti-Spam.*

Gosling, J., Joy, B., Steele, G., & Bracha, G. 2005. *The Java (TM) Language Specification.* Addison-Wesley Professional.

Graham, P. 2003. Better bayesian filtering. *In: Proceedings of the 2003 Spam Conference.*

Grandison, T., & Sloman, M. 2000. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, **3**(4), 2–16.

Gray, A., & Haahr, M. 2004. Personalised, collaborative spam filtering. *In: Proceedings of 1st Conference on Email and Anti-Spam.* Citeseer.

Gregory, M., White, B., Fisch, EA, & Pooch, UW. 1996. Cooperating security managers: A peer based intrusion detection system. *IEEE Network*, **1**4(4).

Gyöngyi, Z., & Garcia-Molina, H. 2005. Web spam taxonomy. *Adversarial Information Retrieval on the Web.*

Haerder, T., & Reuter, A. 1983. Principles of Transaction-Oriented Database Recovery. *ACM Computing Surveys (CSUR)*, **1**5(4), 317.

Hansen, T., Crocker, D., & Hallam-Baker, P. 2009 (July). *DomainKeys Identified Mail (DKIM) Service Overview.* RFC 5585 (Informational).

Hashii, B., Malabarba, S., Pandey, R., & Bishop, M. 2000. Supporting reconfigurable security policies for mobile programs. *Computer Networks*, **3**3(1-6), 77–93.

Heberlein, L. T., Dias, G. V., Levitt, K. N., Mukherjee, B., Wood, J., & Wolber, D. 1990. A network security monitor. *Pages 296–304 of: 1990 IEEE Computer Society Symposium on Research in Security and Privacy, 1990. Proceedings.*

Herlocker, J. L., Konstan, J. A., & Riedl, J. 1999. An Algorithmic Framework for Performing Collaborative Filtering. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 230–237.

Hershkop, S., & Stolfo, S. J. 2005. Combining email models for false positive reduction. *Pages 98–107 of: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining.* New York, NY, USA: ACM.

Herzberg, A., Mass, Y., Mihaeli, J., Naor, D., & Ravid, Y. 2000. Access control meets public key infrastructure, or: Assigning roles to strangers. *Pages 2–14 of: IEEE Symposium on Security and Privacy.* IEEE Computer Society.

Housley, R., Ford, W., Polk, W., & Solo, D. 1999 (Jan.). *Internet X.509 Public Key Infrastructure Certificate and CRL Profile.* RFC 2459 (Proposed Standard). Obsoleted by RFC 3280.

IEEEProject802. 1986. Local and Metropolitan Area Network Standard: Overview, Interworking and Systems Management. *IEEE 802.1, Draft D*, August.

ISO/IEC. 1995. *Open Distributed Processing – Reference Model.* Tech. rept. ISO/IEC-10746. International Standards Organization, Geneva, Switzerland.

Jajodia, S., Samarati, P., & Subrahmanian, V. S. 1997. A logical language for expressing authorizations. *Pages 31–43 of: IEEE Symposium on Security and Privacy.* IEEE Computer Society.

Janakiraman, R., Waldvogel, M., & Zhang, Q. 2003. Indra: A peer-to-peer approach to network intrusion detection and prevention. *Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings*, 226–231.

Johansson, E. S., & Dawson, K. 2003. *Camram*. `http://www.camram.org/`. Web resource no longer available as of December 2009.

Jøsang, A., & Pope, S. 2005. User centric identity management. *In: AusCERT Asia Pacific Information Technology Security Conference*.

Keller, A., & Ludwig, H. 2003. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, **1**1(1), 57–81.

Kemmerer, R. A., & Vigna, G. 2002. Intrusion detection: a brief history and overview. *Computer*, **3**5(4), 27–30.

Kendall, M. G. 1938. A new measure of rank correlation. *Biometrika*, **3**0(1-2), 81–93.

Kleinrock, L. 1961. Information Flow in Large Communication Nets. *RLE Quarterly Progress Report*, **1**.

Kohl, J., & Neuman, C. 1993 (Sept.). *The Kerberos Network Authentication Service (V5)*. RFC 1510 (Proposed Standard). Obsoleted by RFC 4120.

Kolan, P., & Dantu, R. 2007. Socio-technical Defense against Voice Spamming. *ACM Transactions on Autonomous and Adaptative Systems*, **2**(1), 2.

Kong, J. S., Boykin, P. O., Rezaei, B. A., Sarshar, N., & Roychowdhury, V. P. 2005. Let Your CyberAlter Ego Share Information and Manage Spam. *Arxiv preprint physics/0504026*.

Kong, J. S., Rezaei, B. A., Sarshar, N., Roychowdhury, V. P., & Boykin, P. O. 2006. Collaborative Spam Filtering Using E-Mail Networks. *Computer*, **3**9(8), 67–73.

Lathia, N., Hailes, S., & Capra, L. 2008. The effect of correlation coefficients on communities of recommenders. *Pages 2000–2005 of: Proceedings of the ACM Symposium on Applied Computing*. ACM New York, NY, USA.

Laurie, B., & Clayton, R. 2004. proof-of-work proves not to work. *In: Proceedings of the The Workshop on Economics and Information Security.*

Law, A. M., & Kelton, W. D. 1997. *Simulation Modeling and Analysis.* McGraw-Hill Higher Education.

Lee, W., Stolfo, SJ, & Mok, KW. 1999. A data mining framework for building intrusion detection models. *Pages 120–132 of: Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on.*

Lee, W., Cabrera, J. B. D., Thomas, A., Balwalli, N., Saluja, S., & Zhang, Y. 2002. Performance adaptation in real-time intrusion detection systems. *Lecture notes in Computer Science*, 252–273.

Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., Roberts, L. G., & Wolff, S. 2001. A Brief History of the Internet. *Contributions In Librarianship and Information Science*, **9**6, 3–24.

Levchenko, K., Paturi, R., & Varghese, G. 2004. On the difficulty of scalably detecting network attacks. *Pages 12–20 of: Proceedings of the 11th ACM conference on Computer and Communications Security.* ACM New York, NY, USA.

Libbie, M., & Ludemann, P. 2006. Algorithmically determining Store-and-forward MTA Relays using DomainKeys. *Proceedings of Third Conference on Email and Anti-Spam (CEAS), July 27-28, 2006.*

Lilliefors, H. W. 1967. On the Kolmogorov-Smirnov test for Normality with Mean and Variance unknown. *Journal of the American Statistical Association*, **6**2(318), 399–402.

Limewire. 2008. *org.limewire.statistic.StatsUtils class.* Limewire source code.

Linington, P. F. 1999. Options for Expressing ODP Enterprise Communities and Their Policies by Using UML. *Pages 72–82 of: Third International Enterprise Distributed Object Computing Conference.*

Liu, D., & Camp, L. J. 2006. Proof of work can work. *In: Fifth Workshop on the Economics of Information Security.*

Lobo, J., Bhatia, R., & Naqvi, S. 1999. A policy description language. *Pages 291–298 of: Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*. American Association for Artificial Intelligence.

Lottor, M. 1987 (Nov.). *Domain administrators operations guide*. RFC 1033.

Lowd, D., & Meek, C. 2005. Good Word Attacks on Statistical Spam Filters. *In: Proceedings of the 2nd Conference on Email and Anti-Spam*. Citeseer.

Lunt, T. F., Jagannathan, R., Lee, R., Listgarten, S., Edwards, D. L., Neumann, P. G., Javitz, H. S., & Valdes, A. 1988. IDES: The enhanced prototype, A real-time intrusion detection system. *SRI International, Computer Science Laboratory, Menlo Park, Calif, USA*.

Lyon, J., & Wong, M. 2004. Sender ID: Authenticating E-Mail. *Internet Engineering Task Force Draft IETF, Oct*.

Lyon, J., & Wong, M. 2006 (Apr.). *Sender ID: Authenticating E-Mail*. RFC 4406 (Experimental).

Maier, G., Sommer, R., Dreger, H., Feldmann, A., Paxson, V., & Schneider, F. 2008. Enriching Network Security Analysis with Time Travel. *In: Proceedings of the ACM SIGCOMM*.

Marsh, S., & Briggs, P. 2008. Examining Trust, Forgiveness and Regret as Computational Concepts. *Computing with Social Trust*.

Mathworks. 1984. *MATLAB – The Language of Technical Computing*. `http://www.mathworks.com/products/matlab/`.

Mengshu, H., Xianliang, L., Xu, Z., & Chuan, Z. 2005. A Trust model of P2P System based on Confirmation Theory. *ACM SIGOPS Operating Systems Review*, **3**9(1), 62.

Moore, B. 2003 (Jan.). *Policy Core Information Model (PCIM) Extensions*. RFC 3460 (Proposed Standard).

Moore, B., Ellesson, E., Strassner, J., & Westerinen, A. 2001 (Feb.). *Policy Core Information Model – Version 1 Specification*. RFC 3060 (Proposed Standard). Updated by RFC 3460.

Natu, M., & Mirkovic, J. 2007. Fine-grained Capabilities for Flooding DDoS Defense using Client Reputations. *Pages 105–112 of: Proceedings of the 2007 Workshop on Large Scale Attack Defense*. ACM New York, NY, USA.

NIST. 1995. *FIPS-180-1 – Secure Hash Standard*. `http://www.itl.nist.gov/fipspubs/fip180-1.htm`.

Object Management Group. 1997. *Unified Modeling Language*. `http://www.w3.org/XML/`.

Olson, M. A., Bostic, K., & Seltzer, M. 1999. Berkeley DB. *Pages 183–192 of: Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference*.

Oracle. 2006. *Oracle Berkeley DB Java Edition*. `http://www.oracle.com/database/berkeley-db/je/index.html`.

Ortalo, R. 1998. A flexible method for information system security policy specification. *Lecture Notes in Computer Science*, **1**485, 67–84.

Pang, R., & Paxson, V. 2003. A high-level programming environment for packet trace anonymization and transformation. *Pages 339–351 of: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM New York, NY, USA.

Paxson, V. 1999. Bro: A system for detecting network intruders in real-time. *Comput. Networks*, **3**1(23), 2435–2463.

Perrif, A., Song, D., & Yaar, A. 2003. *StackPi: A New Defense Mechanism Against IP Spoofing and DDoS Attacks*. Tech. rept. Tech Report CMU-CS-02-208, Carnegie Mellon University, School of Computer Science.

Porras, P., & Shmatikov, V. 2006. Large-scale collection and sanitization of network security data: risks and challenges. *Page 64 of: Proceedings of the 2006 workshop on New security paradigms*. ACM.

Porras, P. A., & Neumann, P. G. 1997. EMERALD: Event monitoring enabling responses to anomalous live disturbances. *Pages 353–365 of: Proceedings of the 20th National Information Systems Security Conference*. Citeseer.

Porras, P. L. P., & Shmatikov, V. 2004. Privacy-Preserving Sharing and Correlation of Security Alerts. *Pages 239–254 of: Proceedings of the 13th Usenix Security Symposium, Usenix Association.*

Postel, J. 1981 (Sept.). *Transmission Control Protocol.* RFC 793 (Standard). Updated by RFCs 1122, 3168.

Prakash, V. V. 2007. *Vipul's Razor.* `http://razor.sourceforge.net/`.

Raymond, P. R. 2004. System and method for discouraging communications considered undesirable by recipients. *US Patent*, **6**.

Recordon, D., & Reed, D. 2006. OpenID 2.0: A Platform for User-centric Identity Management. *Page 16 of: Proceedings of the Second ACM workshop on Digital Identity Management.* ACM.

Return Path. 2009. *Email Delivery Optimization (previously known as Bonded Sender).* `http://www.returnpath.net/commercialsender/certification/`.

Riverbed. 2009. *Riverbed Cascade: Advanced network and application performance analysis and reporting.* `http://www.riverbed.com/products/cascade/`.

Roberts, L. G. 1967. Multiple Computer Networks and Intercomputer Communication. *Pages 3–1 of: Proceedings of the 1st ACM symposium on Operating System Principles.* ACM New York, NY, USA.

Rodgers, J. L., & Nicewander, W. A. 1988. Thirteen ways to look at the correlation coefficient. *The American Statistician*, **4**2(1), 59–66.

Royal Statistical Society. 1995. *Shapiro-Wilk W test implementation (R94).* `http://lib.stat.cmu.edu/apstat/R94`.

Royston, J. P. 1982. An extension of Shapiro and Wilk's W test for normality to large samples. *Applied Statistics*, 115–124.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. 1998. A Bayesian approach to filtering junk e-mail. *In: Learning for Text Categorization: Papers from the 1998 Workshop*, vol. 62. Madison, Wisconsin: AAAI Technical Report WS-98-05.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. 1996. Role-Based Access Control Models. *IEEE Computer*, **2**9(2), 38–47.

Sauvé, J., Marques, F., Moura, A., Sampaio, M., Jornada, J., & Radziuk, E. 2005. SLA Design From a Business Perspective. *Lecture Notes in Computer Science*, **3**775, 72.

Sebring, M., Shellhouse, E., Hanna, M., & Whitehurst, R. 1988. Expert systems in intrusion detection: A case study. *Pages 74–81 of: Proceedings of the 11th National Computer Security Conference*.

Siegel, S., & Castellan, N. J. 1956. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill New York.

Siemborski, R., & Melnikov, A. 2007 (July). *SMTP Service Extension for Authentication*. RFC 4954 (Proposed Standard). Updated by RFC 5248.

Singhal, A. 2004. Challenges in running a commercial web search engine. *In: IBM's Second Search and Collaboration Seminar*.

Skene, J., Lamanna, D. D., & Emmerich, W. 2004. Precise service level agreements. *Pages 179–188 of: Proceedings of the 26th International Conference on Software Engineering*. IEEE Computer Society.

Sloman, M., & Lupu, E. 2002. Security and management policy specification. *IEEE Network*, **1**6(2), 10–19.

Sloman, M., Lobo, J., & Lupu, E. C. 2001. *Policies for distributed systems and networks: International Workshop*. Springer-Verlag.

Snapp, S. R., Smaha, S. E., Teal, D. M., & Grance, T. 1992. The DIDS (distributed intrusion detection system) prototype. *Pages 227–234 of: Proceedings of the Summer 1992 USENIX Conference: June 8–12, 1992, San Antonio, Texas, USA*.

Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., & Moore, B. 2003 (Nov.). *Policy Quality of Service (QoS) Information Model*. RFC 3644 (Proposed Standard).

Socolofsky, T.J., & Kale, C.J. 1991 (Jan.). *TCP/IP tutorial*. RFC 1180 (Informational).

Sommer, R. 2005. *Viable Network Intrusion Detection in High-Performance Environments*. Ph.D. thesis, Technische Universität München.

Spearman, C. 1987. The proof and measurement of association between two things. *The American journal of Psychology*, 441–471.

Stern, H., Mason, J., & Shepherd, M. 2004. *A Linguistics-Based Attack on Personalised Statistical E-mail Classifiers*. Tech. rept. CS-2004-06. Faculty of Computer Science, Dalhousie University.

Stigler, S. M. 1989. Francis Galton's account of the invention of correlation. *Statistical Science*, 73–79.

Stolfo, S. J., Hershkop, S., Wang, K., Nimeskern, O., & Hu, C. W. 2003. A Behavior-Based Approach to Securing Email Systems. *Lecture Notes in Computer Science*, 57–81.

Tan, W. Y. 2002. Constraints-based access control. *Pages 31–44 of: Proceedings of the Fifteenth Annual Working Conference on Database and Application Security*. Norwell, MA, USA: Kluwer Academic Publishers.

The Apache Software Foundation. 2009. *The Apache SpamAssassin Project*. `http://spamassassin.apache.org/`.

The SANS Institute. 2009. *Computer Security Training, Network Research & Resources*. `http://www.dshield.org/`.

Tosic, V., Esfandiari, B., Pagurek, B., & Patel, K. 2002. On requirements for ontologies in management of web services. *Lecture Notes in Computer Science*, 237–247.

Trend Micro. 2009. *Email Reputation Services*. `http://www.trendmicro.com/services/rbl/`.

Tsirtsis, G., & Srisuresh, P. 2000 (Feb.). *Network Address Translation - Protocol Translation (NAT-PT)*. RFC 2766 (Historic). Obsoleted by RFC 4966, updated by RFC 3152.

USC Information Sciences Institute. 2009a. *The Network Simulator – ns-2*. `http://www.isi.edu/nsnam/ns/`.

USC Information Sciences Institute. 2009b. *The ns-3 network simulator.* `http://www.nsnam.org/`.

Verisign Labs. 2009. *Personal Identity Portal.* `https://pip.verisignlabs.com/`.

Vixie, P., & Schryver, V. 2000. *Distributed Checksum Clearinghouses.* `http://www.rhyolite.com/dcc/`.

Wakeman, I., Chalmers, D., & Fry, M. 2007 (November). Reconciling Privacy and Security in Pervasive Computing: The Case for Pseudonymous Group Membership. *In: 5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing.*

Wei, S., & Mirkovic, J. 2007. Building Reputations for Internet Clients. *Electronic Notes Theoretical Computer Science*, **1**79, 17–30.

Wei, S., Mirkovic, J., & Kissel, E. 2006. Profiling and clustering Internet Hosts. *In: Proceedings of the 2006 International Conference on Data Mining.* Citeseer.

Wittel, G. L., & Wu, S. F. 2004. On attacking statistical spam filters. *In: Proceedings of the 1st Conference on Email and Anti-Spam.* Citeseer.

Wong, M., & Schlitt, W. 2006 (Apr.). *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1.* RFC 4408 (Experimental).

World Wide Web Consortium. 1996. *Extensible Markup Language.* `http://www.w3.org/XML/`.

Wu, B., & Davison, B. D. 2005. Identifying link farm spam pages. *Pages 820–829 of: International World Wide Web Conference.* ACM New York, NY, USA.

Yaar, A., Perrig, A., & Song, D. 2004. Siff: A Stateless Internet Flow Filter to Mitigate DDoS flooding attacks. *Pages 130–143 of: Proceedings of the IEEE Symposium on Security and Privacy.*

Yang, X., Wetherall, D., & Anderson, T. 2005. A DoS-limiting network architecture. *ACM SIGCOMM Computer Communication Review*, **3**5(4), 252.

Yerazunis, Bill. 2009. *The CRM114 Discriminator – the Controllable Regex Mutilator*. `http://crm114.sourceforge.net/`.

Zheng, W., & Jin, L. 2009. Online Reputation Systems in Web 2.0 Era. *Page 296 of: Value Creation in E-business Management: 15th Americas Conference on Information Systems, Amcis 2009, Sigebiz Track, San Francisco, Ca, USA, August 6-9, 2009, Selected Papers*. Springer.

Zimmermann, P. R. 1995. *The official PGP user's guide*. MIT Press Cambridge, MA, USA.