A Framework for Developing and Sharing Client Reputations

Anirban Basu, Ian Wakeman, and Dan Chalmers

School of Informatics, University of Sussex
A.Basu, ianw, D.Chalmers@sussex.ac.uk

Abstract. Due to the open nature of the Internet, the abuse of the provided services has become widespread. Recent research has been conducted on behavioural profiling and reputations, mostly at the network level, to aid detection of malicious clients. We build on this research and propose a novel generalised framework for developing and globally sharing client reputations formed from behavioural histories. Promising initial simulations show a high level of success of our framework.

1 Introduction

The Internet has made possible the interconnection of servers and clients to provide services that have enriched the lives of millions of people around the world. But the boon becomes a bane since the open nature of the Internet allows servers to be attacked and abused. The attacker could abuse the terms of a service, e.g. attempting to open more than an allowed number of TCP connections. A Denial of Service (DoS) attack can use malicious connections to exhaust or significantly limit a network resource, leaving insufficient resources for the genuine connections. DoS can be distributed (DDoS) when the attack originates from multiple network entities, all operating with the same common malicious objective. Alternatively, the attacker could send an usually large number of unsolicited messages, which can have financial or legal implications to the victims. Similar to DDoS, the attacker could have multiple Sybil [1] identities working with the same malicious intent.

There are various defences against such attacks, e.g. *network intrusion detection*, where audit trails of network packets traces over a network are analysed in retrospect and in real-time to detect anomalies or patterns of known attack signatures. A related approach is *connection classification* where a particular network interaction is classified as malicious or genuine by analysing activity at the end-to-end network layer. For example, an email message sent to an SMTP server is checked for spam content. The value of these mechanisms can be improved by recording interactions between clients and servers across all servers and clients, allowing the generation of reputations. The level of interaction with a network client identity can then be decided based on its past local and/or global records of the behaviour and/or reputation. This allows forgiveness of occasional misbehaviour while identifying repeat offenders.

Our work is primarily motivated by [2–4] that suggest using and sharing client behavioural information and/or client reputations in order to protect network services from attacks. We briefly present the key related work in §2. Instead of devising new solutions for various types of services on the Internet or relying solely on low-level packet analysis, we investigate: *can a generalised high-level 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?* This approach is described in §3. We present results of high level simulations in §4.

2 Related work

Allman, et. al. [2] present an architecture of large scale sharing of behavioural history of network actors, e.g. hosts or email addresses, in an effort to inform policy decisions on how to treat future interactions. Building on this work, a client reputation system is proposed in [3] that can aid service providers in deciding whether to accept or decline interactions with a 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 can also be used for traffic prioritisation.

Both [2, 3] propose frameworks for use of client behavioural history and reputation at the packet-level, without knowledge of the application semantics in the end-to-end network layer. [5] describes profiling the behaviour of Internet hosts and then applying clustering techniques to categorise those hosts in order to develop an Internet-wide host reputation system. It is based on the hypothesis that most users (representing hosts) over the Internet exhibit slow-changing patterns of their behaviours over long periods of time. [6] extends the concepts of capabilities developed in [7] by adding reputation-based granting of capabilities tickets. There are a number of commercial and open source products working in a similar vein e.g. Arbor's PeakFlow [8], Riverbed's Cascade [9], SenderBase [4] and DShield [10].

3 The proposed framework

Behavioural histories and reputations of clients are associated with their identities. Various network identity schemes exist, e.g. IPv4 and IPv6 addresses, or the Public Key Infrastructure [11], amongst others, e.g. [12]. Some of these are long-lived, obtained through "strong" allocation and verification processes while others are short-lived and "weak". We assume a strong and long-lived identity scheme (e.g. public keys) throughout the rest of the paper. We shall describe later in this section that weak, short-lived identities, e.g. IPv4 or IPv6 may be used with our framework, although this is primarily left for future work.

We identify three types of network entities – *clients*, *servers*, and the *Global Reputation Analyser (GRA)*. Servers provide services, and record behavioural

histories and reputations of their clients. Servers report those local reputations to the GRA. Other servers can obtain interpretations of global reputations of clients from the GRA. The GRA may be implemented as a distributed overlay of closely administered and trusted nodes. Global and local (if available) reputations are useful to servers in determining levels of services, which helps filtering of malicious clients while rewarding genuine clients.

The proposed framework has five functional stages: *behaviour analysis*, *building of local reputation*, *global reputation reporting*, *global reputation query* and *decision making for service level*. In this paper, we do not present mathematical details of the framework, which can be found in [13].

Behaviour analysis: The interpretation of "good" or "bad" behaviour is dependent on policies implemented by servers. Behaviour can be observed through a variety of monitoring mechanisms, such as a packet filter or a Bayesian spam filter. More than one monitoring system may be used at a time to gather information about a client identity. Data obtained, at any instant, from each of these monitoring systems is generalised and expressed as a tuple. The behavioural history is a vector of such tuples. The size of this recorded history depends on implementation. Using this behavioural history as input, a policy-specific *behaviour analyser* is implemented. It evaluates all or part of the vector to output the interpretation of behaviour in quantised form, which is fed into the reputation system. The behaviour analyser is *open-ended* and can augment existing network activity monitoring systems.

Building of local reputation: Similar to the concept of *situational trust* in [14], reputation in our model is context-aware. Thus, reputations developed for "SSH" as the application context are independent of "email". Semantic dependencies between application contexts constitute an area of future work. The reputation developed is a policy-dependent function – *the reputation response* – of time and of cumulative behaviour values generated by the behaviour analyser. Reputation values are in the continuous range $[-1 \quad 1]$. For evaluation, we use an illustrative logarithmic reputation response policy. This policy enables fast reaction to bad behaviour and relatively slow response to good 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 any further change in behaviour tends to saturate the reputation further. A time-based decay in our logarithmic reputation response lets bad clients shed off bad reputations slowly with time, and have the reputations of good clients lowered with long periods of inactivity.

When IP addresses are frequently re-used (i.e. short-lived identities), the time decay policy can ensure that the the reputations are decayed to default values within the DHCP [15] minimum lease period.

Global reputation reporting: Servers submit local client reputations (and additional information) to the GRA, either at the ends of sporadic services or during

on-going services. The timestamp of the report, various reputation response policy parameters, amongst others are also submitted with each report. Age-based scavenging is applied on those reports to ensure that reputations that have been generated by servers with tougher reputation to behaviour response conditions are scavenged slower than the ones with more lenient conditions. Multiple submissions are allowed so long as the evidence of service interaction between a server and a client is valid for each submission. This evidence is an *authorisation token*, which the client provides to the server. This token may contain information relevant to the implementation but will at least contain a timestamp signifying the expiry of the token and the identity of the server as well as 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. It is essential that prior to this reporting mechanism, the identities of the client and the server are known to the GRA. Globally sharing reputations instead of behavioural history ensures a level of privacy protection for clients.

The steps from reputation query to global reputation reporting are illustrated in algorithm 1. Note that the steps in reporting global reputation are based on the assumption that a strong public-key identity infrastructure is in place. For other identity infrastructures, the concept of token signatures will need to be adapted.

Global reputation query: 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 particular application context. Each global reputation value may be associated with a quantitative measure of how much a (querying) server can be expected to believe in that particular reputation value. This is the context-aware *confidence* that one server has on another and is in the continuous range $[-1 \quad 1]$. Various methods [16, 17] exist for estimating trust based on similarity of opinion. We use, as an example policy, a statistical correlation coefficient to determine the context-aware confidence.

What the querying server does with the reputations and the confidences is implementation and policy specific. Global reputation may be used when the querying server has prior knowledge of the client and when it does not. The querying server may, for example: *a*) alter parameters of the reputation response such that the response is made either more strict or more lenient; and *b*) initialise the local reputation for the client from a statistical measure (e.g. least deviation from the last known local value, highest confidence) of the data obtained from the global reputation query.

Determining service levels: The globally queried or locally recorded reputation of a client is one of the factors in determining the service level that is provided to the client. There can be other factors, external to our model, affecting levels of service, such as the class of the client where two different clients having the same

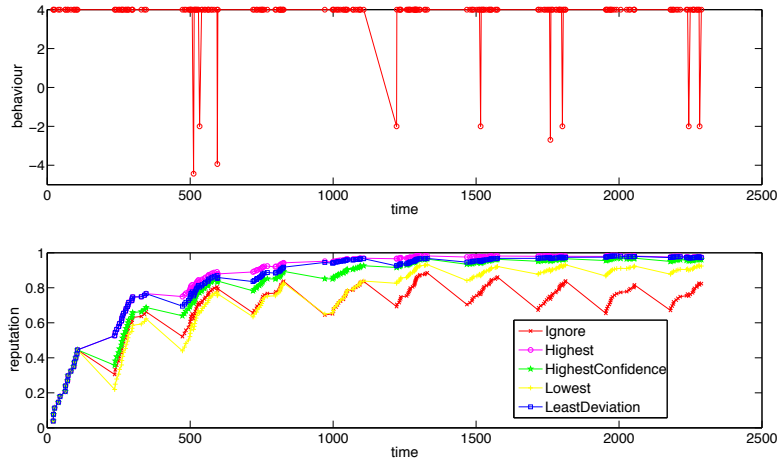
Algorithm 1 Global reputation reporting

1: The client sends a signed authorisation token to the GRA, for a server that it will request service from.
2: If the GRA accepts the token, the client also sends the same signed authorisation token to the target server. The GRA will not accept the token if it already has another (possibly unused) token for the same application context and server.
3: The server signs the authorisation token again to query the global reputation of client from the GRA.
4: The GRA performs a token authenticity check confirming that: the two copies of the 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.
5: On successful authenticity check, the GRA scavenges out-of-date reputation reports, and returns an anonymised vector of global reputations of the client.
6: The server makes inferences from the global reputations of the client.
7: The server provides service to the client and records its local reputation of the client which is used to alter service levels.
8: Either at the end of a sporadic service or during an on-going service, the server signs the authentication token, and sends it with its local reputation of client and other necessary parameters to the GRA.
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 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.
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 on-going service, a client could be required to re-issue another authorisation token as soon as the server reports its reputation to the GRA.

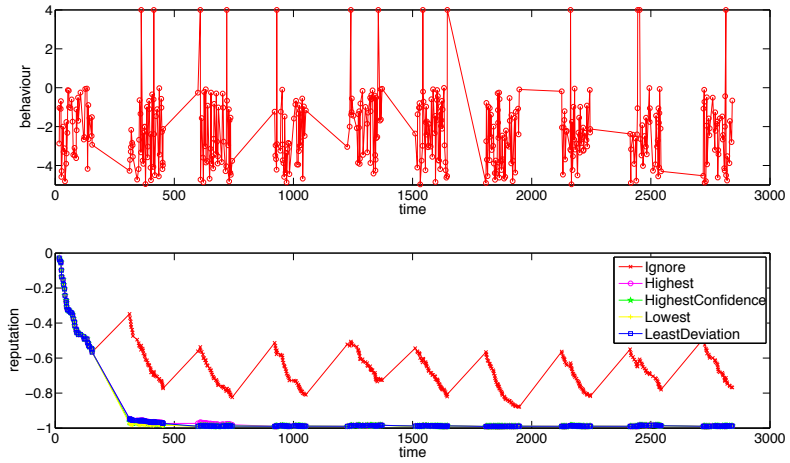
reputation but belonging to different classes (e.g. "premium" and "standard") may not be entitled to the same service level.

4 Evaluation

We have evaluated our framework with example policies and synthetic behaviour data on a multi-threaded discrete event simulator. In the absence of per-actor behaviour statistics, we have defined quantised behaviour per actor classes in agreement with population statistics of email behavioural data available from [4]. Those actor classes are: usual email sender (*Type 1*), spammer (*Type 2*), cautious email sender (*Type 3*) and malicious sender (*Type 4*). Type 3 is essentially Type 1 with very low frequency of bad behaviour. Type 4 represents an equal mix of good and bad behaviour with malicious intent. We compare the following policies for interpretation of global reputation: *ignore global reputation*; use the global



(a) Type 1 client



(b) Type 2 client

Fig. 1. Graphs of cumulative behaviour and of reputation versus time.

reputation that *deviates the least* from the last-known local reputation; use *the highest* global reputation; use *the lowest* global reputation; and use the global reputation with *the highest confidence*.

We present in figure 1, the graphs of cumulative behaviour (upper subplot) and of reputation (lower subplot) versus time for the a client of Type 1 class and another of Type 2. These represent a small subset of the various experiments we have conducted with our framework.

For the Type 1 client, we observe that the behaviour is generally good with occasional slips. The reputation, particularly the one with global reputation ignored, responds to such deviations in behaviour and also to inactivity over time. In the case of the Type 2 client, we notice that despite being consistently bad, there are some incidents of good behaviour but, the reputation response is less forgiving and it nearly saturates at the negative reputation. With global reputation ignored, the cyclic increase of reputation from negative values is caused by the time decay.

Certain adjustments can be made through policies to optimise the reputation response parameters. The cyclic decrease of reputation of the Type 1 actor due to certain periods of inactivity only may be interpreted to be of no malicious intent. The time decay parameter can be reduced to slow-down the decay. This situation is reversed for the Type 2 client, where it does not prove itself worthy, in subsequent interactions, of reputation increases (due to time decay). Therefore, reducing the time decay parameter will ensure that reputation is gained with longer periods of inactivity.

There are adjustments of policy parameters, which are described in [13] along with further detailed simulations and analyses of other actor types and of various attacks on our model. From those simulations, we observe that the uses of the local and the global reputations allow the servers to form fairly accurate views about their clients. In particular, the *least deviation* and the *highest confidence* policies are efficient in interpreting global reputations of clients that are consistent across all genuine servers that they interact with. In case of attacks, with attempts to falsify global reputations, the local reputation policies remain unaffected and hence resilient.

Based on our general experiences with simulation, we propose the following framework-level improvements that are left for future work:

1. Servers should turn on time decay only after a service contract has ended cleanly, i.e. with a successful report of global reputation to avoid undue effects of time decay.
2. The GRA should queue the creation of the new authorisation token if there is an unused one for the same client, server and application context. A global reputation lookup request from the server using the new token should be honoured only when the server reports the reputation of the client with the so-far unused token. This will ensure synchronisation of previously unreported global reputations during any dropped communication between the server and the GRA, e.g. due to a DoS attack.

5 Conclusion

In this paper, we propose a high-level generalised framework for locally developing and globally sharing client reputations from their behavioural histories in client-server interactions. Our experiments reveal that our framework is effective in weeding out malicious clients while rewarding genuine ones. Future work, to bring this framework to fruition, includes design of a resilient distributed GRA and semantics of application contexts, amongst others.

References

1. Douceur, J.R.: The Sybil Attack. In: Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS), Springer (2002)
2. Allman, M., Blanton, E., Paxson, V.: An Architecture for Developing Behavioral History. In: Proceedings of the Workshop on Steps to Reducing Unwanted Traffic on the Internet. (2005)
3. Wei, S., Mirkovic, J.: Building Reputations for Internet Clients. Electronic Notes Theoretical Computer Science **179** (2007) 17–30
4. Cisco Systems: Cisco IronPort Senderbase Security Network. `http://www.senderbase.org/` (2009)
5. Wei, S., Mirkovic, J., Kissel, E.: Profiling and clustering Internet Hosts. In: Proceedings of the 2006 International Conference on Data Mining, Citeseer (2006)
6. Natu, M., Mirkovic, J.: Fine-grained Capabilities for Flooding DDoS Defense using Client Reputations. In: Proceedings of the 2007 Workshop on Large Scale Attack Defense, ACM New York, NY, USA (2007) 105–112
7. Anderson, T., Roscoe, T., Wetherall, D.: Preventing Internet denial-of-service with capabilities. ACM SIGCOMM Computer Communication Review **34**(1) (2004) 44
8. Arbor Networks: Arbor Peakflow X: Enterprise Network Monitoring, Protection and Visibility. `http://www.arbornetworks.com/en/peakflow-x.html` (2009)
9. Riverbed: Riverbed Cascade: Advanced network and application performance analysis and reporting. `http://www.riverbed.com/products/cascade/` (2009)
10. DShield: DShield – Cooperative Network Security Community. `http://www.dshield.org/` (2009)
11. Adams, C., Lloyd, S.: Understanding Public-Key Infrastructure: concepts, standards, and deployment considerations. Macmillan Technical Publishing (1999)
12. Moskowitz, R., Nikander, P.: Host Identity Protocol (HIP) Architecture. RFC 4423 (Informational) (May 2006)
13. Basu, A.: A Reputation Framework for Behavioural History. PhD thesis, University of Sussex, UK (January 2010)
14. Marsh, S., Briggs, P.: Examining Trust, Forgiveness and Regret as Computational Concepts. Computing with Social Trust (2008)
15. Droms, R.: Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard) (March 1997) Updated by RFCs 3396, 4361, 5494.
16. Herlocker, J.L., Konstan, J.A., Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering. Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1999) 230–237
17. Breese, J.S., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Learning **9** (1998) 309–347