CHAPTER 5

IDENTITY-BASED TRUST COMPUTATIONS

5.1 INTRODUCTION

In the proposed Hierarchical Broker Architecture, the consumers who are in need of resources communicate their requests to the RRA along with their requirements. The RRA selects a suitable trustworthy RP without any bias, since it does not gain any monetary benefits from every individual transaction. Consumers requesting for resources may be genuine or malicious. In order to identify the malicious consumers who are prone to provide Distributed Denial of Service attacks (DDoS) at the RRA’s site, a DDoS Defeat Engine comprising Authenticators is proposed. The RRA processes only the requests of the genuine consumers, not that of the unauthenticated DDoS consumers, based on identity-based trust, thus increasing resource utilization for processing of genuine requests.

The architecture is further authenticated using DDoS attack defense mechanism and spam filtering mechanisms that endeavor to optimize resource utilization with the identification of genuine entities. An overlay network of Anomaly Detection Agents (ADA) is set up at the RRA level which enables the deployment of collaborative defense mechanisms. The DDoS defense consists of sensory registers that capture and analyze the traffic characteristics, short-term memory for local detection of DDoS attacks and a long-term memory for collaboration. A layered filter approach consisting of origin-based and statistical content-based filtering techniques is used to filter
spam. The results of simulation show that the presence of these mechanisms minimizes the wastage of resources which can be put to effective use in grid applications.

5.2 AUTHENTICATION MECHANISM IN HIERARCHICAL BROKER ARCHITECTURE

Consumers requesting for resources are dispersed over various domains in grid environment. Therefore, it is difficult to predict the behavior of consumers and verify their requests for legitimacy. The DDoS attacks target at exhausting the resources of the grid environment making it unavailable to its intended genuine consumers. The DDoS consumers who aim at attacking the grid infrastructure imitate genuine consumers and flood the RRA with illicit requests. Processing of these requests is an overhead at the RRA’s site which results in wastage of resources that can be utilized for processing genuine requests.

This led to the proposal of the enhanced Hierarchical Broker Architecture with the DDoS Defeat Engine, comprising one or more Authenticators as shown in Figure 5.1. An Authenticator authenticates the consumers and helps in detecting the DDoS attackers. The DDoS Defeat Engine envelops and protects the entire architecture from the illegitimate traffic that streams in at the RRA’s site. The RRA entity in this architecture acts as the main coordinator between the brokers and RP, providing trustworthy and satisfactory resources to the consumers.

Each Authenticator of the DDoS Defeat Engine may be associated with more than one RRA. It is possible to embed the DDoS Defeat Engine along with the actual functionality of the RRA. However, the objective of any grid architecture is to allow for better reuse (and sharing) of resources.
Though an additional layer of Authenticators incurs delays and communication costs, it allows more than one RRA to be associated with an Authenticator, leading to better resource reutilization. It is necessary that each entity at every level be associated with at least one entity at its higher level to enable the formation of the hierarchical architecture. The consumers requesting for the resources approach the top layer of the architecture. The consumers are also required to register themselves with at least one RRA in order to request for the resources. The following sections discuss how this architecture addresses the above problems.

5.2.1 Need for DDoS Defeat Engine

To shield the grid architecture from threats posed by the consumer community, a DDoS Defeat Engine is proposed. The DDoS Defeat Engine consists of Authenticators which interacts, authenticates and certifies the consumers. The Authenticator detects and differentiates the genuine consumer from the DDoS consumer and sends only the genuine consumer’s request to the RRA for further processing. It is shown in Figure 5.1 that the doors of the authenticators are open only to the genuine consumers and remain closed to the DDoS attackers. By virtue of the DDoS Defeat Engine, RRA processes only genuine authenticated requests forwarded by the Authenticators. This leads to better utilization of the system resources without wasting these resources on processing the DDoS attacks.

5.2.2 Working Principle

When a consumer arrives with the request for a resource, the Authenticator module in the DDoS Defeat Engine initially verifies the consumer. If the consumer is an unauthenticated DDoS consumer, the request for registry into the RRA is rejected else the registry of the consumer is
accepted on finding it to be genuine. The authentication mechanism used by the Authenticators within a DDoS Defeat Engine to authenticate and detect DDoS consumers relies on the symmetric key encryption algorithm. An Algorithm of authentication mechanism is explained in section 5.2.3.

![Hierarchical Broker Architecture with Authentication Mechanism](image)

**Figure 5.1 Hierarchical Broker Architecture with Authentication Mechanism**

Only the authenticated consumers’ requests are allowed to be processed by the RRA thus eliminating the wastage of grid resources. The RRA then selects a suitable trustworthy RP which complies with the DDoS free consumer’s request. The RRA intimates the selected RP about the details
of the consumer’s request through the broker, who nominated this RP. The consumer is also notified of the selection.

The consumer utilizes the resources of the RP and after the completion of transaction, the entities involved (consumer and broker) are summoned to provide their feedback on the RP’s resources at the RRA’s site. Finally, the nature of the feedbacks (genuine or malicious) provided by both consumer and broker are evaluated by the RRA using the reference set, and the trust value of the involved entities (consumers, RP and brokers) is updated accordingly.

5.2.3 Algorithm for Authentication Mechanism

1. The consumer and the Authenticator possess a shared key (K_{s(c-a)}) that is created and exchanged through a secured channel. Further, communication between Authenticator and the consumer is established using this shared key. This protects the system from the intrusion of the attackers.

2. When a consumer files a request with no authentication information, the Authenticator sends back a challenge string consisting of the ID of the Authenticator (A_{id}), the sequence number (S_{num}), and the validity time (V_{time}).

3. Secure Hash Algorithm (SHA1) is used to calculate the digest on the challenge string by the Authenticator. This digest encrypted with the private key of the Authenticator (K_{pr.a}), forming the signature of the Authenticator (Sig_{a}) is sent back to the consumer.
4. The consumer decrypts the signature of the Authenticator (\(\text{Sig}_a\)) using the public key of the Authenticator (\(K_{\text{pb-a}}\)) and retrieves the digest. This digest is again encrypted using the shared key (\(K_{\text{s(c-a)}}\)) forming the signature of the consumer (\(\text{Sig}_c\)) which is then sent to the Authenticator.

5. The Authenticator decrypts the signature of the consumer (\(\text{Sig}_c\)) with the shared key (\(K_{\text{s(c-a)}}\)), and compares the obtained digest with his own digest for similarity. If similar, the consumer is identified as a DDoS free consumer and an Authenticated Certificate is issued. The same is used for further transactions.

A DDoS free consumer might send multiple resource requests to the same authenticator. In this case, the Authentication token created for the first resource request can be used in the subsequent requests. This avoids the need for the Authenticators to send challenge strings in the subsequent requests. Since the subsequent requests can be authenticated immediately without the need of exchanging the challenge string and the authentication token, the response time to the subsequent requests can be shortened.

A DDoS consumer might intercept a resource request with a valid authentication token and replay the intercepted message from multiple attacking sites. To counter this kind of attack, two measures are taken. The first measure is that an authentication token is only valid for a limited period of time. The length of the period is determined by the Authenticator based on the application. This measure avoids a token being used by an attacker for an unlimited period of time. If an authentication token attached to a request has expired, the Authenticators will send a new challenge string to the consumer. To check whether an authentication token has expired, the Authenticator
checks whether validity time ($V_{\text{time}}$) in the token is within a certain time interval. $V_{\text{time}}$ is the time limit generated by the originator of the token. In order to check whether a token has expired, $V_{\text{time}}$ should be normalized (e.g. using the Greenwich Mean Time). This is because the Authenticators might be located in different time zones. The other measure is to limit the number of times that an authentication token can be used for validating requests. This measure ensures that even if an attacker uses a token that has not expired, only a limited number of the attacker’s requests can pass the authentication check.

The originator of the token, the Authenticator, is responsible for maintaining the count on the number of times that the token has been used to validate the resource requests. Thus, only the DDoS free consumer’s requests are allowed to be processed under any of the RRA.

The Authenticators can dynamically de-authenticate a consumer if it is identified as malicious during the course of transaction. This de-authentication is based on the trust-index of the entity. If the trust-index of the consumer which was earlier authenticated by the Authenticator drops below a certain threshold due to its continuous malicious behavior, the Authenticator de-authenticates it and updates its database accordingly. Such a consumer is re-authenticated only after its change of profile is verified to be genuine with increased trust-index. Though the authentication is initially done using a cryptography algorithm, the authentication information of consumer is later updated dynamically based on the trust of the consumer, which is explained in chapter 4.
5.3 DDoS PROBLEM

DDoS attack can be characterized as a large-scale, coordinated attack that is launched indirectly through multiple compromised hosts (called zombies) on victim network resources, with the purpose of preventing legitimate users from using those resources. A DDoS attack occurs when multiple compromised systems flood the bandwidth or resources of a targeted system. These systems are compromised by attackers using a variety of methods.

Many security systems are researched to secure the grids in authentication and encryption areas. However, availability and access control in grid computing still need more research. The recent well-publicized DDoS attacks have made people realize the importance of availability of data and service grids. The vulnerabilities of DDoS in grid computing arise from both the network environment that it relies on, and the algorithms that are applied. Flooding attacks are the most possible attacks that target network vulnerabilities in grid computing while low-bandwidth attacks target the algorithms running at the application level. Each grid has limited resources that can be exhausted by sufficient number of users. Thus when attacks are successful, the grid goes out of service.

Filtering is often ineffective, as the route to the filter will normally be swamped so that only a trickle of traffic will survive. However, by using an extremely resilient stateful packet filter that will inexpensively drop any unwanted packets, surviving a DDoS attack becomes much easier. When such a high performance packet filtering server is attached to an ultra-high bandwidth connection (preferably an internet backbone), communication with the outside world will be unimpaired as long as not all of the available bandwidth is saturated, and performance behind the packet filter will remain
normal as long as the packet filter drops all DDoS packets. It should be noted however, that in this case the victim of the DDoS attack still would need to pay for the excessive bandwidth. The price of service unavailability thus needs to be weighed against the price of truly exorbitant bandwidth / traffic.

In a grid, the availability of resources is crucial and the effect of a DDoS attack could be annihilating. Hence, a DDoS defense mechanism is a must in any grid architecture. Also, the defense must be carried out at wire-speed such that the server need not hibernate or be shut down (both of which could severely hamper the QoS of the grid). Thus, a DDoS defense mechanism must be able to distinguish attack packets from legitimate ones with high accuracy and minimal resource consumption. The complexity of DDoS problem suggests that the solution will require a collaborative defense mechanism.

For this purpose, a DDoS attack defense mechanism for packet filtering based on a three-level hierarchical cognitive memory model has been proposed. At the base level, sensory registers capture and analyze the traffic characteristics and pass it onto Short-term memory at the next level which stores these patterns and tries to detect a DDoS attack locally. The results of the analysis are communicated to the Long-term memory present at the highest level which is used for global detection of an attack using collaboration.

5.4 SPAM PROBLEM

An article in Electronic Spam (2007) suggested that spamming is the abuse of electronic messaging systems to indiscriminately send unsolicited bulk messages. Spam has pervaded the internet. The California legislature found that spam cost United States organizations alone more than
$13 billion in 2007, including the lost productivity and the additional equipment, software, and manpower needed to combat the problem. This article proposed that spam's direct effects include the consumption of computer and network resources and the cost in human time and attention of dismissing unwanted messages.

In a grid system where resources are of prime importance, spam is not just annoying but devastating as it consumes precious memory as well as CPU cycles. Hence, a spam filtering solution is a must for any grid system.

Spam filtering techniques include origin-based filtering, content-based filtering, context-based filtering, heuristics-based filtering and statistical filtering. Origin-based filters such as blacklist and white list use network information such as IP addresses and e-mail domains to classify messages. Attempts to stop spam by blacklisting the IP addresses of senders still allow a small percentage through. Most IP addresses are dynamic, i.e. they are frequently changing. An ISP, or any organization directly connected to the Internet gets a block of real Internet addresses when they register in the DNS. Within that block, they assign individual addresses to customers as needed. A dial-up customer may get a new IP address each time it connects. By the time that address appears on blacklists all over the world, the spammer will have new addresses for the next run. There are 4 billion possible IPv4 addresses on the Internet.

Content-based filters examine the message contents and try to “understand” them to detect spam. Several statistical techniques are merged with content-based filtering for accurate classification. Bayesian spam filtering, a form of e-mail filtering, is the process of using a Naive Bayes classifier to identify spam email. Many modern mail programs now implement Bayesian spam filtering. Particular words have particular
probabilities of occurring in spam email and in legitimate email. For instance, most email users will frequently encounter the word Viagra in spam email, but will seldom see it in other email. The filter does not know these probabilities in advance, and must first be trained so it can build them up. After training, the word probabilities (also known as likelihood functions) are used to compute the probability that a mail with a particular set of words in it belongs to either category.

Each word in the mail contributes to the mail’s spam probability. This contribution is called the posterior probability and is computed using Bayes' theorem. Then, the spam probability of the email is computed over all words in the email, and if the total exceeds a certain threshold, the filter will mark the email as a spam and can be automatically moved to a "Junk" email folder, or even deleted outright. The advantage of Bayesian spam filtering is that it can be trained on a per-user basis. A layered filtering mechanism consisting of origin-based filters (blacklist and white list) followed by statistical content-based filters (Bayesian filter) is followed here since these filters are able to detect most possible spam messages adaptively, according to the requirement of the entities, with minimal false positives.

5.5 PROPOSED ARCHITECTURE

In a grid environment where the availability of resources is important, detection and defense against DDoS attacks and spam filtering are key issues. This thesis proposes an intuitive mechanism based on Atkinson-Shiffrin’s cognitive memory model to detect DDoS attacks. Defense against these attacks is provided by designing a filter that can detect and mitigate a DDoS attack based on the current traffic pattern by dynamically assigning a score to the packet and discarding it if it exceeds a certain threshold. This score is based on the frequency of occurrence of the various attribute-value
pairs present in the packet. Ideally, a DDoS detection mechanism must be detected before it can hog the resources of the network. However, it is most beneficial to cater to all requests till the system utilization improves much. The scheme presented here takes the greedy approach to overload-control. The defense mechanism is placed at the border of the system-perimeter of RRA site.

In order to provide a spam filtering solution for the distributed setup of grid, this proposal utilizes collaborative spam filtering techniques. Logically, each entity has to be protected from spam. However, most of the interaction with outside world takes place at the RRA level. Although brokers and RP interact with the outside world, they do so under the constant supervision of RRA. Placing spam filtering mechanism at broker or the RP site would slow down the response time as they are smaller entities compared to the RRA site. Hence, spam filtering mechanism is placed at the top level of the hierarchy, at the perimeter of the RRA site, thereby filtering spam before it can actually enter the network and consume resources.

5.5.1 Extended Hierarchical Broker Architecture

In the existing Hierarchical Architecture, a network of Anomaly Detection Agents (ADA) is incorporated. This is responsible for spam filtering and DDoS defense and is shown in Figure 5.2. An ADA is a software agent that may be embedded as a plug-in at any entity. The DDoS defense and spam filtering mechanisms have been placed at the ADA. An ADA is capable of running multiple threads. It can also be extended to provide the full functionality of an Intrusion Detection System (IDS).
Figure 5.2 Extended Hierarchical Broker Architecture
An overlay network is defined on top of the network connecting RRA. The overlay network designates some RRAs as super-peers while the others are denoted as peers. An ADA is placed at the perimeter of each RRA site similar to a firewall. Each ADA associated with peer RRA is associated with local databases that are used in spam filtering techniques and memory tables for DDoS defense mechanism.

Certain techniques are applied locally while others utilize the collaboration in the overlay network. In order to monitor the messages and requests exchanged between the various ADA, a centralized control is required. The super-peer RRA provides centralized control over the network and maintains the centralized databases. An ADA associated with a super-peer RRA thus performs superfluous tasks in addition to those performed by the other ADA.

Grid is essentially distributed in nature. Providing centralized control on the other hand, introduces a single point of failure. To overcome this disadvantage, redundancy is brought in by allowing more than one RRA to act as super-peer. Each ADA at the super-peer level contains identical, synchronized copies of the central databases.

5.5.2 DDoS Defense Mechanism in Hierarchical Broker Architecture

The DDoS defense mechanism used to secure the proposed Hierarchical Three-Tier Broker Architecture is explained in this section.

5.5.2.1 Atkinson-Shiffrin’s memory model

Atkinson-Shiffrin’s cognitive memory model is one of the models used to represent the human brain that has been extensively studied in the
field of Artificial Intelligence. Sensory information stored in the human brain is linked to neurons. Information in some cells can be preserved only for a short term and these constitute the Short Term Memory (STM). There are other cells in the human brain which can store information for a long duration, in the order of years. Such cells make up the Long Term Memory (LTM). The Atkinson-Shiffrin’s model shown in Figure 5.3 consists of a three-layered structure of memory consisting of sensory registers, STM (also called active memory) and LTM.

The sensory registers are large capacity storage that can save information with high accuracy. However, they also decay at a fast rate to keep provisions for entry of new information. STM are fragile but can hold information with significant strength for quite sometime. As they have a fast access-time they are used during inference generation as well. Part of the information stored in the STM is copied into the LTM. LTM has large capacity and can hold information for a longer duration, sometimes spanning years. This model can be compared with the model of memory systems in computers.

Figure 5.3 Atkinson-Shiffrin’s cognitive memory model
An intuitive hierarchical defense mechanism has been proposed. Registers and STM are implemented at all RRA sites while LTM is present only at super-peer RRA.

### 5.5.2.2 Level 1 - Registers

The registers are used to capture and analyze the traffic characteristics. A set of attributes ‘A’ is chosen and their values are mapped in a hash map data structure called the memory table. In case another packet with the same value for a particular attribute arrives, the count of that value is incremented. For a packet, with different value for an attribute, collision is resolved by chaining. Thus, the memory table contains values and frequencies of occurrence of those values for the specific set of attributes in ‘A’. A score is computed per packet based on the frequency of attribute values at the STM to classify genuine and attack packets.

The memory table is refreshed periodically by making the registers ‘forget’ the old attribute values. This is achieved by uniformly decreasing the frequency of all attribute-value pairs present in the memory table. The memory table is refreshed every $T_s$ second where $T_s$ is a short interval that is chosen based on the bandwidth of the traffic.

### 5.5.2.3 Level 2 – Short Term Memory (STM)

The STM is used to detect an attack locally. The STM at an entity copies the data from the local register to its hash map and computes a per attribute score. The per-attribute score is computed using Equation (5.1) as a function of standard deviation of values and the fraction of packets arriving with that attribute.
\[ \omega = (\sigma * \rho_a) / \rho_i \]  \hspace{1cm} (5.1)

Here \( \rho_a \) is the fraction of packets containing an attribute, \( \rho_i \) is total number of packets that have arrived in the interval \( T_\omega \). \( \omega \) is the Per-attribute score and \( \sigma \) is standard deviation of the values obtained in that interval. This attribute score reflects the pattern in the packet header. Standard deviation indicates the deviation of the packet characteristics from the existing experience. If most packets show similar deviation, then the experience has to be changed to reflect that. This ensures that the defense is adaptive.

A score is computed per packet based on the frequency of attribute values. The idea is that packets causing a flooding attack tend to increase the frequency of a certain set of attribute-value pairs. If the attribute-value pairs present in a packet correspond to that in the set, then the frequency of those values in the memory table will increase. An upper limit is fixed on the frequency of a value. This limit varies dynamically based on traffic characteristics.

Initially, the score of a packet is initialized to zero. Every time an attribute-value pair of a packet causes the frequency to exceed the limit, its score is incremented. This score is compared with the threshold and discarded if it is above the threshold. Otherwise, the packet is considered genuine.

For example, consider a generic flooding attack with port number 13, packet length 50 and some typical header field values. During the flooding attack, all attack packets that arrive at the site, contain similar header values. If the set of attributes ‘A’ contains port number and packet length and then memory table counts the frequency of occurrence of values 13 for port number and 50 for packet length amongst other values, then it can be
observed that the frequency of occurrence shoots up for the attribute values present in the packet header of attack packets.

Initially, the score of a packet is initialized to zero. If $\alpha$ attributes are present in $A$, assuming equal weight for each attribute $(1/\alpha)$ then each attribute contributes $(1/\alpha)$ to the score. If the weights are chosen dynamically, the score of the packet is incremented accordingly. Every time an attribute-value pair of a packet causes the frequency to exceed the limit, its score is incremented by $(1/\alpha)$. This score is compared with the threshold and discarded if it is above the threshold. Otherwise, the packet is considered genuine. For example, if the threshold is 0.5, and at least $\alpha/2$ attributes have crossed the frequency limit, then the packet is more probable to be an attack packet. The default value is assumed to be 0.5. It can increase or decrease dynamically based on the nature of attack.

The threshold is computed by considering the characteristics of packets encountered in this epoch. It is possible that genuine requests contain some packet header values similar to attack packets. For example, genuine packet of length 50 may arrive that has a different port number. But this cannot be classified as attack. Hence, a minimal number of frequency overflows must be tolerated. In an epoch when the generic flooding attack discussed above is in progress, the threshold will be high, and only those packets that are very similar to the attack pattern get discarded. This ensures minimal misclassification. If the attack has stopped, then the number of packets with this pattern will decrease, and the threshold is lowered again. This ensures that the defense is adaptive.

The STM also stores the current traffic characteristics which are used to modify the frequency limit, the refresh rate and the threshold
dynamically. Other details such as total number of packets received, number of attack packets, and number of genuine packets are also stored at the STM.

The scores from each STM are sent to the LTM every $T_p$ seconds i.e., after the elapse of an epoch. The epoch is computed as $T_p = k \times T_s$, where $k$ is any integer, computed and this is common to all STMs present in the architecture.

5.5.2.4 Level 3 – Long Term Memory (LTM)

LTM is present at the ADA at the super-peer RRA site and stores information for the long run. LTM is also responsible for the collaboration between the RRA. Every $T_p$ seconds, data from various STM arrive at an LTM. The statistics present have to be stored at the LTM and a new threshold has to be computed that acts as the global discarding threshold. For example, consider three peer RRAs registered under a certain super-peer RRA. If the incoming values are [0.5, 0.5, 0.9] and the local threshold at the super-peer RRA is also 0.5, then it is possible that the attack is taking place at the local site only and the global threshold can remain at the neutral 0.5. Else if it is a global attack, a new threshold is computed that takes into account all these threshold values. This global threshold is sent to the peer RRA which chooses the minimum of locally computed threshold and the global threshold as the new threshold. The score of all packets arriving after this update is compared with the new threshold.

This is nothing but the sequential summation, to which normalization is performed to obtain a value between 0 and 1. This value is used as global threshold. The mechanism of the proposed DDoS attack defense is as follows: Choose a specific set of attributes, ‘A’, for analyzing the packets. Create a memory table indexed by attribute names. This abstracts
the sensory registers. The following steps are executed in parallel at each RRA:

**Step 1:** For each packet, capture the packet header. The registers remember the values of attributes in ‘A’. Any collision due to different values in the memory table is resolved by chaining. The packet header is compared with the already existing patterns in the memory table.

If the frequency of pattern occurrence exceeds a threshold, then the probability of it being an attack packet is high.

**Step 2:** Every $T_s$ seconds, decay the registers by ‘forgetting’ the old patterns present in the memory table.

**Step 3:** Every $T_p$ seconds, after the elapse of an epoch, refresh the registers by computing new threshold.

For this, a miniature table is created and sent as an experience to STM for pattern comparison with existing experiences. This is used for local detection of attacks. The detection results, packet feature data and traffic statistics are sent to LTM where spatial and temporal correlation of data is carried out to compute global thresholds. The global threshold is sent to all RRA who then chooses the minimum of local and global thresholds as the threshold for the next epoch.

The new thresholds reflect the current load at each site as well as the nature of incoming traffic (attack or genuine). The statistics and thresholds of previous epochs are entered in STM as ‘experience’. The sequence diagram of the DDoS defense mechanism in Hierarchical Architecture is given in Figure 5.4.
5.5.3 Spam Filtering in Hierarchical Architecture

A consumer request describes a job, specifies the policy requirements, budget and schedule in the prescribed format. However, malicious consumers can attempt to spam the RRA site by either sending illegal request files causing runtime exceptions at the RRA site. Brokers can advertise their presence by publishing themselves at the RRA site. Brokers may misbehave by sending bogus advertisements or goofed messages. Filtering such spam advertisements is crucial to the architecture as a broker could advertise with false credentials weakening the trustworthiness of the architecture.

Another class of special messages is the feedback provided by the consumer and broker to the RRA after the completion of each transaction.
Sending spam in the guise of feedback has serious consequences as the trust value of the RP may not be calculated in fairness. This message has to be filtered by the spam filter at the RRA site. Filtering spam at the server level prevents wasteful consumption of resources in the grid. Viruses can also be sent to disrupt the RRA site or the RP site where the Resource Specification Language (RSL) file is forwarded along with other details such as policy constraints, budget and schedule. These messages are also filtered to prevent the damage from further propagating through the grid system. Thus, placing the spam filtering techniques at the top of the hierarchy is effective as well as efficient.

5.5.3.1 Layered defense mechanism

The spam filtering thread at the ADA is implemented as a layered defense technique. The concept of layered defense has been suggested here as being much more effective at spam filtering than any of the individual solutions. In this solution, first, the source address of the incoming request or feedback message is checked against the local white list database maintained at the corresponding ADA. If the domain is listed in the white list, the message is assumed as good message.

At the next level, the source domain is checked against a centralized blacklist repository preserved at the ADA of super-peer RRA. If the origin is present in the blacklist, the message is classified as spam and moved to Junk folder to await further action. On the other hand, if the domain address is cleared in the centralized check, statistical content-based filtering has to be performed. These provide a simple and inexpensive solution for easy filtering. The more costly (in terms of processing) technique of Bayesian filtering is applied at the next step to classify messages that have been sent by a sender who is neither present in the blacklist nor in the white list.
Spammers can confuse filters by obfuscating text. For example, “spam” can be written as “s-p-@-m”. While a user can perceive this visually, it is not easy for a system to understand. Hence, at the preprocessing step, a de-obfuscating algorithm in Lee et al (2007) may be included.

The request or feedback message after de-obfuscation is passed through a Bayesian spam filter and the spam probability of that is calculated. If the value is above 0.9, the request or feedback message is classified as spam and moved to the Junk folder. Otherwise, the request or feedback message is moved to the genuine messages folder for further processing. Sufficient arguments on how Bayesian filtering is most suited for filtering spam is presented in White paper GSI software (2007). The accuracy of the classification can be improved using Chi-squared and inverse Chi-squared filters discussed in O’ Brien and Vigel (2003). A detailed analysis of spam filtering using neural networks as well as Bayesian filters is presented in Yang and Elfayoumy (2007) where an alternative solution is to introduce a feed-forward back-propagation neural network algorithm classifier is also mentioned.

The request or feedback messages classified as spam are present in the junk folder and the trust value of these entities got reduced as punishment for their spammicity. These messages can be added to the training data set that is used to re-train the Bayesian filter making it adaptive. The accuracy can be further improved by manual inspection and classification of messages present in the junk folder which shall be added to the training data set. This ensures fewer numbers of false positives (mails that are falsely classified as spam). The sequence diagram for spam filtering in Hierarchical Broker Architecture is given in Figure 5.5.
Figure 5.5 Sequence Diagram for Spam Filtering

5.5.3.2 Spam filtering techniques

Origin-based filtering and statistical content-based filtering are used in this solution. Two of origin-based spam-filtering techniques are blacklist and white list. Blacklist contains a list of addresses whose messages should always be blocked. White list contains a list of addresses and messages coming from which should always be allowed. Bayesian filters are used for classifying the spam based on the content of the messages. Bayesian filters can accurately classify over 98% of spam as discussed in White Paper, GSI software (2007).
**Bayesian filtering:** Spam filtering using Bayesian filter consists of two major phases. The first phase is a training phase where the filter is trained according to the user requirement based on the training dataset given by the user. The second phase is the actual filtering phase where spam filtering takes place and the filter adapts itself over time.

The training phase consists of three steps.

Step 1: Two input corpora are used - SPAM and HAM. Each word in a corpus is put into the respective table along with its frequency of occurrences.

Step 2: A hash table is created with input from these frequency tables and probability of the word being spam is calculated using Equation (5.2) as

\[
Pr(spam | \text{words}) = \frac{Pr(\text{words} | spam)Pr(spam)}{Pr(\text{words})}
\]  

(5.2)

This probability value assigned to each word is commonly referred to as spamicity, and ranges from 0.0 to 1.0. A spamicity value with greater than 0.5 indicates that the message containing the word is likely to be spam while a spamicity value with less than 0.5 indicates that a message containing the word is likely to be ham. The spamicity value of 0.5 is neutral, meaning that it has no effect on the decision as to whether a message is spam or not.

Step 3: On finalizing the hash table, the filter is trained. This is usually the cold start phase since considerable time is taken to train a filter before it can be operational.
The filtering phase consists of the following steps.

Step 1: The message is tokenized.

Step 2: The fifteen most interesting words in the message are determined, i.e., those words that are most significant in identifying whether the message is spam or not. Here, the interesting word is measured by how far the spam probability of the word is from a neutral 0.5.

Step 3: The total probability for the message to be spam is computed according to the formula.

Step 4: The message is treated as spam if this algorithm gives it a probability of more than 0.9 of being spam.

De-obfuscation: Spammers often obfuscate words in order to confuse the spam filters. For example, ‘mortgage’ may be written as ‘m-o-r-t-g-a-g-e’ or ‘disclaimer’ may be obfuscated as ‘d!sc1@!mers’. Although the human eye can easily comprehend the obfuscated words, it is difficult for a spam filter to de-obfuscate words by itself. Bayesian filters can recognize such words only if the filter has been trained with all probable obfuscated versions of spam words. However, it is not possible to list all possible obfuscations of a word, leave alone obfuscations of all possible spam words. Once spammers get hold of the common spam probability table, which is readily available in the internet, they can always find ways to circumvent the filter. Hence, a de-obfuscation technique may be used in a pre-processing step. A de-obfuscation pre-processor could be based on Hidden Markov Model. The Lexicon Tree-Hidden Markov Model can be used for spam de-obfuscation. However, it has a huge number of states making the filtering solution costly. Dynamically-weighted Hidden Markov Model (DW-HMM)
discussed in Lee et al (2007) has been shown as being more efficient. DW-HMM methodology may be used in a de-obfuscation pre-processor. However, due to resource utilization constraints and cost loss incurred by the consumers due to processing delay, this simulation does not include the pre-processing technique although in machines capable of more computing power, de-obfuscation increases the efficiency of the filter.

5.5.3.3 Trust Computation

An overview of computation of the trust has been presented below. An RRA is a neutral entity like Open Grid Forum (OGF) and can be considered trustworthy by default. However, all other entities in Hierarchical Architecture have a trust value associated with them.

Trust of a broker

The initial trust value of a broker is assigned by the associated RRA as a function of the presence-index of broker. The presence-index is based on the information of infrastructure (I-index) and policy (P-index) of the broker. Further, trust is updated by considering the behavior of the broker, e.g. while publishing or providing feedbacks whether the messages are spam messages or not, if the feedback is biased or genuine, etc.

Trust of a consumer

Consumer trust is calculated based on the feedback provided for that resource by the consumer, spaminess of the request or feedback and the commitment of consumer to pay for the resource. The initial trust value of the consumer is a threshold value of 0.5 as a median value. A request from a consumer with a trust value less than threshold is not entertained.
Trust of an RP

The initial trust value of an RP is a function of the presence-index of the RP which is based on information about infrastructure (I-index) and policy (P-index). Further, trust value of RP is updated based on the feedback given by the broker and/or consumer and feedback of RRA if required for the transactions of services provided by that RP as discussed in section 4.2.

Spamicity computation of entities

This research defines a new function to compute the spamicity of an entity using Equation (5.3) based on the spamicity of the request or feedback messages sent by that entity.

\[
\text{Spamicity of an entity} = f(\text{spamicity of message}) \tag{5.3}
\]

The choice of this function is crucial as this directly affects the trust-index of that entity. An entity that has sent spam messages in the recent past is more likely to send another spam message in the near future. Hence, the spamicity function must take into account the old spamicity of the entity as well as the spamicity of the most recent message from that entity at the same time assigning more weight to spamicity of recent message. This can be achieved by computing the spamicity of an entity as an average of the “aged” spamicity value of that entity and the spamicity of recent message sent by that entity. The spamicity of an entity is aged by normalizing the exponential of the existing old spamicity. Thus, the spamicity value of the consumer entity is given by Equation (5.4) as

\[
\psi'(E) = \frac{1}{2} \left[ \frac{\psi'_{\text{old}}(E)}{e} + \psi'_{\text{recent}}(m) \right] \tag{5.4}
\]
Here, $\psi(E)$ is the new spamicity of an entity $E$ (broker or consumer), $\psi_{\text{recent}}(m)$ is the spamicity of the recent message, and $\psi_{\text{old}}(E)$ is the existing old spamicity of the entity $E$.

An entity is listed in the blacklist if most of the recent messages from that entity are spam. However, only the most trustworthy entities such as the some trusted brokers or consumers shall be enlisted in the white list.

The initial spamicity of an entity is taken as zero assuming that an entity will not send any spam at the very beginning. A new formula to compute the trust value of an entity is obtained by incorporating the spamicity of that entity. Since the spamicity is computed based on request or feedback messages, the trust-index of an RP is not affected by spamicity. However, the identity-based trust indices of consumers as well as brokers have been changed to reflect the spamicity of that entity. In the new model, the identity-based trust value of a consumer is the non-spamicity of the consumer and the identity-based trust value of a broker is the average of non-spamicity of the broker along with the average trust of all RPs associated with that broker. The identity-based trust values of consumer or broker is computed based on non-spaminess of the entities since the genuineness of the entity is identified from the non-spaminess. The trust value of consumer is given by Equation (5.5) as

$$TI(E_c) = 1 - \psi(E_c)$$

(5.5)

Here, $TI(E_c)$ is the identity-based trust value of consumer entity, and $\psi(E_c)$ is the spamicity of the consumer. The trust value of a broker is given by Equation (5.6) as:
Here, \( TI(E_{br}) \) is the identity-based trust value of a broker, \( \psi(E_{br}) \) is the spamicity of the broker, \( TI_i(E_{RP}) \) is the trust-index of \( i^{th} \) RP and \( m \) is the total number of RPs associated with that broker. The trust value computation of the RP does not change based on the spamicity as the RP in the system is not capable of spamming the system.

5.6 SIMULATION AND RESULTS

The Hierarchical Broker Architecture is simulated with 100 consumers, 100 RPs, 50 brokers, 10 RRAs and 5 Authenticators. Currently ten different resource types with fixed criticality rates are considered for the purpose of simplicity; however, this can be easily extended without losing generality. Given the total number of requests to be generated and the mean of the distribution, the generation of number of requests to be issued during the \( i^{th} \) time instant is according to a Poisson distribution. Here, we used 1000 requests with the mean value of 30.

The percentage of resource utilization with and without the Authentication module (DDoS Defeat Engine with Authenticators) in the proposed architecture is plotted against various time instances as shown in Figure 5.6. The resource utilization is measured in terms of CPU utilization time. When the consumer requests are not processed by the Authenticator, the architecture is prone to process the unauthenticated malicious DDoS requests. When the requests are processed via the Authenticator, most of the malicious DDoS requests are detected at the earlier stages and the resources
are utilized only for the genuine requests. Therefore, the utilization of system resources for legitimate requests processing is maximized.

![Resource utilization performance graph](image)

**Figure 5.6 Graph with and without Authentication mechanism**

The spam filtering test set contains 1000 messages (genuine + spam) from brokers and consumers which are fired dynamically. The white list contains 10% of the entities (consumers and brokers) and blacklist contains 30% of the entities (consumers and brokers) in the database initially for all the test runs. Though the white list is static throughout, the blacklist adapts itself over time during the simulation and is dynamic. This is because an entity all of whose messages are spam gets added to the blacklist. And an entity that has stopped spamming shows a reduction in spamicity value over a period of time and is removed from the blacklist since the trust of that entity is improved. The good and bad corpora used for training the Bayesian filter contain 1000 messages each. The architecture is simulated for each individual filtering mechanism as well as for the combined solution and the various
graphs are plotted. The spam detection rate graph plots the efficiency of each filter while the graph depicting false-positives describes the effectiveness of each filter. The percentage of resource consumption is plotted against the number of request and feedback messages for different percentages of spamicity of the entities with and without the spam filtering solution at the ADA.

A DDoS flooding attack is simulated by sending requests from external entities. For simulation only the source address, destination address, protocol flag and QoS of the packet are extracted. A packet capturing tool JpCap is used in this simulation to capture the packets and access all its headers. A greedy approach is followed in the defense mechanism which allows an entity to receive and respond requests till it reaches its resource utilization limits. A resource consumption table of each entity is maintained at the RRA for this purpose.

The graphs in Figures 5.7 and 5.8 have been plotted with 50% spam messages in a test data set of 1000 messages. Figure 5.7 shows the number of false positives generated by the individual filters. Bayesian filter gives the least percentage of false positives. Combined layered filter gives lesser percentage of false positives compared to other filters as the number of messages increases. As the number of messages increase, the percentage of false positives reduces much as the filter adapts itself over time by re-training.

Figure 5.8 shows the spam detection rate of each filter as well as the combined filter. It can be seen that though white lists can detect 100% spam, they generate a huge number of false positives. This is because white list is highly prejudiced, allowing only messages which are from the addresses listed in the white list database and classifying all other messages as spam.
Blacklists, on the other hand, are lenient allowing messages from all addresses other than those listed in blacklist database. If the size of the blacklist database is large, too many messages may be discarded as spam generating a huge number of false positives and if the size is small, blacklist generates a huge number of false-negatives bringing down the spam detection rate. An adaptive blacklist performs better. Thus, both these techniques must be combined with a more effective technique such as Bayesian classifier. Even though white lists are prejudiced and blacklists lenient, they help reduce resource consumption during the detection of spam messages as the more complex Bayesian Classifier need not be performed for all the messages.
An entity (consumer or broker) listed in white-list is considered implicitly trustworthy and the spamicity of that entity remains constant (what it was initialized with). Similarly, an entity listed in blacklist is considered implicitly untrustworthy and the spamicity value does not change, though the entity may stop spamming. This drawback can be overcome by making the blacklist dynamic by computing just the Bayesian probability for a message from an entity that has been blacklisted. If the entity stops spamming, the spamicity value decreases and over a period of time, the entity may be removed from the blacklist. It can be seen that the spam detection rate of the blacklist improves over a period of time as the blacklist adapts itself. Thus, this model makes use of a static white list and dynamic blacklist combined with a Bayesian Classifier to combat spam. It can be seen that the combined filter solution is effective in terms of spam detection rate and also generates the least number of false positives as can be seen from the graphs.

**Figure 5.8 Spam detection rate of filters**
Figure 5.9 accounts for the resource consumption of the system for different percentages of spam messages in a simulation of the trustworthy architecture with and without spam filtering solution. It can be seen that the initial resource consumption is greater in the simulated architecture incorporating the spam filtering solution. This is because of the training phase which consumes extra resources. However, as the number of messages increases, the resource consumption in the system with spam filtering stabilizes. Without spam filtering more resources are wasted in processing spam requests or feedbacks. Sometimes runtime exceptions occur at the RRA site causing the RRA site to crash. The graphs in Figure 5.9 are plotted with 10%, 50%, 90% spam messages in the testing dataset. In the long run, the resource consumption is minimal in the architecture with spam filtering solution thus increasing the overall efficiency of the system.

![Resource consumption for different % of spam](image)

**Figure 5.9  Resource Consumption graph for different percentages of spam messages**
Figure 5.10 is the resource depletion graph that illustrates the percentage of resources that are wasted, i.e., consumed but not for problem solving. For this simulation, 30% of the requests are assumed to be spam and 20% of the requests are attack packets. It can be seen that resource depletion without the filtering and defense mechanisms has a linear relationship with the number of messages. As the number of request and feedback messages increase, more resources are wasted.

![Resource depletion graph](image)

**Figure 5.10 Resource Depletion Graph highlighting wastage of resources**

However, with the filtering and defense mechanisms in place, although more resources are depleted initially over time, as more requests are processed, the resource wastage stabilizes. This overhead is due to the threads that are alive throughout. Thus, we infer that presence of the DDoS defense and spam filtering solution minimizes the resource wastage and ensures availability of resources.
5.7 CONCLUSION

The consumers who enter the Hierarchical Broker Architecture with the motive of attacking the system and paralyzing it from functioning are detected by the proposed DDoS Defeat Engine, with the help of the Authenticator modules. The Authenticator identifies and rejects the fake requests coming from unauthenticated DDoS consumers and allows only the genuine requests to the RRA for further processing. Thus, resource utilization is maximized for processing the genuine requests. Wastage of resources for processing unauthenticated DDoS requests is eliminated by the proposed architecture with bearable overhead.

This chapter extends the existing trustworthy Hierarchical Broker Architecture by creating an overlay network of RRA for collaborative spam filtering and DDoS attack defense mechanism. An ADA that runs the DDoS defense mechanism and the spam filtering thread is placed at the perimeter of each RRA site. An intuitive three-level hierarchical model that consists of registers to capture and analyze the traffic characteristics, a short term memory to perform local attack detection and a long term memory for global detection and collaboration has been proposed.

A layered spam filtering solution, consisting of white list, blacklist and Bayesian filter, has been proposed. These mechanisms are incorporated at the perimeter of the RRA site at the highest tier in the architecture. This firewall-like protection ensures that the smaller entities need not be burdened and that the spam messages as well as attack packets can be filtered before they enter the architecture.

The effectiveness of this filtering solution can be seen from the resource consumption graph which clearly shows that fewer resources are
exhausted in a system with the filtering mechanisms in place, as against the system with only trust based resource selection, with less false positives. The resource depletion graph further reinstates the combined effectiveness of DDoS defense mechanism and the spam filtering solution in bringing down the resource wastage improving the grid resource availability. This simulation uses a static set of attributes to track the frequencies of the attribute-value pairs. This could be made more dynamic to adapt to varying attacks. Future work consists of including inference engine to define dynamic firewall rules automatically based on the goals defined at LTM at ADA to counter DDoS attacks more effectively. Also more complete intrusion detection features could be included to further secure the trustworthy Hierarchical Broker Architecture.