Adaptive Distributed Mechanism Against Flooding Network Attacks Based on Machine Learning

Josep L. Berral
Computer Architecture Deptartment
Technical University of Catalonia
berral@ac.upc.edu

Nicolas Poggi
Computer Architecture Deptartment
Technical University of Catalonia
npoggi@ac.upc.edu

Javier Alonso
Barcelona Supercomputing Center (BSC-CNS)
Computer Architecture Deptartment
Technical University of Catalonia
alonso@ac.upc.edu

Ricard Gavaldà
Deptartment of Software,
Technical University of Catalonia
gavalda@lsi.upc.edu

Jordi Torres
Barcelona Supercomputing Center (BSC-CNS)
Computer Architecture Deptartment
Technical University of Catalonia
torres@ac.upc.edu

Manish Parashar
Dept. of Electrical and Computer Engineering,
Rutgers University
parashar@rutgers.edu

ABSTRACT

Adaptive techniques based on machine learning and data mining are gaining relevance in self-management and self-defense for networks and distributed systems. In this paper, we focus on early detection and stopping of distributed flooding attacks and network abuses. We extend the framework proposed by Zhang and Parashar (2006) to cooperatively detect and react to abnormal behaviors before the target machine collapses and network performance degrades. In this framework, nodes in an intermediate network share information about their local traffic observations, improving their global traffic perspective. In our proposal, we add to each node the ability of learning independently, therefore reacting differently according to its situation in the network and local traffic conditions. In particular, this frees the administrator from having to guess and manually set the parameters distinguishing attacks from non-attacks: now such thresholds are learned and set from experience or past data. We expect that our framework provides a faster detection and more accuracy in front of distributed flooding attacks than if static filters or single-machine adaptive mechanisms are used. We show simulations where indeed we observe a high rate of stopped attacks with minimum disturbance to the legitimate users.

AISec'08, October 27, 2008, Alexandria, Virginia, USA.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: General—*security and protection*

General Terms

Algorithms, Management, Reliability, Security

Keywords

Machine Learning, Flooding Attacks, DDoS, Autonomic Computing, Cooperative, Intrusion Detection

1. INTRODUCTION

Nowadays, networks and autonomous systems have reached a high level of sophistication, and the same way failures, attacks, and the corresponding protection mechanisms have reached a high level of complexity too. Autonomic computing and its network-oriented area autonomic networking, are working on self-organizing, self-optimizing, self-healing, and self-defense methods, in order to protect networks against these attacks, abuses, and failures. Also machine learning and artificial intelligence are finding their place as important ingredients to achieve these self-* capabilities with higher accuracy, allowing adaptation and adjustment when simple and static algorithms can not be applied for. For this automatization, data mining tehcniques are being used to allow systems and networks to recognise patterns in the system's behavior and act accurately. Many techniques are available for categorizing and mining such patterns when they occur at a single site. But on large distributed systems, interesting patterns may be itself themselves distributed, i.e., not visible at once from a single site, so harder to detect.

Distributed attacks, floodings, and abuses are a forms of attacks that sometimes are hard to detect because their resemblance to legitimate connections or traffic. Distributed denial of services and flooding attacks overwhelm networks

and resources sending a great amount of usual-looking packets, opening many usual-looking connections or requesting many hard queries to specific services, simultaneously from a large amount of requesters. This scheme makes difficult to detect each single attacker because of the apparently normal behavior of each individual requester. Also, it cannot easily be stopped from the bordergate access points of the victim network because each router in the network only sees a part of the attack, even a negligible part in the case of access points. For this type of attacks, one must look for methods where information about suspected attacks spreads quickly among the intermediate elements before the attack reaches the victim in full.

DDoS attacks can occur in popular sites and networks, motivated by interest or vandalism. DDoS became famous in 1996 when SYN floods took down Web servers by attacking the root nameservers. With only 1000 *zombie machines* and the trin00 DDoS toolkit , a critical server from the University of Minessota was disabled and denied access to a very large university network. The increasing appearance of DDoS toolkits and trojans make it necessary to protect networks and critical services from these attacks, because with a reduced number of machines and a downloadable toolkit a distributed denial of service can be performed easily against any given site.

In our approach we present a mechanism where, using information sharing and machine learning, a network is able to stop and avoid a distributed attack, abuse, or flooding. Each element shares with its neighbors information about the status of the network, and aggregates its local information and the one received to model and classify the traffic it is receiving. Our mechanism lets each element learn about the behavior of its portion of network, adjusting its classifiers to its location in the network and the traffic it typically sees.

In Section 2 we discuss previous work and approaches to this area. In Section 3 the architecture of the framework and each mechanism are detailed. In Section 4 we explain the how the classifiers work and how they adapt to their environment. In Section 5 we describe some experiments to validateour approach. Finally, in Section 6 we draw some conclusions and anticipate some of our future work.

2. RELATED WORK

Our main reference is the work of Zhang and Parashar . They defined a cooperative framework for the detection of DDoS attacks, in which a subset of the network nodes (the overlay network) is selected. Nodes in the overlay network maintain detection mechanisms for abnormal situations, and exchange and aggregate information about traffic condition in different parts of the network. The results were very satisfactory in the sense that information sharing between intermediate network nodes allow them to detect attacks with more accuracy and timeliness.

More technically, the detection mechanism used in Zhang and Parashar's previous work was the well-known CUSUM (cumulated sum) algorithm applied to traffic volume. When a node detected that local traffic became suspiciously high, it immediately gossiped to p selected neighbors the key-pair $\langle victim, confidence \rangle$, indicating the victim target and the confidence (or degree of suspicion) computed by its detectors. Also each node used the sum of confidences of the received gossips and the confidence of its own detector to declare that a victim is under attack. Therefore, note that there are at least three parameters that have to be set, either for all nodes at once, or for each node separately: the threshold that triggers gossiping, the amount p of gossip spread, and the threshold on the gossips that triggers an attack declaration. It is, in general, hard to anticipate *a priori* the best values for these parameters.

With a different goal in mind, Poggi et al. proposed AUGURES, an architecture used to model users and streams of user requests to websites. The system prioritizes incoming users and sessions according to their expected utility to the system (e.g., expected revenue) so that, in case of a system overload, only users with high utility are effectively admitted to the system. They used machine learning techniques (such as decision trees or Naive Bayes classifiers) to learn to assign priorities from weblogs of past activities, and showed that even with simple learners one can have nontrivial predictive power about user's future intentions.

Other techniques have been previously developed in order to stop denial of service attacks. In , nodes closest to the source are equipped with traffic ratio detectors. When the traffic ratio mismatches with the usual one, traffic is limited for the specific connection retaining the weight of the attack as far from the victim as possible. Also, a technique using machine learning has been developed in using as information the ratio of TCP flags and TCP connections. Furthermore, heuristics and data mining mechanisms, such as MULTOPS , have been developed in order to stop bandwidth attacks using statistical information about the traffic.

For our approach, we will use the infrastructure purposed in , complementing it with the power of adaptivity provided by machine learning induction, and in particular applying techniques similar to those in . Furthermore, in order to restrict the attack strike to the network borders we have added to the model a backward warning system (not present in) to stop the attack as close to the source as possible. So, our network elements will learn to determine whether the situation is normal or abnormal, and use this status to determine whether a message to a particular node should be forwarded or blocked, if a flooding attack to that node is suspected. This way, attacks can be stopped much before they reach the victim, and even very close to the borders of the network.

3. NETWORK ARCHITECTURE

3.1 The Overlay Network

In our approach, a subset of the intermediate network nodes are chosen to belong to an overlay network. Nodes in the overlay network will be equipped with detection and classification capabilities, and will exchange gossiping about possible threats and warnings about declared threats. It is important that nodes in the overlay network are key nodes (like backbone routers, firewalls, bordergate nodes, and high adjacency nodes) that see most of the traffic for the most requested services. In the extreme case, all routing nodes from the network would be chosen to belong to the overlay network, although this may be unfeasible in practice.

3.2 Classifiers and Detectors

We use detectors and classifiers to distinguish patterns of normal traffic patterns from attacks and abuses. Classifiers are models that produce predictions, and in particular

those that can be trained using statistical or machine learning techniques. Detectors react to changes in specific traffic measures, such as "amount of traffic towards a given node", "relation between amount of traffic sent/received to a particular service", or others depending on protocols, message headers, or specific behavior patterns. In this work we have used only the first, amount of traffic towards a given node. In particular, for the moment we do not analyze at all the headers or contents of the messages. This is partly due to the complexity of performing this analysis, but also because we want to concentrate on the particular case of distributed flooding attacks which, by definition, have "large amount of traffic" as the only common feature. However, our framework could handle well additional information about the packets, if available, and this could be potentially be very useful in protecting from flooding and other types of attacks.

In this work we will use CUSUM algorithm to determine when the traffic towards a particular node changes significantly, but other detection algorithms can be used. As classifiers, we will use the well-known Naive Bayes method, which will integrate the local information plus that shared with the neighbors to declare when an attack is occurring. More details about classifiers and how they are used at the nodes are given in Section 4.1.

3.3 Information Sharing

To improve the accuracy in classification, each node must have the maximum global knowledge about traffic. For this approach there are two kind of messages that will be shared: gossips, indicating suspicion of a flooding attack, and warnings, indicating high certainty of an ongoing attack. An important difference is that gossips are sent to and received from neighbors in any direction of the overlay network, while warnings are sent only in the direction of (apparent) attackers, in order to strangle attacks as close to the source as possible.

When a detector (the CUSUM algorithm for traffic volume towards a particular victim) considers that traffic to a victim is increasing in an abnormal way, a gossip is spread among the neighbors indicating that a victim can be under attack. In one extreme, all neighbors will receive the gossip, and in the other extreme, only the next-step-node will receive the gossip. The message contains the possible victim ID and the confidence, that is, the number of gossips received referring to the same possible victim. Also, when a classifier determines that a victim is under attack while classifying a message destinated to it, a warning is sent to node from which the message was received (i.e., towards the source of the attack), and sent too to the neighbors if required. Each node uses the aggregated gossips and warnings received as inputs to its classifier.

In other words, the idea of this scheme is the following: With the warnings, victim-closest nodes will indicate to the bordergate nodes to stop the attack flow, and with the gossips, the source-closest nodes will indicate to the intermediate network nodes that the aggregated traffic is still attempting to enter the network, instead of letting the attack flow in. The result should be both to improve the accuracy in classification, but also, when an attack is detected, to stop undesired traffic as close to the sources as possible, thus freeing resources in the network.

```
For each received message do:
    if accumulated mean > destination threshold:
        -increment gossip for destination
    endif
    if time unit changes:
        if gossips for destination > 0:
            -send gossip with confidence to neighbour node
        endif
        -renew the traffic mean and accumulated mean
        -clean gossips and warnings
    endif
    -evaluate line <message, gossips for dest, warnings for src>
    if classified as attack:
        -increment warning for source
        -send warning to backward node
    else
        -forward message to destination
    endif
endfor
```

Figure 1: Message evaluation algorithm in each node

3.4 Node and Network Algorithm

Each node contains the algorithm for classifying each message passing through it. The algorithm compares the accumulated sum of means for each time unit with a characteristic threshold for each destination. When a message arrives, the accumulated mean for its destination is renewed and if it reaches the threshold, a gossip is added for that destination. Then, each node accumulates self-gossips and neighbor-gossips and for each time unit the number of confidences is checked. If confidences for a service are above a concrete threshold, a gossip is sent to the next node in the service path, and also is sent to p neighbors. Also, for each time unit (empty time units included) means and accumulated means are updated according the last time-unit statistics.

After getting all the statistics of traffic to the message destination, a classifier evaluates the message attributes including the gossips for the destination and the warnings for the source. If classifier concludes that the message belongs to an attack, the message is not forwarded and also a warning is sent to the backward node. The evaluation algorithm is given in Figure 1.

4. NETWORK LEARNING AND CLASSIFICATION

Given that each node on the network has different characteristics depending on the context and the network situation, configuring with accuracy a classifier becomes a problem. The classifier must be adjusted to the node detectors, the level of trust to a concrete gossip, and the amount or type of usual traffic passing through the node.

4.1 Training the Network

Each node is equipped with a learner/classifier, so machine learning can give each node the ability to adapt their thresholds of gossip trusting and detection mechanisms for each victim or source to the node situation. The only thing that needs to be done is to train each node on its typical traffic and in situations denoting an attack. For that, a training data set must be obtained capturing samples of traffic and samples of attacks or abuses. If not available, simulations

or workloads can be used, that clearly reflect different the volumes of traffic corresponding to attacks and non-attacks.

To model the traffic and the context of the intermediate network, for reduce each message to a tuple of only four attributes: destination, source, warnings received about the source being an attacker, and gossips about suspicion of attack to the victim. One could add more relevant attributes depending on the level at which the message is analyzed (link, IP, TCP, application, ...), although we have not done it so far in our experiments.

We next describe the particular learning and classifier we have used, the Naive Bayes method. In this method, the different message attributes are supposed to be statistically independent. Although obviously they are not, this assumption leads to an efficient and simply classifier that usually does well in practice, so it is a good starting point before more complex classifiers are tried out.

To describe the method, we use the following notation for converting a boolean value to an integer: given two numbers a and b, we denote

$$a \lesseqgtr b = \begin{cases} 1 & \text{if } a > b, \\ 0 & \text{if } a \leq b. \end{cases}$$

Given a message m, regarded a a tuple $\langle src, warn, dest, gossip \rangle$, the Naive Bayes method computes estimates of $Pr(attack|m)$ and $Pr(noattack|m)$ and simply predicts "attack" if $Pr(attack|m) \lesseqgtr Pr(noattack|m)$, and "no attack" otherwise. These two probabilities are computed using Bayes' Theorem, e.g.:

$$\Pr(attack|m) = \frac{\Pr(m|attack) \cdot \Pr(attack)}{\Pr(m)} \qquad (1)$$

Observe that $\Pr(m)$ can be desregarded because we only care about the ratio of $\Pr(attack|m)$ and $\Pr(nonattack|m)$. $\Pr(attack)$ and $\Pr(nonattack)$, for the purposes of training the classifier, are simply the proportion of packets labelled "attack" and "nonattack" seen at each node over time. Therefore, they can estimated from the training data, locally to each node. To estimate $\Pr(m|attack)$ we use the independence assumption, as follows:

$$\begin{aligned} \Pr(m|attack) &= \Pr(\langle src, warn, dest, gossip \rangle | attack) \\ &= \Pr(src, warn|attack) \cdot \Pr(dst, gssp|attack). \end{aligned}$$

Note that we have grouped source and warning together since naturally a message arrives to this node with information about its source and the number of warnings for the source, while the level of current gossip for a particular destination is added to it by the node from the table kept locally. Thus, it does not make much sense to further split the probability in four. Now again, these two probabilities can be estimated from the empirical distribution of messages in the training data as follows:

- $\Pr(src, warn|attack) = \frac{\Pr(src, warn, attack)}{\Pr(attack)}$.
- $\Pr(dst, gossip|attack) = \frac{\Pr(dst, gossip, attack)}{\Pr(attack)}$.
- $\Pr(attack)$ is the attack ratio at this node.

All in all when an attack occurs, one can expect the following three phases, which we have indeed observed in our simulations:

- A ramp-on phase, where the malicious load increases abnormally. The attack is not yet detected but the of gossips increases. Nodes aggregate all the knowledge about neighborhood traffic against the victim.
- The detection and reaction phase. The number of gossips triggers some classifiers and attacking traffic is increasingly. Traffic decreases but gossips and warnings keep circulating in substitution of the attack load (volume of gossips and warnings is extremely lower than the attack volume).
- The stabilization of the attack. Warnings arrive to bordergate nodes, and the attack is stopped very close to the border. Gossips keep all the bordergate nodes informed about the continuation of the attack.

5. EXPERIMENTS

In this section we describe our experiments in this framework. The goal of these experiments is to check that the detection and classification algorithms behave as we expect, and to test the effectiveness of the method at detecting and stopping attacks early.

For the detection algorithms, we will trace the volume of traffic produced during a normal status and during an attack, to check the evolution of gossip messages and confidence, and the relation with the volume of normal traffic. For the classifier we will check the effectiveness of this during the whole test and monitor both true detections and false alarms (false positives).

5.1 Environment

For the experimentation, a simulated network using the *Objective Modular Network Testbed in C++* (OmNET++) has been built. We use the OmNET online simulator because of its generic and flexible architecture, and the capability of simulating real networks with maximum detail, controlling time variables and having implemented protocols from each TCP/IP stack layer. We have used the high-level topology of the network of the Technical University of Catalonia, and as traffic model we have used logs of normal HTTP 1.1 sessions from different entrance point of the network to an HTTP server inside. The network will run in normal status until a distributed denial of service attack begins, flooding the path to the service and interfering with normal connections. Also, we have equipped the intermediate network nodes and the bordergate nodes with the detection mechanisms and the classifiers described in the previous sections. The paths clients-to-service are shown in Figure 2.

For simplicity, we have chosen a single destination (service) as potential victim (on the left of the figure), and view the rest of the network as a tree-like structure from a bunch of clients (on the right of the figure) to the service. Therefore, the distributed-attack critical points, where the attack will be detected first, will be those closest to the service. The network consists of a gigabit ethernet. It contains the bordergate routers, the access points to our network, as well as intermediate nodes connecting these access points to services. Each client can be understood as an access point for several clients, and not only as a single and final client, just like the attackers can be understood as access points for several attackers. The gossiping level for our test is just gossiping the confidence to the successor node in the path to

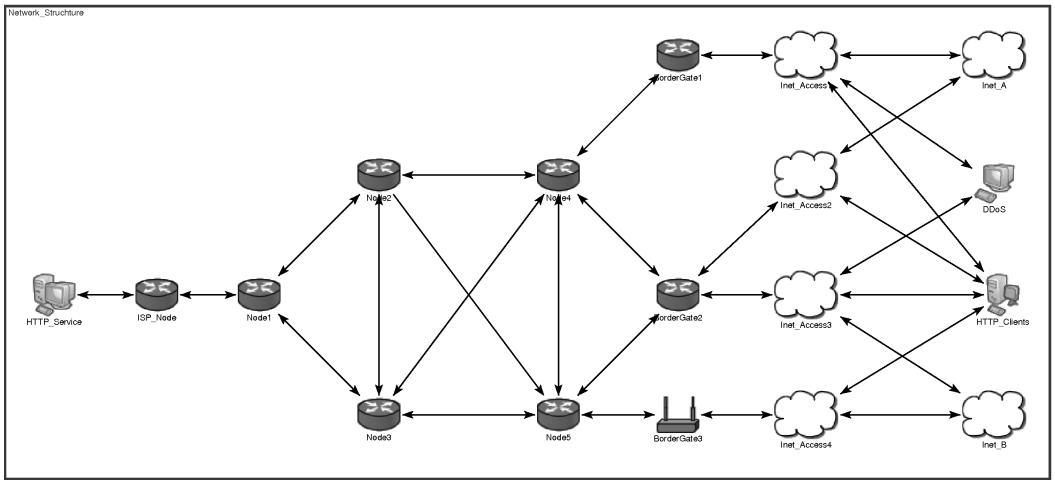

Figure 2: The Network Structure, with the victim service, intermediate network, exterior network, and clients and attackers

the victim, and also for the warning level, sending warnings only to the previous node. Also, clients have been modeled as HTTP clients, with variable thinking times and session times. The attackers are configured to make requests continuously without waiting any response, with a reason of 500-1000 attack packets against each legitimate package.

5.2 Experiments and Results

For these experiments, an attack will be performed over the testbed network. We will follow the pattern described next, which is quantitatively similar to the patterns we have observed in networks within our university and roughly agrees with those described in, e.g., . When no attack occurs, clients are mainly HTTP 1.1 and HTTP 1.0 behaved clients, with an average of 350 bytes per request, thinking times between 4 and 7 seconds, and idle times of 300 seconds. At a certain moment, we simulate the start of an attack: HTTP 1.0 flooders will start flooding the network with requests towards a specific victim (a generic HTTP-web server, with an average of 2000 bytes per response), as many as 500-1000 times the usual number of requests directed to it; this factor is, again, consistent with our experience and with . Once the attack is performed we will measure the number of packets that reached the target, and how many of them where attackers. As the classifiers discrimine the traffic packet by packet, we are interested in the number of attacking packets reaching the target and the number of non-attacking packets that are misclassified and discarded.

First of all, once the system has been set up with the detection and information sharing devices, it must be trained. For this learning phase we will run the network with well-behaved client request streams to collect non-attacking data, and some attacks to collect the attack behaviors to be modeled. The advantage of using OmNET is that this toolkit comes with usual client and service patterns, and these patterns can be adjusted to HTTP, FTP and other existing or wanted models. Also, random factors can be added in order to randomize in a common range some client and service variables like thinking time, start time, request sizes, intervals between sessions, and more, so each run becomes lightly different letting us to not overadjust our models and not being restricted to a only one concrete situation. So, training must be done while gossips are enabled, and warnings should be set temporarily disabled (we are not depending on classifiers yet). Each node will collect information from each packet about the source, the destination, the confidences to this destination and the real class of the packet (attacker, non-attacker), in order to train its classifier.

Looking at the results of the Naive Bayes machine learner, it can be observed what was expected: First, the attack probability for warnings increases with the number of warnings, while the probability of non-attack decreases attack. Second, for a low number of gossips, the probabilities remain stable until a relative high number of gossips, where the attack probability increases, becoming zero the no attack probability. Also, the source and destination conditional probabilities follow the expected behavior too. The fact that each attribute has its own probability chart will ensure to us that the system will not only memorize the attack destinations for classifying traffic.

For traffic behavior, Figure 3 shows the mean number of received messages for each node. Comparing the network with and without discrimination mechanism at the nodes, we can observe that messages are reduced during the attack, except on the bordergate nodes, where all attack arrives but does not pass through. Also, bordergate nodes receive the extra amount of messages referring to the retransmission from rejected messages. Evidently a gossip message can not be compared to the flooding messages. And for the gossip system, we tested the behavior of the algorithm in front of usual traffic and attacks, and it was just as expected too. Also the accumulated traffic behavior is shown, and we can observe that without the mechanism, during an attack all nodes receive attacking traffic, while using the classification mechanism only the bordergate nodes receive its impact, but stop it almost completely, thus freeing the intermediate network and the victim server of being flooded.

In Figure 4 we show the evolution of attack confidences (level of gossip) over time. The two plots in each represent the accumulated traffic at bordergate nodes and at interme-

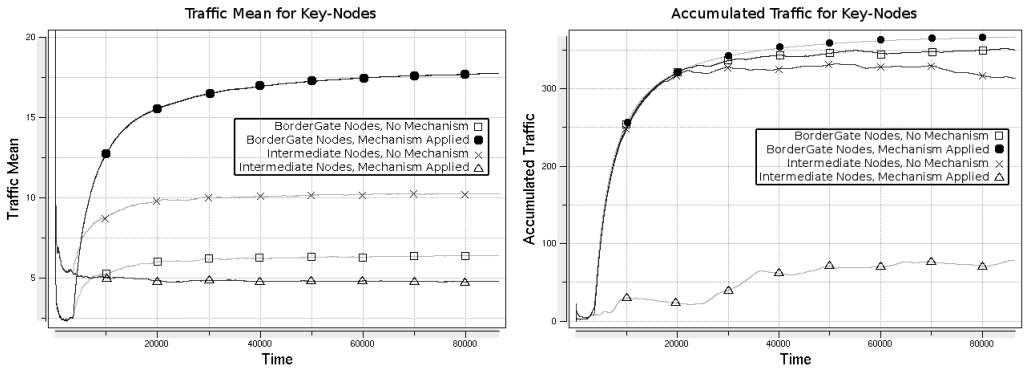

Figure 3: Messages received and Aggregated Traffic for each node

diate nodes. At time 400, an attack starts flooding the paths to the target service. The level of traffic at the bordergate nodes remains high until the end of attack. Without the detection mechanism, it can be seen that the accumulated traffic at intermediate nodes is essentially the same as that received in the bordergates, since no messages are blocked. On the other hand, with the detection mechanism, the traffic seen by the intermediate nodes is a fraction of the traffic at the bordergate nodes.

Finally, for the machine learner, observing the results of classification we can see that the NaiveBayes classifier has high accuracy (above 95%) and all classification mistakes are false negatives, that is, some attacking messages are let through. We observe no false positives: when no attack is occurring, all traffic arrives to the destination service. Moreover, most of the mistakes in one node are corrected in the next node in the path to the victim.

As a result, our experiments have demonstrated that detection and classification systems work as well, stopping all flooding traffic as close to the source as possible, freeing the intermediate network of flooding traffic and preventing service overload during the attack. The classifiers classified flooding traffic with accuracy above 95%, with no false positives. The false negatives (attacking messages not classified as such) at the bordergate nodes are stopped in the intermediate network, but far enough from victim service. More precisely, in these experiments less than 1% of the attacking messages arrived at its destination.

6. CONCLUSIONS AND FUTURE WORK

In order to avoid distributed denial of service attacks and flooding attacks or abuses, we have presented a mechanism based on sharing information and machine learning, which enables the network to stop these attacks early and close to the sources. Previous approaches that we are aware of used static techniques or adaptive techniques based on collecting information and treating it with predefined, pre-tunes, models. In our approach, a learning component lets the system create, adjust, and renew the behavior models. Each element of the network learns from its local traffic patterns and shares this information with the other elements so that each one has aggregated information about the network. This information is collected in a local model or classifier. At prediction time, again information about the state of the network circulates among the nodes, but the information is this time passed through the classifier, which determines with high confidence whether messages for a given destination belong to an attack.

We have tested the method simulating the topology of the Technical University of Catalonia and real volumes of attacking and nonattacking traffic with OmNET++ and attacking it. Our mechanism produces good results: attacking traffic is identified as such with accuracy over 95%, and most of it is stopped essentially at the bordergate nodes. Less than 1% of the unwanted traffic arrived to the victim server, thus essentially posing no thread. When no attack is taking place, all of the legitimate traffic arrives to its destination.

There are a good number of issues to be investigated further. One could of course try more advanced methods for message classification. Given the very sketchy information we now consider about each message (source, warnings, destination, and gossips) there is little point in trying to improve on the (extremely high) accuracy. Observe that one of the main drawbacks of our approach so far is that, when a service is under attack, *all* traffic to it is blocked. This is unavoidable if only the traffic volume is taken into account. So the problem of interest is to open up the messages and use information the packets inside for finer classification, that is, really discriminating on a packet-per-packet basis whether it is legitimate or malicious. Also this will help to reduce false positives when a legitimate user has an infected computer performing an attack, and increase the detections when attackers use evasion techniques like spoofing or bouncing. This classification problem has of course been studied elsewhere; our goal in this work was, for the moment, designing and validating the information-sharing scheme.

Another issue before the system becomes practical at large networks is that of memory usage at each node. Right now, each node keeps a table of all source and destination nodes it has seen. This is all right for nodes of moderate size, where the number of access points and destination services is in the order of, say, hundreds. For large-scale networks, this is of course impossible. We plan on using so-called *sketching* techniques for keeping only information about those addresses generating and receiving the highest volumes of traffic at any given period, which are precisely those that are involved in flooding attacks.

7. ACKNOWLEDGMENTS

This research work is carried out in part under the FP6 Network of Excellence Core-GRID funded by the European

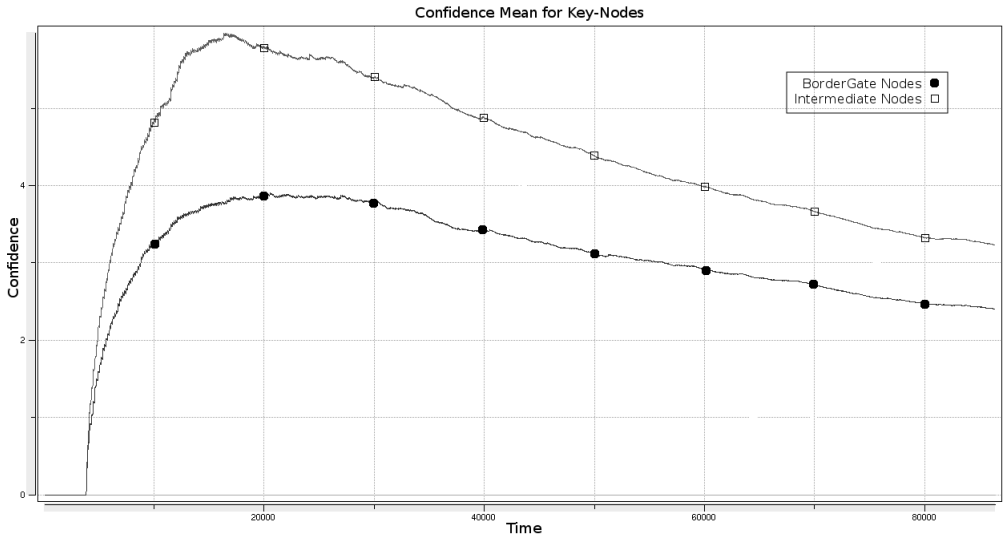

Figure 4: Evolution of Confidences at nodes

Commission (Contract IST-2002-004265). Also this work has been supported by the Spanish Ministry of Education and Science (projects TIN2007-60625). Professor R. Gavaldà is partially supported by the EU PASCAL2 Network of Excellence and by the DGICYT MOISES-BAR project, TIN2005-08832-C03-03.

8. REFERENCES

[1] N. S. A. Quiroz, M. Parashar. Decentralized clustering analysis and online anomaly detection for peer grid systems. *Technical Report, CAIP Rutgers, 2006*, 2006.

[2] D. Dittrich. The dos project's trinoo distributed denial of service attack tool, October 1999.

[3] T. M. Gil and M. Poletto. MULTOPS: A Data-Structure for bandwidth attack detection. In *Proceedings of the 10th USENIX Security Symposium*, pages 23–38, 2001.

[4] P. Jelena and M. Greg. Attacking ddos at the source. In *Proceedings of the IEEE International Conference on Network Protocols10 2002*, 2002.

[5] F. Kargl, J. Maier, and M. Weber. Protecting web servers from distributed denial of service attacks. In *World Wide Web*, pages 514–524, 2001.

[6] A. Keromytis, V. Misra, and D. Rubenstein. Using overlays to improve network security. In *Proceedings of SPIE ITCom Conference on Scalability and Traffic Control in IP Networks II 2002*, 2002.

[7] Y. W. M.S. Srivastava. Comparison of ewma, cusum and shiryayev-roberts procedures for detecting a shift in the mean. *Annals of Statistics*, 21:645–670, 1993.

[8] S. Noh, C. Lee, K. Choi, and G. Jung. Detecting distributed denial of service (ddos) attacks through inductive learning. In *IDEAL*, pages 286–295, 2003.

[9] R. Nou, J. Guitart, V. Beltran, D. Carrera, L. Montero, J. Torres, and E. Ayguade. Simulating complex systems with a low-detail model. In *Proceedings of the 16th Paralelism Meeting 2005, Spain*, 2005.

[10] N. Poggi, T. Moreno, J. Berral, R. Gavalda, and J. Torres. Web customer modeling for automated session prioritization on high traffic sites. In *Proc. 11th Conf. on User Modelling (UM2007). Springer Lecture Notes in Computer Science 4511*, pages 450–454. User Modelling Inc., March 2007.

[11] K. Rieck and P. Laskov. Language models for detection of unknown attacks in network traffic. *Journal in Computer Virology*, 2(4):243–256, 2007.

[12] W. W. Streilein, D. J. Fried, and R. K. Cunningham. Detecting flood-based denial-of-service attacks with snmp/rmon. In *Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, Fairfax, Virginia, USA*, 2003.

[13] A. Varga. The omnet++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference*, pages 319–324, Prague, Czech Republic, June 2001. SCS – European Publishing House.

[14] H. Wang, D. Zhang, and K. Shin. Detecting SYN flooding attacks. In *Proceedings of IEEE INFOCOM 2002*, 2002.

[15] S. Williams, B. Parry, and M. Schlup. Quality control: an application of the CUSUM. *British Medical Journal*, 1992.

[16] G. Zhang and M. Parashar. Cooperative defense against ddos attacks. *Journal of Research and Practice in Information Technology (JRPIT), Australian Computer Society Inc.*, February 2006.