RIP – A robust IP access architecture

Djamel H. Sadok, Eduardo Souto, Eduardo Feitosa, Judith Kelner, Lars Westberg

1. Introduction

For several years, Internet security research has been of the view that protecting data could only be obtained when using known techniques such as digital signatures and encryption. Such oversimplified assumption has been abandoned since it assumes that the devices responsible for encrypting, forwarding, and sending packets are themselves trustworthy and fails to consider broader security issues (Chakrabarti and Manimaran, 2002). On several occasions, network infrastructure (e.g. routers, links, and servers) has been compromised by malicious adversaries or even people simply seeking a new challenge, according to reports made to the CERT advisories (CERT Coordination Center, 2007). Pervasive connectivity and the wide range of new Internet services have also given technically advanced intruders the opportunity to carry out a variety of increasingly sophisticated and complex malicious attacks, thereby threatening the integrity of its infrastructure and violating user privacy. The days that perimeter security was seen sufficient to protect business assets are gone.

There are an overwhelming number of strategies proposed in this domain comparing them is often pointless as they differ in scope and concern. They can be seen as point actions that target specific problems. Although mostly good at their tasks, they miss the benefits of the big picture. A well thought of arrangement of these established and tested building blocks, not only alleviates the user from dealing with individual technologies, but has shown to be more efficient than the sum of isolated contributions. Security is defined in this work as a multi-stage and collaborative effort where each security module, or sensor, specializes in a given area of expertise. The opinions of such modules should be opportunistically sought and a smart decision process must be used to flag and report security problems. An analogy could be that of a patient seeking second and third opinions about a health issue, one who many times is directed to specialized medics...
The present work builds an architecture that mitigates some of these known Internet threats. We neither claim that it is complete nor that it is the only way for achieving similar levels of protection. We do however see it as a step in a promising direction.

1.1. Threats to our security

It has been a cat and mouse game all along the last decades where we are still learning new lessons sometimes costly ones. Security management cannot be seen as a static process. The same administrator utilities were also used as the first attack targets in order to avoid detection. While often packaged and shared with other attackers, they would allow even those with little knowledge of malicious techniques to perform harmful security breaches. Virus writers sent encrypted email viruses using metamorphic engines to change their syntactic properties of decryption routines in order to avoid detection. Some of these viruses known as blended threats launched multiple types of attacks once inside a system.

Security threats have taken a number of facets including: information theft, system attacks that exploited vulnerabilities in used encryption techniques, address and domain spoofing, fishing and identity theft. To handle these problems, a number of works have pointed to what they called best practices for controlling the impact of such threats. Among these they highlight the need to maintain antivirus software up to date, filter out inappropriate content including viruses and spam email, adequate configuration of firewalls to prevent the exploring of back doors or launching DoS attacks, the establishment of selective trust relationships and common black lists, subscribing to news from security monitoring centers, the use of intrusion detection software and to specify and deploy effective security policies to control access to resources. Users interested in finding out more on security attacks and possible solutions to them should consult (Peng et al., 2007; Mirkovic et al., 2005; Mirkovic and Reiher, 2004).

1.2. A robust view

Pioneering IP Networks architectural design had given little interest to security problems in the first place. Its resilience was limited to physical attacks to its routers which lead consequently to the Internet adopting a datagram type of subnet. Unlike telecommunication networks, IP networks do not distinguish between the traffic used for controlling communications (signaling traffic) and data traffic (or user traffic). This somehow purposely adopted naive and simplistic design turns IP networks very vulnerable in nature, prone to various types of abusive attacks and their signaling open to user manipulation and misuse. TCP-SYN and ICMP based attacks are examples of such exploits.

It is clear from both the range and sophistication of the existing threats that only a well drawn and unified collaborative approach which coordinates the actions of antivirus, firewall and IDSs, among other mechanisms, is capable of better securing our assets. In other words, security mechanisms not only need to be present in a solution but equally importantly, they need to operate in a synchronized manner to be more effective. This view is at the heart of our RIP approach. Note that scope for such collaboration can be: a) local to a corporate network, domain or institution; b) commonly shared among different networks that have the same security interests while also agreeing on some basic security policies and a code of conduct through established policy level agreements; c) finally, one may see such collaboration extended to a global Internet basis, where common efforts may be harmonized to identify, fight against and mitigate security threats as early as possible. Web services may be a key facilitator for such coordination. In this work, we limit ourselves to the local scope.

A new access architecture designed to turn access control more robust than it is today is described. Robust IP looks at new access mechanisms capable of increasingly tightening network security without ignoring performance and scalability. In order to achieve a flexible architecture, we have adopted a policy-based approach as a means of implementing run-time adaptive systems for the management of Internet services, networks, and security systems.

We chose to focus on network access infrastructure rather than the routed backbone as changes here are easier and more effectively introduced by the owners of such networks and end systems. A number of key technologies and practices are considered as part of the proposal.

RIP looks also at the cooperation between a host and a network it accesses as a means to enhance security. Checking the software and hardware clearance of a host, a server or endpoint, and ensuring that it is in compliance with current corporate or domain specific security policies is a first step towards fighting some of these threats. Such access control is often distributed across many heterogeneous components such as firewalls, databases, filtering routers, and switches.

Further, security has traditionally been seen as an important functional area in network management and with the advent of Policy-Based Management (PBM) some new security work has also followed suit. PBM provides a way to allocate and monitor network resources according to defined network policies. Policy definitions are an answer to relatively high level questions such as: Who can access which resources on the network? A policy-based management system is governed by a set of rules that determines the action course to be taken based on some conditions (Verma, 2000). Fundamentally, a network policy consists of rules and controls that determine network operation. The evaluation of a policy is triggered by an event which results in a policy decision being enforced on specific network device(s) or services. Policies are declarative, i.e., they can be adapted at run-time to flexibly control system behavior and are therefore becoming increasingly popular in adaptive, run-time configurable networks and information systems. The IETF lead the way and defined a complete PBM framework (Westerinen et al., 2001).

In the context of admission control, policy-based admission control was generated by the need from network managers and service providers to monitor, control, and enforce the adequate use of network resources and services. The resulting action may be admitting a user into the requested network service, user forwarding to a different one, access denial, among other possible outcomes determined by the network policies in place. Access policy parameter values...
include user identifier(s) and applications, user preferences and credentials, traffic/bandwidth requirements, security considerations, time-of-day/week, access interface used, etc.

The RIP architecture considers policies as an important concept to build upon its security decisions and control of its resources.

2. RIP architecture

The RIP architecture is designed to be modular, to ease future modification and the seamless addition of new components. As shown in Fig. 1, the architectural components were broken down in accordance with their roles.

The main entities that compose a RIP access network are:

- **End Point (EP)** – represents any endpoint device or system used to gain access to an IP network.
- **Policy Enforcement Point (PEP)** – the system entity that performs, a la IETF, access and traffic control, by making decision requests and enforcing them.
- **Traffic Management Module (TMM)** – the system entity that characterizes and separates legitimate from non-legitimate user traffic using traffic classification algorithms and heuristics.
- **Policy Management Module (PMM)** – at the heart of RIP, there is this entity that issues decisions and actions based on information obtained from other components of a network. It is responsible for evaluating applicable policies and returning a decision to continually ensure network security. This entity is local to its domain. In future extensions of RIP, it may be distributed across different networks to increase its response effectiveness.
- **External Services (ES)** – represent additional services that are integrated into RIP. Examples of these services are AAA (Authentication, Authorization, and Accounting), a policy editor, log files and remediation systems. The latters are responsible for checking user devices to guarantee that they have the latest security patches and antivirus software applied to them.

We believe that this collaborative communication style between the components represents a significant step in the direction of self-defending networks and innovation in the area. A standard multivendor event logging and interchange format is needed to enable seamless interworking and facilitate deployment, see Section 2.4 for more information on this.

2.1. Policy enforcement point

The Policy Enforcement Point (PEP), an IETF defined concept, can be seen as a front-end agent that represents where the policy decisions are actually enforced. The PEP can be any network element that interfaces with the endpoints, for example, a switch, firewall, router or even a VPN system.

Within the RIP architecture, an endpoint must identify itself to the PEP system associated to it before obtaining access to network resources. The PEP system then consults the policy

Fig. 1 – RIP architecture overview.
management module to know whether this particular device is allowed to join the network, what services it is able to use, and what control already it exerts or wants to exert over it. Through PEP (our man on the field), RIP enforces security policies and ensures that each user attached to its network is compliant, protected, and safe to a degree. The criteria for allowing local or remote endpoints to enter the network are specified using access control policies that are managed by the PMM and usually represented using the eXtensible Access Control Markup Language (XACML), an XML access control extension.

2.2. Traffic management module

The main goal of this module is to identify and ultimately notify the likely existence of any anomalous traffic behavior to the PMM module. As shown in Fig. 2, this module is composed of traffic capture and a set of classification components.

The traffic capture component is responsible for capturing and aggregating packets based on flows. These flows will be fed into the classification component which will be responsible for identifying any anomalous behavior based on known statistical or mathematical behavior.

When abnormal traffic is detected, the PMM is then notified of this state. It starts assessing other indicators to help reaching a correct decision capable of mitigating attacks or reacting to them as stipulated by the network policies. As part of TMM, the authors have adopted a methodology for profiling Internet backbone traffic similar to one suggested in Xu et al. (2005b) and Crovella et al. (1998). A comprehensive overview of the main points of this methodology is presented in Section 3.3.

2.3. Policy management module

Network configuration can no longer be achieved by individually configuring each network element, but rather by specifying the configuration and service policies that act over the entire network at a central element (Verma, 2000). Manual device configuration remains prone to human errors and does not scale well. Hence core to the RIP architecture is its Policy Management Module (PMM). Given the knowledge of network state at hand, the PMM is able to take independent decisions based on predefined policies. As a result, the PMM is required to maintain communication with all RIP components.

Examples of the main actions taken by a PMM include:

- Locating and evaluating a set of rules that is applicable to a given PEP being managed by this PMM.
- Keeping a register about which endpoints (EPs) are associated to what PEPs (e.g. for the session control).
- Monitoring the network, listening to events triggered by other components or network elements (for example, recent events registered by the log system such as router down events, line failures, etc.).
- Enforcement of new policies based on information sent by the TMM module or by external services (for example, a device diagnosis submitted by the remediation system).
- Keeping a policy database updated and notifying PEPs of policy changes.
- Resolving policy conflicts when possible, and reporting them to a policy management console and administrator(s).

Furthermore, PMM can also be responsible for coordinating the execution of other functions/services such as: AAA (authentication, authorization and admission control), bandwidth management, event logging, remediation actions, firewall setup rules, quarantining, etc. Both centralised and distributed solutions may be used to implement RIP.

2.4. Components interactions

The different RIP components should be capable of correctly exchanging and interpreting information. The question is how to deal with a legacy of “Babel” languages provided by a variety of independent solutions (switches, routers, firewalls, remediation system, IDS, etc.) often from a multitude of vendors. In an attempt to help dealing with this problem, a number of policy languages have recently been proposed. Examples of these are XrML (ContentGuard Inc., 2001), PONDER (Damianou et al., 2001), XACL (Kudo and Hada, 2000), SAML (OASIS, 2005a), XACML (OASIS, 2005b), and ODRL (Open Digital Rights Language Initiative, 2001). RIP uses the XACML for specifying control access policies and managing access to resources.

![TMM components](image)
XACML is an XML-based standard or DTD (document type definition) that provides a policy language and an access control decision and response language for managing access to network resources. It is designed to provide a universal language for authorization policy to enable interoperability with a wide range of administrative and authorization tools. By doing so, it allows for security policies to be applied consistently across different domains and vendor products. The current XACML standard is in version 2.0 (OASIS, 2005b), although a working draft of the version 3.0 is under way (Oasis, 2005c).

During our initial tests, some limitations involving the use of the XACML standard language within the context of RIP were detected. First, the XACML communication model only permits synchronous messages in the form of a request and response communication. However, some scenarios require that the interaction between RIP components must be conducted asynchronously. Second, the existent XACML policy-combining algorithms are unable to deal with different levels of policy priority. Consequently, RIP adopts some important extensions to the XACML communication model to enable it to accept asynchronous messages (notification messages) and defines a new priority-based policy-combining algorithm. Note that such extensions may also be useful for other application domains that use policy-based management in general.

The main components of proposed architecture (TMM, PMM, and PEPs) were naturally designed to work in a distributed manner preventing single points of failure, nonetheless within a single corporate network. For example, TMM heterogeneous sensors may probably sit at different network segments. Furthermore, a fault tolerance scheme may be adopted by replicating some of these modules (within clusters) to prevent possible single points of failure.

Together, the RIP components stand a better chance in facing the varying threats. Every element in the network acts as a point of defense. Fig. 3 provides a snapshot of the collaboration potentials between the components of the RIP architecture.

Under the RIP architecture, all elements act as sensors able to supply the policy management module with relevant information and help the latter reaching the right decisions in a timely manner. For instance, during the host admission process, information about the software installed in the client machine needs to be sent to PMM to provide health policy validation. PMM then carries out sanity tests, to determine ongoing health policy compliance, and decide if limited network access is necessary to enforce. As an example, Bell Canada and other ISPs are deploying “Walled Gardens” – VLAN for quarantining an infected host into its own little sandbox from where it can access only windows update, anti-virus update sites and the ISP’s support pages, nothing else, on any port. During a security breach situation, system log information, router and traffic management mechanisms can be required to elaborate the policy enforcement decision.

2.4.1. A collaboration example within RIP
We present the interaction between the traffic management module, policy management module and firewall implemented in our RIP prototype. Fig. 4 draws the overall information exchanged between these network elements.

The TMM components, and associated traffic pre-processors or sensors, namely, Analyzer-PX, Profiling, and ChkModel, periodically exchange information about the two way

![Fig. 3 – Collaboration among RIP components.](image-url)
traffic. For instance, the TMM continuously sends to PMM information in the form of a sorted list containing the IP addresses that have recently gained access to network resources. This list indicates if the clients are well behaving, suspicious or bad according to the classification method implemented by sensors such as Profiling, ChkModel or other traffic processors that may be added to the RIP architecture in subsequent releases. A close examination of security attacks has interestingly shown that only 0.6–14% of IP addresses encountered in a DDoS attack had some sort of connection with the victim in the past. This work dwells on this idea to build the Trusted IP List (TIL). This information, as well as the one exchanged between RIP components, is described using an extension of the XACML language with added support to asynchronous messages. A snapshot of these messages generated by Profiling and ChkModel components is presented in Fig. 5.

Anyone of these notification messages triggers the processing at the PMM of information associated to hosts included in the alert message and a decision is made on whether or not to include the target IP address in the TIL. During an attack situation, this list will be used by the PMM module to assist in taking future decisions. Enforcement messages elaborated by the PMM are sent to Policy Enforcement Points for execution. Next, we present a brief description showing the processing carried out by the policy decision point as part of the PMM, seen as the brain behind RIP.

2.4.2. Policy decision point processing

PMM information processing could follow finite state machines such as the finite state automata, Markov chains, or stochastic regular grammars. In our RIP prototype, we used a finite state machine to correlate the information received by TMM components in possible states that a network client could take. These states could be used in future enforcement. Table 1 shows the machine states that can be assigned to IP addresses.

There is a straight relation between the state machine and TIL since the classification stored into TIL is influenced by changes in the machine state. Table 2 illustrates this process.

As observed in Table 2, there are two basic operations carried out on the TIL table. The set operation that assigns a classification to an IP address and remove operation that removes an IP address from the TIL.

The main idea consists in attributing to an IP address a classification in accordance with the information sent by TMM components. The trust level is useful during policy enforcement. For example, a network administrator can establish bandwidth distribution policies in conformity with the trust level of IP addresses. Fig. 6 depicts the mapping between the states assigned to an IP address and the RIP policy enforcement.

When under attack, the PMM interrupts any changing states and takes the following actions: 1) it requests to TMM a list containing all IP addresses from the moment that this attack was detected (i.e. a recent network snapshot). 2) The addresses not stored within TIL will have their traffic limited or even blocked. 3) The next step is to walk through TIL while assigning to each IP address a bandwidth limit according to its current state (traffic shaping action). The following section gives more details on the interaction between PMM and PEP enforcement.

2.4.3. Policy enforcement point

In order to put in practice the decisions taken by PMM, we built an enforcement agent that translates the PMM decisions to the particular native language of the PEP responsible for applying policies such as firewalls, routers and information
sent by PMM consists of a high level description of the decisions (written in XACML). In practice, a decision usually is composed by IP addresses and an action identifier for each address. For example, in our RIP prototype we assigned actions related to network access that are applied at a firewall. Possible values to the action identifier are: Limit, Extreme-Limit and Block that will be translated by the agent as different degrees of bandwidth shaping for a set IP addresses. An example of the enforcement message sent by PMM is showed in Fig. 7.

Fig. 5 – XACML messages sent from TMM to PMM. (a) Alert notification sent by the profiling component to the PMM. (b) Alert notification sent by ChkModel traffic sensor to PMM.
When receiving an enforcement message, the firewall agent translates it to firewall rules as shown in Fig. 8. In the first rule the IP address 192.168.0.14 will suffer a bandwidth limitation whereas the second rule generated by the agent request blocks all traffic from host 192.168.0.32.

### 3. Implementation

An initial RIP prototype has been developed. It consists of the three main modules, namely, context handler for enabling the communications with PEPs and other network components, a policy management module, and a traffic management module.

### 3.1. Context handler

The RIP context handler is responsible for constructing XACML messages containing also additional information, such as subject, action, resource and environment related attributes. Then, the XACML message is transmitted to the PMM, which decides upon which actions must be taken. For example, Fig. 9 depicts the context handler used in the access control process. In order to guaranty security and reliability, RIP employs industry standards, such as the IEEE 802.1x (IEEE), IETF RADIUS (Rigney et al., 2000) and the IETF EAP (Aboba et al., 2004) protocols for host access negotiation with network devices.

The context handler is located between PEP and PMM and is responsible for translating between RADIUS and XACML messages.

<table>
<thead>
<tr>
<th>State</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Set address as &quot;Good&quot;</td>
</tr>
<tr>
<td>B</td>
<td>Set address as &quot;Transient&quot;</td>
</tr>
<tr>
<td>C</td>
<td>Set address as &quot;More-Transient&quot;</td>
</tr>
<tr>
<td>D</td>
<td>Remove from TIL</td>
</tr>
</tbody>
</table>

Table 1 - Representation of information sent by TMM components to PMM in states.

<table>
<thead>
<tr>
<th>Profiling</th>
<th>ChkModel</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>Good</td>
<td>State-A</td>
</tr>
<tr>
<td>Good</td>
<td>Transient</td>
<td>State-B1</td>
</tr>
<tr>
<td>Bad</td>
<td>Good</td>
<td>State-B2</td>
</tr>
<tr>
<td>Good</td>
<td>Bad</td>
<td>State-C1</td>
</tr>
<tr>
<td>Bad</td>
<td>Transient</td>
<td>State-C2</td>
</tr>
<tr>
<td>Bad</td>
<td>Bad</td>
<td>State-D</td>
</tr>
</tbody>
</table>

Table 2 - Operations performed in TIL according to PDP States.

![Fig. 6 - Mapping states and policy enforcement decisions.](image)

![Fig. 7 - PMM message to a PEP.](image)
pass out from 192.168.0.14 to any queue(limit_queue)
block in from 192.168.0.32 to any

Fig. 8 – Example of a firewall translation.

3.2. The PMM server

In order to increase code portability, the PMM has been implemented using Java and XACML technology. The internal PMM architecture basically consists of a module responsible for communicating with the other architectural components and by a Policy Decision Engine (PDE), whose role is to determine which policies are applicable to a specific or set of network elements.

The communication model employed in a RIP PMM server is based on request and decision (used for example, to consult other PEPs and give endorsement through policy decisions). However, it is stateful and thus may make decisions even if no requests are received, i.e. it additionally employs the communication model based on notifications. For example, the RIP traffic analyzer tool can unexpectedly send notifications about attack attempts generated by an end system within the network.

Communication with other architectural components is achieved using two specific interfaces implemented through transport sockets, one of them is used to wait for requests and the other one for notifications. The process of parameter marshalling and unmarshalling is achieved whenever information arrives at one of these sockets. Next, the objects are forwarded to the PDE.

The PDE evaluates the policies stored in the policy repository, determines which policies are applicable, and returns a response (in case of the request) or a decision set (in case of the notification) to the PEP(s) associated to the received requests or notifications. When a request is sent to the engine, both PEP and user information are extracted from the request message to carry out some verifications. An example of such verification is to determine if the PEP is a valid entity, the engine maintains a register of all PEPs in its network. Another offered PDE service is maintaining a control of the sessions in progress.

In the present prototype, policies are stored in a lightweight directory service (LDAP directory), we designed a common format (scheme) that is used to specify policies stored in the directory. This scheme defines a representation of the XACML objects that can be created within the directory. Furthermore, we also created a simple management tool that converts the abstract policy into entries that can be populated into an LDAP Directory. Once the LDAP entries are written into the directory, the PMM can gain access to them, read the entries and determine the appropriate policy to be applied. Discovery of applicable policies in the PDE is made by a “PolicyFinder” application. Basically, it retrieves specific policies in a policy repository. This mechanism is modular and supports both access and control policies.

3.3. The TMM server

The TMM server is a key component to the RIP architecture. It has the responsibility of capturing raw traffic information, processing, and extracting data related to suspicious and anomalous events. As depicted in Fig. 2, TMM is composed by both traffic capture and different classification modules. We developed a capture tool, named Analyzer-PX, responsible for packet capture and flow aggregation. This tool is written in C++ and makes use of the known packet capture library (PCAP) that provides a high level interface to packet capture systems (PCAP Public Repository).

The Analyzer-PX has two basic roles: packet collection, which can be done in either on-line or off-line modes; and the aggregation of frames into flows. We define a flow as a unidirectional series of IP packets with the same source and destination addresses, port numbers and protocol number. The Analyzer-PX works by dividing time into administrative time slots represented by ta. This is used to check if flows have terminated their transmission and copies these from data structures (in the form of hash tables to speed access to them) into files. In our experiments we used a ta of 60 s. The flows are stored in a local file system using a binary format. In order to ensure compatibility with other traffic capture tools, the Analyzer-PX is also able to convert these binary files into a text format.

3.3.1. Profiling anomalies

The classification module has the role of automatically digging through massive traffic and to provide plausible means to understand and quickly recognize anomalous traffic events. In order to achieve this goal, we have implemented a methodology for profiling Internet backbone traffic similar to the one suggested in Xu et al. (2005a). This methodology (herein called profiling method) works by examining communication patterns of end hosts (addresses and ports) that account for a significant number of flows in a time period.

The profiling method relies on information-theoretic techniques to extract and classify flows. In particular, this approach places great emphasis on the entropy concept. Entropy can be seen as the measurement of information of a given dataset, which essentially quantifies “the amount of uncertainty” contained in that dataset (Shannon and Weaver, 1949).

Fig. 9 – Context handler.
We could use different metrics to calculate the entropy in our methodology, such as the quantities of flows and bytes. Our profiling approach implements an identification methodology, which consist of four related basic stages:

1. Preprocessing: deals with the definition of the domain data that will be processed. In other words, in this stage the packets are captured and aggregated into flows. In our experiments, these tasks were carried out by our home grown DPI level traffic capture and aggregation tool, known as the Analyzer-PX;
2. Extracting significant clusters: determines the clusters for four features or dimensions. This procedure aims to reduce and facilitate dataset behavior inspection through the identification of its most significant or principal elements. The extraction of significant clusters deals with a four dimensional feature space composed by the four attributes srcIP, dstIP, srcPrt and dstPrt. Considering these elements, we can identify two relevant types of network communication behavior. Firstly, there is a relationship between IP addresses (srcIP and dstIP), one that determines the communication pattern between hosts. Secondly, there is also the behavior built from port/service (srcPrt and dstPrt) usage patterns.
3. Clusters classification: classifies each cluster’s element into behavior classes based on similarities and dissimilarities of communication patterns (ports and IP addresses).
4. Communication patterns interpretation: defines a set of behavior classes capable of better describing given applications and services. According to the results presented in Section 4, we have been able to build a small set of behavior classes that represent anomalous traffic.

Fig. 10 shows the interconnection among the four stages and how they are fed and interact. This process allows an automated or supervised adaptation of parameters. For instance, the iterations allow that information coming from communication patterns interpretation to affect the decisions taken on preprocess stage.

After traffic classification, the TMM sends to the FMM a file containing IP addresses, port and class behavior. This information is processed by the FMM and used according to existing policies to be enforced by RIP.

3.3.2. Check model
ChkModel can be seen as another point of defense between the attackers and their DDoS victims. It monitors and polices the incoming and outgoing traffic from the network. Placing the defense at the border exposes the defense system to moderate-to-high traffic volumes, thus enabling control of the majority of DDoS attacks.

The ChkModel sensor was designed to distinguish between well and badly intentioned traffic, and also to identify possible service resource problems. It observes the total traffic between clients and servers, at connection and socks granularity. A connection is represented by the combination of the following elements: IP addresses and port numbers of the clients and servers. A sock is defined as a collection of connections that have the same IP address and port number of the server (Stevens and UNIX Network Programming, 1998). Connection classification is used to create the good client list, and socks classification is used to detect attacks or resource problems. Fig. 11 illustrates the concepts of a connection and a sock.

Fig. 12 depicts a ChkModel deployment at the network edge. Rather than establishing the legitimacy of individual packets, ChkModel observes connection and sock behavior and classifies them as being legitimate or attacks. As shown in Fig. 12, the ChkModel is basically composed of the traffic observation and classification components. The traffic observation component is responsible for capturing and aggregating packets based on connections and socks. These are then fed into the classification component responsible for identifying any anomalous behavior based on known legitimate connection and socks models. When FMM is notified of a problem, it starts assessing other indicators or sensors (i.e. seeking other opinions) before taking a decision capable of mitigating attacks or reacting to them as stipulated by the network policies.

The traffic observation component is responsible for collecting real traffic at one or more points on a target network and for gathering traffic statistics at connection and socks granularity which are then stored in the Conn–Hash and the Sock–Hash tables respectively. These statistics are continuously read by the classification component that compares them with existing legitimate connection and socks models. The current implementation of the ChkModel verifies only the legitimacy of TCP traffic, which currently reaches as much as
90% of traffic in the Internet. Judging a TCP connection as legitimate is based on the two-way communication paradigm of the TCP protocol. A TCP connection is modeled by the ratio of the number of packets sent to and received from a specific destination (Stevens and UNIX Network Programming, 1998). This relationship will be used as a baseline for the implementation of legitimate connection and sock models, as described in more details next.

3.3.2.1. Legitimate TCP connection model. Data flows from the source to destination during a TCP session are controlled by the constant flow of acknowledgments in the reverse direction. Congestion can be perceived if the flows of acknowledgments decrease. In this situation, TCP reduces the sending rate. This explains why normal TCP communication can be modeled by the ratio of the number of packets sent to and received from a specific destination. A legitimate TCP connection model defines two thresholds, ConnThr1 and ConnThr2 as the maximum ratio (number of packets sent to or received from) for a healthy TCP connection and the maximum ratio for a suspicious connection respectively. ChkModel classifies a connection as good if its ratio is below ConnThr1, as suspicious if its ratio is between ConnThr1 and ConnThr2 otherwise it is classified as being definitely bad. Because of the dynamic nature of network behavior, some normal traffic may sometimes be miss-classified as bad. In order to deal with this problem we created a second threshold which provides a new classification called suspicious for borderline cases for use in our sensor.

3.3.2.2. Legitimate TCP sock model. ChkModel’s legitimate TCP sock model is similar to the previous one. The main
difference is that it defines only one threshold namely SockThr, seen as the maximum ratio for a normal sock. Recall that sock is defined as a collection of connections that have the same IP address and port number of the server. A sock is classified as normal if its ratio is below SockThr and as an attack otherwise.

When a TCP sock is low, it is an indication that a determined server is overloaded, possibly by an attack, and through the TCP connection model, the presence of an aggressive sending host in the Conn–Hash table signals the possibility that a host can be participating in an attack.

3.3.2.3. CheckModel INTERACTION with PMM. In the normal state, two basic messages are exchanged with PMM:

- Client List message: informs the PMM of the clients classifications. The classification attribute can be set to “good”, “bad” or “transient”. These classifications are passed through XACML messages to the PMM that applies enforcement policies and rate limit anomalous packets, ensuring the forwarding of legitimate packets. Fig. 13 depicts an example of this message.

- NetWork Status – this message informs the PMM of the status of the access network. This status can be “Normal” or “Under Attack”. In the “Normal” State, this message is created whenever a sock is classified as being an attack, then a message is sent to the PMM with the attribute value equal to “Under Attack” state.

When an attack is identified, the ChkModel state is changed to “Attack” and its classification processes are interrupted during a time interval named backoff period defined by the network administrator. The backoff period is important to ensure that the decisions taken by the PMM achieve their full effect. Moreover, this period avoids the processing overhead resulting from the huge amount of packets sent by attackers. After the backoff period, if the attack persists the ChkModel starts a new backoff period otherwise it returns to the Normal state. In this last case, a NetWork Status message is sent to PMM with the attribute value set to “Normal” to inform a change in the ChkModel’s state machine.

4. Evaluation and results

We divided this evaluation into two experiments. In the first one, we used a controlled environment to inject attack traffic to investigate the architecture’s ability of discovering and taking actions to mitigate DoS attacks. In a second experiment, we used real packet traces containing anomalies and attacks to validate the efficiency and precision of the RIP architecture in detecting massive traffic attack situations and other anomalies.

4.1. Experimenting with generated traffic

This section presents an evaluation of the RIP architecture when submitted to attack traffic injection within a controlled testbed environment. To cover the variety of possible attacks and to fairly evaluate the robustness of this architecture, we allowed changes to a number of different aspects of the testbed: (1) test topology, (2) background workload of legitimate traffic, and (3) more importantly attack characteristics. The
following sections give more details about the methodology used.

4.1.1. RIP testing environment
RIP testing was purposely confined to an isolated testbed consisting of real machines within our laboratory. The idea was to create a controllable environment that resembles as much as possible a realistic DoS network topology. As depicted by Fig. 14, our testbed contains 16 PCs, 3 Cisco switches with 24 10/100 Mbps interfaces and two Giga uplink interfaces. The PC nodes are used as edge nodes running different user applications, to simulate routers, and application level traffic generators.

The RIP prototype consists of 4 servers (all of them running the FreeBSD operational system), while each one having a distinct role: PMM server, TMM server, firewall/gateway, in addition to other network servers. The choice of FreeBSD is justified by its known superior performance with regard to packet capture, firewall rules handling and its high integration of the libraries and security features. We used four attackers, of which two were inserted in the internal network (RIP network) and the others in the external network.

4.1.2. Legitimate traffic
The goal of the RIP architecture is to prevent ingress and egress anomalous traffic into and out of the network. Legitimate traffic during the experiments must be representative of what one expects to encounter in real networks. To meet this requirement we designed a traffic generation script that produces such traffic. This script is in the form of a set of bash (Unix shell) commands responsible for initiating connections assigned to a set of local and foreign hosts. The object here is to establish communications between hosts with addresses within the local subnet prefix (or protected RIP network) and those in the outside world (the unprotected Internet) and find out how much of it is intercepted as unwanted malicious traffic. The traffic will be generated both as outward queries (from external clients to the local server) and as inward queries (from the local clients towards foreign servers). The density of these queries changes in accordance with their type or underlying protocol: DNS, HTTP, ICMP, NTP, etc.

The legitimate traffic script randomly generates Telnet, FTP, HTTP, DNS, and NTP traffic. During Telnet connections, data is exchanged using small-size packets over a long time period (more than five seconds), whereas during FTP sessions, a data connection is maintained until the whole file is transferred. Similarly, during HTTP connections, data is maintained until the information of the whole page has been transmitted. This duration may range between three and twenty seconds depending on the location of the HTTP server in our testbed. UDP and ICMP sessions are simpler. In our tests, only DNS and NTP traffic is used to produce UDP traffic and ICMP traffic is generated through invoking the Packet Internet Gropper (ping) command.

As far as Telnet traffic is concerned, the traffic generator script uses a common (real) telnet program. It is executed in a remote host client and basically establishes a connection with the server and waits randomly (between 5 and 15 s)
before releasing it. For FTP and HTTP sessions, the traffic generator script uses the publicly available wget program. FTP is executed by a local host client to establish the connection with the server and then to copy (get) a file. The file size follows a Pareto distribution with $0.9 \leq \alpha \leq 1.1$. HTTP is executed in local and remote host clients. The idea is to establish connections to web servers within the RIP network or the Internet and then to copy (get) the index page to the clients. The HTTP connection starting times and durations are heavy-tailed and follow a Pareto distribution with $\alpha = 1.12$ as in Crovella et al. (1998) and Barford and Crovella (1998). To deal with DNS queries, the RIP network includes a DNS server. The traffic generator script uses the nslookup command to query local and external DNS servers. Further, DNS queries can be executed for both local and remote host clients. For NTP traffic, the ping echo service is invoked by both local and remote host clients. The targets of ping are highly popular (commonly accessed) web sites and the existing RIP servers.

Fig. 15 shows the average of the amount of traffic sent and received (flows) as created by our script. The plotted line in the graphic represents the amount of flows sent and received by 16 sources pertaining to the RIP network during a one hour time.

4.1.3. Malicious traffic generation

In order to test different attacks, we employed a packet injection tool, called the Packet Analysis and Injection Tool (Packit) (Intrusense Packit). We used it to create customizable DoS and DDoS attack scripts. Packit is a network tool designed to customize, inject, monitor, and manipulate IP traffic. It allows the spoofing of nearly all TCP, UDP, ICMP, IP, ARP, RARP, and Ethernet header options. Packit is useful for testing firewalls, intrusion detection/prevention systems, port scanning, simulating network traffic, and general TCP/IP auditing.

4.1.4. Results

Different levels of UDP, TCP and ICMP attacks were tested over the topology shown in Fig. 14 to determine the minimum flow rates needed to create a denial of service effect. In the case of UDP and ICMP attacks, the targeted resource is a 10 Mbps link in the RIP network. In the case of TCP attacks, the targeted resource is the victim’s connection buffer.

In order to show the effectiveness of our architecture in detecting attacks we present the TCP SYN flooding attack experiment where a web server is the attack target. Fig. 16 clearly shows the increase of flow numbers seen by the RIP firewall/gateway relatively to before and after the attack is started.

The line plotted in the graphic depicts the amount of UDP, ICMP, and TCP aggregated flows. Before the attack taking place, the mean number of flows was 4003 flows per second whereas it suddenly increased to 15,738 flows per second once the attack was launched. Therefore, the attackers generated more than 10,000 new flows towards the target.

Fig. 17 gives a closer view (lasting five minutes) of the attack suffered by the HTTP server. The mean number of flows before this attack was 839 flows per second and increased to an unusual 8000 flows per second when under the attack. With the activation of the RIP detection profiling heuristic, the attack effects only lasted 120 s. This time represents the average time that the architecture requires to detect and take an action to mitigating it. From then on, only the flows previous to the attack are allowed to go through the network. Further tuning of the RIP architecture may be needed to lower this detection delay though.

4.2. Experimenting with real traces

This section presents an evaluation of the RIP architecture using trace data obtained from CAIDA (Cooperative Association for Computers & Security)
Internet Data Analysis) (Shannon et al.). This dataset consists of a huge amount of IPv4 packets sent by DoS attack victims in response to spoofed attack traffic. In spite of the trace being unidirectional (i.e. contains only responses from the attack victim) the profiling methodology relies on examining communication patterns of hosts, which is useful to understanding victim populations, and to validate algorithms for detecting or classifying malicious traffic. This backscatter from the victims was collected by the UCSD Network Telescope in the months May, August, and December 2005. The main characteristics of these traces are summarized in Table 3.

The basic insight exploited by the traffic profiling methodology implemented into TMM is that interactions between hosts display specific communication patterns (i.e. behavior). Our task is to find out such behavior and consequently generate the necessary rules that can be enforced and interpreted by a policy enforcement point.

First, a set of suspected communication patterns representing DoS attack responses present into this trace file was chosen. We visually captured these communication patterns using graphlets that reflect the most common behavior for a particular attack (Karagiannis et al., 2005). A graphlet can be used for representing network flow characteristics corresponding to different applications, by capturing the relationship between the use of source and destination ports, the relative cardinality of unique destination ports and IPs as well as the magnitude of these sets. Hence, for a given fixed dimension, the victims’ source IPs, there is a limited number of graphlets which can represent the resulting packet flows. Fig. 18 displays the graphlets used in our evaluation used to identify the DoS attack flows from victim’s responses.

Fig. 18(a) displays the behavior of a victim which communicates with many destinations IP on many destination ports using one or few source ports. As it turned out, this graphlet came up to be the best in describing the answers of victims to DoS attacks, because this communication pattern perfectly matches with its traffic description. In addition, we have evaluated similar communication patterns such as those graphlets described in Fig. 18(b)–(e). For instance, Fig. 18(b) displays a graphlet where a DoS victim uses one or few source ports to communicate with specific destination port on many destinations ports. Fig. 18(c) represents an attacked host using one or few source ports to access many IP destinations on one or few ports. Similarly, we also represent extreme relative communication patterns represented in Fig. 18(d) and (e). Fig. 18(d), for instance, displays a victim host which talks to one or few hosts with one or more services using a single or few ports. Finally, Fig. 18(e) displays the second extreme case where a victim host communicates with many destination hosts on many destination ports using several sources ports.

After the possible identification of a set of behavior patterns represented by graphlets, we next introduced some heuristics and rules for use as attack blocking strategies based on the characteristics of the described patterns. In order to improve the accuracy of this profiling method we also considered the minimum number of flows per minute (m), a threshold we introduced to identify aggressive sources.

Note that such threshold may change with the speed of the interface being considered. Considering the description of the trace file used, where all datasets consist of huge amounts of packets sent by DoS attack victims in response to DoS attack traffic, we adopted a completeness metric to evaluate the capacity of the architecture in detecting attacks. This metric reflects the percentage of the traffic classified by a classification method implemented in TMM (i.e. the profiling method in this particular work).

We carried out experiments on each trace while diversifying the values of the flow rate m (100, 300 and 500). Due to space limitations and the similarity of the results, we only plot the graphics related to m = 100 as the flow rate. In Fig. 19, we plot the completeness for each trace. In general, our architecture is able to identify most of the attack traffic with high effectiveness. In the first trace (December, 1) with m = 100, we achieved 87.9% of completeness, against 86.3% with m = 300 and 85.4% with m = 500. This decrease is intuitive as the flow rate increases it becomes more difficult to single out all attacks. Similar results also were obtained through the other traces. In the second trace, our architecture reached 91.4%, 90.1% and 89.7% completeness percentages with m = 100, 300 and 500 respectively. In the last trace, we had 89.1% of completeness with m = 100, 88.2% with m = 300 and 87.2% with m = 500. To sum up, in the all traces analyzed our architecture was able to detect more than 85% of the DDoS attacks. These results are encouraging and may be enhanced by further tuning of the software performance.

### Table 3 – Characteristics of the DoS traces.

<table>
<thead>
<tr>
<th>Trace</th>
<th>Date</th>
<th>Duration</th>
<th># Packets (millions)</th>
<th># Bytes (GB)</th>
<th>Src IP (millions)</th>
<th>Dst IP (millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>December 1, 2005</td>
<td>08 h 34 min</td>
<td>33.2</td>
<td>1.55</td>
<td>1.88</td>
<td>9.28</td>
</tr>
<tr>
<td>2</td>
<td>August 25, 2005</td>
<td>23 h 59 min</td>
<td>96.26</td>
<td>3.90</td>
<td>0.05</td>
<td>14.94</td>
</tr>
<tr>
<td>3</td>
<td>May 25, 2005</td>
<td>16 h 29 min</td>
<td>119.88</td>
<td>4.83</td>
<td>0.15</td>
<td>12.51</td>
</tr>
</tbody>
</table>
We also represent graphically the data with pie charts to give an intuitive vision of the participation of each behavior pattern in the percentage of traffic classified. As shown in Fig. 20, the majority of classified flows correspond to behavior patterns represented by graphlet (a) pie charts. For the first trace, the amount of classified flows as graphlet (a) reached 82.3%, 81.6% and 81.1% with $m = 100$, 300 and 500 respectively. Despite the lower percentage of flows falling into the graphlet (e) on Trace 1, many flows were identified by this graphlet within traces 2 and 3. For example, for the third trace, the percentage of identified flows by graphlet (e) was around 30%.

4.3. Examples using the CheckModel sensor

The results of the ChkModel tests demonstrated that this approach presents good performance against massive DDoS attacks, great scalability and low consumption of system resources. Recall that CheckModel compares both traffic directions for a flow and detects possible attacks when this is below a given minimum threshold. We setup a testbed with 52 desktops, 2 24 ports switches (10/100/1000 Mbps) and 2 servers (Athlon XP 4200 + 64bits, with 2 Gb of RAM and 160 Gb of hard disk) acting as a router, gateway and firewall. Linux distributions (Gentoo, Ubuntu, Debian, Red Hat and Slackware) were used. Fig. 21 shows the topology built for testing purposes.

Fig. 21 has 4 PCs used to generate the attack traffic using a script that invokes the Packit packet spoofing tool. Three attacks per hour were carried out using 1 kb packets with a rate ranging between 500 and 20,000 packets per second. As metrics, we observed CPU and memory processing, re-incident IP addresses and efficiency in identifying DDoS attacks.

Fig. 22 shows the daily average CPU processing. This has been below 1% of the other processes running at the server. Memory consumption was also less than 5% of the 2 Gb total memory.

Between the 11th and 14th days, one sees that both memory and CPU consumptions do settle down. First, we think that this is due to the fact that with time, IP hash tables receive less updating as the trusted traffic becomes more known. Second, on the 10th day valid timestamps expire for many non reused addresses leading to their removal from these tables.

4.3.1. Re-incident IPs

Our results confirm the studies in Jung et al. (2002) who also show that a large portion of valid IPs continues reestablishing.
connections with the server for several days. The sudden fall in re-incident IPs observed on the 27th and 28th results from the low access experienced during the weekend, see Fig. 23. Please note that our capture of re-incidence was only started on the second day of the tests.

4.3.2. Efficiency

Fig. 24 shows the flows crossing the router on the 8th of May between 15:34 and 15:42. This timeframe has been selected to launch DDoS attacks from four sites with 20,000 packets per second each.

Before the attack, the average traffic from trusted IPs was 15,670 packets per second. As a result of the attack this number fell to 11,930. Since our policy was to block un-trusted IPs, the situation stabilized after 1 min and 15 s, the time necessary for CheckModel to react. This lead to 76% efficiency in terms of protecting legitimate traffic.

4.4. Orchestrating attack detection

The essence of our work lies in the power gained from the clever combination and coordinated orchestration of different attack detection modules. As a proof to our approach we combined the use of two previously introduced detection strategies, namely, the ChkModel and Profiling. They both ran at different network interfaces while a set of attacks was

![Diagram](attachment:image.png)
generated. The results of the analysis made by each of these
techniques for three types of attacks are presented.

This evaluation process was made using traffic submitted in 20 min intervals or cycles. The first 5 min corresponded to normal user traffic while both detection strategies are running and without injecting attack traffic. At the fifth minute attacks start being injected into the local area network. Hence, we expect anytime now, traffic classification, attack detection and their mitigation to start taking place. Fifteen minutes into the experiment, the traffic is halted. Note that the same topology earlier described is used in this part of the tests. Next, we describe the three types of unwanted traffic used in the controlled experiment.

4.4.1. Flood TCP SYN joint detection
We flooded the network with TCP SYN attack packets towards a Web Server target at port 80. Spoofed IP addresses and random originator ports were used. Fig. 25 shows the TCP SYN attack at the router interfaces. It shows that once the first attack is initiated the captured traffic increases dramatically. For the CheckModel processor the increase in the number of flows towards the same Web server IP address is rapidly identified as an attack and communicated to the policy management module or PMM. Consequently, TMM takes the initiative of limiting the offending IP source addresses (spoofed ones) pointed out by the CheckModel. Recall, that the PMM decision is taken after consulting its trusted IP list (TIL). Since the attack lasts for 10 min, the ChkModel continues informing the PMM about the attacking IP addresses. This may continue until the PMM may decide to block entirely these addresses.

Fig. 25 shows it takes the Profiling strategy more time to detect this type of attacks than the ChkModel. This is due to the fact that Profiling needs obtaining a large quantity of flow information in order to enable it to identify anomalous traffic with a high level of accuracy. This also why the Profiling strategy takes a harsher action against anomalous traffic by blocking it as opposed to ChkModel’s progressive actions.

4.4.2. Spam joint detection
Next, we created an e-mail spam attack towards an external SMTP Server. We used forged IP subnets and addresses with randomly allocated client port numbers in the originator’s addressing fields.

Unlike the previous TCP SYN attack, Fig. 26 shows that only the Profiling strategy was capable of detecting the attack. This can be explained by the fact that for the ChkModel, spam e-mail is legitimate traffic as it looks just like normal e-mail when making use of the TCP handshake mechanism for connection establishment. On the other hand, Profiling detects the attack as it senses a sudden increase of the number of flows targeting a single Web server or IP address. However, the response time for Profiling remains relatively high as it borders the two minutes.

4.4.3. Detection of low rate TCP SYN flooding
This time we use the same previous TCP SYN flooding attack but at a low rate with around 48 packets a minute. This type of attack is harder to detect as it generates a low number of flows and hence slips through the control of the Profiling technique.

ChkModel on the other hand, sees a number of TCP SYN packets going through towards the same destination server.
and therefore is capable of detecting the attack as shown in Fig. 27.

These latest experiments confirm the earlier observation that a coordinated approach that combines a number of techniques is necessary. We have seen that the same attacks may be jointly identified and in some other cases missed by a given strategy. One may adopt a strategy whereby harsher decisions are made when two or more methods detect a given attack, or that further techniques may be called upon when a possible attack has been flagged by one the approaches. It is this type of coordination that enhances the strength of detection techniques and leads to more effective ways of protecting network traffic and services.

5. Related works and our contributions

Most of existing security approaches have been designed to operate individually, including IP traceback (Aljifri, 2003; Bellovin et al., 2003), packet filtering (Kim et al., 2004), sophisticated intrusion detection systems (Ning et al., 2001) and DDoS defense schemes (Kandula et al., 2005; Mirkovic et al., 2002). These approaches have been slow in detecting and responding to distributed attacks generating sizeable traffic volumes and diversity. We review here only those approaches that provide some form of cooperative defense between different nodes or share other strong similarities to RIP.

Hwang et al. (2007) propose that such collaboration be done through the combination of signature-based intrusion detection systems aiming to cover known attacks and anomaly detection systems (ADS) to detect novel unknown attacks. This work is interesting since it integrates the flexibility of ADS with the accuracy of a signature-based system. This integrated approach first filters out the known attack traffic by IDS (e.g. SNORT) through signature matching with a database. The remaining traffic containing unknown or burst attacks is fed to the episode-mining engine to generate frequent episode rules. The episodes that do not match the normal profiles or match them with unusually high frequency are labeled as anomalous and are used to generate signatures which capture the anomalous behavior. The advantage of using rules is that they tend to be simple and intuitive, unstructured and less rigid. Among their drawbacks: they are difficult to maintain, and in some cases, are inadequate to truly represent many types of events (including attacks). Moreover, the approach requires a priori knowledge of what rules and patterns are interesting.

In their attempt to increase the attack detection and lower false alarms, Siaterlis and Maglaris (2004) combine the output of several sensors utilizing a data fusion algorithm based on the Dempster–Shafer’s Theory of Evidence. The sensors are autonomous but are collaborating by sharing their beliefs about the network’s state, i.e. whether it is coming under attack or not. Although flexible, they make use of a large set of network states (Frame of Discernment in the “Theory of Evidence” terminology) leading to higher processing cost since Dempster’s rule of combination requires finding all possible states that induce a hypothesis.

Other works explore collaborative communication to handle issues of scalability, adaptability and performance.
However, no one can guarantee this cooperation and willingness among routers owned by competing ISPs, domains or different organizations to deploy the same sensors in their own networks. On the other hand, the RIP architecture was designed to focus on both the source and victim defenses and does not assume any additional network infrastructure changes. Second, the multicast communication style between the nodes adopted by DefCOM and COSSACK is not scalable, mainly in an attack scenario. Moreover, handling damaged or subvert nodes in the overlay network may be hard, and DefCOM is likely to misfunction if they are not handled. In contrast, we use a relatively simple cooperative mechanism among the security components to avoid unnecessary message broadcast. In our architecture, each endpoint must identify itself to the PEP system associated to it before obtaining access to network resources. Furthermore, the access control mechanism employs periodic checking to know if it is in compliance with current corporate or domain specific security policies. Third, the classifier proposed by DefCOM and COSSACK will not work with current attack traffic stemming from spams, worms and P2P because of the lack of distinct signatures.

RIP differs from the proposals presented in this section in that it builds an extensive traffic analysis mechanism. RIP adopts some of the latest advanced traffic classification techniques based on the behavior of aggregated flows in contrast to packets, obtaining better results while actually processing less information. In other words, RIP strives on looking to reduce memory consumption and CPU processing and achieving similar results to when using costly deep packet inspection. Finally, the RIP architecture is an ongoing project for which many refinements and extensions are planned. For example, in order to speed up the network packets processing up to the wire speed, it is strongly recommended to investigate the possible gains from network hardware. Although it is possible to accomplish software architectures that scale to service a high packet throughput, it is not a simple task to manage and deal with on-line processing. We plan to undertake some adaptations in our classification methods by implementing some packet processing over specialized hardware support. Other future optimizations include the use multicore off the shelves commodity PCs, the exploiting of parallel threads, cache optimization and kernel level drivers. The addition of new components for dealing with other types of unwanted traffic such as spams and worms is also part of our current extension work.

The differences between RIP and existing works such as COSSACK can be used to increase the robustness of this proposal. For example, detecting a DDoS attack from an edge network and blocking this at the firewall or similar devices does not turn the attack ineffective since resources such as bandwidth are still being consumed. In other words, collaboration with other networks is necessary to zero-in and mitigate the effect of such attacks. One could extend RIP’s coordinated collaboration to interact with other Internet wide efforts such as COSSACK. Consequently, COSSACK may see the whole RIP architecture as a single COSSACK watchdog or sensor, and vice versa, RIP may also see the whole COSSACK collaborative system as a special sensor that gives it early warning and important feedback about unwanted traffic.

Shyu et al. (2007) and Xie et al. (2006) propose a collaborative two-layer architecture for an agent-based distributed intrusion detection system. The first layer, the host layer, uses lightweight host agents to detect anomalies and then forwards them to the classification layer, exclusively composed of dedicated hosts, to execute misuse detection on received information and advises others hosts and classification agents in the network about special types of attacks. Agents are used to reduce the computational load on the system by dividing it between different hosts and facilitate system reconfiguration. There are however several types of security threats targeting vulnerable agent based IDS. They take advantage of the execution of exploits of different agents, malicious activities against running platforms, etc.

The DefCOM (Defensive Cooperative Overlay Mesh) (Mirkovic et al., 2003) and COSSACK (Coordinated Suppression of Simultaneous Attacks) (Papadopoulos et al., 2003) architectures provide an integrated defense solution that enables filtering and admission challenges. In these architectures, the nodes collaborate by exchanging messages, marking packets for high or low priority handling, and prioritizing marked traffic. Under normal conditions, no filtering or admission challenges are required. When an attack begins, the DefCOM and COSSACK networks form a group of defense nodes that are deployed at source and victim networks and cooperate in filtering the attack. A drawback of these approaches is that they are unable to handle attacks from legacy networks that do not deploy special elements configured with their defense mechanisms.

RIP differs from the proposals presented in this section by its capability to support updates or insertions of new components permitting to increase the security level. For example, the Hwang’s proposal could be seen as a new point of defense or component of RIP architecture like CheckModel and Profiling Method used in our experiments. In our RIP prototype, we used a finite state machine to correlate the information received by TMm components in possible states that a network client could take. This correlation could easily follow the Dempster-Shafer’s Theory of Evidence as suggested by Siaterlis and Maglaris.

DefCOM and COSSACK have some similarities to RIP in the sense that both architectures employ traffic correlation and analysis. However, our solution differs in several aspects. First, they chose to deploy detectors at the victim’s side and send alerts to a filter or rate limiter located at the source side. However, no one can guarantee this cooperation and
6. Conclusions

The architecture presented does not claim to be complete. It remains an ongoing research tool and is likely to evolve with the inclusion of new strategies such as heuristics to increase its effectiveness in detecting anomalous traffic. Our current proof of concept implementation currently combines the use of two strategies for classifying Internet backbone traffic known as profiling and checkmodel for protocol verification.

Currently, the average time it takes RIP to detect and act on mitigating DDoS attacks remains relatively high and future tuning of the RIP architecture may be needed to lower this detection delay. There is however a tradeoff between this time and what we called here as the completeness of RIP. It is clear that the more anomalies we want to detect, the less aggressive is RIP’s reaction. The authors plan to introduce new heuristics and evaluate their effectiveness in future RIP implementations as well as optimize the code over new multicore PCs.

The feasibility of using RIP and similar architectures over a multi gigabit interface in real-time network interface can be questioned. Our initial studies have pointed to some encouraging signs coming from recent software and hardware developments. In this context, RIP may make use of the following:

- Off the shelf adapter boards capable of packet sniffing at such speeds. Examples of these include some PC Intel boards.
- The use of multi processor and multicore PC architectures and their large caches to speed up the processing of incoming traffic and reduce the execution of RIP’s heuristics and algorithms.
- Rewriting the packet capture device drivers and the use of more optimized packet capture libraries instead of libpcap to reduce processor kernel interrupts and the use of parallel threads.
- Other optimizations include investigating the use of packet sampling, reducing the communications between threads and costly packet copying across adapter, kernel and user spaces as in Schneider (2007) and Deri (2007).

Further plans include offering UDP and ICMP support into the ChkModel and conducting more elaborate evaluations.

Acknowledgments

This work was supported by the Research and Development Centre, Ericsson Telecomunicac¸o˜n, Brazil.

REFERENCES

ContentGuard, Inc. eXtensible rights markup language, XrML 2.0; 2001.
IEEE. 802.1x: port based network access control. Available online: http://www.ieee802.org/1/pages/802.1x.html.
Kandula S, Katabi D, Jacob M, Berger A. Botz-4-Sale: surviving organized DDoS attacks that mimic flash crowds. In: Second symposium on networked systems design and implementation (NSDI), Boston, MA; May 2005.
OASIS. Assertions and protocols for the OASIS security assertion markup language (SAML) V2.0; March 2005.
OASIS. eXtensible access control markup language (XACML) version 2.0; February 2005.
Oasis. XACML v3.0 administrative policy, working draft; December 2005.
ACM Computing Surveys April 2007;39(1).


Professor Djamel H. Sadok teaches Computer Networks at the Federal University of Pernambuco, Brazil since 1993. He has obtained his PhD from the University of Kent at Canterbury. His interests include Mobile Networks, Traffic Measurement and Security.