DIADEM Firewall: Web Server Overload Attack Detection and Response

Gerhard Münz, Ali Fessi, Georg Carle, Olivier Paul, Dušan Gabrijelčič, Yannick Carlinet, Sherif Yusuf, Morris Sloman, Vrizlynn Thing, Jan van Lunteren, et al.

DIADEM Firewall: Web Server Overload Attack Detection and Response

Gerhard Münz, Ali Fessi, Georg Carle
{muenz|fessi|carle}@informatik.uni-tuebingen.de
University of Tübingen, Germany

Yannick Carlinet
yannick.carlinet@francetelecom.com>
France Telecom

Oliver Paul
olivier.paul@int-evry.fr
Grandes Ecoles des Télécommunications, France

Sherif Yusuf, Morris Sloman, Vrizlynn Thing
{sy99|m.sloman|vrizlynn.thing}@doc.ic.ac.uk,
Imperial College London, UK

Dušan Gabrijelčič
dusan@e5.ijs.si
Jozef Stefan Institute, Slovenia

Jan van Lunteren, Patricia Sagmeister, Gero Dittmann
{jvl|psa |ged}@zurich.ibm.com
IBM Zurich Research Laboratory, Switzerland

Abstract

High-profile web servers often become the victim of web server overload Distributed Denial-of-Service (DDoS) attacks. Motivations of such attacks range from technical challenge (e.g. script kiddies) to financial profit (e.g. blackmailing the web server's owner).

This paper presents the DIADEM Firewall architecture that allows an ISP to protect its customers from being the target of a DDoS attack. Additionally, it provides protection against usage of customer hosts for attacks. Furthermore, the use-case of the web server overload attack detection and response mechanism will be explained in more details. Finally, we outline the integration an FPGA based high-speed classifier engine integrated into the Linux Netfilter firewall as well as its deployment during a response action against the DDoS attack.

1. Introduction

Today's networking environments are becoming more and more complex. The number and diversity of deployed network devices and configurations is rising and the diversity of services offered and used is increasing. On the other hand, broadband Internet access is becoming a commodity for more and more Internet users.

However, networking environments have not only increased the utility provided for the users, they have also become prone to various kinds of malicious activities and security threats. Attacks and the tools used to perform them are getting more and more sophisticated and harmful by exploiting the increasing number of permanently connected and often badly secured customer hosts. Such hosts are susceptible to worm and virus infections, and once infected, they can be easily misused as attack sources in Distributed Denial-of-Service (DDoS) attacks. Firewalls as well as Intrusion Detection and Prevention Systems (IDS/IPS) are used to detect such malicious activities and to protect the networking environment and other potential victims, such as web servers, corporate networks etc. However, these systems are usually operated as standalone devices protecting an isolated subsystem or network only. Possibilities for coordinated operation are very limited and mostly realized on a manual basis. Hence, the existing attack detection and prevention systems are not sufficiently effective as countermeasures against sophisticated attacks and upcoming threats.

In the scope of the DIADEM[1] Firewall project [1], we are developing an architecture that enables an ISP (Internet Service Provider) to better protect its own networking environment as well as the connected hosts and servers of its customers. The main goal of the project is to establish a tight loop of detection, decision and response across multiple, distributed firewall and networking devices.

In this paper, we present our approach on the basis of one selected use-case, the web server overload DDoS attack. During such an attack, multiple distributed attack sources attempt to overload a web server by simultaneously requesting a given object located on the server. Depending on the number of attackers and the performance of the web server, the attack causes service degradation or even Denial-of-Service for legitimate users. This attack is common today and has various properties we can expect from new and forthcoming attacks.

The rest of this paper is structured as follow. An overview of the overall DIADEM Firewall architecture is presented in section 2. The general attack detection architecture and particularly the web server overload attack detection mechanism are explained in section 3, followed by a description of the response architecture and the web server use-case related response actions in section 4. Section 5 outlines how a high-speed packet classifier engine implemented on an FPGA is deployed for efficient mitigation of the attack. Finally, conclusions and directions for future work are given in section 6.

2. Overall Architecture of DIADEM Firewall

This section gives an overview of the architecture of DIADEM Firewall and presents its high-level components. As shown in Figure 1, we separate the different components of the architecture according to three levels: data level, element level and administrative level.

2.1. The Data Level

The data level contains the equipment that deals with the actual traffic in the operator's network. Considered

[1] short for DIstributed Approach for a high-speed programmable firewall and attack DEtection systeM.

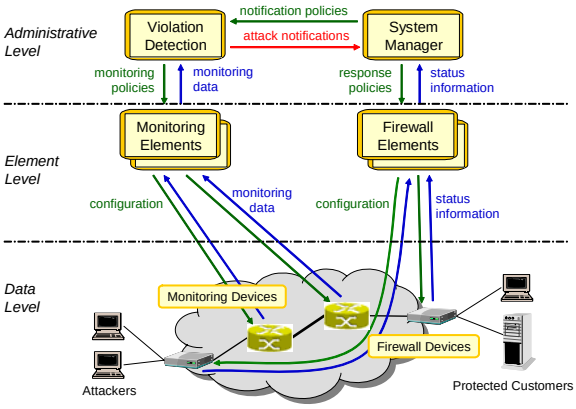

Figure 1: DIADEM Firewall Components

devices include routers, network monitors, and firewalls that are used for network monitoring and response purposes. Project results include implementations of innovative components with special monitoring and response functionality that are evaluated in scenarios with commercial and open-source (Linux) routers and firewalls.

2.2. The Element Level

The element level is an abstraction layer for the data level equipment, which hides device specific details by providing abstract interfaces (APIs) to control the equipment in the data level independently of its specific control interface. Two types of components are considered: Monitoring Elements and Firewall Elements.

Monitoring Element (ME)

The Monitoring Element is an abstraction of a router or a metering device that monitors traffic at a specific point in the network and exports both flow and packet-based monitoring data to a remote system called *Violation Detection* (c.f. section 2.3) for further analysis. Flow-based data consists of statistics such as the number of packets and the number of octets observed during the reporting interval for a specific source-destination flow. Packet-based information includes packets that are selected with filtering and packet sampling methods. The Monitoring Element can also provide statistics about packets with specific header fields, for instance, the number of observed TCP SYN packets or ICMP echo request packets within a given time interval. The Monitoring Element can also be configured to focus on application specific data such as HTTP traffic to detect web server attacks. Section 3 explains the corresponding monitoring and detection process in more detail.

The Monitoring Element provides a *Monitoring API* to the administrative level for dynamically setting and changing the monitoring configuration. Thus, the network monitoring process and exported monitoring data can be adapted to the current needs of the Violation Detection system.

Monitored data is exported from the Monitoring Element to the Violation Detection system using the IPFIX (IP Flow Information Export) protocol [2] which is currently standardized in the IETF. The strength of the IPFIX protocol resides in its capability to carry any kind of data records in a bandwidth-efficient way.

The Monitoring API is realized using the Netconf protocol [3] which is also currently standardized in the IETF. The Netconf protocol is very flexible and can be easily adapted to different configuration tasks.

The Firewall Element (FE)

The Firewall Element is an abstraction for a router or a firewall device that allows enforcing response policies in order to stop or mitigate an attack. Response policies may require that one or more specific data flows are blocked, redirected or rate-limited. The Firewall Element offers a *Firewall API* and a *Response API* to the administrative level. Hiding the device-specific details of the data level components, the Firewall API can be used to trigger immediate actions. On the other hand, the Response API delivers response policies to the Firewall Element that allow for making fast local decisions.

2.3. The Administrative Level

The administrative level allows the system administrator of to enter new policies, new attack counter-measure logic, and take decisions when needed for a counter-measure. It comprises two components: *Violation Detection* and *System Manager*.

Violation Detection (VD)

The Violation Detection system receives the monitoring data from Monitoring Elements which are distributed in the network. It then performs a variety of attack detection methods deploying both signature-based and anomaly detection techniques. If an attack is detected, information about the type of attack, the target, recognized attack sources etc. is encoded into IDEMF (Intrusion Detection Exchange Message Format) [4] messages and sent as an alert notification to the System Manager.

The Violation Detection is described in more details in Section 3.

System Manager (SM)

The System Manager integrates a policy service for specifying, storing and distributing policies to Policy Management Agents (PMAs). PMAs can be deployed in the Violation Detection, the Firewall Elements, and in the System Manager itself.

PMAs receive events via a publish-subscribe system. Upon reception of a new event, a PMA searches for a matching policy and enforces the corresponding action(s) if necessary. For example, the PMA of the System Manager might receive an attack notification event from the Violation Detection. A policy specifies how to react, e.g. by calling the Firewall API in order to perform a specific response action on the Firewall Element, or by triggering a reconfiguration of the Violation Detect to adapt its detection strategy.

The functionality of the System Manager is explained in more details in Section 4.

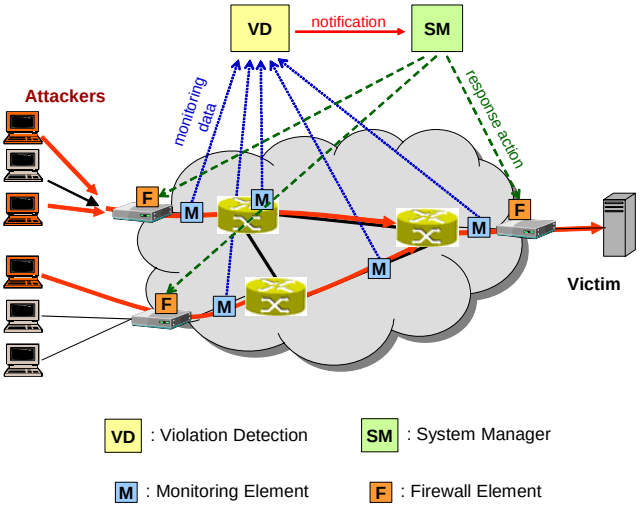

Figure 2: Deployment of the DIADEM Firewall Architecture

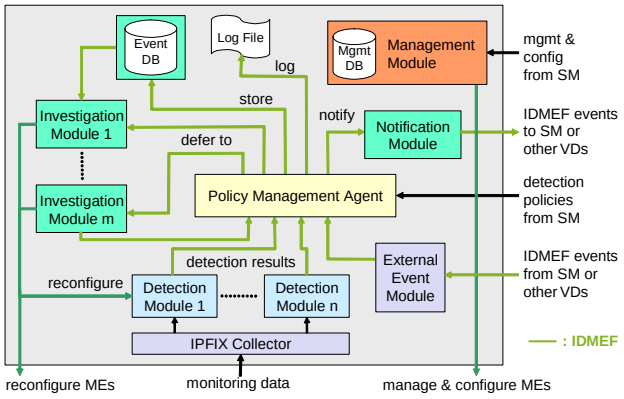

Figure 3: Violation Detection Architecture

2.4. Deployment

Figure 2 sketches the deployment of the DIADEM Firewall architecture in an operator's network. The network is equipped with several highly distributed Monitoring Elements that deliver monitoring data to the Violation Detection which analyzes the data in order to detect ongoing attacks. If an attack is detected or suspected by the Violation Detection, a notification message is sent to the System Manager which uses this information to determine and trigger the appropriate response to the attack. A typical response consists of blocking, redirecting, or rate-limiting the suspicious traffic. This requires the reconfiguration of the appropriate Firewall Elements.

Ideally, if the source(s) of the attack can be correctly identified, the attack traffic can be blocked as close as possible to the source generating the attack. This prevents attack traffic from reaching the victim and reduces unwanted traffic within the network.

Like the Monitoring and Firewall Elements in the element level, the components of the administrative level (Violation Detection and System Manager) can also be distributed for performance reasons or to avoid single points of failures.

2.5. Inter-Domain Deployment

Cooperation between different administrative domains is advantageous for network monitoring as well as for attack detection and response. However, a strong trust relationship between cooperating network providers is required. Handing out monitoring data would also reveal the network topology and information about network performance and utilization etc. Furthermore, customer's privacy comes into question here. Less difficult is the dissemination of alert notifications between distinct administrative domains. Upon receiving an alert from another domain, the System Manager can decide to enforce appropriate countermeasures immediately, or to ask the local Violation Detection at first if the attack can also be observed in the own domain. Depending on the trust relationship with the foreign domain, the System Manager might also decide to ignore a foreign alert completely. Note that a System Manager will not usually be permitted to enforce response actions or policies on Firewall Elements belonging to a different domain.

3. Violation Detection

This section describes the Violation Detection architecture within DIADEM Firewall. In particular, a detection module for web server overloading is presented.

3.1. Architecture

As represented in Figure 4, the Violation Detection system is organized in several modules:

- The IPFIX collector receives monitoring data from Monitoring Elements.
- Detection modules apply detection methods on the monitoring data. Detection results are then passed to the PMA as IDEMF events. Existing detection tools (e.g. Snort [5]) may be integrated in the violation detection function as detection modules.
- The Policy Management Agent (PMA) stores and interprets detection policies which may be triggered by IDMEF events or can be used to filter events. Possible actions are notify, log, store, defer the event to an investigation module, or discard the event. Other actions might be to reconfigure the Monitoring Elements to get more information about current traffic.
- The event database is used to store interesting IDMEF events for later investigation and correlation with other events.
- The investigation modules examine events that cannot be clearly classified by detection modules. Possible investigation methods include:
  - compare/correlate an event with earlier events stored in the event database
  - reconfigure detection modules and/or Monitoring Elements

  If the investigation module concludes that an event is an attack, a new IDMEF event is passed to the PMA.

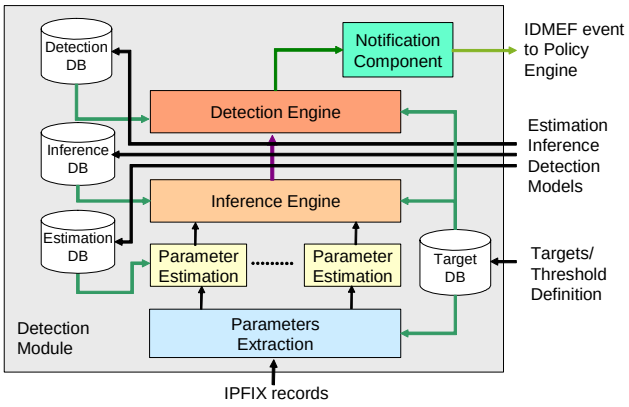

Figure 4: Web Server Overloading Detection Module

- The notification module sends IDMEF events (alarm notifications) to the System Manager or other Violation Detection components.
- The external event module constitutes the counterpart of the notification module. It receives IDMEF events from the System Manager and other Violation Detection components and controls which events are passed to the PMA.
- The management module exports a management and configuration interface to the System Manager. It provides a start-up configuration to all modules in the Violation Detection and to the associated Monitoring Elements and maintains management data in management database.

3.2. Web Server Overload Detection Module

The overall behavior of the *Web Server Overload Detection Module* is to build a view of application level actions from the network-level measurement parameters provided by the IPFIX collector. These inferred application level actions are used to detect overloading attacks by matching detected results with a detection model. When an attack is detected, an IDMEF event is sent to the PMA within the Violation Detection system. Each Web Overload Detection Module is expected to monitor traffic directed to and originating from a specific web server. As a result, models and training data used by each module are server specific.

The Web Server Overload Detection Module consists of four components as shown in Figure 3.

- The parameters extraction component receives IPFIX records from the IPFIX collector, selects those that are relevant for the monitored targets, and extracts the useful measurement parameters. Monitored targets include potential victims (web server address and port number) as well as potential attackers (e.g. IP addresses). The definition of monitored targets is included in the target database.
- The parameter estimation components transform the extracted measurement parameters into input parameters that are adapted to the inference engine. For example, IP addresses may be transformed into country codes in order to limit the size of the inference model. Each parameter may need a different form of correction or adaptation. For each parameter, the estimation function looks up the corresponding model from the estimation database which defines the necessary transformation depending on the type of parameter as well as the target web server. In the case of the IP address to country code transformation, this model would be a data structure representing relationships between these two parameters.
- The inference engine infers information about the invoked HTTP operation (HTTP method, targeted URI, operation result, etc.) from the parameters received by the estimation components. In contrast to classical application level processing schemes, the inference engine retrieves application level information based on probabilistic models applied to the measurement results from lower levels. Thus, the inference approach does not require expensive packet processing including reassembly and pattern matching and allows for a faster, yet less precise application level parameters estimation [6]. For each estimated element, the inference engine selects the corresponding inference model from the inference database. This model derives a tuple including the identity of the user (IP address), the performed action (HTTP header values), and the target web server from the estimated input parameters. Although inference models are target specific, we separate the definition of targets from the definition of models for efficiency reasons. The definition of targets may change frequently (several times an hour) while the modification of models should happen less frequently (for example, once a month or more often depending on the frequency of the server's content update).

The detection engine checks if the tuple received from the inference engine conforms to normal target behavior. Therefore, profiles of normal behavior are stored in the detection database. The detection engine retrieves the appropriate profile from the database and compares it to the received tuple. A profile represents the normal HTTP server behavior as well as the acceptable deviation. Profiles are built incrementally using received tuples and a change detection algorithm. The profile is used, along with the identity of the source, to decide whether an overload attack is going on. If an attack is detected, an alert is sent to the notification component which sends a corresponding IDMEF event to the PMA. The IDMEF event includes the identity of the victim (address and port), the identity of the attack sources (IP addresses and port numbers), the type of attack (used method, impacted URI, operation result etc.), the level of deviation between the attacker's behavior and a normal behavior, and the number of packets and bytes related to the communication.

4. Attack Response Mechanisms

This section describes the System Manager architecture and how it communicates with other modules in the DIADEM Firewall architecture in Figure 1.

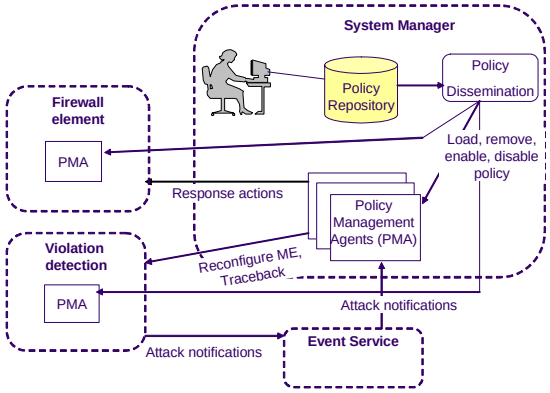

Figure 5: Overview of the System Manager

4.1. The System Manager

The System Manager mainly communicates with the Violation Detection system and the Firewall Elements, as indicated in Figure 5.

The System Manager receives IDMEF notifications from the Violation Detection system indicating the type of attack, the target and the presumed source(s) of the attack, the signature of the attack etc. The System Manager uses this information to determine the appropriate countermeasures against the attack. Notifications may result in the reconfiguration of the Violation Detection or the Firewall Elements in order to respond to the detected attack. The appropriate response actions are specified by response policies enabled by the System Manager. When notifications related to a given attack are no longer received over a given period of time, the System Manager stops the enforced response actions and restores the prior state of the system.

The System Manager provides reports to the DIADEM Firewall architecture administrator as well as to the impacted customer regarding the detection and the treatment of the attack.

4.2. Response Policies

Response policies specify the actions needed to be undertaken in order to stop or mitigate the attack, making use of any of the parameters provided by the triggering alert notification. In case of an automated response, they equate to one of the following:

- perform an action on an Firewall Element, such as redirection, rate limiting etc.,
- perform an action on the Violation Detection component; e.g. reconfigure the Violation Detection for further investigation of the attack, or
- generate and send a new event to the Violation Detection system or to a Firewall Element.

In addition, human network administrators can be informed if manual intervention is required.

4.3. Policy Management Agents

Automated policy enforcement is entrusted to Policy Management Agents (PMAs) that act as local policy interpreters in the Violation Detection system and the Firewall Elements. Once policies are compiled, they are loaded and enabled in the PMAs. The Elvin event system [7] is used for communication with and between the PMAs. PMAs only subscribe to those events which are relevant to their stored policies. On receiving an event, the PMA queries a domain service to determine the target objects on which to perform the policy action(s). A policy can optionally specify constraints which must evaluate to true before the action can be performed.

```
inst oblig /SMPolicies/signatureBasedAlert {
        on webAttack(attack_signature, confidence);
        subject /PMA/SM;
        target t = /PMA/FE;
        do t.closeTCPConnection(attack_signature);
        when confidence > MIN_CONFIDENCE; }
```

Figure 6: Trigger Connection Closing

4.4. Web Server Overload Response

When the Violation Detection system detects a web server attack, it notifies the System Manager, which inspects the attack signature and may decide to reconfigure the Violation Detection to improve the detection result. For example, the Violation Detection system can be asked to look for a more specific attack signature or to increase the confidence in the detection result in order to reduce the probability of false positives. As a consequence, it might be necessary that the Violation Detection system also reconfigures the associated Monitoring Elements to provide more detailed monitoring data.

If the alert notification is severe enough, the corresponding response policy may trigger a response mechanism on the Firewall Elements. The web server overload attack requires established TCP connections between the attackers and the victim. Therefore, the sources of the attack can be identified easily. An effective response consists in:

- setting new firewall rules that drop all packets originated by the attacking sources and directed to the web server and
- closing the TCP connection by inserting a TCP FIN or RST packet towards the server. Note here, that the inserted packet must carry the right sequence number.

While the firewall rules prevent the attackers from sending more HTTP requests, closing the existing TCP connections immediately releases the web server by stopping the currently served requests. Both functionalities are provided by a Linux based firewall implementation. In order to speed up the packet filtering, we make use of a packet classifier engine as presented in section 5.

Figure 6 shows an example of the policy that triggers the closing of a TCP connection under the condition that the confidence in the detection result exceeds a predefined minimum confidence level.

Whether a response to an attack is successful or not, does not only depend on the functionality of the firewall devices but also on the points in the network where the

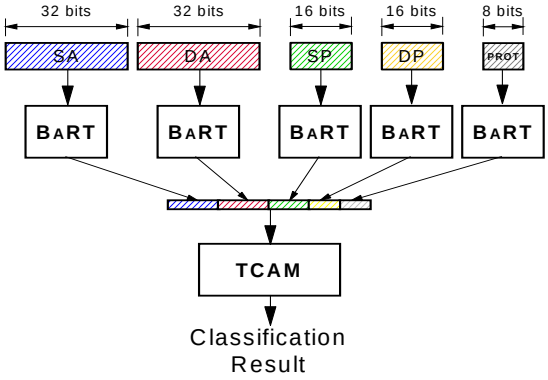

Figure 7: Parallel Packet Classification Scheme (P2C)

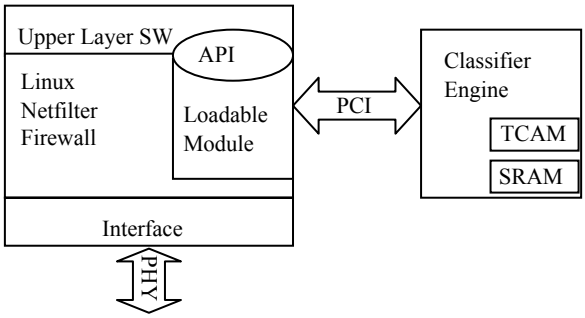

Figure 8: Linux Netfilter and Classifier Engine

firewall devices are located. In general, it is desirable to install countermeasures as close as possible to the attack sources. Thus, the System Manager has to identify those firewall devices that are closest to the sources and that provide the required functionality.

Finally, response actions should be stopped when the attack has finished. Though, it is often not possible to determine if the attack has finished as long as the response actions are enforced. In case of the web server overload attack, the firewall rules are removed after a certain period of time. Incoming traffic from the attacker will be analyzed further for a faster detection and response in case the attack persists.

5. High-Speed Packet Classification

As described in section 2.2, the Firewall Element is an abstraction for a router or a firewall device that allows enforcing response policies in order to stop or mitigate an attack. Such a firewall device can be a software firewall such as Linux Netfilter or a commercial hardware firewall.

In the scope of the DIADEM Firewall project, a high-speed packet Classifier Engine (CE) is developed and integrated into a Netfilter-based software firewall in order to offload the processing-intensive packet classification function from the firewall system into hardware for very high-speed links. The CE is based on the P2C algorithm [8] which is a multi-field packet classification scheme that searches the fields independently in a first phase and determines the final classification result using a TCAM in a second phase. The P2C concept is shown in Figure 7 for 5-tuple classification.

A key element of the P2C scheme is novel encoding of the intermediate search results, which allows a significant reduction of the storage requirements and minimizes the dependencies within the field search structures, thus enabling fast incremental updates. The encoding involves several styles that can be applied simultaneously and allow several performance aspects to be balanced at the granularity of individual rules. The scheme can be used in combination with various search algorithms and memory technologies. The P2C implementation for the CE performs the field searches using the Balanced Routing Table search (BaRT) algorithm [9]. BaRT is a scheme for exact- and prefix-match searches that achieves wire-speed search performance in combination with high storage-efficiency and fast dynamic updates, through application of a novel hash function. BaRT can efficiently process the search key in segments of about 8 bits, requiring in the worst-case only four memory accesses to search a 32-bit IP address and two accesses for a 16-bit port number, all of which can be performed efficiently in a pipelined fashion. A detailed description of BaRT is presented in [9].

FPGA prototypes based on the combination of P2C and BaRT have demonstrated the feasibility of achieving wire-speed classification performance for OC-192 and OC-768 with off-the-shelf technology. The prototype supported 1733 rules of a commercial firewall using only 2 KB of SRAM in combination with 5 KB of TCAM.

The CE will be implemented on an FPGA board and connected to a Linux Netfilter firewall through a PCI bus. It will be configured and initialized through the upper layer software of the Firewall Element. The lookup tables, which are a set of packet filter rules, are embedded in SRAM. The multi-field search, currently performed through a TCAM can also be implemented using TCAM emulation in SRAM. The communication between the Firewall Element and the CE is realized through an API located in user space and a loadable kernel module embedded in the Linux kernel. The basic architecture can be seen in Figure 8.

6. Conclusions

In this paper, we discussed DDoS attack detection and response, in particular the web server overload DDoS attack. To date, no efficient and cost-effective solution for the detection and mitigation of DDoS attacks is available. This is mostly for the reason that DDoS attacks can be detected at the victim, but effective countermeasures need to be enforced closer to the attack sources. Thus, protection systems deployed at a single point in the network cannot cope with DDoS attacks efficiently.

The DIADEM Firewall architecture is a significant improvement as it provides distributed detection and mitigation of DDoS attacks within the network. We did present how to deploy the DIADEM Firewall architecture as a protection against web server overload attacks, which present a common threat to high-profile web servers. Finally, we showed how response actions are accelerated using a high-speed packet classifier engine.

Next steps of the project include experiments in a test environment that resembles an operational broadband network. Furthermore, the DIADEM Firewall architecture

will be used to implement efficient detection and response mechanisms against other highly distributed DoS attacks, such as TCP SYN flood attacks, profiting from the distributed nature of the DIADEM Firewall architecture.

Acknowledgments

We gratefully acknowledge support from EU FP6 for DIADEM Firewall Project FP6 IST-2002-002154.

References

[1] DIADEM Firewall Homepage, http://www.diadem-firewall.org/

[2] Claise, B. et al., "IPFIX Protocol Specifications", Internet Draft, Work in Progress, draft-ietf-ipfix-protocol-15.txt, 2005.

[3] Enns, R. et al., "NETCONF Configuration Protocol", Internet Draft, Work in Progress, draft-ietf-netconf-prot-06, 2005.

[4] Debar, H. et al., "The Intrusion Detection Message Exchange Format", Internet Draft, Work in Progress, draft-ietf-idwg-idmef-xml-14, 2005.

[5] Snort Homepage, http://www.snort.org/.

[6] Paul, O., Kiba, J.-E., "Tradeoffs for Web Communications Fast Analysis", *IFIP Networking Conference (NETWORKING) 2005*, May 2005.

[7] Elvin: Content Based Messaging, http://elvin.dstc.edu.au/.

[8] J. van Lunteren and T. Engbersen, "Fast and Scalable Packet Classification," *IEEE Journal on Selected Areas in Communications,* vol. 21, No. 4, May 2003.

[9] J. van Lunteren, "Searching Very Large Routing Tables in Wide Embedded Memory," *IEEE GLOBECOM'01*, vol. 3, pp. 1615-1619, November 2001.