DoS Protection in the Cloud through the SPECS Services

Valentina Casola, Alessandra De Benedictis
DIETI
University of Naples Federico II
Napoli, Italy
{casolav,alessandra.debenedictis}@unina.it

Massimiliano Rak
DIII
Second University of Naples
Aversa, Italy
massiliano.rak@unina2.it

Umberto Villano
DING
University of Sannio
Benevento, Italy
villano@unisannio.it

Abstract—Security in cloud environments is always considered an issue, due to the lack of control over leased resources. In this paper, we present a solution that offers security-as-a-service by relying on Security Service Level Agreements (Security SLAs) as a means to represent the security features to be granted. In particular, we focus on a security mechanism that is automatically configured and activated in an as-a-service fashion in order to protect cloud resources against DoS attacks. The activities reported in this paper are part of a wider work carried out in the FP7-ICT programme project SPECS, which aims at building a framework offering Security-as-a-Service using an SLA-based approach. The proposed approach founds on the adoption of SPECS Services to negotiate, to enforce and to monitor suitable security metrics, chosen by cloud customers, negotiated with the provider and included in a signed Security SLA.

Index Terms—Cloud security monitoring; DoS attacks; OS-SEC; Security Service Level Agreement;

I. INTRODUCTION

Owing to wide diffusion of the cloud computing paradigm, more and more organizations and individual customers are relying upon cloud services to carry out their business. Unfortunately, cloud service customers (CSCs) do not have control over cloud infrastructures, and so they cannot eliminate or even mitigate possible structure vulnerabilities. Moreover, a CSC has no possibility to react to intrusions or to traditional network attacks targeting availability, confidentiality and integrity of cloud resources and services.

Intrusion Detection systems (IDSs) are widely adopted for identifying malicious behaviors targeted to protected hosts or network environments. Several open-source and commercial IDS products have been developed and are widely used in traditional and cloud environments to monitor the level of security of deployed infrastructures and services. It is clear that the ability of monitoring security through suitable tools, as the above-mentioned IDSs, is fundamental for cloud service providers (CSPs) aiming at protecting their infrastructures (and their reputation), but it is also of uttermost importance for cloud customers. A CSC should be provided with the capability of directly monitoring his services and resources, and of configuring monitoring tools according to his requirements. The proposal of an Intrusion Detection-as-a-service (IDSaaS), dealt with here, is a first step in this direction.

In this paper we present a security monitoring system offering IDSaaS, able to protect cloud resources and services against intrusion attempts and denial of service attacks (DoS). The proposed monitoring system is automatically configured and activated based on a Security Service Level Agreement (Security SLA), negotiated, agreed upon and signed by cloud customer and provider. Security SLAs are contracts regulating the conditions under which target services are to be delivered to customers, and include security-related terms and guarantees that specify the level of security these services have to guarantee. According to our approach, security requirements regarding the protection from intrusions and DoS attacks are negotiated by the customer and included in a Security SLA, and they are automatically enforced in the target service’s supply chain by the activation of suitable security mechanisms. The security mechanisms, which include monitoring systems, are configured on the basis of the signed SLA, and of the corresponding Service Level Objectives (SLOs) with associated security metrics to be used for monitoring purposes.

In fact, the work presented in this paper is part of a wider work carried out in the FP7-ICT programme project SPECS, whose main objective is the design and development of a framework offering Security-as-a-Service using an SLA-based approach.

This paper is organized as follows. In Section II we present related work. In Section III, we deal with the SPECS framework and its approach to Security-as-a-Service based on a Security SLA. In Section IV we introduce the proposed security mechanism for providing DoS detection capabilities, and in Section V we show its behavior in the case of attacks. Finally, in Section VI, we draw our conclusions.

II. RELATED WORK

There is a relatively large literature on the use of IDS in clouds to prevent DoS and DDoS attacks coming from the outside and targeting resources in the cloud [1], [2], [3], [4], [5]. An interesting solution of IDS working at hypervisor level, which does not require additional software to be installed in virtual machines is presented in [6]. Good surveys on these topics are [7] and [8]. There is also on-going, yet unpublished,

\footnote{A supply chain consists of the set of components invoked to provide the negotiated target service, along with their configurations and needed security services.}

\footnote{http://www.specs-project.eu}
work on the reverse problem, i.e., on the use of cloud resources to perform DoS attacks externally to the cloud.

On the other hand, to the best of our knowledge, not so much work has been done in the area of configuring security requirements specified through SLA documents. Karjoth et al. [9] introduce the concept of Service-Oriented Assurance (SOAS). SOAS adds security provisioning assurances (an assurance is a statement about the properties of a component or service) as part of the SLA negotiation process.

Smith et al. [10] present a WS-Agreement approach for a fine-grained security configuration mechanism to allow an optimization of application performance based on specific security requirements. Brandic et al. [11] present advanced QoS methods for meta-negotiations and SLA-mappings in Grid workflows. Meta-negotiations are defined by means of a document where each participant may express the pre-requisites to be satisfied for a negotiation, the supported negotiation protocols and document languages for the specification of SLAs. In the pre-requisites there is the element <security> that specifies the authentication and authorization mechanisms that the party wants to apply before starting the negotiation.

Ficco et al. [12], [13], [14] proposed architectures to detect intrusions on cloud services with a focus on cloud-specific attacks, and considered the possibility to use SLAs in the IDS service offering.

However, none of the approaches reported in the literature is similar to the one proposed in this paper, which is focused on the need to map security mechanisms to security features under monitoring control. Furthermore, no existing work correlates the monitoring system to the user security requirements and to the desired Service Level Objectives.

III. The SPECS Framework

The main goal of the SPECS project is the development of a framework for the management of the SLA life cycle, intended to build applications (SPECS applications) offering services whose security features are stated in and granted by a Security SLA [15], [16].

The SPECS framework addresses both CSPs’ and CSCs’ needs by providing techniques and tools for (i) enabling user-centric negotiation of security parameters in a cloud SLA, by providing a trade-off evaluation process among customers and CSPs, in order to compose cloud services guaranteeing a minimum security level; (ii) monitoring in real-time the fulfillment of the SLA agreed upon, notifying both CSCs and CSPs when an SLA is not being fulfilled, and (iii) enforcing the agreed SLAs in order to keep a sustained Quality of Security (QoS) that fulfills the specified security parameters. The SPECS enforcement framework is also able to “react and adapt” in real-time to fluctuations in the QoS, by advising and/or applying suitable countermeasures.

In the typical SPECS usage scenario there are three main involved parties:

- **SPECS Customer**: the end-user, i.e., the CSC of the cloud services covered by Security SLAs;
- **SPECS Owner**: the provider of the cloud services covered by Security SLAs;
- **External CSP**: an independent (typically public) CSP, which is unaware of the SLAs, and provides just basic resources without security guarantees;

As for the interactions among such parties, the SPECS Customer uses the cloud services offered by the SPECS Owner, which mainly acts as a broker by acquiring resources from External CSPs and by reconfiguring/enriching them in order to match the customer’s security requirements. The SPECS Owner may possibly be supported by a Developer, a cloud service partner, which helps the SPECS Owner in the development of SPECS applications and in the delivering of cloud services.

The SPECS framework enables to easily enrich an existing cloud service with Security SLAs, by re-using a set of available security mechanisms and by exploiting a set of services (Core services) devoted to the management of Security SLAs’ life cycle. As shown in Figure 1, a SPECS application orchestrates the SPECS Core services dedicated to Negotiation, Enforcement and Monitoring, respectively, to provide the desired service (referred to as “Target Service” in the picture) to the SPECS Customer (i.e., to the end-user). The Core services run on top of the SPECS Platform, which provides all the functionality related to the management of Security SLAs’ life cycle and needed to enable the communication among Core modules. In addition to this functionality, provided by the “SLA Platform services”, the SPECS Platform also provides support for developing, deploying, running and managing all SPECS services and related components. In the figure, these services are referred to as “Enabling Platform services”.

Security-related SLOs are negotiated (Step 1) based on the SPECS Customer’s requirements. A set of compliant offers, each representing a different supply chain to be implemented, is identified with the help of an interoperability layer (represented by the SPECS SLA Platform services), which is
also responsible for their validation (e.g., for verifying their actual feasibility based on the current system configuration) (Step 2). Of course, given a set of security requirements expressed by the SPECS Customer, multiple supply chains may be identified, each characterized by its own cost and associated security level. The resulting supply chains may be ranked to help the SPECS Customer choose the desired configuration. The agreed terms are included in a Security SLA that is signed by the SPECS Customer and the SPECS Owner (Step 3). Afterward, the agreement is implemented through the Enforcement services, which acquire resources from external CSPs and activate suitable components that provide, in an as-a-service fashion, the security capabilities needed to fulfill the SLOs included in the signed Security SLA (Steps 4 and 5). At the same time, suitable services and agents are activated for the monitoring of the specific parameters included in the Security SLA (Step 6). Monitoring data are collected by the SPECS Monitoring module and analyzed based on a monitoring policy: if needed, they are forwarded to the Enforcement module, which performs a diagnosis to verify whether they reveal an incoming (or already occurred) violation of the signed SLA. As a consequence, countermeasures may be adopted, consisting in re-configuring the service being delivered, or applying remediation actions defined together with the security mechanisms.

The Security SLA format [17], defined in the SPECS project based on the WS-agreement standard [18], represents security features using the following concepts:

- **security capabilities**: the set of security controls [19] that a security mechanism is able to enforce over the target service;
- **security metrics**: the standard of measurement adopted to evaluate security levels of the services offered;
- **SLOs**: the conditions, expressed over security metrics, representing the security levels that must be respected according to the SLA.

A security mechanism is a software devoted to grant security features over the target service. In practice, in SPECS, it is developed as a Chef cookbook. Chef\(^3\) is a tool for the automation of cloud systems and infrastructures, which allows to automatically configure and activate services and applications on cloud resources. The information associated to a security mechanism is maintained in the mechanism metadata. Metadata are defined by the mechanism developer and include all information needed to automate the mechanism deployment and configuration, according to the SLA content, and to monitor its behavior at run-time. In light of the above, the development of SPECS security mechanisms mainly consists in the development of suitable Chef cookbooks, possibly obtained by adapting existing security software, and in the definition of the associated metadata.

### IV. SPECS Security Mechanism for DoS Protection

In this section, we will tackle the main object of this paper by presenting a realistic example, i.e., the use of a SPECS application offering a cloud service that instantiates web containers protected against DoS attacks. Let us first consider the scenario without SPECS. At the state of the art, a web developer that wants to deploy such a service on resources leased from a public CSP is responsible for applying suitable security configurations. There are indeed existing appliances that offer predefined services (e.g., a pre-configured web server), but there is no standard way to check for the security features that are possibly provided by CSPs. Hence, the web developer has to (i) manually check each CSP’s offers, (ii) evaluate one by one the offers and compare them to his own security requirements, and (iii) apply suitable configurations by means of external tools if the required security level is not natively supported (and this is almost always the case). But the hardest task is to monitor continuously the running service, in order to spot security issues. This is unsupported by CSPs, and is completely up to the web developer.

This is the point where SPECS comes profitably into play. Providing the service through a SPECS application gives a number of significant advantages. As a matter of fact, the SPECS application (i) offers a single interface to select among different offers of multiple providers, (ii) enables the web developer to specify explicitly the needed security capabilities on the target web container by negotiating and signing an SLA with the CSP, (iii) automatically acquires and configures the resources (i.e., virtual machines) to enforce the security controls requested, (iv) enables continuous monitoring of the security metrics negotiated, and (v) automatically finds and applies remedies to (some of) alerts in the case of SLA violations.

Protecting a web container against DoS attacks requires the implementation of a DoS detection and mitigation capability (more on this later). In order to provide such capability, by exploiting the SPECS framework, we developed a security mechanism based on the popular OSSEC tool\(^4\). OSSEC has been integrated with the SPECS Enforcement services, providing the possibility to configure and to activate it based on a signed SLA. In the following subsection, we deal with the mentioned DoS detection and mitigation capability. Then (Subsection IV-B), we provide a brief overview of the OSSEC tool. The integration of OSSEC with the SPECS framework is discussed in Subsection IV-C. In Section V, we instead describe the testbed used in our experiments, presenting a running example of the behavior of our system under a well-known attack.

It is worth pointing out that the SPECS application offering secured web containers is publicly available on BitBucket\(^5\).

---

\(^3\)https://www.chef.io/chef/

\(^4\)http://www.ossec.net/

\(^5\)https://bitbucket.org/specs-team/specs-enforcement-webpool
OSSEC detects anomalous activities in a system monitoring, rootkit detection, real-time alerting and active response that performs log analysis, file integrity check, policy monitoring, rootkit detection, real-time alerting and active response. OSSEC detects anomalous activities in a system mainly by log analysis; it collects, analyses and correlates all logs recording the activity of running processes, generating alerts in case of suspicious behavior.

From an architectural point of view, OSSEC is based on a client-server paradigm. A Manager (the server) stores all configuration options, the databases used for integrity checking, the logs and the auditing system entries, and executes the main analysis logic. Small Agents, typically residing on the systems to be monitored (the clients), collect all relevant information and forward them to the Manager. However, it is possible to configure the Manager in order to act in an agentless mode, for those systems that do not allow the installation of Agents on acquired resources.

Both the Manager and the Agents execute a set of processes in background. On every host running an Agent, a LogCollector collects generated logs, and an Agentd process compresses and encrypts them before forwarding them to the server. Here, an Analysisid process sends them to a filtering chain: in a pre-decoding phase, static variables (e.g., hostname, program name, timestamp, . . .) are first extracted. In the actual decoding phase the key variables (e.g., the IP addresses that try to contact the host) are processed. The filtering is performed on the basis of specific rules or patterns: if an alarm is generated, OSSEC can provide either a passive or an active response. In the former case, an e-mail is sent to the system administrator, while in the latter it is possible to invoke specific actions specified in a script (e.g., isolation of a malicious IP).

### C. Integration of OSSEC

In order to make OSSEC available on-demand as an automatically-enforceable security mechanism, we developed an OSSEC Adapter. This is a REST interface that allows to invoke and to control remotely both the server and the client components. The OSSEC Adapter acts as a gateway that interfaces with the OSSEC Manager and receives all the events that the OSSEC Manager generates. In this way, it is possible to automatically create and configure several Agents and to bind them to the Manager.

For each rule match that occurs during the filtering process, the OSSEC Manager triggers an alert, and generates a message in the format reported in Table III. Such message is translated in order to be compliant with the format adopted by the Event Hub component, residing in the SPECS Monitoring module and devoted to collecting all event notifications coming from the deployed monitoring systems. This component is

---

**TABLE I**

<table>
<thead>
<tr>
<th>Control Framework</th>
<th>Control Family</th>
<th>Security Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIST-800-53r4</td>
<td>CA - Security Assessment and Authorization</td>
<td>CA-5 Continuous Monitoring</td>
</tr>
<tr>
<td></td>
<td>SC - System and Communication Protection</td>
<td>SC-5 Denial of Service Protection</td>
</tr>
<tr>
<td></td>
<td>SI - System and Information Integrity</td>
<td>SI-4 Information system monitoring</td>
</tr>
</tbody>
</table>

**TABLE II**

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Metric Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection_latency</td>
<td>It represents the time interval between the first symptom of a (detected) attack and the generation of a message event</td>
</tr>
<tr>
<td>False_positives</td>
<td>It reports the number of detected false positives in a predefined time interval</td>
</tr>
<tr>
<td>Detected_attacks</td>
<td>It reports the number of attacks detected in a predefined time interval</td>
</tr>
<tr>
<td>Attack_Report_Max_Age</td>
<td>It represents the frequency of attack report generation</td>
</tr>
</tbody>
</table>

**B. OSSEC**

The security mechanism developed in the SPECS framework to grant the security capability described above exploits OSSEC, an open-source host-based Intrusion Detection System that performs log analysis, file integrity check, policy monitoring, rootkit detection, real-time alerting and active response. OSSEC detects anomalous activities in a system.
TABLE III
OSSEC ALERTS

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crit</td>
<td>the level of severity associated with the alert</td>
</tr>
<tr>
<td>id</td>
<td>identifies the rule that was violated</td>
</tr>
<tr>
<td>component</td>
<td>the location of the log file that triggered the alert</td>
</tr>
<tr>
<td>classification</td>
<td>identifies the group or the groups that the violated rule belongs to</td>
</tr>
<tr>
<td>description</td>
<td>a textual description of the violated rule</td>
</tr>
<tr>
<td>message</td>
<td>the log string or group of strings that triggered the alert</td>
</tr>
<tr>
<td>acct</td>
<td>a user identifier of the machine that was found responsible of the alert (e.g., the username of the attacking machine)</td>
</tr>
<tr>
<td>src_ip</td>
<td>source IP address that was found responsible of the alert</td>
</tr>
<tr>
<td>src_port</td>
<td>source port used for perpetrating the attack</td>
</tr>
<tr>
<td>dst_ip</td>
<td>destination IP address of the machine where the alert was generated</td>
</tr>
<tr>
<td>dst_port</td>
<td>port of the agent where the alert was generated</td>
</tr>
<tr>
<td>file</td>
<td>path to the file where a change was detected</td>
</tr>
<tr>
<td>md5_new</td>
<td>MD5 hash after file modification</td>
</tr>
<tr>
<td>sha1_new</td>
<td>SHA1 hash after file modification</td>
</tr>
<tr>
<td>sha1_old</td>
<td>SHA1 hash before file modification</td>
</tr>
<tr>
<td>src_city</td>
<td>geographical location of the machine that caused the alert</td>
</tr>
<tr>
<td>dst_city</td>
<td>geographical location of the machine where the alert was generated</td>
</tr>
</tbody>
</table>

TABLE IV
EVENT HUB EVENT FORMAT

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>identifies the specific OSSEC Agent that originates the message.</td>
</tr>
<tr>
<td>object</td>
<td>identifies the specific log that generated the alert.</td>
</tr>
<tr>
<td>labels</td>
<td>labels the event to enable aggregation and filtering (e.g., ossec-switch-user to identify that the event is related to an attempt to change user). The Adapter will add standard OSSEC labels to every message received from the OSSEC Manager.</td>
</tr>
<tr>
<td>type</td>
<td>identifies the type of the payload represented by the “data” field, in order to enable its parsing (e.g., syslog or JSON for OSSEC logs).</td>
</tr>
<tr>
<td>data</td>
<td>contains the alert message payload, as generated by OSSEC.</td>
</tr>
<tr>
<td>timestamp</td>
<td>contains the message timestamp. The Adapter will report the time at which the event is generated.</td>
</tr>
</tbody>
</table>

All OSSEC modules, along with the modules for communication and for the management of the alerts have been packed to enable automatic configuration and deployment. In particular, two different packages are available, for the client and the server side, respectively.

V. EXPERIMENTATION

In this section, we propose a brief description of the web container SPECS application behavior when the DoS protection is activated, i.e., when the DoS detection and mitigation capability is enforced on the top of a web server provided by a CSP. In particular, we show an example of operation in case of an attack. It should be noted that in the following we will not deal with the whole SPECS framework deployment, focusing only on the components directly used by our IDSaaS solution: the OSSEC Manager, the OSSEC Agent and the Event Hub. The Event Hub is a standard SPECS component devoted to send alert notifications to the SPECS monitoring module and to the web server. It is based on Heka\(^6\), an open source stream processing software system developed by Mozilla, useful for loading and parsing log files and for performing real time analysis, graphing, and anomaly detection on any data flow.

Following up the negotiation process of the SPECS application, we acquired a pool of three virtual machines on Amazon AWS (t2.micro instances), pre-configured with the OpenSUSE operating system distribution, hosting respectively:

- the OSSEC Manager package;
- the OSSEC Agent package together with an Apache v2.2.2 instance (the client);
- the OSSEC Event Hub Adapter;
- the Event Hub component.

Moreover, we acquired an additional instance of VM with a different user to run the attacker software. As for the type of attack to be performed, we searched for exploits of Apache’s vulnerabilities enabling the execution of unauthorized code, administration privilege upscaling and also leading to the unavailability of the service. We identified several exploit databases, classified according to the vulnerabilities present in the CVE dictionary\(^7\), from the Exploit Database\(^8\) we choose the well-known Slowloris attack\(^9\). This attack tries to open many connections to the target web server, and to hold them open as long as possible. It accomplishes this by opening connections to the target web server and sending partial requests, so that affected servers will keep these connections open, filling their maximum concurrent connection pool, eventually denying additional connection attempts from clients.

We prepared the attacker machine so as to run the Slowloris attack and, on the defense side, we prepared a script for counteracting the attack (by closing all active ports) when the Manager requires an active response to a detected alert. It is worth pointing out that the Manager already has several built-in responses that can be applicable for this attack, but they are not satisfactory to stop a running attack. Therefore, we built our own solution. The script was included in the client package so as to be deployed together with the Agent on the web server machine. It is activated when a specific rule is matched on the Manager (too many 400 error codes returned) and closes all the active ports on 80/TCP by means of the command `fuser -k 80/TCP`. After, it re-activates the Apache web server. This way, the malicious connections

\(^6\)https://hekad.readthedocs.org/en/v0.9.2/
\(^7\)https://cve.mitre.org
\(^8\)http://www.exploit-db.com/
\(^9\)http://ha.ckers.org/slowloris/
are shut down and the normal operation is re-established.

In our experiments, we run the attack on the attacker machine by means of the command: `./slowloris.pl -dns www.example.com -port 80 -timeout 2 -num 500 -tcp 5`

The command parameters identify the address and port to attack (dns and port respectively), the re-transmission time period (timeout), the number of sockets that are opened to send packets (num), and timeout window of TCP (tcpto).

The result of the attack without the DoS Detection and Mitigation capability would be the unavailability of the server, as the tool opens several sockets and periodically sends requests with high frequency. This would be registered in the Apache access_log with several entries showing multiple requests from the same IP with response code 400-Bad Request.

With the capability activated, the high number of 400-Bad Request responses is instead notified to the Manager, which finds a match with one of its rules and activates, on the Agent, the script to stop the attack. The Manager also activates the OSSEC Adapter, which notifies the detected attack to the SPECS EventHub; it in turn collects the events and counts the number of detected attacks. The actual attack detection latency, which is measured and sent as an event to the EventHub by the OSSEC Adapter, depends on the request timeout of Apache, which has to expire before a 400-Bad Request is generated. The SPECS application enables the Customer to monitor the web container, reporting the actual value of the security metrics agreed in a dedicated web page, thus allowing the continuous monitoring of the agreed SLA.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have described the implementation of a security mechanism for Denial of Service protection and mitigation, based on OSSEC and exploiting the SPECS services. The solution presented enables to offer such security capability in an as-a-service fashion, as stated in the Security Service Level Agreement agreed upon and signed by cloud service provider and customer. We have also described the integration of the mechanism in the SPECS framework and demonstrated its use launching a DoS attack against a protected web server.

As for our future work, we aim at extending the DOS protection capability by improving the definition of the associated security metrics and measurements, and by providing solutions to automate the tuning of the protection based on ongoing events.

ACKNOWLEDGMENT

This research is partially supported by the grant FP7-ICT-2013-11-610795 (SPECS).

REFERENCES