An Approach for Adaptive Security of Cloud Applications within the ATMOSPHERE Platform

Jorge L. M. da Silva[1], Alexandre M. Braga[1], Cecília M. F. Rubira[1] and Ricardo Dahab[1]

[1] Institute of Computing, State University of Campinas
Av. Albert Einstein, no. 1251, Cidade Universitária Zeferino Vaz
Campinas, SP, Brazil, Zip Code 13083-852

jorgem@unicamp.br, [alexbraga|cmrubira|rdahab]@ic.unicamp.br

Abstract. *Self-protecting systems can reduce response time to known attacks by automating decision-making processes in security operations. This paper briefly describes the ATMOSPHERE platform for monitoring and enforcement of trustworthiness in cloud systems, as well as proposes a way to enhance its Trustworthiness Monitoring & Assessment framework (responsible for analyzing and planning adaptation actions) in order to instantiate the concept of adaptive security for self-protecting cloud infrastructures and applications. The paper approaches adaptive security based upon adaptive Web Application Firewalls, and enhances a software-based, feedback control loop (named MAPE-K) for monitoring and analysis of security events, as well as the planning and execution of adaptation actions for securing cloud applications. This is a work in progress, currently under development, to be integrated to ATMOSPHERE's framework.*

1. Introduction

Nowadays, in highly automated software environments that apply continuous deployment, it is quite common to see insecure applications being deployed into cloud infrastructures, despite the presence of known vulnerabilities in those applications. In a common scenario, ready-to-deploy applications may still have *known* vulnerabilities, which remain in code due to insufficient security tests or lack of opportunity to correct security issues. Nevertheless, they are still considered good enough for deployment, from a feature value point of view.

This may happen because, as cloud systems become more interconnected and diverse, and time-to-market constraints impose tight schedules, software architects are less able to anticipate and design proper configurations of security features. As such, these issues are left to be dealt with at run time, delegating design decisions concerning security to production environments, eventually resulting in easily compromised environments. In this context, adaptive security for self-protecting cloud system and applications means that security features in cloud systems should be able to dynamically adapt their behaviour, hardening their security settings according to perceived changes on vulnerability context and exposure surface of deployed cloud applications. In a concrete example detailed later, an adaptive Web Application Firewall can detect the deployment of an application with known vulnerabilities (such as SQL Injection), and dynamically change its own behavior to initially monitor and then to block malicious inputs for that insecure application.

The main objective of this paper is to discuss the concept of adaptive security for self-protecting systems in the context of the ATMOSPHERE's cloud platform. Its main contribution is in explaining important design decisions and implementation details anticipated so far, aiming at a viable implementation of this concept. The text is organized as follows. Section 2 contains background and related work. Section 3 overviews the ATMOSPHERE platform and explains its framework for Trustworthiness Monitoring and Assessment (TMA), while Section 4 discusses, with a running example, the approach for adaptive security based upon the TMA framework. Finally, Section 5 discusses the implementation and Section 6 concludes the text.

2. Background and Related Work

Security in cloud systems [Vacca 2017, Pearson and Yee 2013] involves not only protecting the infrastructure against attacks, but also hardening applications (if possible, in advance) in order to make them fault tolerant against direct attacks and infrastructure malfunctioning or compromise. For instance, sometimes, hardening cloud applications may involve protecting exposed APIs, adopting security configurations for containers, or even using secure hardware to grant secrecy and privacy near to the bare metal [Arnautov et al. 2016].

The challenge of dynamically secure applications resides in providing appropriate *feedback loops* to developers, administrators, and (adaptive) systems, with appropriate tools and supporting infrastructures, improving their ability in handling security issues at run time, generally within short response time. Thus, the missing link that connects automated security controllers and self-protecting systems is the proper feedback from the environment to assist developers or deployed systems in making safe and secure choices about security actions at run time.

Adaptive systems [Kephart and Chess 2003, Kephart 2011, Lalanda et al. 2013, IBM 2006, Brun et al. 2009] can help to solve issues related to delayed design decisions regarding security to production environments where overwhelmed administrators are traditionally unable to reconfigure an applications' design in response to environmental changes. This way, adaptive security systems can be understood as any solution able to protect systems, users, or data against run time threats via the enforcement of adjustable defensive strategies [Tziakouris et al. 2018]. These systems are usually supported by anomaly detectors [Wang et al. 2018], and encompass activities ranging from data capture, management and analysis, to automated decision making and system operations.

Adaptive systems can adopt the feedback-loop architecture known as MAPE-K [IBM 2006] (**M**onitor, **A**nalyzer, **P**lanner, and **E**xecutor over a **K**nowledge base) which, among other features, makes feedback loops explicit in software systems. MAPE-K's architecture organizes a control loop into four components that share a common knowledge base [IBM 2006], as follows:

- **Monitor** collects, aggregates, filters and reports details (such as metrics) collected, from managed resources, by **probes**.
- **Analyzer** correlates and models recurring events in such a way that it can learn about the environment and predict future changes.
- **Planner** constructs action chains needed to achieve a new stable state and uses policies to guide its work in generating recipes for execution.
- **Executor** controls the execution of a plan performed by **actuators** and, sometimes, considers dynamic updates to running plans.

3. ATMOSPHERE's Trustworthiness Monitoring and Assessment Framework

ATMOSPHERE [ATMOSPHERE 2018a] is a 24-month project aiming at the design and development of an ecosystem comprised of a **framework** and a **platform** enabling the implementation of next-generation trustworthy cloud services on top of an intercontinental hybrid and federated resource pool. The framework considers a broad spectrum of trustworthiness properties and their measures, while the platform supports the development, build, deployment, measurement and evolution of trustworthy cloud resources, data management services and data processing services, and is demonstrated by sensitive application scenarios for cloud-enabled secure and trustworthy applications.

The design and implementation of ATMOSPHERE's framework and platform rely on lightweight virtualization, hybrid resources and Europe&Brazil federated infrastructures. The part of ATMOSPHERE's platform discussed in this paper is the framework for Trustworthiness

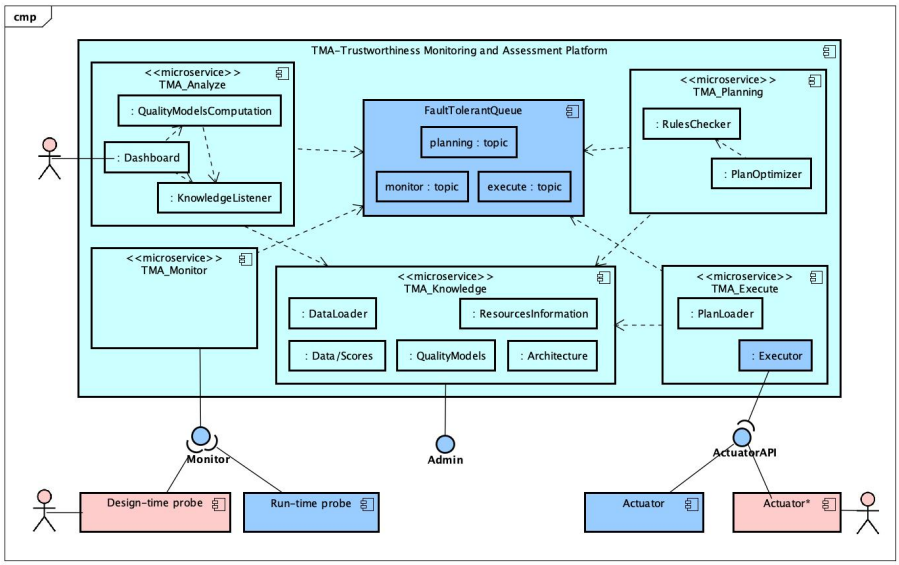

Figure 1. Architecture of TMA based on MAPE-K (from [ATMOSPHERE 2018b]).

Monitoring and Assessment (TMA), which architecture is based upon a MAPE-K's feedback control loop. Figure 1 is a UML diagram showing TMA's architecture, which is internally designed using micro services and each component represents a MAPE-K functionality. Briefly speaking, TMA's **Monitor** provides a generic interface that can be used by **probes** installed in different layers of the system to send data about the collected measurements and events. These data are then stored at TMA's **Knowledge**, and kept to be used later by other TMA's components. Similarly, TMA's **Planner** exposes an interface that allows for TMA's **Analyzer** to notify it about the need for adaptation, while TMA's **Executor** provides an interface through which TMA's Planner can submit an adaptation plan to be performed. Finally, **actuators** provide an interface with predefined adaptations that can be activated by TMA's Executor. The three main components sequence their interaction by notifying other components by a fault tolerant queue (e.g., Apache Kafka).

4. Adaptive Security for Self-Protecting Cloud Infrastructures and Applications

One way to quickly mitigate known security risks of deployed applications is to dynamically reduce the chance of vulnerability exploitation by shielding insecure applications against those known vulnerabilities, while gaining time for developers to correct these vulnerabilities in future deployments. However, in general, dynamic shields are compensatory controls that do not fully substitute secure coding techniques for avoiding known vulnerabilities, since dynamically shielding applications consume computational resources (e.g., memory, CPU), impose delays to application's communications and, depending on the rigor of attack detection, produce too many false alarms, possibly overloading response mechanisms.

Traditional Web Application Firewalls (WAFs) are common examples of application shields. WAFs can monitor application's HTTP interfaces and block malicious requests or inspect responses against data leaks according to predefined rules. Modern WAFs apply the concept of virtual patching, meaning that a WAF's rule set can be enhanced to protect against specific attacks found in deployed applications. That patch is said to be virtual because it is applied to the WAF's context for the targeted application, but not to the application itself, which remains the same. In general, ordinary WAFs are not adaptive and virtual patches are performed with human intervention.

Non-adaptive WAFs cannot dynamically adapt to newly deployed applications with known vul-

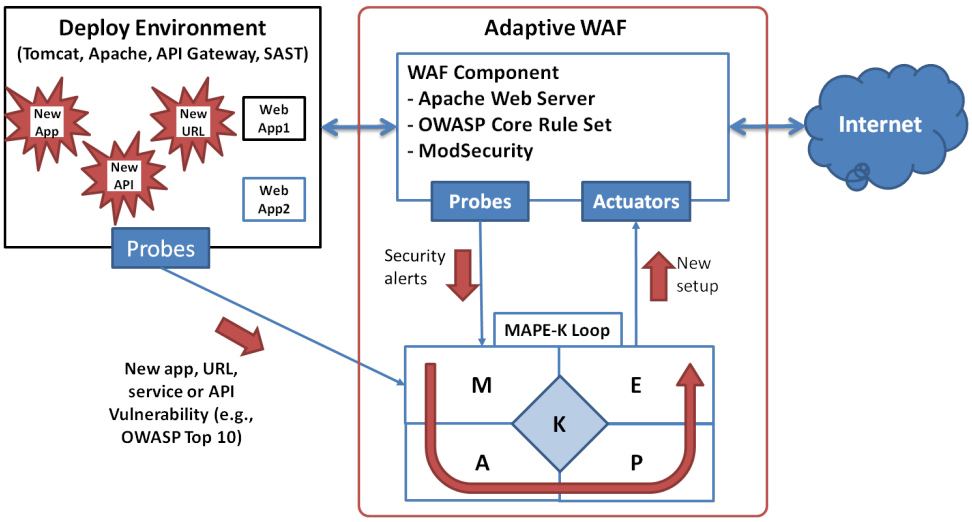

Figure 2. An Adaptive Web Application Firewall with a feedback control loop.

nerabilities. That is, ordinary WAFs cannot dynamically change their behaviour by automatically updating their security settings based on malicious content detected on HTTP requests/responses or context changes in web containers, such as the deployment of new and probably insecure applications. On the other hand, the **Adaptive WAF** is an enhanced WAF able to dynamically modify its rules to harden protections in front of insecure applications within its security domain, while still being able to weaken its security settings for monitored applications that had corrected their issues.

Differently from ordinary WAF, the main advantage of adaptive WAF is perceived in highly automated (e.g., DevOps) environments, where WAFs have to automatically adapt to knowingly insecure applications deployed into production environments, allowing enough time for developers to actually patch vulnerabilities automatically detected prior to deployment. In a possible scenario depicted in Figure 2, an Adaptive WAF can be implemented by integrating an ordinary WAF and a MAPE-K control loop (such as ATMOSPHERE's TMA framework) for monitoring and analyzing security alerts, as well as planning for appropriate actions to be taken to modify (the rigor of) WAF's behaviour in response to possible (known) cyber-attacks.

Figure 2 illustrates an Adaptive WAF placed in front of deployed applications, midway between deployment environment and the Internet. This Adaptive WAF is composed of a WAF component and a MAPE-K loop. These two components interact by means of probes (signaling alerts to MAPE-K loop) and actuators (updating WAF's settings). Also, probes placed at deployment environment notify MAPE-K about changes on deployed applications and their vulnerabilities.

This work adopts ModSecurity [Trustwave and contributors 2019], an open-source WAF used by security practitioners in many actual deployments [Folini and Ristić 2017], as the WAF component for Adaptive WAF. ModSecurity has monitoring features that can be used to implement sensors to feed probes with alert's data addressed to TMA's Monitor. Also, ModSecurity can interface with probes and actuators through its API and plug-ins. Initially, ModSecurity acts as a sensor and produces security alerts on potential attacks against applications. Then, ModSecurity receives control instructions (i.e., commands) from actuators to modify its security settings.

In a **running example**, when deploying an application with known vulnerabilities (e.g., SQL Injection) to a web server (e.g., Apache) or an application container (e.g., Tomcat) in production environment, Adaptive WAF is warned by probes attached to Static Analysis Security Tools (SAST) that a knowingly insecure application has just been deployed under its security domain.

For instance, this initial warning can be generated by a design-time probe parsing a report generated by a SAST during the build process, just before an actual deployment. Then, Adaptive WAF, through its MAPE-K loop, automatically activates its probes and actuators by dynamically configuring ModSecurity to monitor inputs for that application (according to OWASP's Core Rule Set [Project 2019]), looking for malicious content related to SQL Injection in HTTP requests.

Continuing the example, internally, Adaptive WAF behaves as follows. When security alerts are first fed to TMA's control loop by Monitor's probes in the deployment environment, Analyzer realizes that incoming alerts refer to a solvable issue (e.g, SQL Injection) and decides to listen to application's inputs before blocking (instead of blocking requests anyway) and instructs Planner to adapt ModSecurity settings accordingly. Different plans accomplish different security policies and configure ModSecurity in different ways, ranging from adding URLs and domains to a monitored location, to switching from monitoring mode to blocking mode, to turning existing rules on and off, to actually building and enabling new rules. Planner builds a recipe to modify ModSecurity's setup (e.g., install a new context for the targeted application and start listening to its inputs) and selects the appropriate Executor to perform these actions. Executor uses the plan to activate actuators that interact with ModSecurity through its API or by modifying configuration files.

Finally, in a simple implementation that concludes the loop, an actuator replaces ModSecurity's configuration file by a new one including the needed configuration to listen to application's inputs, loading those new settings. In further rounds of MAPE-K's loop, upon detecting that malicious input is actually being injected, Adaptive WAF may decide to block malicious requests.

5. Implementation of the Solution

ATMOSPHERE has a fully functional deployment which serves as integration environment for probes and actuators, as well as a baseline for further enhancements of domain-specific knowledge, such as the representation of WAF's alerts and vulnerabilities as models for a MAPE-K cycle. The current implementation of MAPE-K's control loop is being used in other contexts, such as privacy, elasticity and reliability, being able to monitor and analyze incoming telemetry received from probes, as well as plan for adaptations and execute adaptation actions by means of actuators in the above mentioned contexts. This is work in progress and an actual implementation of Adaptive WAF is under development. At the time of writing, sensors and probes were being developed in order to capture malicious actions shown at security alerts generated by ModSecurity.

For instance, ModSecurity generates audit logs when intercepting possible attacks and generates alerts to give operators (e.g., sysadmins) forensic information about malicious requests. These logs contain detailed information (sometimes, in JSON format) about requests, that can be used by automated decision-making processes, such as: header information, request payload, port information, response content, audit data, and matched rules. For this, Apache Kafka Connectors are being used to listen for changes that occur in the logs generated by ModSecurity, and pull in those changes automatically. Once the connector is setup, data in text file is imported to a Kafka Topic as JSON messages. Any further data appended to the text file creates an event. These events are being listened by the Connector and written to the Kafka Topic. Also, probing and monitoring can be accomplished from within an application or in a separate thread which listens for events to come and dispatches them to various registered listeners. Furthermore, as ModSecurity works both as a probe and as an actuator, is has to be able to keep states between rounds for different adaptation cycles, similar to a simplified mechanism for session management.

Finally, there are other implementation details to observe in order to correctly use ModSecurity as part of an Adaptive WAF. First, ModSecurity should be used as a reverse proxy attached to a web server running exclusively for it. This way, ModSecurity does not compete with running

applications for computational resources in the same web server [Ristic 2005]. Also, while an ordinary WAF within a web server has to suffer a reset in order to apply an updated rule set, an Adaptive WAF is restricted to only soft resets on ModSecurity's web server in order to preserve availability [Ristic 2005]. A soft reset only reloads configuration files without stopping the web server. This prevents applications from becoming unavailable during a reset.

6. Concluding Remarks

This text proposes an approach for adaptive security of self-protecting cloud systems and applications in the context of the ATMOSPHERE framework for trustworthiness of cloud systems. The concept of Adaptive WAF materializes the abstract notion of adaptive security as a dynamic shield composed by an ordinary Web Application Firewall integrated to a software-based, feedback control loop, named MAPE-K cycle. This text contributes to better understand self-protecting systems and software-based control loops by explaining (with a running example) several design decisions and implementation details that support a concrete case for adaptive security in the context of application security for cloud systems. As future work, an actual implementation of the approach will be released as part of the ATMOSPHERE platform.

Acknowledgements. This work was financially supported by **Project ATMOSPHERE** (www.atmosphere-eubrazil.eu), funded by European Commission under Horizon 2020 grant agreement no. 777154, and Brazilian MCTIC and RNP, cooperation agreement no. 51119.

References

[Arnautov et al. 2016] Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., Lind, J., Muthukumaran, D., O'keeffe, D., Stillwell, M., et al. (2016). Scone: Secure linux containers with intel sgx. In *OSDI*, volume 16, pages 689–703.

[ATMOSPHERE 2018a] ATMOSPHERE (2018a). Adaptive, trustworthy, manageable, orchestrated, secure, privacy-assuring hybrid, ecosystem for resilient cloud computing. *URL: https://www.atmosphere-eubrazil.eu/project*.

[ATMOSPHERE 2018b] ATMOSPHERE (2018b). Trustworthiness monitoring & assessment framework. *URL: https://github.com/eubr-atmosphere/tma-framework*.

[Brun et al. 2009] Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., and Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*, pages 48–70. Springer.

[Folini and Ristić 2017] Folini, C. and Ristić, I. (2017). *ModSecurity Handbook*. Feisty Duck.

[IBM 2006] IBM (2006). An architectural blueprint for autonomic computing. Technical report, IBM.

[Kephart 2011] Kephart, J. (2011). Autonomic Computing: The First Decade. *Proceedings of the 8th ACM International Conference on Autonomic Computing*, pages 1–2.

[Kephart and Chess 2003] Kephart, J. and Chess, D. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

[Lalanda et al. 2013] Lalanda, P., McCann, J., and Diaconescu, A. (2013). *Autonomic Comp.* Springer.

[Pearson and Yee 2013] Pearson, S. and Yee, G. (2013). *Priv. and Sec. for Cloud Comp.* Springer.

[Project 2019] Project, O. M. C. (2019). Owasp modsecurity core rule set. *URL: https://coreruleset.org*.

[Ristic 2005] Ristic, I. (2005). *Apache security*. O'Reilly Media.

[Trustwave and contributors 2019] Trustwave and contributors (2019). Open source web application firewall. *URL: http://www.modsecurity.org*.

[Tziakouris et al. 2018] Tziakouris, G., Bahsoon, R., and Babar, M. A. (2018). A survey on self-adaptive security for large-scale open environments. *ACM Computing Surveys*, 51(5):100:1–100:42.

[Vacca 2017] Vacca, J. (2017). *Security in the Private Cloud*. CRC press.

[Wang et al. 2018] Wang, T., Xu, J., Zhang, W., Gu, Z., and Zhong, H. (2018). Self-adaptive cloud monitoring with online anomaly detection. *Future Generation Computer Systems*, 80:89–101.