Semantic-based policy management for cloud computing environments

Krishna Sindhuja Kalusani

USC ID: T25568677

Cloud environment being provided by various cloud service providers needs different access control mechanism and diverse policies. The access control policy is complicated because it includes all the policies by different cloud service providers. Administration is a tough job due to this heterogeneity on the policies. This can be addressed by semantic web technologies for interoperability of different cloud service providers. In this paper, the same concept is used for policy management such that any user has the flexibility of having one control point to access data from any source. This is facilitated by the proposed semantic-based policy management framework. The striking features of semantic web technologies over non semantic access control methods are that they can provide runtime extensibility, adaptability and ability to analyse policies at different levels of abstraction. Semantic web uses Web Ontology Language for simulation which can be translated to any policy languages and formalisms. Semantic-based policy management framework is built on the concept of centrally expressing a user's security requirements that are applied to a user's resources scattered across the cloud. The two components of the proposed architecture are Cloud Service Provider and the Semantic based Policy Management Service.

- A CSP offers one or more cloud services that are used by cloud customers and controls access to the protected resources. It evaluates access requests made by a requester against applicable policies and is in charge of enforcing access decisions when a requester attempts to access the protected resources. Therefore, a CSP acts as a policy decision point (PDP) and policy enforcement point (PEP)
- A semantic-based policy management service (SBPMS) enables the cloud users to specify, edit and manage their access policies. It also conducts a conflict resolution on the policies to find and resolve possible conflicts and finally exports the policies into target CSPs. Therefore, SBPMS acts as policy administration point (PAP) and a policy information point (PIP)

The proposed framework uses Protege to provide an optimised solution to efficiently manage a large knowledge base with authorisation information.

When the proposed architecture is implemented to check its operation, the results have proven the semantic based policy management as the natural solution to the problems in cloud management.

Automated Testing for SQL Injection Vulnerabilities: An Input Mutation Approach

The popularity of SOA applications can be attributed to their continuous availability, interoperability, and flexibility. However, this also makes them attractive to malicious users. Web Services are gaining popularity with the e-commerce, social media etc. For the increasing need for the security of applications on internet, the security for web services has also increased. The number of reported web vulnerabilities is growing sharply. The major attack a web service faces is that targeting SQL injection vulnerabilities. Testing for vulnerabilities is as important as the development of the web service. In this paper μ4SQLi, an automated testing approach is proposed that can produce inputs to bypass application firewalls for SQL injection vulnerabilities. SQLi attacks target database-driven systems by injecting SQL code fragments into vulnerable input parameters that are not properly checked and sanitised. These injected code fragments could change the application's behaviour if they flow into SQL statements exposing or changing the system's data. Conventionally, these vulnerabilities are checked manually and the solution is inserted dynamically into the source code called as white box testing which is practically not feasible most of the times. The proposed approach is kind of black box testing. Starting from "legal" initial test cases, our approach applies a set of mutation operators that are specifically designed to increase the likelihood of generating successful SQLi attacks. More specifically, new attack patterns are likely to be generated by applying multiple mutation operators on the same input. Moreover, some of our mutation operators are designed to obfuscate the injected SQL code fragments to bypass security filters, such as web application firewalls (WAF), while others aim to repair SQL syntax errors that might be caused by previous mutations. As a result, our approach can generate test inputs that produce syntactically correct and executable SQL statements that can reveal SQL vulnerabilities, if they exist. By producing SQLi attacks that bypass the firewall and result in executable SQL statements we ensure to find exploitable vulnerabilities as opposed to vulnerabilities that cannot be exploited. This approach is automated and supported by a tool called Xavier. When tested for evaluation, it is more feasible, faster and more likely to detect vulnerabilities than other traditional approaches.