Science of Computer Programming 163 (2018) 160–173

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

A practical approach for detecting multi-tenancy data interference

Andrei Furda *, Colin Fidge *, Alistair Barros

Science and Engineering Faculty, Queensland University of Technology, Brisbane QLD 4001, Australia

ARTICLE INFO

Article history:
Received 22 August 2016
Received in revised form 29 April 2018
Accepted 30 April 2018
Available online 3 May 2018

Keywords:
Multi-tenancy
Microservices
Cloud computing
Multi-tenant data separation

ABSTRACT

This paper presents a practical solution for identifying tenant data interference defects when migrating single-tenant legacy code to multi-tenant components or multi-tenant microservices.

The paper explains the concepts of multi-tenant components and microservices, elaborates a formal definition of the multi-tenancy data interference problem based on information flow control theory, and presents a practical method to identify potential defects by analysing the code statically. The method has been implemented as a prototype developer support tool for PHP code.

The implemented support tool prototype demonstrates the method's effectiveness for supporting the transformation of single-tenant legacy source code into multi-tenant components or microservices. It could also be used to confirm that multi-tenant components or microservices are free of data interference defects. The prototype implementation has been validated in a case study with code from the open-source enterprise application SugarCRM. Results indicate that the developed approach significantly increases the efficiency of multi-tenancy transformation in larger code bases by pointing out potential defects.

1. Introduction

Single-instance (so-called *native*, or *application-level*) multi-tenant software-as-a-service (SaaS) applications offer significant cost benefits over single-tenant SaaS applications [19,15]. Such applications are based on multi-tenant components (Fig. 1), or the more recent architectural style of multi-tenant microservices (Fig. 2). However, the sharing of component or microservice instances by multiple tenants introduces the risk of data security breaches of confidential data belonging to the accessing tenants.

In this paper, we present a practical method for detecting potential data interference defects that could cause tenant data leaks in multi-tenant components or microservices. We first elaborate a formal definition of multi-tenancy data interference and present the method and its prototype implementation as a PHP programming support tool. The tool identifies potentially unsafe code that could expose sensitive tenant data to other tenants that share the same component or microservice instance. The tool can be used to aid software developers undertaking code modernisation towards application-level multi-tenancy, or during the development of new multi-tenant components or microservices. In particular, its purpose is to help ensure appropriate non-interference (i.e., separation) of tenant data at the code level. The tool identifies source code that

* Corresponding authors.
E-mail addresses: andreifurda@gmail.com (A. Furda), c.fidge@qut.edu.au (C. Fidge), alistair.barros@qut.edu.au (A. Barros).

https://doi.org/10.1016/j.scico.2018.04.006
0167-6423/

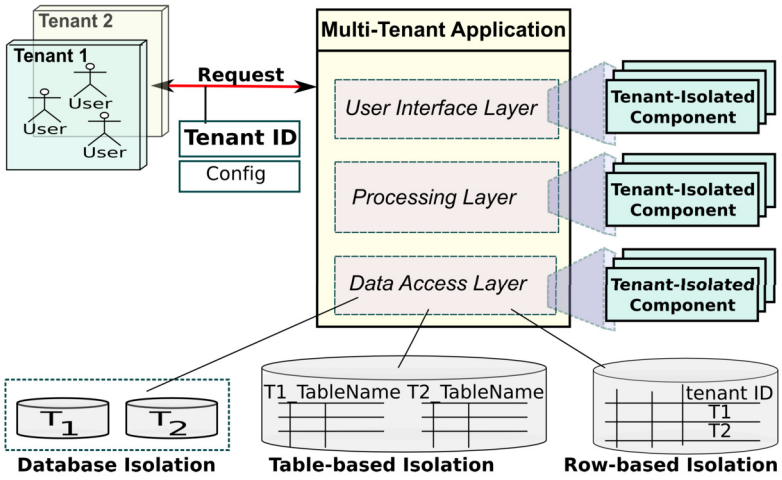

Fig. 1. Multi-tenant component architecture [5,12,9].

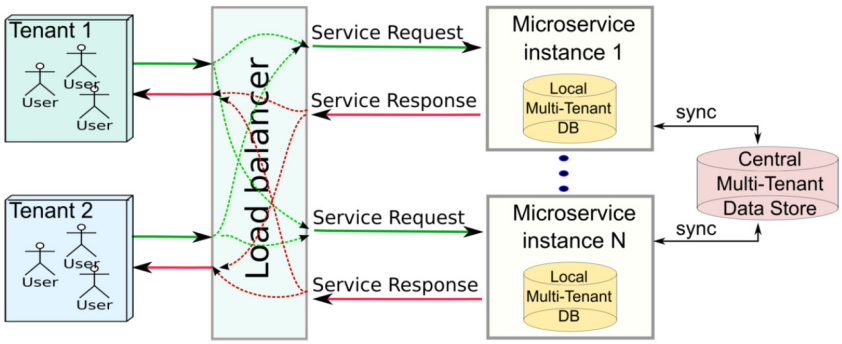

Fig. 2. Multi-tenant microservices architecture [13].

is potentially insecure and needs to be checked or extended by the programmer. It also suggests, by inserting comment annotations into the source code, how to correct the detected defects.

The typically high complexities of static analysis algorithms often make a precise analysis of large code bases impractical for real use, due to unacceptably long execution times [21]. To be practically applicable, most tools aim for a trade-off between precision (i.e., few false positives), performance (i.e., acceptable runtime), and recall (i.e., the ability to find real defects) [2]. Similarly, we have also aimed to maximize the tool's applicability in practice, sacrificing precision for performance.

1.1. Microservices

Microservices are lightweight services that typically implement a single business capability. Microservices are independently deployable and quickly scalable due to a container-based deployment model [11]. They are a promising solution for gradually transforming monolithic legacy applications to benefit from Cloud computing [12].

The desired characteristics for microservices are statelessness, data consistency, and the capability to be consumed by different organisations, i.e., multi-tenancy [13]. Here, we focus on multi-tenancy, namely on determining whether legacy source code that is chosen to be transformed into a multi-tenant microservice contains potential data interference defects that can cause data leaks from one tenant to another.

1.2. Multi-tenancy

Multi-tenant applications and multi-tenant microservices address the combined requirements of multiple groups of users, departments, or companies (i.e., tenants). By allowing tenants to share the same instances, the number of required instances can be reduced, along with the costs for development, maintenance and deployment of such microservices [19,15,20].

A *tenant* is an organizational entity, usually a group of users, who rent access to a multi-tenant application, such as a payroll, or customer relationship management (CRM) application. In this scenario, multiple, possibly commercially competing tenants, interact simultaneously with the application to retrieve, modify, and enter confidential data.