UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TYPE-BASED DECLASSIFICATION EN DART: IMPLEMENTACIÓN Y ELABORACIÓN DE HERRAMIENTAS DE INFERENCIA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

MATÍAS IGNACIO MENESES CORTÉS

PROFESOR GUÍA:
ÉRIC TANTER

MIEMBROS DE LA COMISIÓN:

SANTIAGO DE CHILE
ABRIL 2018

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN
POR: MATÍAS IGNACIO MENESES CORTÉS
FECHA: ABRIL 2018
PROF. GUÍA: ÉRIC TANTER

TYPE-BASED DECLASSIFICATION EN DART: IMPLEMENTACIÓN Y ELABORACIÓN DE HERRAMIENTAS DE INFERENCIA

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Una dedicatoria corta. Por ejemplo, A los creadores de U-Campus

Agradecimientos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Introducción

La protección de la confidencialidad de la información manipulada por los programas computacionales es un problema cuya relevancia se ha incrementado en el último tiempo, a pesar de tener varias décadas de investigación. Por ejemplo, una aplicación web (o móvil) que como parte de su funcionamiento debe interactuar con servicios de terceros y por tanto debe proteger que su información sensible no se escape durante la ejecución de la aplicación a canales públicos.

Muchas de las técnicas de seguridad convencionales como *control de acceso* tienen deficiencias para proteger la confidencialidad de un programa, por ejemplo no restringen la propagación de información[2].

Formas más expresivas y efectivas de proteger la confidencialidad se basan en un análisis estático sobre el código del programa, y se categorizan dentro de *language-based security*. Una de las técnicas más efectivas se denomina *tipado de seguridad* en un *lenguaje de seguridad*, donde los tipos son anotados con niveles de seguridad para clasificar la información manipulada por el programa.

Uno de los mayores desafíos de los lenguajes de seguridad es facilitar el trabajo del programador, utilizando técnicas más expresivas. En esta dirección, Cruz et al. [1] recientemente propusieron *type-based declassification*, una variación de tipado de seguridad que utiliza el sistema de tipos del lenguaje para controlar la declasificación de la información.

Type-based declassification presenta limitaciones en cuanto a su implementación, debido a que el análisis teórico se realiza sobre un lenguaje minimalista que no incluye características básicas de los lenguajes de programación, como instrucciones condicionales y mutabilidad. En este trabajo, se propone una extensión al lenguaje minimalista utilizado en *type-based declassification*, y la implementación en el lenguaje de programación Dart, desarrollando un plugin para los editores de texto más populares, con el objetivo de proporcionar una experiencia interactiva e intuitiva al usuario.

1.1. Motivación

El tema escogido se considera interesante debido a su importancia teórica y práctica en el campo de la seguridad en lenguajes de programación. Los fundamentos teorícos de *type-*

based declassification están bien descritos [1], pero no así su realización práctica, siendo esto reconocido por sus autores.

Se considera que la realización de este trabajo es relevante para demostrar y materializar en un lenguaje de uso general la investigación de type-based declassification. Dicha materalización constituye un desafío importante en términos de complejidad, al tener que entender la teoría subyacente de tipado de seguridad, así como extender un lenguaje real y sus herramientas de análisis estático con nuevas funcionalidades.

1.2. Objetivos

El objetivo de la memoria es realizar la implementación de un sistema de inferencia para *type-based declassification*. Dentro de los objetivos específicos del trabajo, podemos encontrar:

- **Inferencia y verificación estática de type-based declassification**. Se entiende coma la implementación de un sistema de inferencia de facetas de declasificación para *type-based declassification*, en el lenguaje de programación Dart. Dentro del análisis de la inferencia se incluye la verificación de las reglas del sistema de tipos de *type-based declassification*.
- **Plugin para editores**. Mostrar al programador el resultado de la inferencia, por medio de un plugin para los editores de texto que soporten servidores de análisis estático de Dart, ofreciéndole acciones al respecto.

1.3. Resultados y Organización del documento

En términos concretos este trabajo presenta el diseño de un plugin que realiza el análisis de type-based declassification. Los antecedentes teóricos necesarios para entender este trabajo se abordan en el capítulo 2, mientras que la propuesta de solución es desarrollada en el capítulo 3.

Los detalles de diseño de implementación de la propuesta son revisados en el capítulo 4.

Por terminar.

Capítulo 2

Antecedentes

En este capítulo se discuten los antecedentes y el marco teórico necesario para poder entender este trabajo. Además, se muestran las definiciones formales utilizadas en el resto del documento.

2.1. Protección de confidencialidad en computación

Se nombran y caracterizan brevemente distintas técnicas: criptografía, control de acceso, language-based security (LBS). Por qué LBS es más expresivo y/o efectivo.

2.1.1. Tipado de seguridad y control de flujo

Se explica en que consiste, se muestra la lattice de dos niveles de seguridad y se introduce el concepto de information flow control.

Flujo explicito

Se muestra un ejemplo de flujo explicito.

Flujo implicito

Se muestra un ejemplo de flujo implicito.

Trabajos relacionados

Se muestran distintas herramientas de tipado de seguridad, como Jif.

2.1.2. No-interferencia

Se formaliza la propiedad de confidencialidad con noninterference. Se muestra un ejemplo, o se hace referencia a ejemplos anteriores.

2.1.3. Declasificación

Se explica por qué es necesario tener un mecanismo de declasificación, y cómo se puede controlar. Aquí se da el pase-gol a TYPE-BASED DECLASSIFICATION.

2.2. Type-based declassification

Breve introducción.

2.2.1. Tipos de dos facetas

2.2.2. Sistema de tipos

2.2.3. Propiedades

Safe, relaxed noninterference

2.3. Inferencia de tipos

2.3.1. Objetivo y usos

2.3.2. Constraints

2.3.3. Unificación

2.3.4. Decidibilidad

2.3.5. Inferencia de tipos de seguridad

Mencionar por qué es posible y trabajos relacionados al respecto. Explicar y ejemplificar el uso de PC.

2.4. Lenguaje Dart

Qué es Dart, para qué se usa, cuáles son sus características principales y por qué fue escogido.

2.4.1. Dart Analyzer

2.4.2. Analyzer Plugin

2.4.3. Otras herramientas

Se menciona librería de unit test, manejo de dependencias y versión de Dart.

Capítulo 3

Propuesta

En este trabajo se propone realizar la implementación en Dart de un sistema de inferencia de facetas de declasificación, que incluya el análisis de *Type-based Relaxed Noninterference*, mediante la realización de un plugin para entornos de desarrollo integrado (IDE). En este capítulo se detallan los problemas de inferencia a resolver, las estrategias utilizadas para resolverlos, los cambios al trabajo original y el subconjunto del lenguaje soportado.

3.1. Problema de inferencia

Se discute el problema de inferencia a resolver y ejemplos de lo que se busca del sistema a implementar.

3.2. Consideraciones de diseño

Se discuten las alternativas disponibles respecto a la decisión sobre las facetas de métodos que pertenecen al core del lenguaje, y por qué Bot ->Bot es la escogida. Explicar que de todas formas sería un parámetro configurable de la herramienta.

3.3. Generación de constraints de subtyping

Se muestra en palabras la generación de constraints para las distintas expresiones.

3.3.1. Declaración de métodos

3.3.2. Llamadas a métodos

Retorno

Argumentos

Encadenamiento de llamados

3.3.3. Expresión de retorno

3.3.4. Declaración y asignación a variables

3.3.5. Expresiones condicionales

3.4. Resolución de constraints

3.4.1. Simplificación y eliminación de constraints

3.4.2. Agrupación de constraints

Join

Meet

3.4.3. Unificación y substitución

3.5. Extensión de propuesta teórica

Mostrar los cambios y extensiones necesarias al trabajo de type-based declassification para ajustarse a un lenguaje como Dart, y el subconjunto de Dart soportado.

3.6. Interacción con el usuario

Cuál es la interacción con el usuario deseada.

Capítulo 4

Implementación

En esta sección se detalla la implementación de este trabajo, que se dividió en dos componentes principales. Primero, se implementó un sistema de inferencia para type-based declassification. Segundo, se elaboró un plugin para los editores de texto más populares que integra el resultado de la inferencia.

4.1. Implementación de sistema de inferencia

4.1.1. Diagrama de componentes y descripción general

Se explica el funcionamiento general de la inferencia.

4.1.2. Dart Analyzer

Explicar funcionalidades utilizadas de la librería Dart Analyzer. Información contenida en AST, en especial inferencia de tipos. Limitaciones (si es que tiene alguna relevante).

4.1.3. Representación de tipos

Cómo se representaron los distintos tipos (type variables, arrow types, object types, Top, Bot, OrType).

4.1.4. Representación de constraints

En qué consisten las constraint de subtyping implementadas.

4.1.5. Representación de facetas de declasificación

Uso de anotaciones para indicar las facetas. Abstract classes de Dart en archivo sec.dart para declararlas. Mencionar el parsing de facetas declaradas a object types.

4.1.6. Almacenamiento de información relevante

Uso de diccionarios para llevar registro del tipo de ciertos elementos o expresiones.

4.1.7. Fase de generación de constraints

Uso del patrón visitor para recorrer el AST.

Recolección de errores

4.1.8. Fase de resolución de constraints

Recolección de errores

4.1.9. Testing

Cómo se testea la inferencia.

4.2. Implementación de plugin

4.2.1. Diagrama de componentes y descripción general

Explicar funcionamiento general e integración con sistema de inferencia.

4.2.2. Configuración inicial

Uso de la herramienta y creación del archivo sec.dart en primera ejecución del análisis.

4.2.3. Tipos de errores e información

Capítulo 5

Validación y Discusión

5.1. Batería de tests

Se ponen a prueba las reglas del sistema de tipos y la inferencia.

5.2. Repositorio de prueba

Pequeña aplicación segura que usa faceted types y el sistema de inferencia.

5.3. Usabilidad

Comportamiento, performance del plugin.

Conclusión

(Algo de conclusión)

Proyecciones y trabajo futuro

Formalización de inferencia

Extensión del subconjunto soportado

Sugerencias de edición, navegación y completación de código

Bibliografía

[1] Raimil Cruz, Tamara Rezk, Bernard Serpette, and Éric Tanter. Type abstraction for relaxed noninterference. In Peter Müller, editor, *Proceedings of the 31st European Conference on Object-oriented Programming (ECOOP 2017)*, Barcelona, Spain, June 2017. Dagstuhl LIPIcs. To appear.

[2] Andrew C. Myers. Mostly-static decentralized information flow control. Technical Report MIT/LCS/TR-783, Massachusetts Institute of Technology, January 1999.