

# Introduction

Within information security, entire classes of application vulnerabilities arise due to problematic user input handling (Christey and Martin, 2007). This covers cross-site scripting (XSS), injection (SQL, LDAP, etc), insecure deserialisation and file inclusion vulnerabilities – all of which are regularly discovered in major software products.

There are many existing products that aim to detect security issues within an application. Sadowski et al. describe the benefits of SAST (Static Application Security Testing) tooling deployed within Google which allow checks to be performed as part of the compilation process (2018). The authors explain one of the main challenges with developer adoption as *trustworthiness* “users do not trust to results due to, say, false positives” and highlight the importance of reporting issues early: “survey participants deemed 74% of the issues flagged at compile time as real problems, compared to 21% of those found in checked-in code”.

The main objective of this project is to explore the use of a type system which uses regular-expression based refinement types applied to user input. By considering data flow of this user input, it will be possible to make inferences about potential vulnerabilities present in a codebase. Use of refinement types provides for a more informed evaluation of a particular risk than base types alone and should therefore enable reporting of fewer false positive issues.

## Objectives

**i** Primary objectives are expected to be completed during the lifetime of the project. Additional objectives are potential areas for going beyond the original scope of the project, if time permits.

### Primary Objectives

- Formalise a type system that supports types predicated with a regular expression pattern that elements of the refined type will satisfy (be matched by).
- Explore the consequences of typical string operations (e.g. concatenation) and define the type of their return value when applied to elements of the regular expression type.
- Implement such a type system built against a simplified proof-of-concept language that can ensure type safety. Test the implementation against a variety of test cases.

### Additional Objectives

- Apply the theory explored in the primary phase of the project to produce a type analysis tool which works against type annotations applied to a commonly-used language such as C# or Scala.

## Schedule

Table 1 provides a breakdown of the project time into periods for each fortnight, along with the expected work to be completed.

A meeting will be scheduled for each week to discuss progress and any road-blocks that arise with the project supervisor.

| Time Window                                           | Work                                               |
|-------------------------------------------------------|----------------------------------------------------|
| October 1 <sup>st</sup> – October 14 <sup>th</sup>    | Specification, exploration of prior related works. |
| October 15 <sup>th</sup> – October 28 <sup>th</sup>   | TODO                                               |
| October 29 <sup>th</sup> – November 11 <sup>th</sup>  | TODO                                               |
| November 12 <sup>th</sup> – November 25 <sup>th</sup> | Completion of progress report                      |
| November 26 <sup>th</sup> – December 9 <sup>th</sup>  | TODO                                               |
| January 7 <sup>th</sup> – January 20 <sup>th</sup>    | TODO                                               |
| January 21 <sup>st</sup> – February 3 <sup>rd</sup>   | TODO                                               |
| February 4 <sup>th</sup> – February 17 <sup>th</sup>  | Report work, project presentation preparation      |
| February 18 <sup>th</sup> – March 3 <sup>rd</sup>     | Project presentation preparation, report work      |
| March 4 <sup>th</sup> – March 17 <sup>th</sup>        | Project presentation delivery                      |

Table 1: Breakdown of projected work by time period.

## Legal, social, ethical and professional issues

As a project with some security motivation, it is possible that legal, social, ethical and professional issues will arise. In particular, evaluation of existing static analysis tooling must be performed with care to ensure that use of any particular external test cases is permitted by the Copyright, Designs and Patents Act 1988 within the UK.

Additionally, in order to comply with the *Computer Misuse Act 1988*, any static analysis of external code should only be conducted with permission. Discovered issues should be disclosed responsibly.

## References

Christey, S. and Martin, R. A. (2007), ‘Vulnerability type distributions in CVE’.

Sadowski, C., Aftandilian, E., Eagle, A., Miller-Cushon, L. and Jaspan, C. (2018), ‘Lessons from building static analysis tools at google’, *Commun. ACM* **61**(4), 58–66.

**URL:** <http://doi.acm.org/10.1145/3188720>