Bachelor's Thesis

Character-Level Taint Tracking for Strings in Graal.js

Student:	Lukas Weidinger
SKZ/Matr.Nr.:	----------
Email:	----------
Adviser:	DI Jacob Kreindl, Bsc.
Start Date:	March 7, 2022

JOHANNES KEPLER
UNIVERSITY LINZ

DI Jacob Kreindl, Bsc.
Institute for System Software

P +43 732 2468 4349
jacob.kreindl@jku.at

Office:
Karin Gusenbauer
P +43 732 2468 4342
karin.gusenbauer@jku.at

Dynamic taint tracking [1] is a popular program analysis technique that tracks sensitive data as it flows through an executing program by marking it as tainted. Taint tracking is commonly used to prevent attackers from exploiting program vulnerabilities by injecting malicious inputs and to prevent sensitive information from leaking to untrusted third parties. One way of implementing taint tracking is to extend a runtime environment for the targeted programming language. For example, when a program receives a string value over the network from an untrusted source, the runtime may store this string value using a "tainted string" data type. This runtime could then be hardened against code injection attacks by not executing code stored in such a tainted string.

Graal.js is a high-performance JavaScript runtime that is part of GraalVM [2]. Its source code is open source [3]. GraalVM also contains a node.js implementation that is based on Graal.js.

The main goal of this project is to extend Graal.js with support for attaching taint information to String values on a per character basis. A builtin object should provide a method to add taint information to a range of characters in a String value as well as methods to retrieve or remove taint information from a range of characters in a tainted String value. The presence of taint information should not affect program semantics. To this end, tainted String values should support the same operations as String values without taint information. If an operation copies characters from one String value to create another, then any taint information these characters had in the original String value should also be applied to the copies in the new String value.

The implementation should be rigorously tested with regards to correctness of taint propagation and non-interference with program semantics. Correctness of taint propagation may be tested using manually implemented tests for each supported operation. Non-interference may be tested using Graal.js' existing test suite. To this end, the runtime needs to be extended with a new mode in which all String values are created as tainted String values. Running Graal.js' existing tests in that mode would then reveal cases in which the presence of taint information leads to a wrong result, i.e., the taint information interferes with correct program execution.

Lastly, the capability of the taint tracking support should be demonstrated at the hand of an example application. Ideally one of the

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at
DVR 0093696

applications described by Chin et al. [1] can be implemented in a real-world application, but the base application may also be manually implemented.

The concrete goals of the thesis are as follows:

- Implement a new data type for tainted String values in Graal.js. In addition to the character sequence, this new data type must also be able to store an arbitrary object as taint information for any part of that sequence.
- Support the new data type in every operation that supports regular String values. This includes, e.g., String concatenation with '+', but also builtin String functions.
- Implement builtin functions to access taint information:
  - Add an arbitrary taint label to a range of characters in a provided String.
  - Get the taint information of a particular character index in a provided String.
  - Remove taint information from a range of characters in a provided String.
- Implement tests for taint propagation.
- Use Graal.js' existing tests to show that taint information does not affect program results or output.
- Show the capability of the implemented taint tracking support using a suitable demo application.

Optional goals for this thesis are as follows:

- Optimize the implementation of tainted String values for performance.
- Implement one of the use-cases discussed by Chin et al. [1] in a real-world application.

Explicit non-goals of this thesis are:

- Full support for taint propagation in node.js.
- Full support for JavaScript's meta-programming features.
- Find and/or fix issues in Graal.js.

The progress of the thesis should be discussed with the adviser on at least a monthly basis. The adviser should further be notified of the project's progress on at least a bi-weekly basis. The final version of the written thesis should be submitted before September 30, 2022.

[1] Erika Chin and David Wagner. 2009. Efficient character-level taint tracking for Java. In Proceedings of the 2009 ACM workshop on Secure web services (SWS '09). Association for Computing Machinery, New York, NY, USA, 3–12.
DOI:https://doi.org/10.1145/1655121.1655125

[2] Thomas Würthinger, Christian Wimmer, Andreas Wöß, Lukas Stadler, Gilles Duboscq, Christian Humer, Gregor Richards, Doug Simon, and Mario Wolczko. 2013. One VM to rule them all. In Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software (Onward! 2013). Association for Computing Machinery, New York, NY, USA, 187–204.
DOI:https://doi.org/10.1145/2509578.2509581

[3] https://github.com/oracle/graaljs