Projects

CCBD 2019—2020

Jens Nicolay

CC(BD) grading

- CCBD: 40% oral examination
  theoretical examination about topics discussed in the course, and discussion of contents and results of implementation projects

- CCBD: 60% implementation projects
  small implementation projects that demonstrate understanding of concepts and techniques discussed during lectures and practical sessions, complemented with a short report

- CC: 1 implementation project (30%)
  choose from list, or propose your own (must be approved!)

- CC oral exam: implementation project presentation (20%)
  topic introduction, problem + your approach, implementation details, obtained results

Overview

- security
  security policies, browser security, application-level security, dynamic enforcement, static verification
- promises, generators, async/await
- CRDTs
  state-based, operation-based, counters, sets
- reactivity
  behaviors, events, streams, asynchronicity, observables
- blockchain
  Merkle hash, PoX, Bitcoin, Ethereum, Hyperledger Fabric

Security

- illustrate OWASP top 10 attack + protection
  https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- implement dynamic access control using Jalangi2
  https://github.com/Samsung/jalangi2
- experience report of running static analysis tools on web application

Promises, generators, async/await

- write one larger scenario (with intermediate state) using (i) callbacks, (ii) promises, (iii) async/await
- implement actor abstraction on top of generators
- discuss/implement cancelable promises

CRDTs

- implement a directed-graph CRDT in JavaScript
  (or another CRDT mentioned in "Portfolio")
- implement different versions of sets and counters
- demonstrate CRDTs in small web app (shopping app?)
  for example, server replica + replica in local storage
- implement CRDTs using streams for communication

Reactivity

- implement a Twitter popularity tracking add in RxJS
- compare promise chaining to observables
- implement Observables using generators
  e.g., https://medium.com/javascript-scene/the-hidden-power-of-es6-generators-observable-async-flow-control-cfa4c7f31435

Blockchain

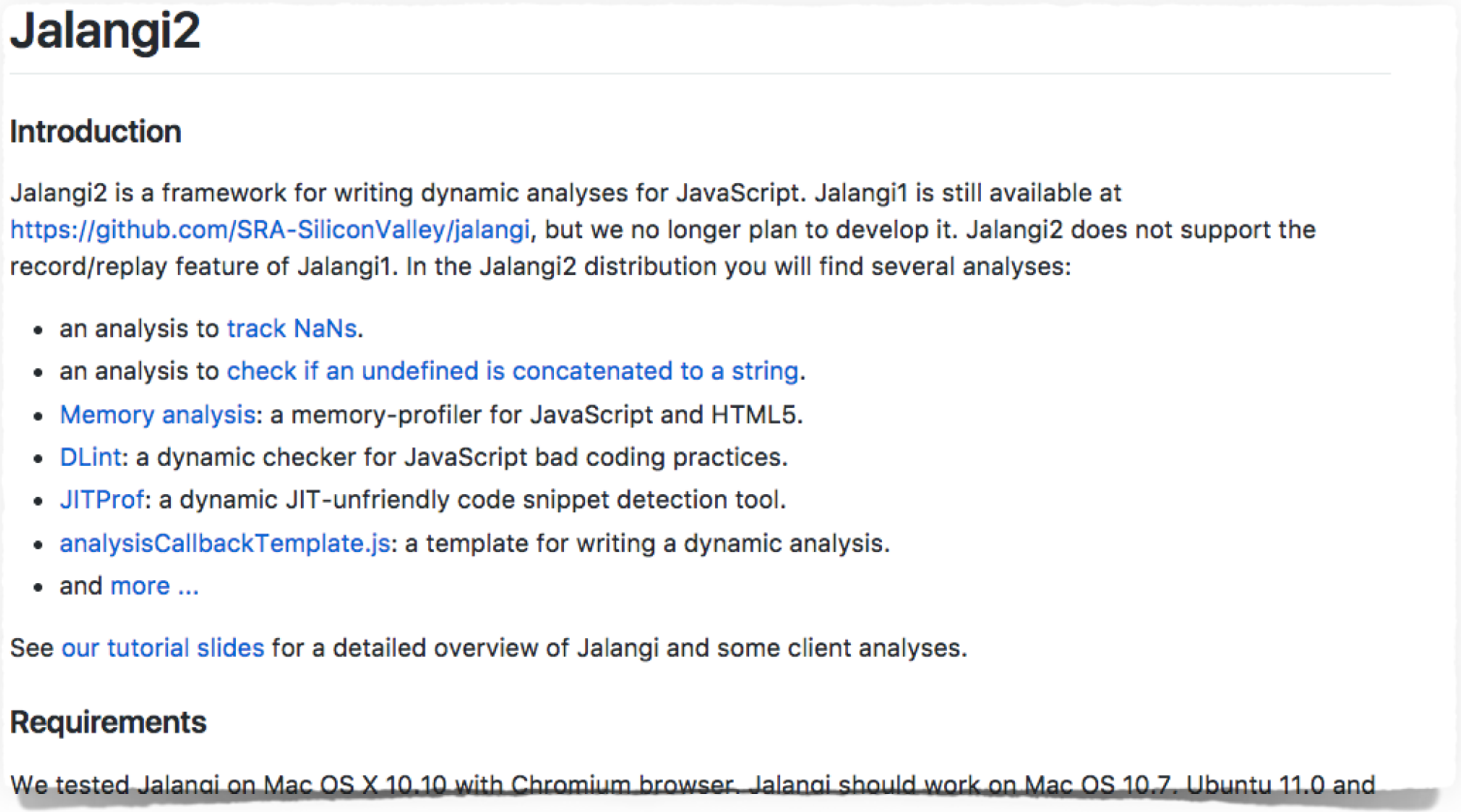
- add PoS or PoSpace to simple blockchain
  https://github.com/lhartikk/naivechain/blob/master/main.js
- develop a diploma registration app on Ethereum/HLF
- implement other interesting Merkle data structures
  (not chain or tree)

Getting started

- pick topic, then...
- start from scratch
- start from existing implementation
  GitHub, tutorials, code from exercise session
- look for research paper(s)

Example: GitHub dynamic access control

https://github.com/Samsung/jalangi2



Jalangi2

Introduction

Jalangi2 is a framework for writing dynamic analyses for JavaScript. Jalangi1 is still available at https://github.com/SRA-SiliconValley/jalangi, but we no longer plan to develop it. Jalangi2 does not support the record/replay feature of Jalangi1. In the Jalangi2 distribution you will find several analyses:

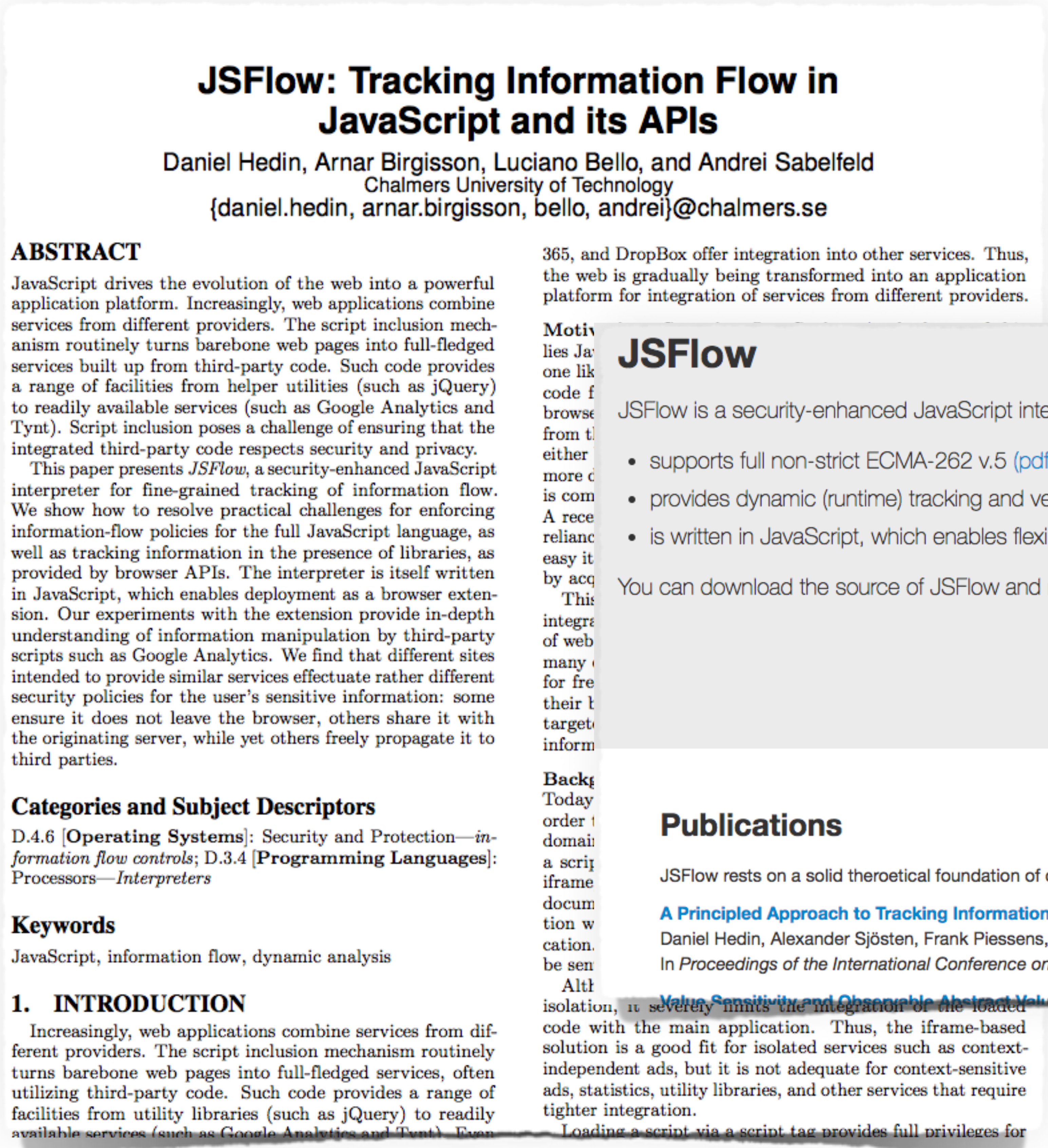
- an analysis to track NaNs.
- an analysis to check if an undefined is concatenated to a string.
- Memory analysis: a memory-profiler for JavaScript and HTML5.
- DLint: a dynamic checker for JavaScript bad coding practices.
- JITProf: a dynamic JIT-unfriendly code snippet detection tool.
- analysisCallbackTemplate.js: a template for writing a dynamic analysis.
- and more ...

See our tutorial slides for a detailed overview of Jalangi and some client analyses.

Requirements

We tested Jalangi on Mac OS X 10.10 with Chromium browser. Jalangi should work on Mac OS 10.7, Ubuntu 11.0 and

Example: paper on security



JSFlow: Tracking Information Flow in JavaScript and its APIs

Daniel Hedin, Arnar Birgisson, Luciano Bello, and Andrei Sabelfeld
Chalmers University of Technology
{daniel.hedin, arnar.birgisson, bello, andrei}@chalmers.se

ABSTRACT

JavaScript drives the evolution of the web into a powerful application platform. Increasingly, web applications combine services from different providers. The script inclusion mechanism routinely turns barebone web pages into full-fledged services built up from third-party code. Such code provides a range of facilities from helper utilities (such as jQuery) to readily available services (such as Google Analytics and Tynt). Script inclusion poses a challenge of ensuring that the integrated third-party code respects security and privacy.

This paper presents *JSFlow*, a security-enhanced JavaScript interpreter for fine-grained tracking of information flow. We show how to resolve practical challenges for enforcing information-flow policies for the full JavaScript language, as well as tracking information in the presence of libraries, as provided by browser APIs. The interpreter is itself written in JavaScript, which enables deployment as a browser extension. Our experiments with the extension provide in-depth understanding of information manipulation by third-party scripts such as Google Analytics. We find that different sites intended to provide similar services effectuate rather different security policies for the user's sensitive information: some ensure it does not leave the browser, others share it with the originating server, while yet others freely propagate it to third parties.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*information flow controls*; D.3.4 [**Programming Languages**]: Processors—*Interpreters*

Keywords

JavaScript, information flow, dynamic analysis

1. INTRODUCTION

Increasingly, web applications combine services from different providers. The script inclusion mechanism routinely turns barebone web pages into full-fledged services, often utilizing third-party code. Such code provides a range of facilities from utility libraries (such as jQuery) to readily available services (such as Google Analytics and Tynt). Even

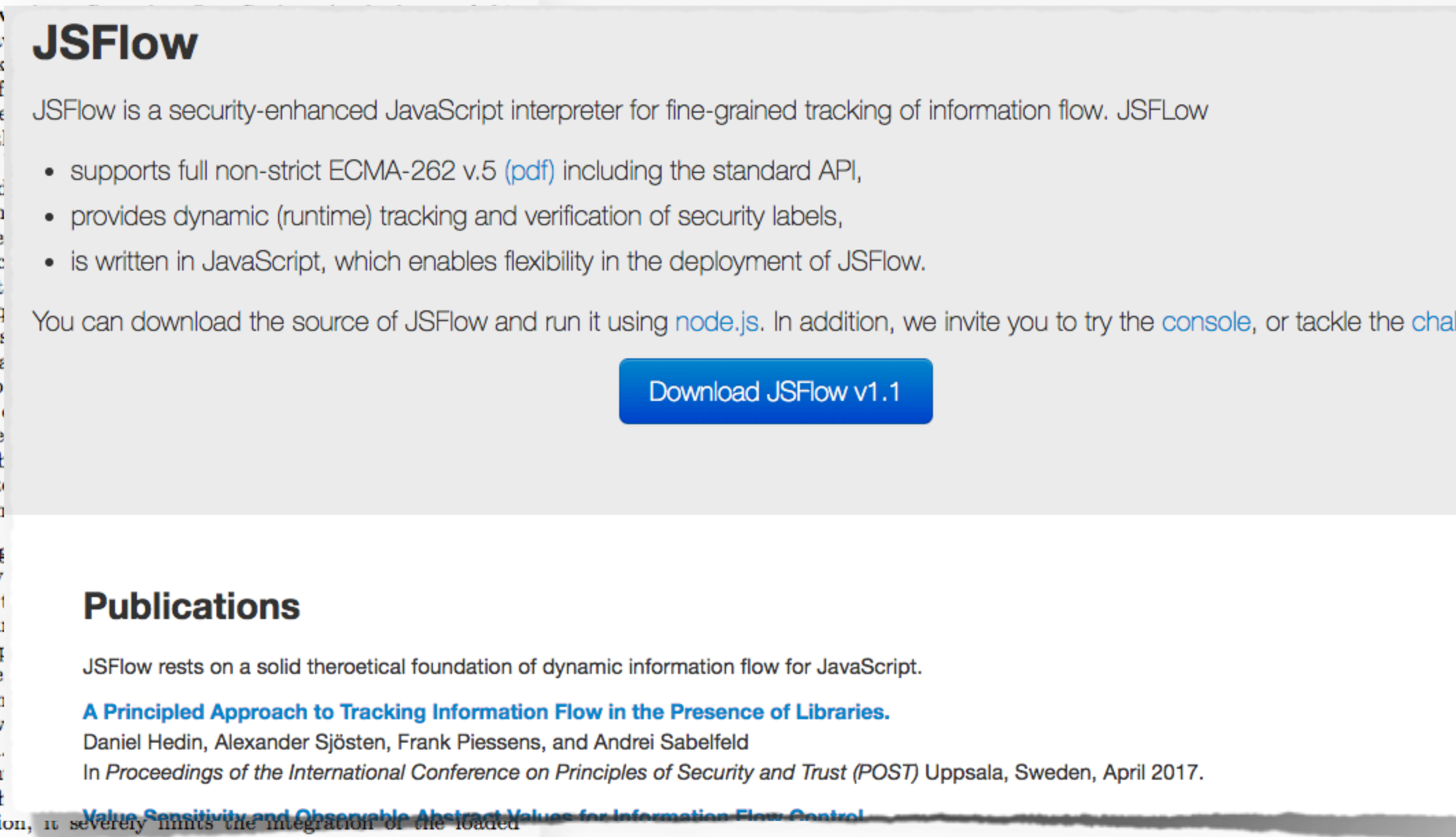
365, and DropBox offer integration into other services. Thus, the web is gradually being transformed into an application platform for integration of services from different providers.

Moti
lies Ja
one lik
code f
browse
from t
either
more d
is com
A rece
relianc
easy it
by acq
This
integra
of web
many
for fre
their b
target
inform

Backg
Today
order t
domai
a scrip
iframe
docum
tion w
cation.
be sen
Alth
isolation, it severely limits the integration of the loaded code with the main application. Thus, the iframe-based solution is a good fit for isolated services such as context-independent ads, but it is not adequate for context-sensitive ads, statistics, utility libraries, and other services that require tighter integration.

Loading a script via a script tag provides full privileges for

http://www.jsflow.net/



JSFlow

JSFlow is a security-enhanced JavaScript interpreter for fine-grained tracking of information flow. JSFLow

- supports full non-strict ECMA-262 v.5 (pdf) including the standard API,
- provides dynamic (runtime) tracking and verification of security labels,
- is written in JavaScript, which enables flexibility in the deployment of JSFlow.

You can download the source of JSFlow and run it using node.js. In addition, we invite you to try the console, or tackle the cha

Download JSFlow v1.1

Publications

JSFlow rests on a solid theroetical foundation of dynamic information flow for JavaScript.

A Principled Approach to Tracking Information Flow in the Presence of Libraries.
Daniel Hedin, Alexander Sjösten, Frank Piessens, and Andrei Sabelfeld
In *Proceedings of the International Conference on Principles of Security and Trust (POST)* Uppsala, Sweden, April 2017.

Requirements

- explicit approval of proposal via email
  short project description

- working code: JS, ClojureScript, Scala.js + related
  make it easy to verify that it works!

- report: 6–10 pages
  - not including full code dump (code illustrations are fine of course)
  - PDF
  - reasonable font size/margins

questions will be 'on-topic'

- oral exam: 10-minute presentation + discussion
  topic introduction (short!), problem, your approach, implementation details, obtained results

Dos

- stay on topic
  no useless features (e.g., nice user interface)

- document as you go

- mention sources

- questions/early feedback: asap

Don'ts

- turn this into a BTh/MTh/PhD
- plagiarize
  mention sources!

all code and reports will be checked!

- deliver a Python project with some JS glue code

Dates

- (initial) topic proposal:
  before the end of **next week**

- implementation + report:
  Sunday January 12 2020 23:59:59.999 CET