Static Analysis, Dynamic Analysis and Symbolic Execution Techniques Applied to Security

Introduction & Case Study

Xinyi Li

May 6, 2020

Overview

Program Analyzer

A program that takes other programs as input and decides whether or not they have a certain **property**[1].

Static Analysis

- Analysis of programs without executing
- Reason for non-trivial properties

Dynamic Analysis

- Analysis of programs by actual executing
- Common testing methods for a desire property

Symbolic Execution

- Analysis of programs by executing with **symbolic** inputs
- Determine what inputs cause each part of a program to execute

[1] Anders Møller and Michael I. Schwartzbach. *Static Program Analysis*. Department of Computer Science, Aarhus University. 2018.

No Free Lunch

Limitations of Program Analysis

> *Program testing can be used to show the presence of bugs, but never to show their absence.*[2]

Rice's theorem[3]

All interesting questions about the behavior (*i.e. non-trivial properties*) of programs (written in Turing-complete programming languages) are **undecidable**.

[2] Edsger W. Dijkstra. "Notes on Structured Programming". circulated privately. Apr. 1970.

[3] Henry Gordon Rice. "Classes of recursively enumerable sets and their decision problems". In: *Transactions of the American Mathematical Society* 74.2 (1953), pp. 358–366.

SAFE-PDF[4]: Detect Malicious Javascript in PDFs

PLAS 2019 / Oracle / University of Sydney

JavaScript programs embedded in PDFs implement some **advanced** features:

1. control embedded multimedia objects
2. interact with the file system or network

However, it may be utilized for **malicious** intentions.

[4] Alexander Jordan, François Gauthier, Behnaz Hassanshahi, et al. "Unacceptable Behavior: Robust PDF Malware Detection Using Abstract Interpretation". In: *Proceedings of the 14th ACM SIGSAC Workshop on Programming Languages and Analysis for Security*. 2019, pp. 19–30.

Code

```
while not q.empty():
    p = q.get()
    p_list = os.listdir(p)
    for i in p_list:
        temp_p = os.path.join(p, i)
        if os.path.isdir(temp_p):
            q.put(temp_p)
            continue
        # do something
        print(temp_p)
```