SBOM.exe: Runtime Integrity for Java

Aman Sharma
amansha@kth.se
https://algomaster99.github.io/

SBOM: Software Bill of Materials

"A Software Bill of Materials (SBOM) is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships."

Source: https://www.ntia.gov/sites/default/files/publications/sbom_at_a_glance_apr2021_0.pdf

`package.json`, `pipfile`, `pom.xml` do not offer a view into transitive dependencies.

```
Metadata (eg. spec information,
producer information)
```

```
Information about the project (eg.
project version, licenses,
checksums)
```

```
"components" : [
   { "group" : "com.sun.activation",
     "name" : "jakarta.activation",
     "version" : "1.2.2",
   } …
 ]
```

Expanded version: https://algomaster99.github.io/talks/microsoft-research-india/slides.pdf (12-15)

What is "Runtime Integrity"?

It means that the data in software is complete, trustworthy and *has not been modified or accidentally altered* by an unauthorised user **at runtime**.

Examples of runtime integrity compromises.

1. Buffer overflows
   - Code Red (discovery 2001) executed instructions in the input.
   - Heartbleed (discovery 2014) affected OpenSSL.
   - ... and many more.
2. Code Injection
   - WannaCry (discovery 2017) encrypted the files on the system.
   - ... and many more.

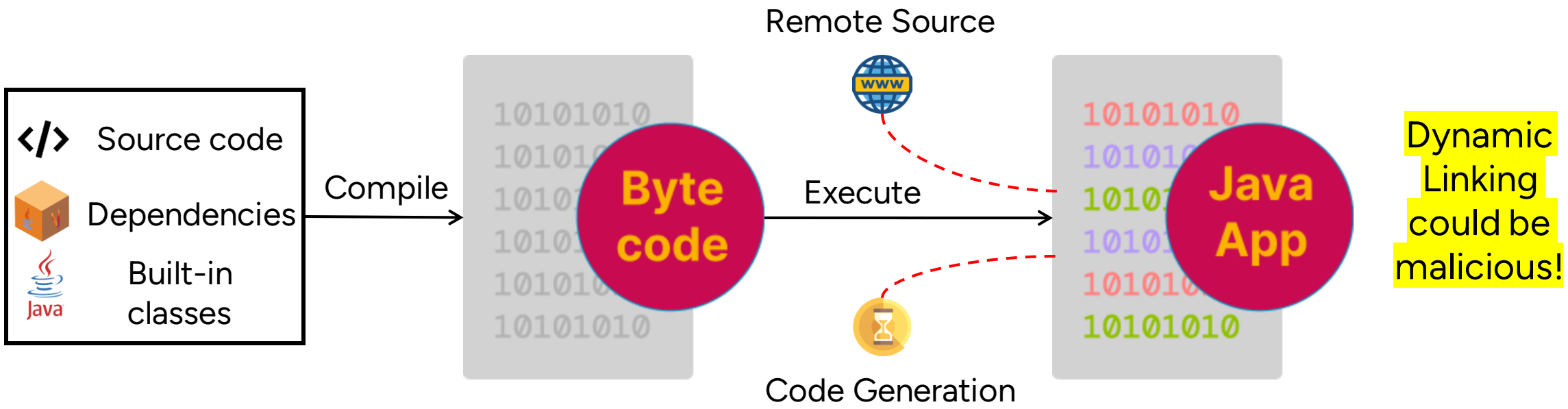
Runtime Integrity in Java

Buffer overflows are difficult in Java due its memory safety properties, but code injection is very much possible in the following ways:

Source: https://en.wikipedia.org/wiki/Memory_safety

- Code can be downloaded at runtime.
- Code can be generated at runtime.

Source: https://docs.oracle.com/javase/specs/jvms/se21/html/jvms-5.html#jvms-5.3



We are the first to use SBOMs for Runtime Integrity

But why?

To protect against attacks that trigger download or generation of unknown code

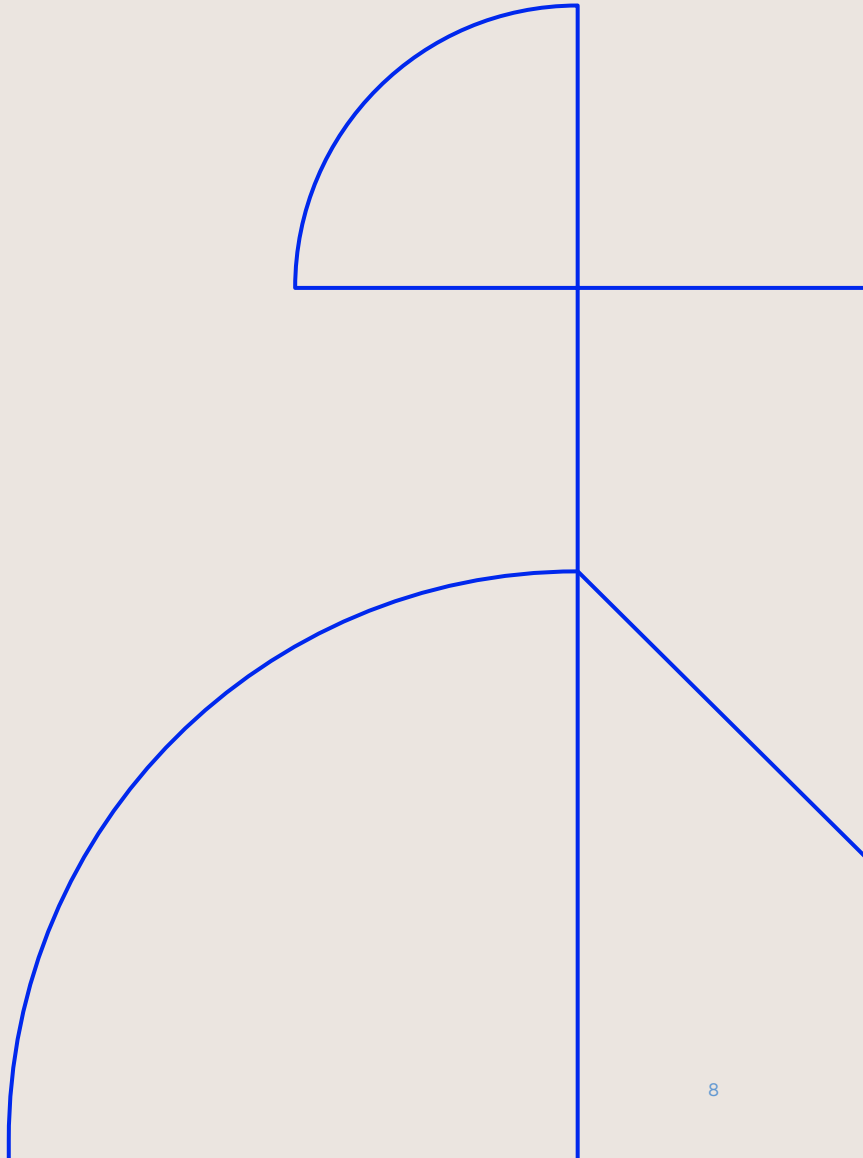
CVE-2021-44228 (Log4shell)

- Malicious code downloaded at runtime.
- Attack on popular logging library Log4J for Java.
- The bug in the library allowed remote code execution.
- Link to attack - https://github.com/cncf/tag-security/blob/main/supply-chain-security/compromises/2021/log4j.md.

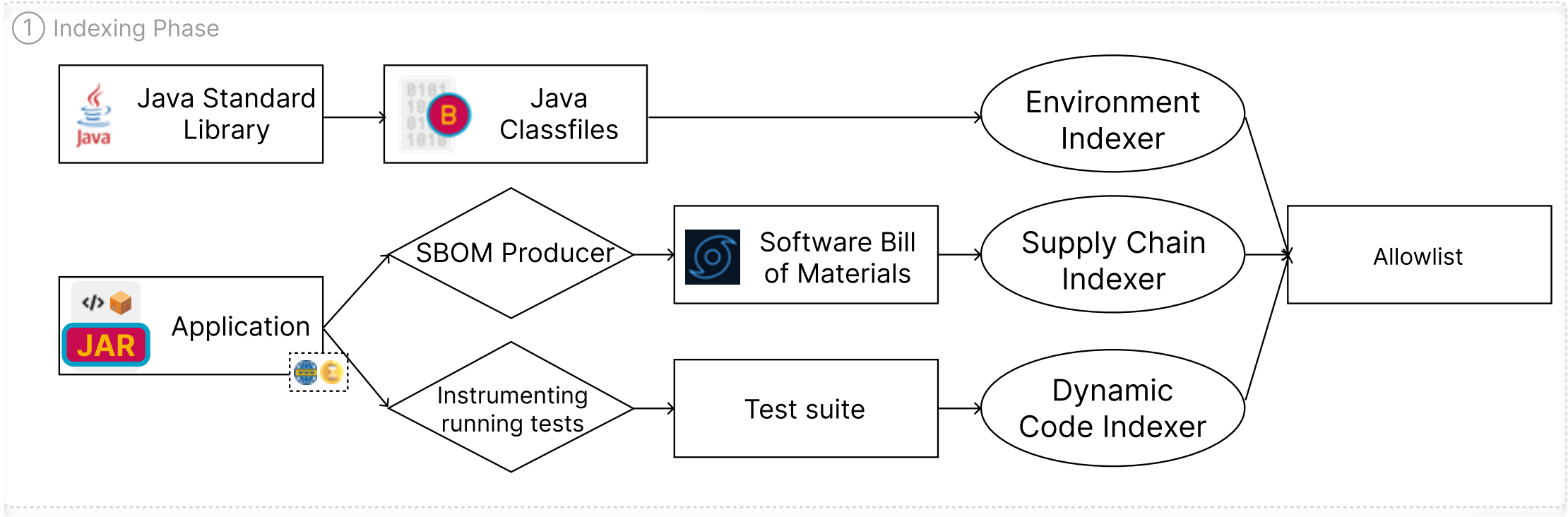
CVE-2022-33980 (Apache Commons Configuration)

- Malicious code generated at runtime.
- It has a feature variable interpolation (`${prefix:name}`) that allows properties to be dynamically evaluated and expanded.
  - For example, it can output java version, date, can encode/decode base64.
  - Complete usage - DefaultLookups (Apache Commons Configuration 2.10.1 API)
- If attacker can inject malicious input in the form `${prefix:name}`, they are able to load the class.
- We also have a demo later!

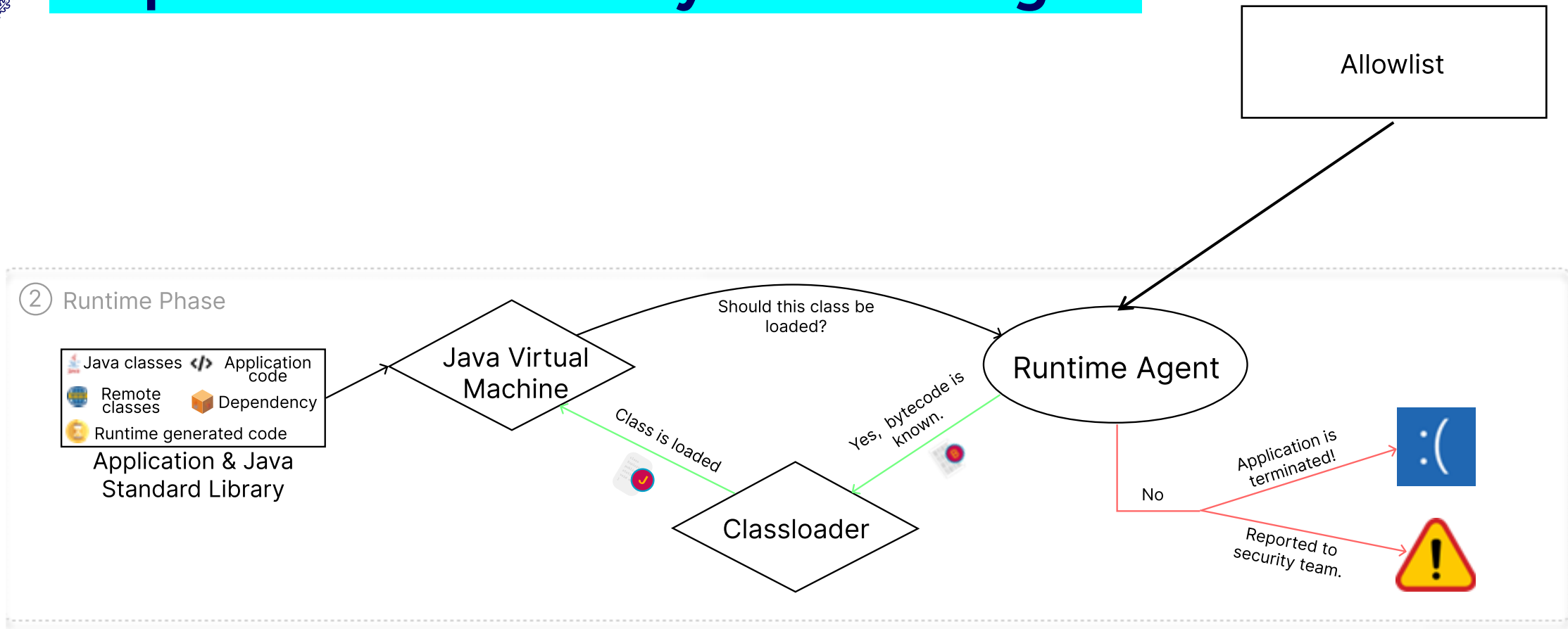
Solution: Create an allowlist of classes and enforce it on Java classloading



Step 1: Indexing



Step 2: Enforcement by Runtime Agent



Demo

CVE-2022-33980 (Apache Commons Configuration)

Source: https://github.com/chains-project/exploits-for-sbom.exe/tree/main/commons-configuration-2022-33980

Related Work

We are not the first to create runtime integrity applications. Here are some example approaches.

1. Permissions Managers: define access permissions for the application at varying granularities.
   - Amusuo et al. *"Preventing Supply Chain Vulnerabilities in Java with a Fine-Grained Permission Manager"*

2. Compartmentalization: different parts of an application are executed in different protection domains
   - Jiang et al. *"Uranus: Simple, Efficient SGX Programming and its Applications"*

3. Integrity Measurement: measuring the application in terms of its control flow, memory, or any kind of execution behaviour and then verifying the measurement.
   - Ba et al. *"RIM4J: An Architecture for Language-Supported Runtime Measurement against Malicious Bytecode in Cloud Computing"*
   - SBOM.exe will also be there soon :)

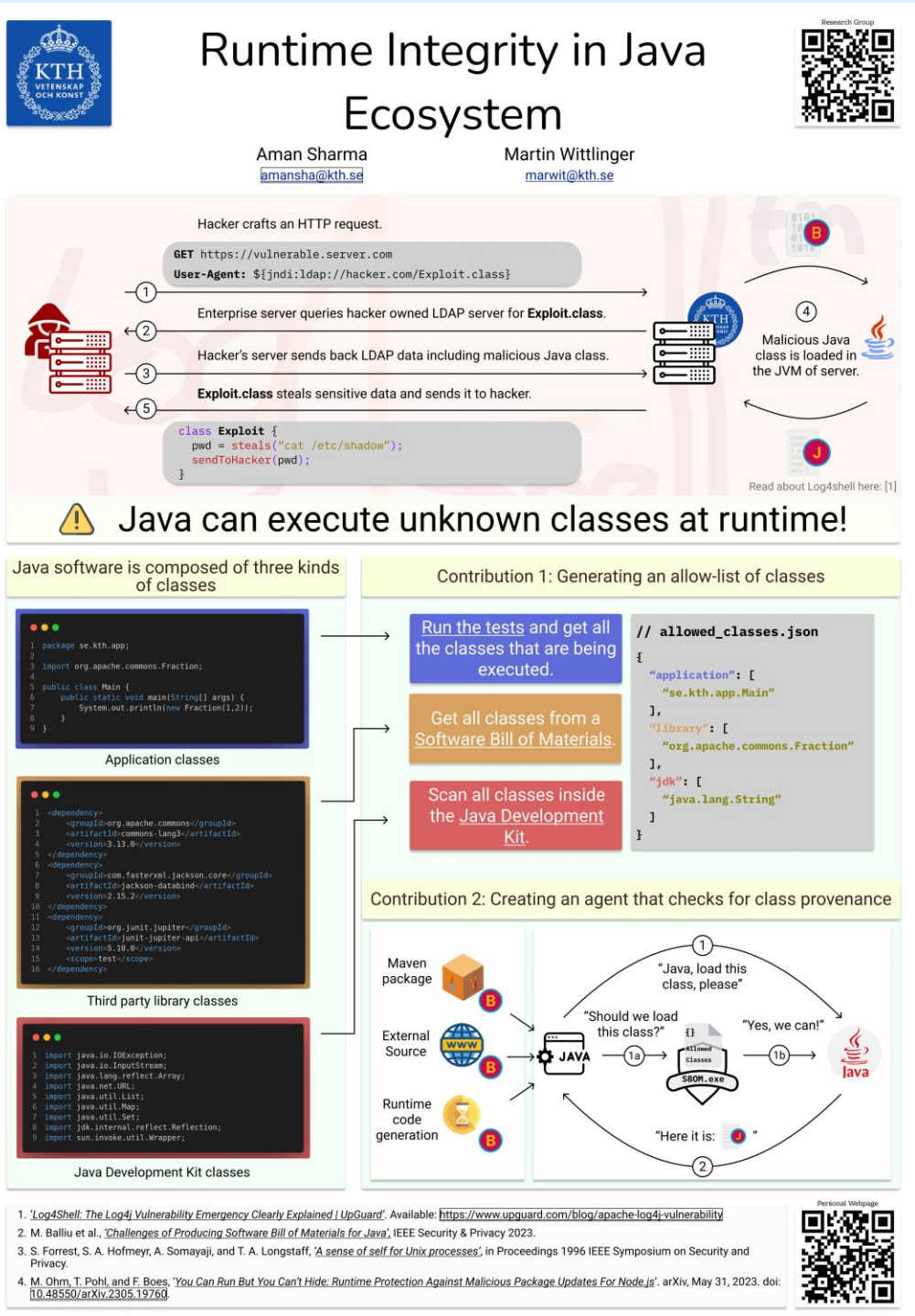
Poster

Presented at IEEE SecDev 2023

Link: https://algomaster99.github.io/posters/runtime-integrity/poster.pdf

26-04-2024

Aman Sharma | amansha@kth.se



Thank you!

Questions?

Personal Webpage

https://algomaster99.gtihub.io

Aman Sharma

amansha@kth.se

Research Group

http://chains.proj.kth.se/