Securing Software Supply Chain at Runtime

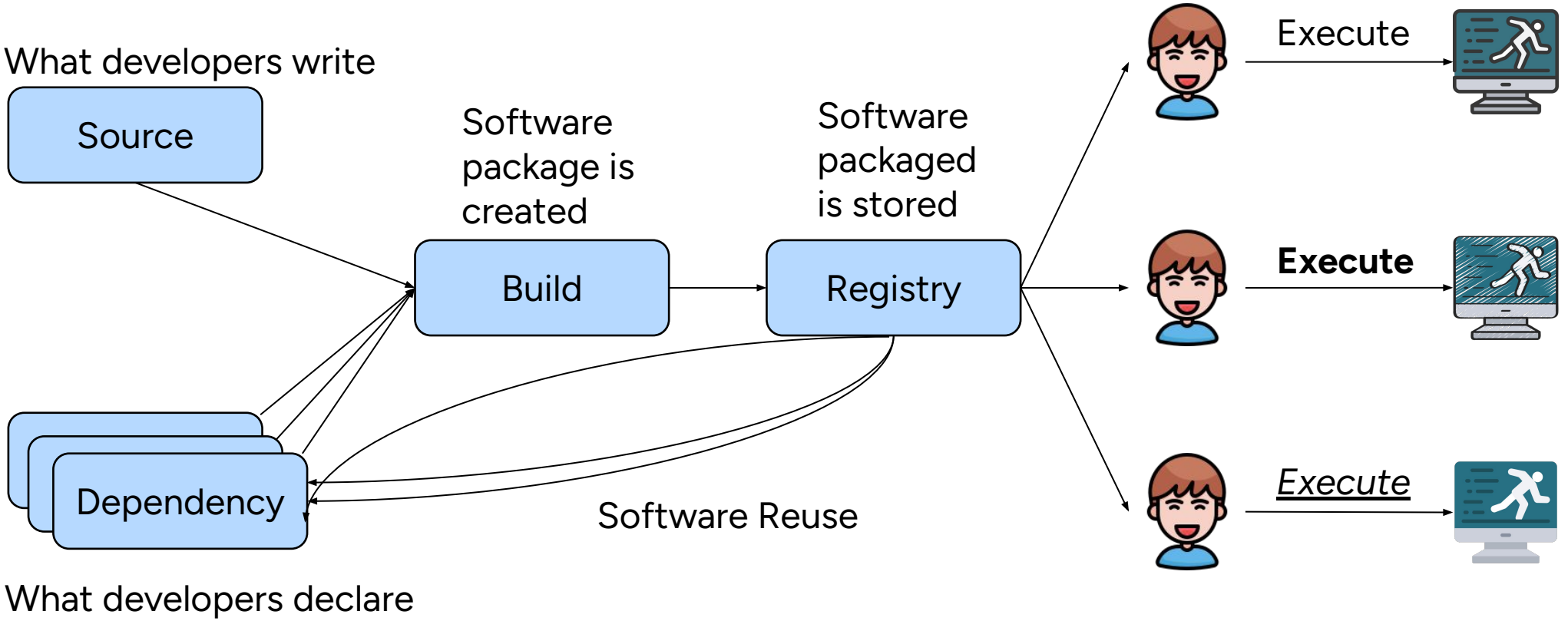
Whitepaper: SBOM.EXE: Countering Dynamic Code Injection based on Software Bill of Materials in Java

Aman Sharma, Martin Wittlinger, Benoit Baudry, Martin Monperrus

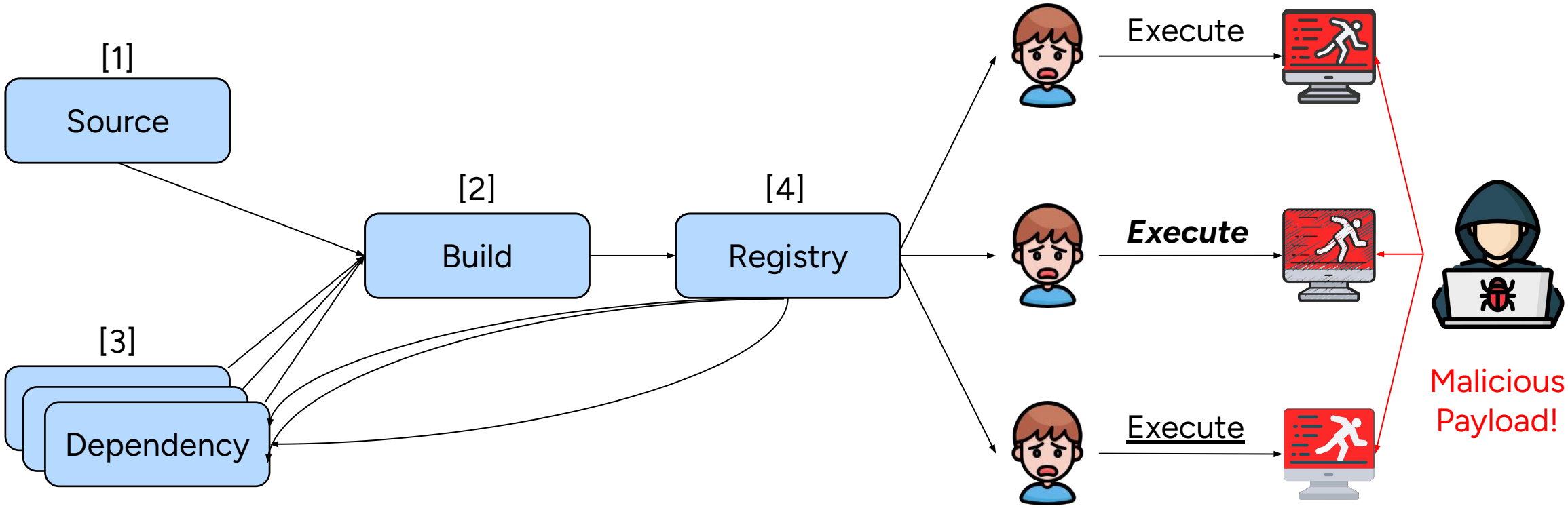
Outline

- Background about Software Supply Chain and relevance to Java
- Demo of Log4shell exploit
- Novel Tool: SBOM.EXE: Countering Dynamic Code Injection based on Software Bill of Materials in Java
- Demo of Log4shell mitigation
- Evaluation
- Conclusion

What is Software Supply Chain?

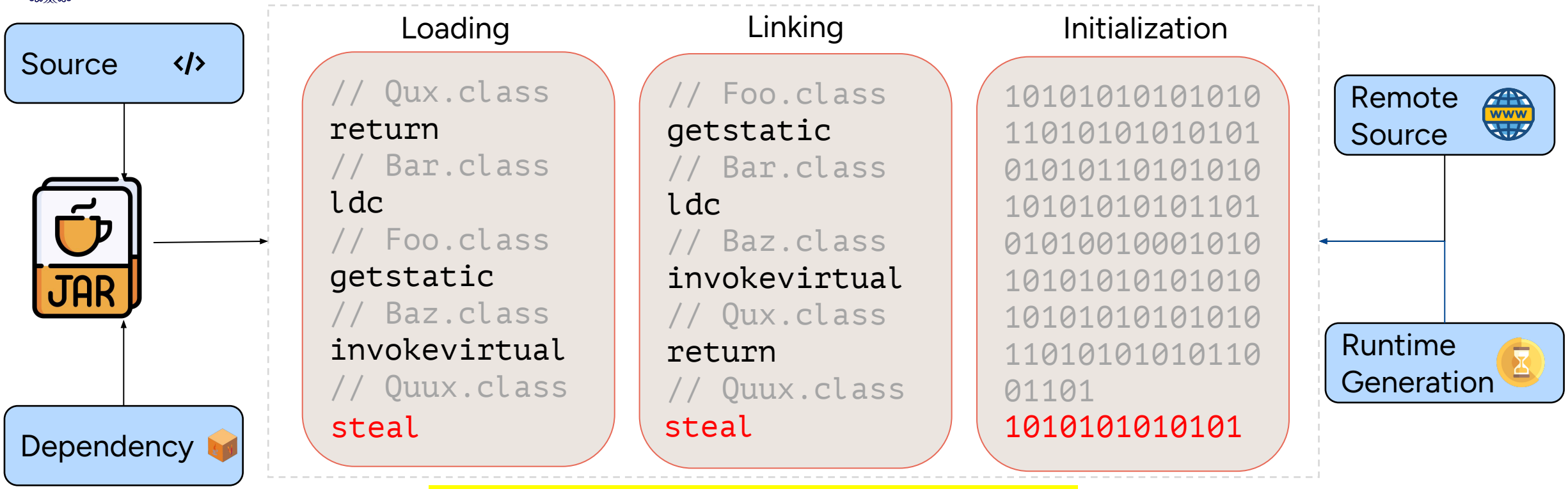


What is Software Supply Chain Attack?



[1] Q. Wu et al. “On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits”, 2021
[2] S. Peisert et al. “Perspectives on the solarwinds incident,” IEEE Security Privacy, 2021
[3] P. Ladisa et al. Towards the Detection of Malicious Java Packages. In Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses, 2022
[4] J. Cappos et al. “A look in the mirror: attacks on package managers,” in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008

How is Software Supply Chain Attack relevant in Java?



Dynamic classloading could be exploited!!!

- Code can be downloaded at runtime.
- Code can be generated at runtime. [5]

[5] Oracle, ClassLoader (Java SE 21 & JDK 21) (oracle.com), 2023, https://docs.oracle.com/en%2Fjava%2Fjavase%2F21%2Fdocs%2Fapi%2F%2F/java.base/java/lang/ClassLoader.html#builtinLoaders

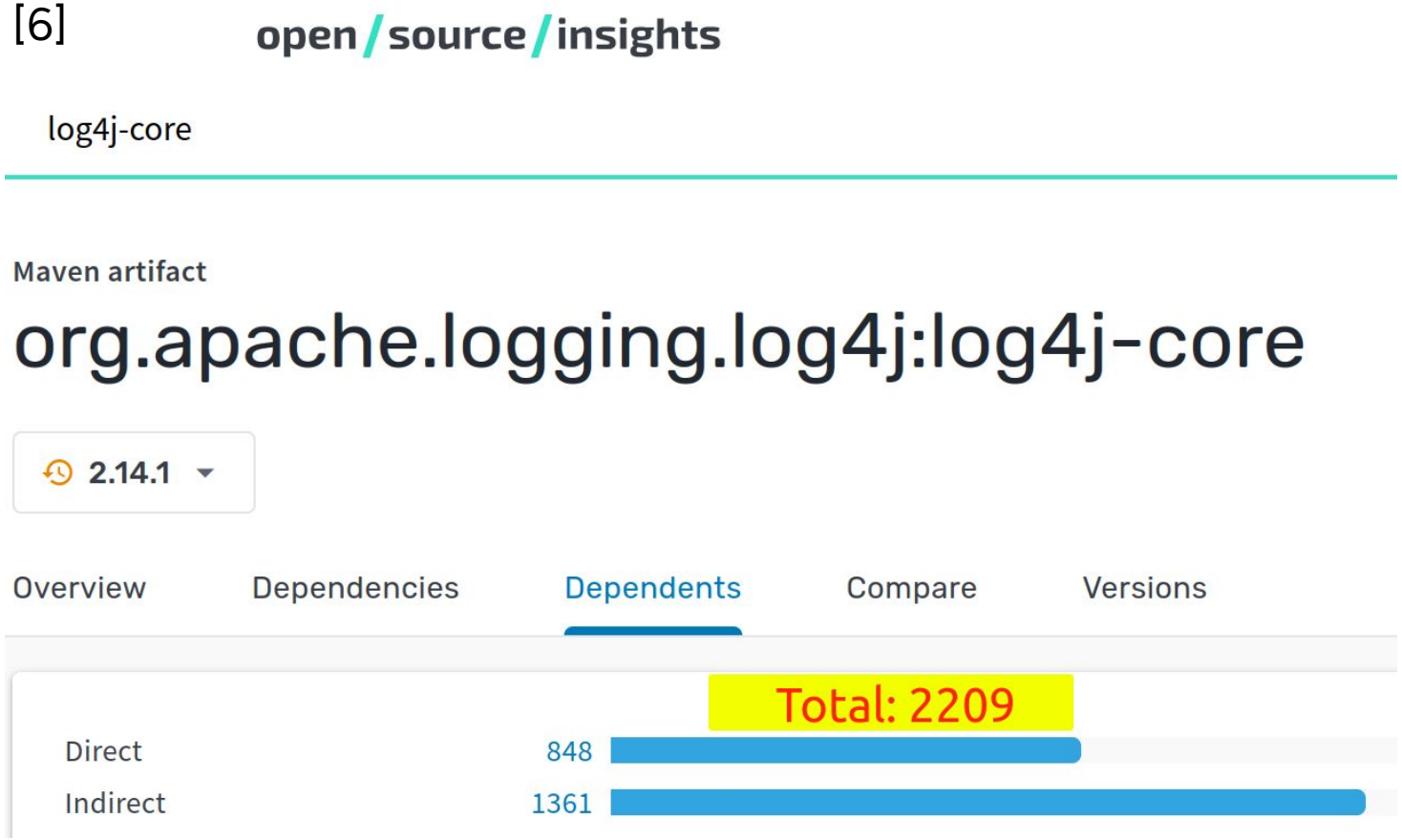
How is Software Supply Chain Attack relevant in Java?

- Code can be g
- Code can be g

[5] Oracle, ClassLoader (Java SE 21 & JDK 21) (oracle.com), 2023, https://docs.oracle.com/en%2Fjava%2Fjavase%2F21%2Fdocs%2Fapi%2F%2F/java.base/java/lang/ClassLoader.html#builtinLoaders

Why is Log4Shell a Software Supply Chain attack?

[6]



[7]

13%

of Log4j downloads are still for known vulnerable versions, nearly 3 years after the vulnerability's discovery.

[6] dev.deps, 'Dependents | org.apache.logging.log4j:log4j-core | Maven | Open Source Insights'. https://deps.dev/maven/org.apache.logging.log4j%3Alog4j-core/2.14.1/dependents
[7] Sonatype, 2024 State of the Software Supply Chain (2024)

Demo: Exploit

CVE-2021-44228 (Log4Shell)

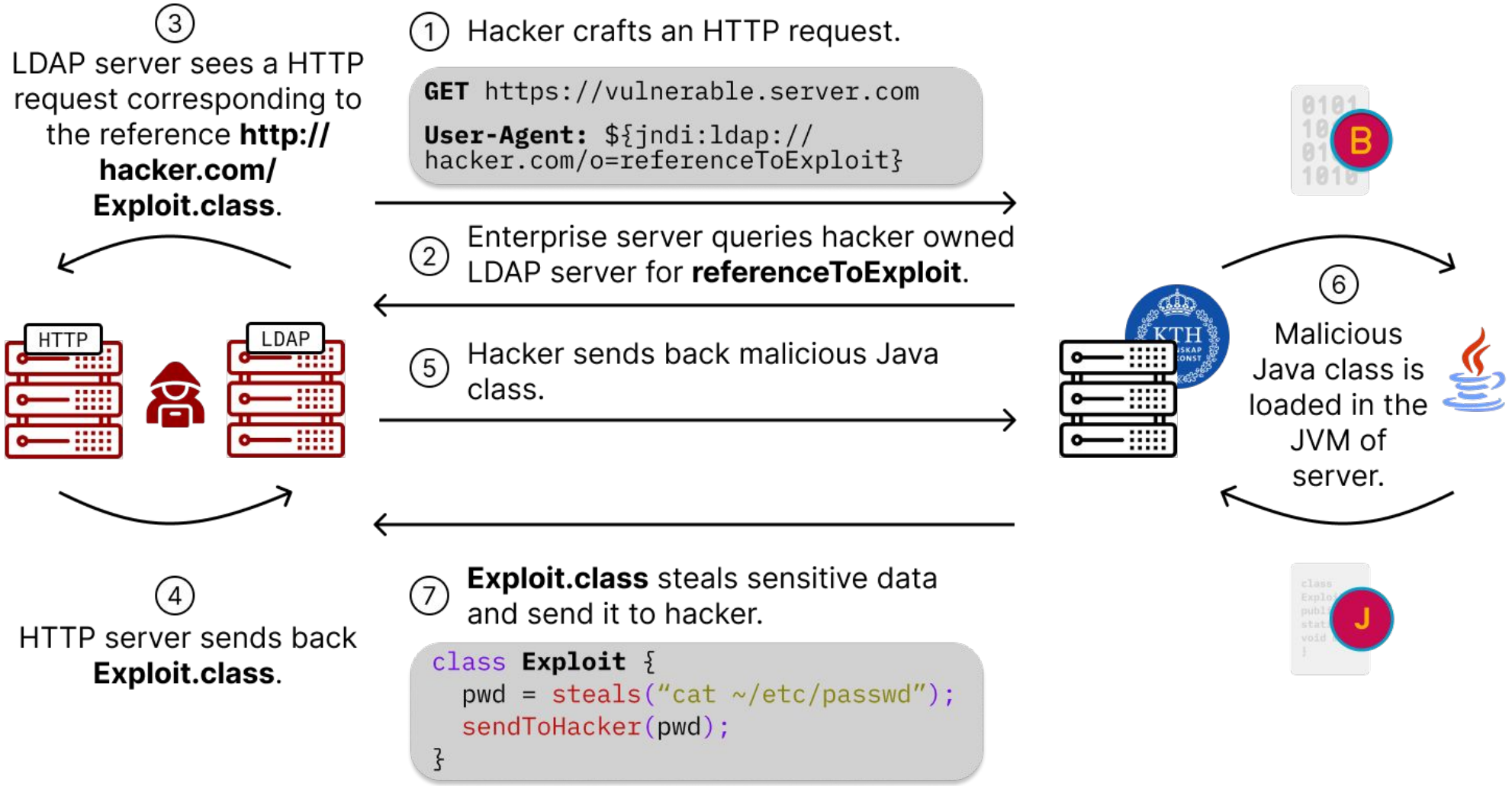
Source: https://github.com/chains-project/exploits-for-sbom.exe/tree/main/rq2/log4shell-2021-44228

Demo Steps (for replication later) for exploit

1. Make sure Java 17 (or earlier) is on `PATH`.
2. Inspect code in *rq2/log4shell-2021-44228/src/main/java* and run `./normal-usage.sh`. This should log “this is an error”.
3. Now startup the LDAP server by going to root of the project and run `java -jar target/RogueJndi-1.1.jar --command "gedit /etc/passwd"`.
    1. This will inject the command argument in the bytecode that will be hosted on LDAP server.
4. Next, go back to the same directory where “normal-usage” was run. Run `./malicious-usage.sh`. This will execute the malicious bytecode.

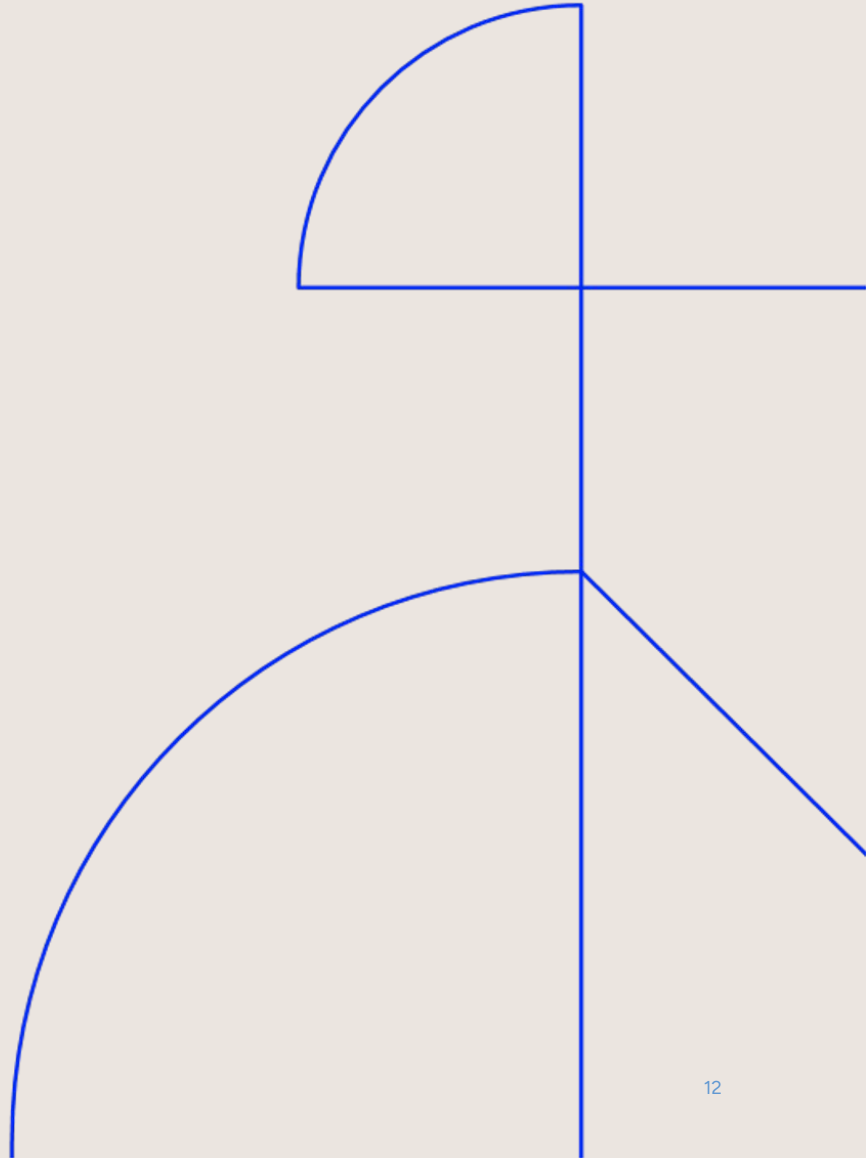
Source: https://github.com/chains-project/exploits-for-sbom.exe/

Log4Shell – a software supply chain attack at runtime



Problem: Java can trigger download or generation of unknown code.

Solution: Create an allowlist of Java classes and only load those classes



Step 1: Indexing

Problem: how to index built-in classes?

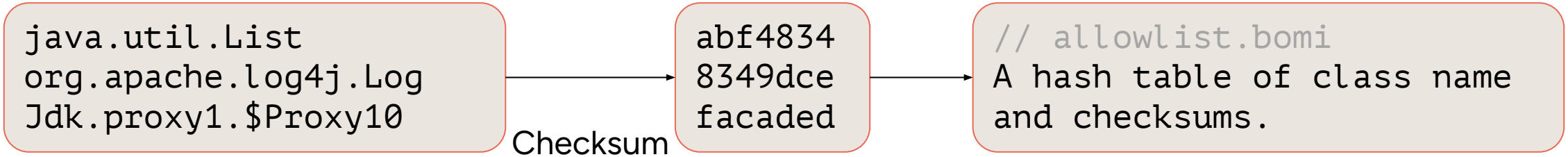
Solution: let's scan all classes using classgraph [20].

Problem: what about source code and dependencies?

Solution: finally, Software Bill of Materials, has one (now implemented) use case.

Problem: and code from remote source and runtime generated code?

Solution: if we execute the code, we can capture them. Let's just run tests.



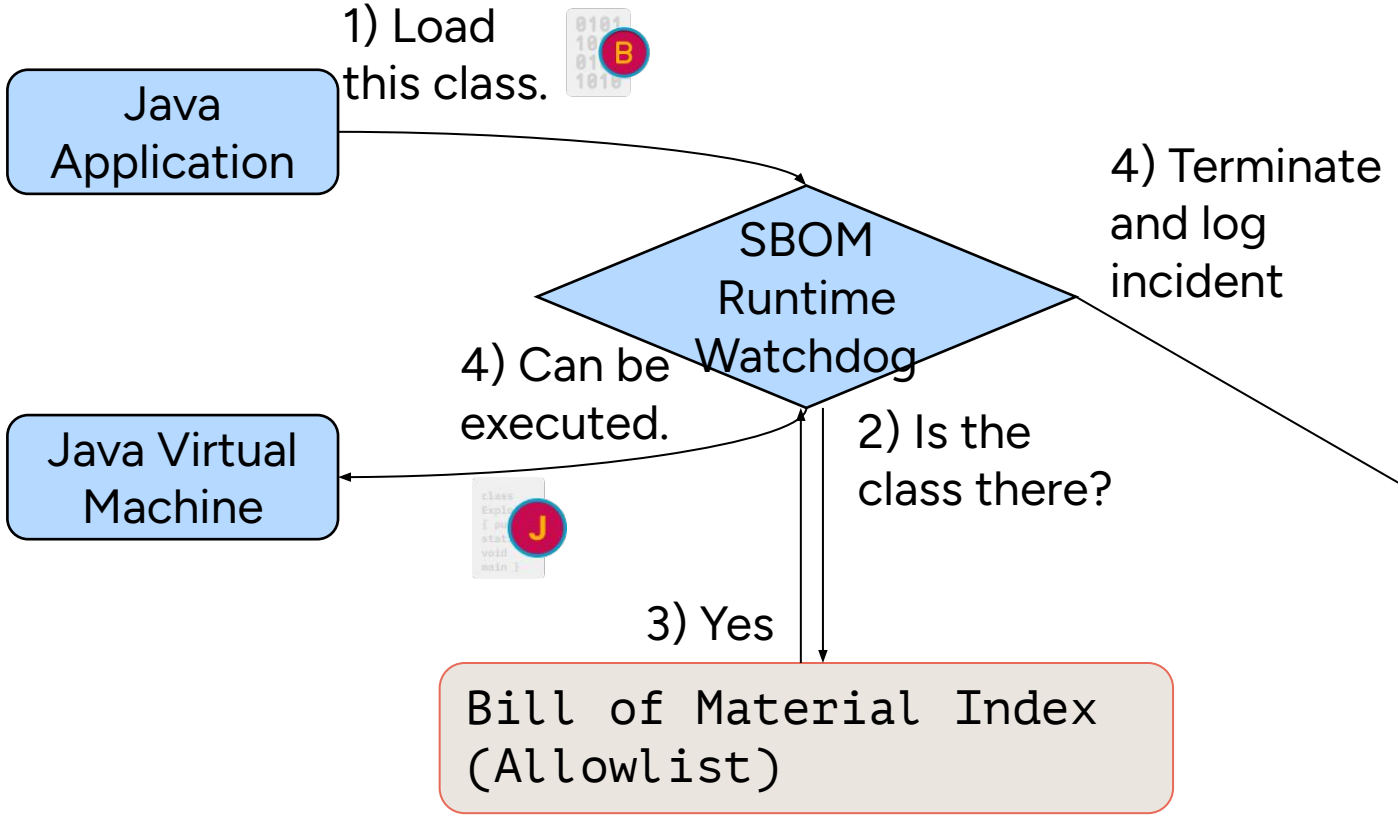
[20] L. Hutchison, Classgraph, GitHub.com, 2024

Part 1 done ✅

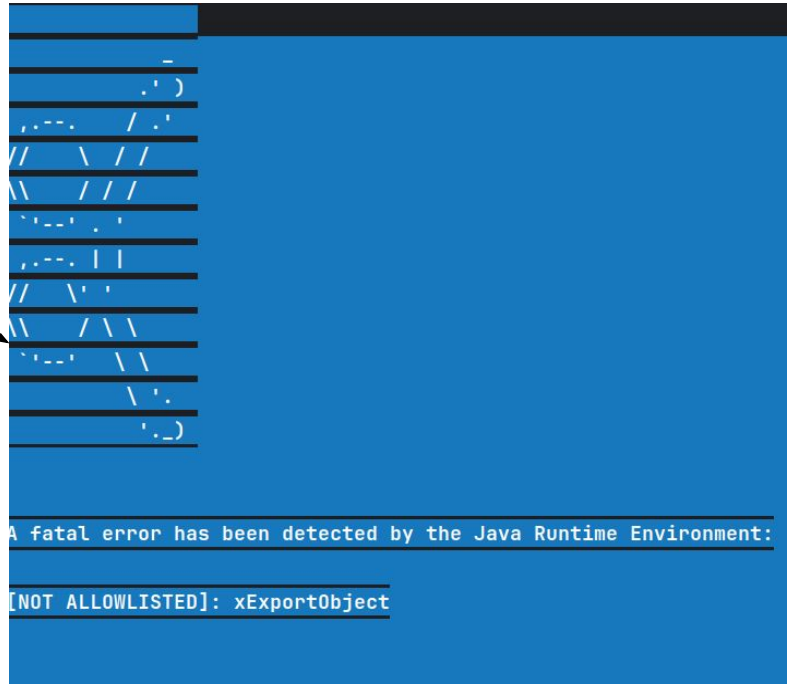
Step 2: Enforcement

Problem: Java class is simply loaded without any integrity.

Solution: We intercept loading and then verify it.



Part 2 done ✅



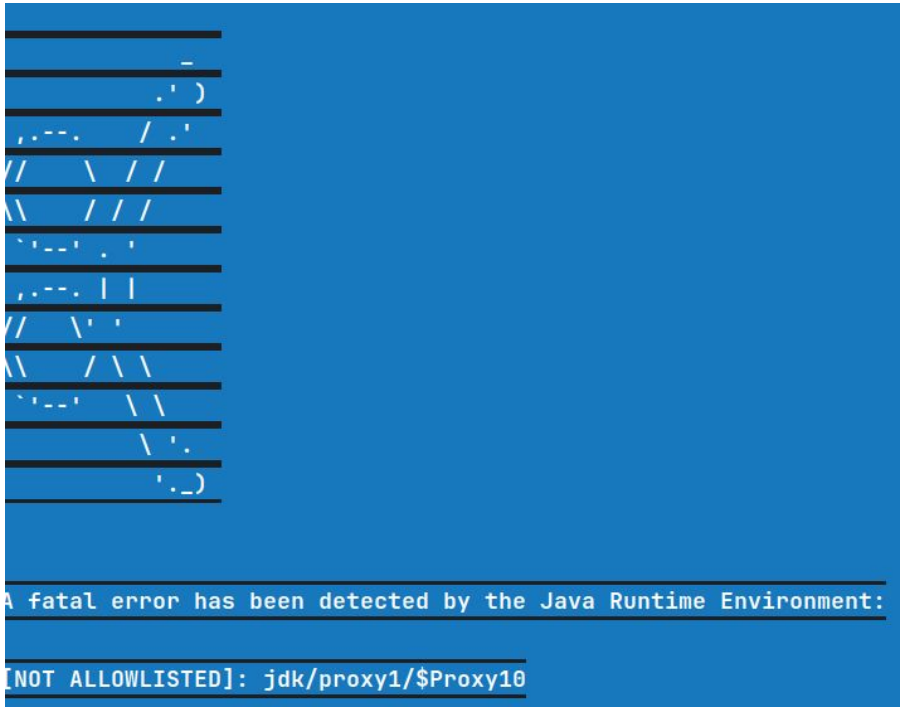
Test run

Okay, we seem to be done. Let's see what happened initially.

Problem 1: There seems to be false-positives. This class was in the allowlist.

Problem 2: There seems to be non-determinism in runtime generated code.

Solution: Let's ignore this non-deterministic features.



Bytecode Canonicalization

- Classnames could change across different executions.
- The type references change.
- The order of method is not fixed.

```
- public class $Proxy10 {
+ public class $Proxy7 {
-     private static $Proxy10.x;
+     private static $Proxy7.x;
-    m1 () {}
+    m3 () {}
-    m3 () {}
+    m1 () {}
}
```

Novel Concepts Summarised

Problem: what to index?

Solution: 3 indexers for built-in classes source code, dependencies, and dynamic code.

Problem: how to load class with verification

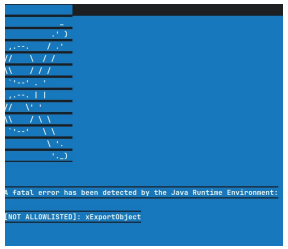
Solution: SBOM Runtime Watchdog is a novel tool to intercept Java classloading and verify integrity of each Java class.

Problem: non-determinism of Java bytecode.

Solution: Bytecode Canonincalization.

Problem: Windows users miss blue screen of death in Linux

Solution:



Demo: Mitigation

CVE-2021-44228 (Log4Shell)

Source: https://github.com/chains-project/exploits-for-sbom.exe/tree/main/rq2/log4shell-2021-44228

Demo Steps (for replication later) for mitigation

1. To run with SBOM.exe protection, we follow two steps:
    1. Run `./generate-index.sh`. This outputs the `index.jsonl` which is the BOMI.
    2. Run `./sbom.exe.sh`. This would terminate the program just before the malicious class is initialized.

Aman Sharma | amansha@kth.se

Evaluation

1. We developed PoC for 3 exploits and mitigated all 3 of them proving that our approach is efficient.
   a. log4shell [8]
   b. authentication with H2 database server [9]
   c. apache commons configuration [10]

2. We integrated our system into 3 real-world applications proving that our approach can mitigate dynamic classloading attacks on them.
   a. PDFBox [11]
   b. ttorrent [12]
   c. GraphHopper [13]

[8] 'NVD - CVE-2021-44228'. https://nvd.nist.gov/vuln/detail/CVE-2021-44228
[9] 'NVD - CVE-2021-42392'. https://nvd.nist.gov/vuln/detail/CVE-2021-42392
[10] 'NVD - CVE-2022-33980'. https://nvd.nist.gov/vuln/detail/CVE-2022-33980
[11] 'Apache PDFBox | Command-Line Tools'. https://pdfbox.apache.org/2.0/commandline.html
[12] M. Petazzoni, mpetazzoni/ttorrent. https://github.com/mpetazzoni/ttorrent
[13] 'GraphHopper Directions API with Route Optimization'. https://www.graphhopper.com/

Takeaways

SBOM.exe can mitigate three high-profile CVEs based on code generation and downloading.

SBOM.exe proposes a strong bytecode canonincalization algorithm which eliminates non-determinism in dynamic classes.

SBOM.exe can work well in production environment as shown by three real world projects.

4th workshop on Software Supply Chain

Full day of discussions about software supply chain on topics:

- code integrity
- reproducible builds
- dependency management
- and many more ... (see agenda)

When: 25th April, 2025

Where: KTH, Stockholm, Sweden

Registration (free of charge):
https://chains.proj.kth.se/software-supply-chain-workshop-4.html

4th CHAINS workshop

https://chains.proj.kth.se/software-supply-chain-workshop-4.html

Thank you!

Rate my talk!

https://www.jfokus.se/rate/2382

Aman Sharma

amansha@kth.se

Project Link: https://github.com/chains-project/sbom.exe

Whitepaper: SBOM.EXE: Countering Dynamic Code Injection based on Software Bill of Materials in Java