String 2.0

MSc Thesis
Introduction

18 August 2020

Christian Zürcher

What have data leaks in common?

They originate from `String` *!*

Current `String` implementations:

- have no security built-in
- are not content-aware and offer no history
- are complex to track
- have static functionality
- functionality is rather basic

What we aim for:

- have no security built-in
- are not content-aware and offer no history
- ~~are complex to track~~ track themselves intrinsically
- have ~~static~~ dynamic functionality
- functionality is ~~rather basic~~ context-dependent

Additional requirements

Must be easy to understand, use, adapt, and extend

Non-intrusive changes to OpenJDK
Maintain Java code compatibility

Never change [behavior of] a running system!

Our approach

Changes to String class of OpenJDK

Compilation of augmented JDK/JRE

JDK/JRE provide ***additional functionality*** without sacrificing any compatibility

Our (current) implementation

specifies entry points in the String class

```
public interface IStringLogic {

    public boolean applyBeforeToString(String s) throws Exception;

    public boolean applyOnInitialization(String s) throws Exception;

    ...

    public String getDescription();

}
```

holds arbitrary Java code run before toString()

holds arbitrary Java code run in constructor

returns description of the corresponding logic

DEMO

Use cases

Security
(prevent password leaks and code injection attacks)

Compliance
(ensure strict value conformance)

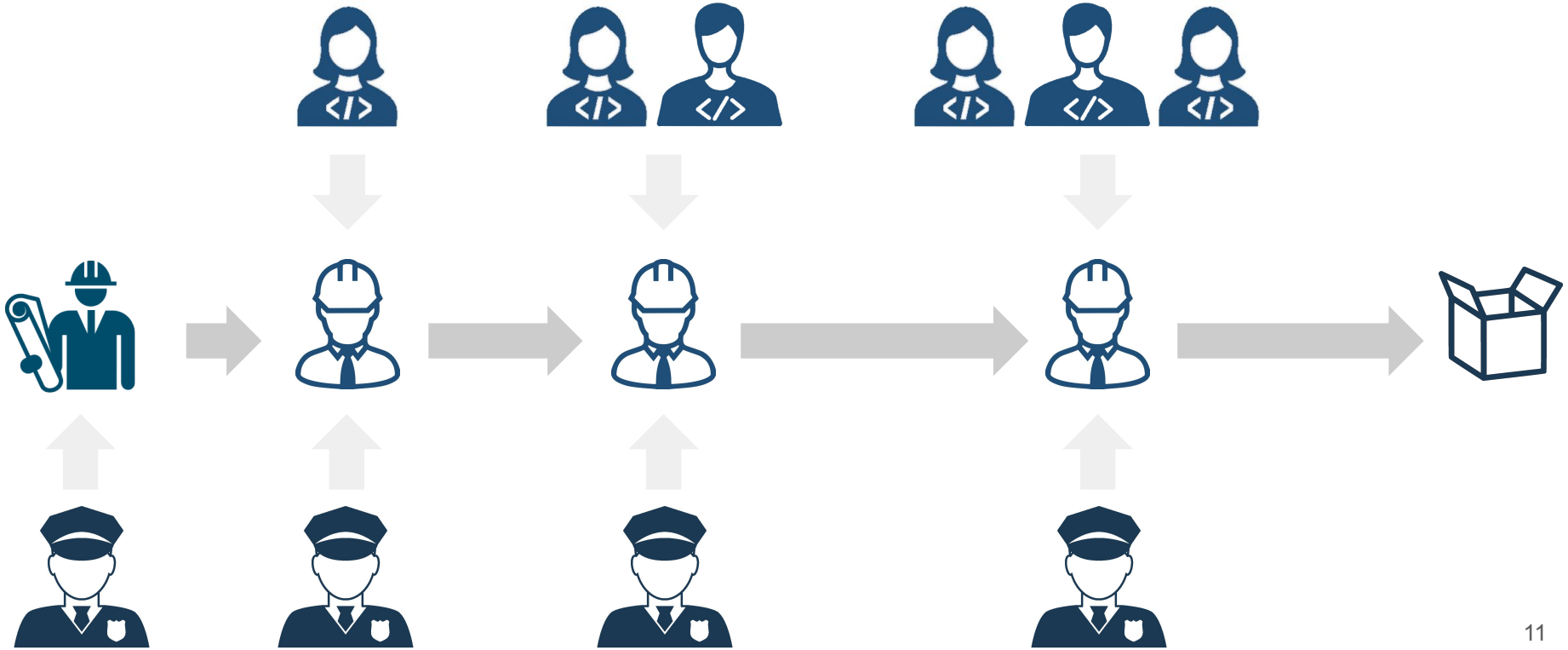
Monitoring
(become notified if certain conditions are met)

Debugging
(use provided information for relaxed debugging)

How could String 2.0 improve traditional workflows?

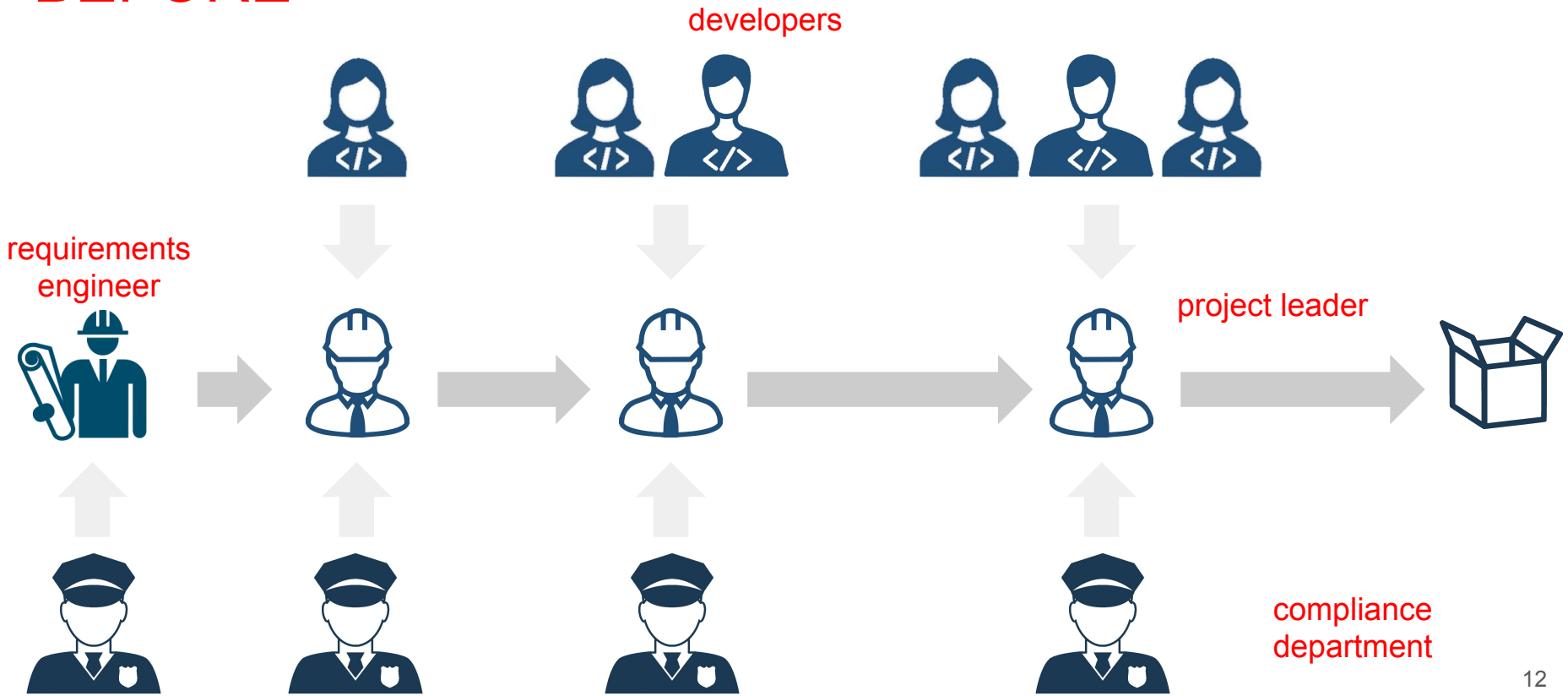
Example: Compliance validation

BEFORE



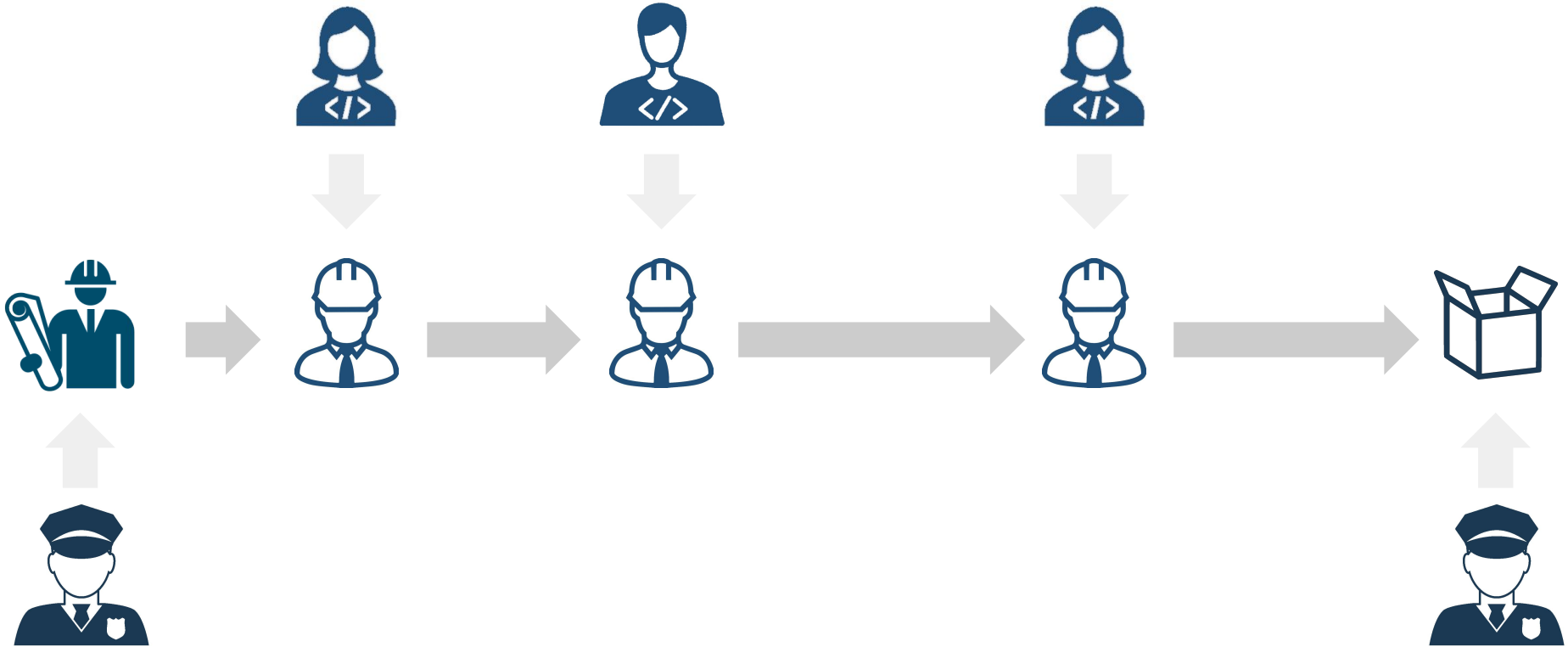
Example: Compliance validation

BEFORE



Example: Compliance validation

AFTER



Next steps

String tracing
(e.g., with a tree)

Logic inheritance
(transfer logic to derived strings)

Evaluation

Summary

What have data leaks in common?

They originate from `String` !

2

What we aim for:

- have ~~no~~ security built-in
- are ~~not~~ content-aware and offer ~~no~~ history
- ~~are complex to track~~ track themselves intrinsically
- have ~~static~~ dynamic functionality
- functionality is ~~rather basic~~ context-dependent

4

Our (current) implementation

specifies entry points in the String class

```
public interface IStringLogic {

    public boolean applyBeforeToString(String s) throws Exception;

    public boolean applyOnInitialization(String s) throws Exception;

    ...

    public String getDescription();
}
```

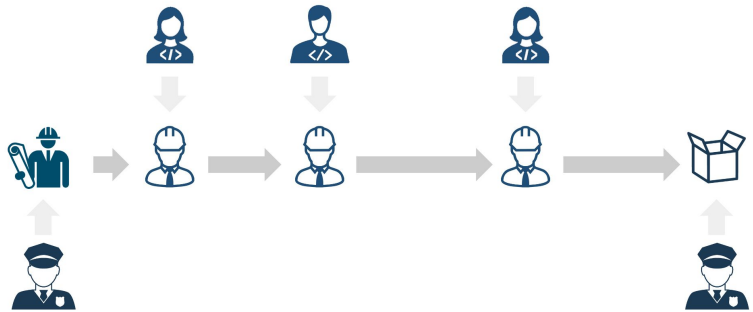
holds arbitrary Java code run before toString()

holds arbitrary Java code run in constructor

returns description of the corresponding logic

7

Example: Compliance validation
AFTER



13