### BString: A String-based Framework to Improve Application Security

MSc Thesis (final presentation)

19 January 2022

**Christian Zürcher** 

#### Two major problems of our time





#### Data leaks

## Remote Code Execution (RCE)

Important: These threats usually originate from String values.

#### Two major problems of our time

2017	2021
A01:2017-Injection	A01:2021-Broken Access Control
A02:2017-Broken Authentication	A02:2021-Cryptographic Failures
A03:2017-Sensitive Data Exposure	A03:2021-Injection
A04:2017-XML External Entities (XXE)	(New) A04:2021-Insecure Design
A05:2017-Broken Access Control	A05:2021-Security Misconfiguration
A06:2017-Security Misconfiguration	A06:2021-Vulnerable and Outdated Components
A07:2017-Cross-Site Scripting (XSS)	A07:2021-Identification and Authentication Failures
A08:2017-Insecure Deserialization	
A09:2017-Using Components with Known Vulnerabilities	A09:2021-Security Logging and Monitoring Failures*
A10:2017-Insufficient Logging & Monitoring	(New) A10:2021-Server-Side Request Forgery (SSRF)*
	* From the Survey

#### Security measures for strings already exist...



Restrictive data types





Taint and data flow analyses

#### ..., but they lack fundamental features



limited interoperability between different tools



no shared configuration

no standardized set up

limited automation

#### Our idea: a generalizable security framework for String

**Step 1:** Improving the String class of OpenJDK

#### *Step 2: Compiling* the modified JDK/JRE

Step 3:Using the additional functionality withoutsacrificing any compatibility

#### **Provided API**

...

public interface IStringBehavior {

public String applyOnCreation(String s);

public String applyOnRead(String s);

public boolean attachToChild();

public boolean recordHistory();

#### **Behind the scenes**



#### Challenges

#### Don't break the system:

- String is used throughout the Java VM
- Changes must not alter the existing behavior
- Recursions are very easy to introduce

#### Native code complications:

- The string pooling uses native code
- Translation between different data types

## **RQ1**: What are the restrictions when used with existing Java code?

#### Compatibility

Project Name			
Apache Comons (IO)			
Apache Commons (Logging)			
Apache HttpClient			
Gson			
JavaMail			
Log4J (core)			
Logback (classic)			
SLF4J			
SLF4J (API)			
Spring			
Square Okhttp			
Square Okio			
Square Retrofit			

Package	Version	Compatible?
Commons-io	2.8.0	✓
Commons-logging	1.2	✓
Org.apache.httpcomponents	4.5.13	✓
Com.google.code.gson	2.8.5	✗ (reflection)
Com.sun.mail	1.6.0	✓
Org.apache.logging.log4j	2.14.1	✓
Ch.qos.logback	1.3.0-alpha5	✓
Slf4j-simple	2.0.0-alpha1	✗ (custom byte buffer)
Org.slf4j	2.0.0-alpha1	✓
Org.springframework	5.3.6	✓
Com.squareup.okhttp3	5.0.0-alpha.2	✓
Com.squareup.okhokio	2.10.0	✓
Com.squareup.okhretrofit	2.9.0	✓

Table 4.1: Evaluation of popular Java libraries

#### Restrictions

#### Native code

Value conversion

Reflection

Concurrency



## **RQ2**: What are the security gains and threats?

#### **Security gains**



#### Date type emulation



In-memory encryption





₿

#### Taint analysis



#### Data flow analysis





String or application hijacking



#### **Developer confusion**

#### **Performance evaluation**

[milliseconds]	String initialization	Read value	Attaching behavior to new String
baseline	4	166	11
without behavior	13	171	20
with empty behavior	16	193	22
with behavior attachment	16	191	32
with history	59	213	87
with encryption on read in IO	2,388	9,331	19
with encryption and decryption	5,112	7,982	timeout

## DEMO

#### Conclusion



only one framework required limited interoperability between different tools







#### Summary







# Security gains Image: Security gains Image: Date type emulation Image: Image: Security gains Image: Off-memory encryption Image: Taint analysis Image: Data flow analysis