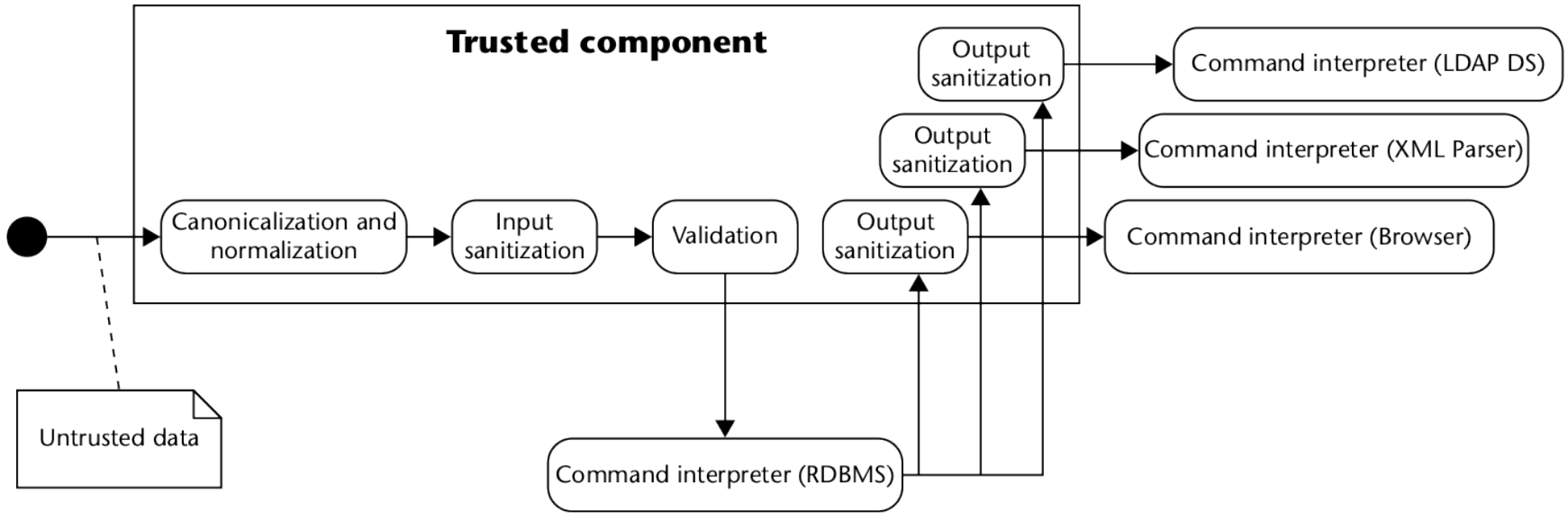
Java
Input Validation and Data Sanitization (IDS)

Mario Alviano

University of Calabria, Italy

A.Y. 2016/2017

Introduction



✗ Never, ever trust external input

✓ Always canonicize/normalize, sanitize and validate it

Prevent SQL Injection

- ✗ Do not form SQL queries from user input
- ✓ Sanitize and validate user input
- ✓ Use parametric queries

```
https://www.securecoding.cert.org/confluence/
display/java/IDS00-J.+Prevent+SQL+injection
```

Prevent XML Injection

- ✗ Do not form XML documents from user input
- ✓ Sanitize and validate user input
- ✓ Use schema validation

```
https://www.securecoding.cert.org/confluence/
display/java/IDS16-J.+Prevent+XML+Injection
```

Prevent XML External Entity Attacks

- ✗ Entity replacement may cause denial of service attacks
- ✓ Only allow known replacements

```
https:
//www.securecoding.cert.org/confluence/display/
java/IDS17-J.+Prevent+XML+External+Entity+Attacks
```

Normalize strings before validating them

- ✗ Unnormalized strings may prevent validation failures
- ✓ Normalization guarantees unique string representation
- ? Unicode normalization forms
  `http://unicode.org/reports/tr15/`

```
https://www.securecoding.cert.org/confluence/
display/java/IDS01-J.+Normalize+strings+before+
validating+them
```

Canonicalize path names before validating them

- ✗ Several representations for the same file/directory
- ✗ Use of `..` may result into path traversal attacks
- ✓ Canonicization gives the real path name

```
https://www.securecoding.cert.org/confluence/
display/java/FIO16-J.+Canonicalize+path+names+
before+validating+them
```

Safely extract files from ZipInputStream

- ✗ ZIP content may explode (ZIP bombs)
- ✗ Possible denial of service attacks
- ✓ Check the size of extracted content

```
https://www.securecoding.cert.org/confluence/
display/java/IDS04-J.+Safely+extract+files+from+
ZipInputStream
```

Exclude unsanitized user input from format strings

- ✗ Format strings are dangerous
- ✗ Do not allow user input become the format string
- ✓ Always pass user input as a string parameter (`%s`)

```
https://www.securecoding.cert.org/confluence/
display/java/IDS06-J.+Exclude+unsanitized+user+
input+from+format+strings
```

Sanitize untrusted data passed to the Runtime.exec() method

- ✗ Do not execute untrusted content!
- ✓ Check the presence of `&` and `;`
- ✓ Use internal code if possible

```
https://www.securecoding.cert.org/confluence/
display/java/IDS07-J.+Sanitize+untrusted+data+
passed+to+the+Runtime.exec%28%29+method
```

Sanitize untrusted data included in a regular expression

- ✗ Quite similar to other injection attacks
- ✓ Whitelist user input
- ✓ Escape malicious characters

```
https://www.securecoding.cert.org/confluence/
display/java/IDS08-J.+Sanitize+untrusted+data+
included+in+a+regular+expression
```

Specify an appropriate locale when comparing locale-dependent data

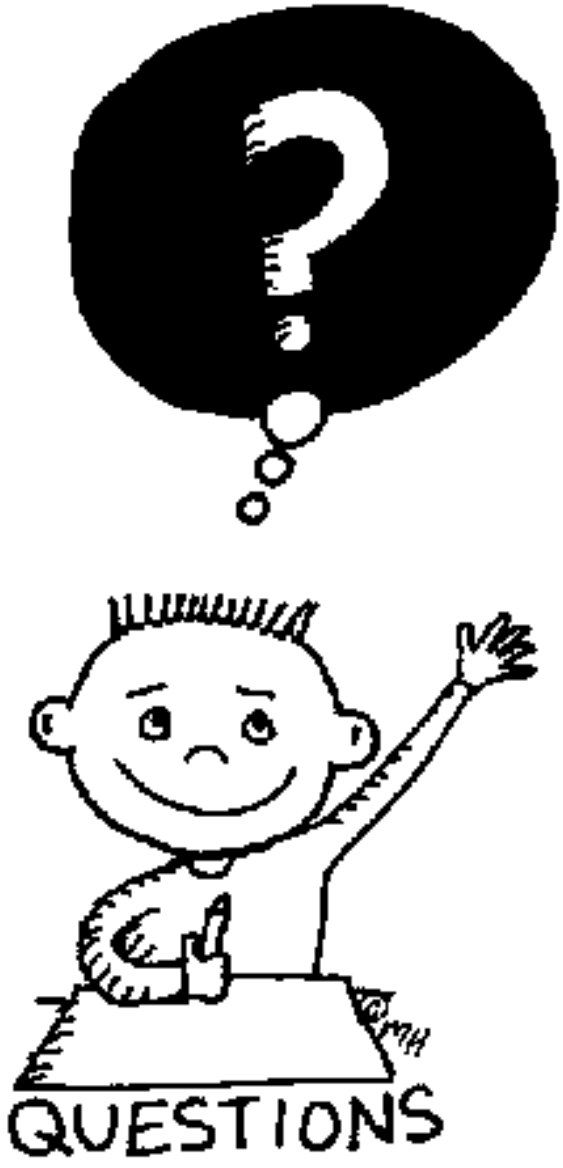
- ✗ Do not underestimate locale differences
- ✓ Pass a locale whenever possible
- ✓ Bypass locale methods if possible

```
https://www.securecoding.cert.org/confluence/
display/java/STR02-J.+Specify+an+appropriate+
locale+when+comparing+locale-dependent+data
```

Perform any string modifications before validation

- ✗ Modifying validated strings may give invalid strings
- ✓ Postpone validation
- ✓ Replace invalid characters instead of removing them

```
https://www.securecoding.cert.org/confluence/
display/java/IDS11-J.+Perform+any+string+
modifications+before+validation
```



END OF THE LECTURE