Lecture 11 – Defense Strategies

[COSE451] Software Security

Instructor: Seunghoon Woo

Spring 2024

Overview

- **Defense strategies**

Defense strategies

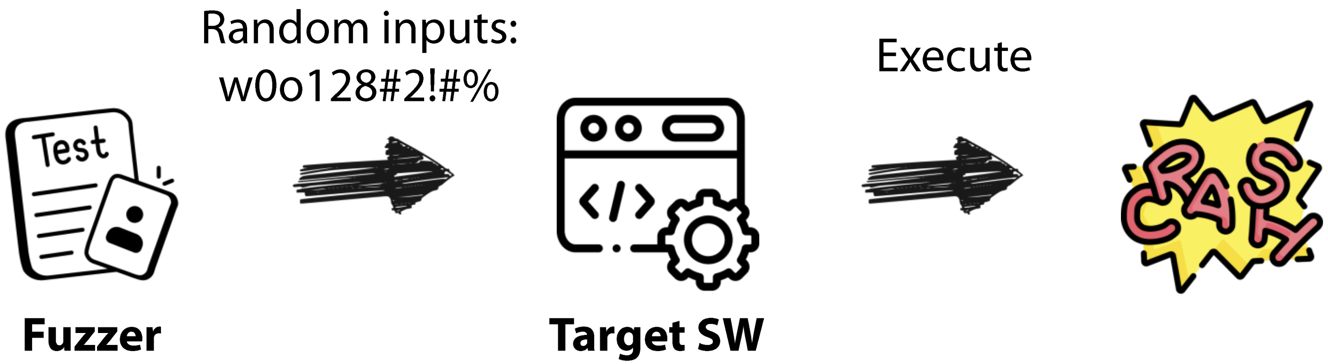
- **Vulnerability detection strategies**
  - **Static analysis**
    - Examining source code without executing it
      - To identify potential security vulnerabilities
    - Also called whitebox testing
  - **Dynamic analysis**
    - Running the program and analyzing its behavior during execution
      - To identify potential security vulnerabilities
    - Also called blackbox testing

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**
  - An automated testing technique to find program inputs that reveal a bug (or vulnerability)
  - How?
    - Generate inputs randomly until program crashes!
  - Fuzzer-found bugs
    - Causes: incorrect arg validation, incorrect type casting, etc.
    - Effects: stack/heap buffer overflows, memory leak, use-after-free, division-by-zero, out-of-bounds, etc.

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**



Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**
  - Purely random data is not a very interesting input!

Target SW

Expected inputs: `$ Target –f build.xml`

```
<project default="dist">
  <target name="init">
    <mkdir dir = "${build}"/>
  </target> …
```

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**
  - Purely random data is not a very interesting input!

Target SW

Expected inputs: `$ Target –f build.xml`

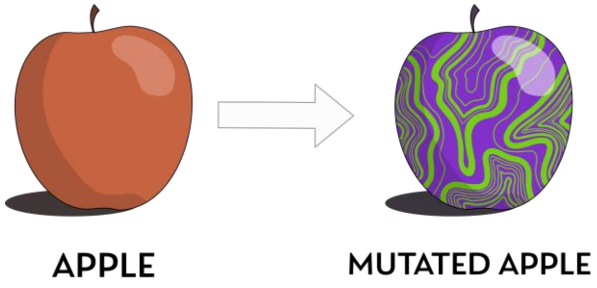
<project default="dist">
  <target name="init">
    <mkdir dir = "${build}"/>
  </target> …

Random inputs: `$ Target –f random.xml`

Afkldjkl123$JKdsnkj!@#Kw;xk
dmakd2K#Nk013kn19dk1CK
#$%mkxd...

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**
  - Using mutation technique!
    1. Collecting legitimate input or input that causes known vulnerabilities
       - These data are called "seeds"
    2. Create testing input by changing some part of the legitimate input



https://graphql.com/learn/mutations/

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**
  - Example: mutated input

Target SW

Expected inputs: `$ Target -f build.xml`

```
<project default="dist">
  <target name="init">
    <mkdir dir = "${build}"/>
  </target> …
```

Random inputs: `$ Target -f mutate.xml`

```
<project def!ault="dist">
  <taWget namD="init">
    <madir dir = "2{build}"/@
  </targe?t> …
```

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**
  - Mutation heuristics
    - Binary input
      - Bit flips
      - Change/insert/delete random bytes
      - Set randomly chosen bytes to interesting values (e.g., INT_MAX, INT_MIN, 0, 1, -1)
    - Text input
      - Change/insert/delete random symbols or keywords from a dictionary

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**
  - Effectiveness of fuzzing
    - Code coverage
      - A measure of how much of the program the input can cover

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**
  - Effectiveness of fuzzing
    - Code coverage
      - A measure of how much of the program the input can cover

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        if (y > 0)
            printf("Here!\n");
    }
    printf("Test end T^T\n");
}
```

- if x = 10, y = 5 then

  * coverage = 4/5 = 80% (ignore curly braces)

Defense strategies

- **Dynamic analysis: fuzz testing (fuzzing)**
  - Effectiveness of fuzzing
    - Code coverage
      - A measure of how much of the program the input can cover

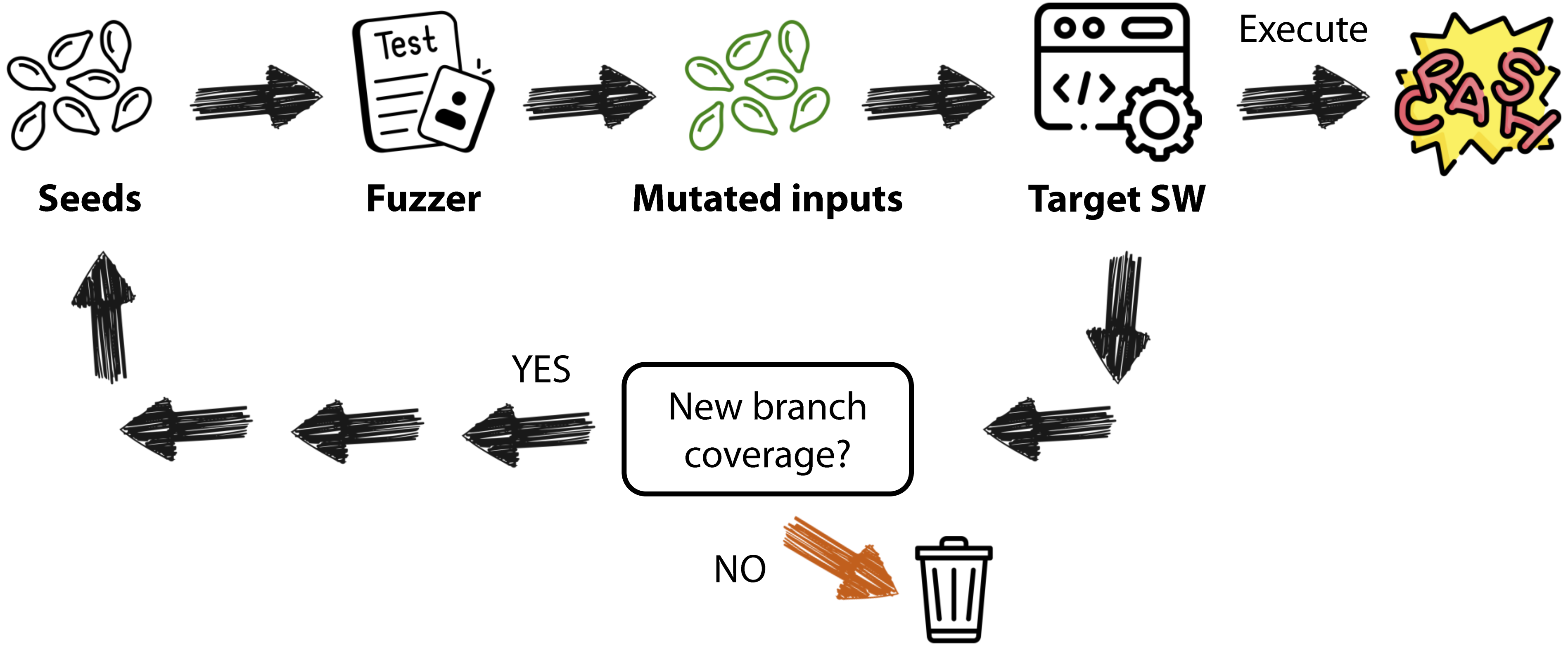
```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        if (y > 0)
            printf("Here!\n");
    }
    printf("Test end T^T\n");
}
```

- if x = 10, y = 5 then
  * coverage = 4/5 = 80%

- if x = 10, y = -5 then
  * coverage = 5/5 = 100%

Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**



Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
    }
}
```

Seed

x, y, z = (5, -1, -5)

Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**

```
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
    }
}
```

Seed
x, y, z = (5, -1, -5)

Input
x, y, z = (**6**, -1, -5)

Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**

```
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
    }
}
```

Seed
x, y, z = (5, -1, -5)

Input
x, y, z = (**6**, -1, -5)

Coverage
4/7 = **57%**

Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
    }
}
```

Seed

x, y, z = (5, -1, -5)

Input

x, y, z = (**6**, -1, -5)

Coverage

4/7 = **57%**

Not new branch coverage

-> Do not add to seed

Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
    }
}
```

Seed

x, y, z = (5, -1, -5)

Input

x, y, z = (5, **100**, -5)

Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**

```
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
    }
}
```

Seed
x, y, z = (5, -1, -5)

Input
x, y, z = (5, **100**, -5)

Coverage
6/7 = **86%**

New branch coverage
-> Add to seed

Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
    }
}
```

Seeds

x, y, z = (5, -1, -5)

x, y, z = (5, 100, -5)

Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**

```
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
    }
}
```

Seeds

x, y, z = (5, -1, -5)

x, y, z = (5, 100, -5)

Input

x, y, z = (5, 100, **10**)

Defense strategies

- **Dynamic analysis: coverage-guided fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
    }
}
```

Seeds

x, y, z = (5, -1, -5)

x, y, z = (5, 100, -5)

Input

x, y, z = (5, 100, **10**)

Coverage

7/7 = 100%

Crash detect!

Defense strategies

- **Dynamic analysis: directed fuzzing**
  - Fuzzing techniques to quickly reach a specific point rather than coverage
  - Typically done as a greybox testing
    - Source code + binary
  - For example
    - The closer the fuzzing input gets to a certain point, the higher the score is given

Defense strategies

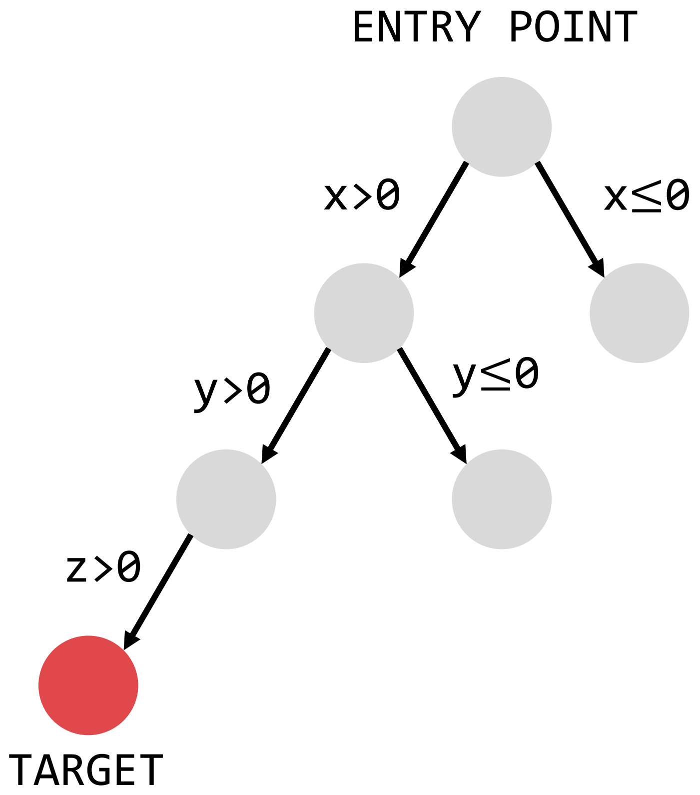
- **Dynamic analysis: directed fuzzing**

```
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```

Defense strategies

- **Dynamic analysis: directed fuzzing**

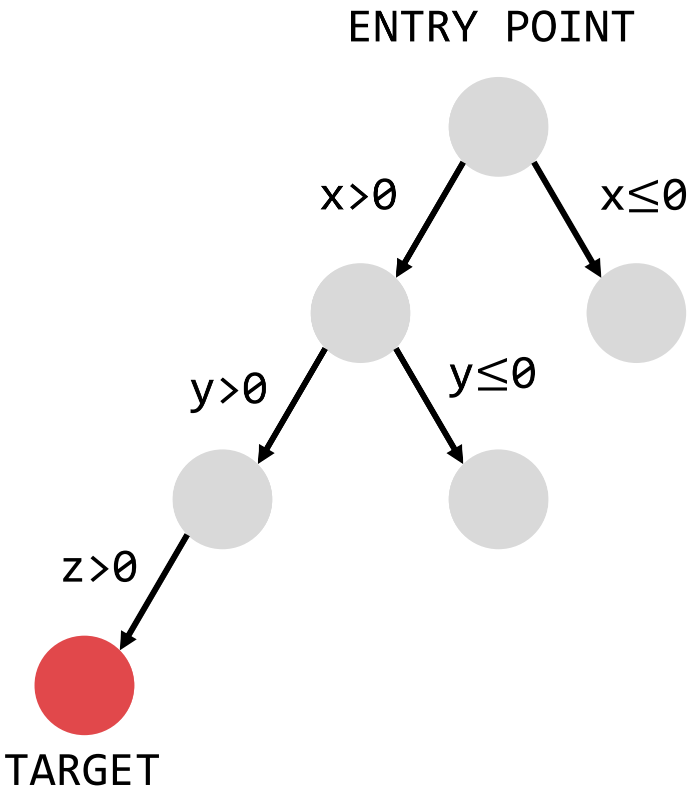
```
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```



Defense strategies

- **Dynamic analysis: directed fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```



Seeds

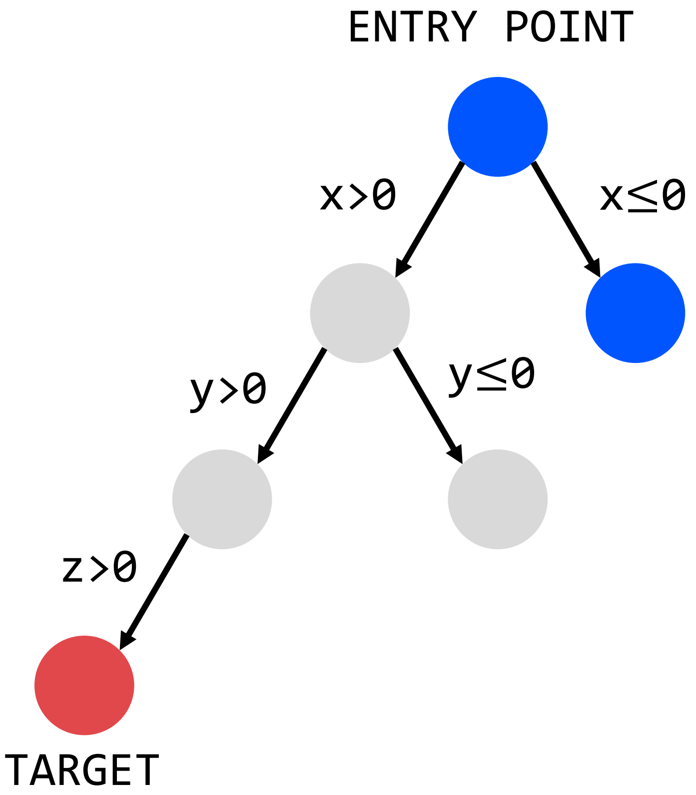
① x, y, z = (-5, -1, -5)

② x, y, z = (5, -1, -5)

Defense strategies

- **Dynamic analysis: directed fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```



Seeds

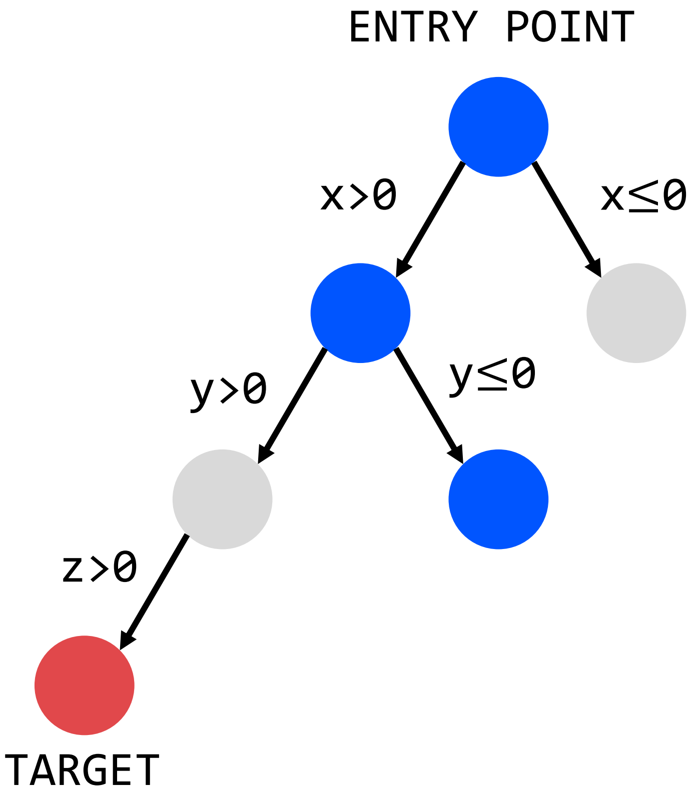
① x, y, z = (-5, -1, -5)

② x, y, z = (5, -1, -5)

Defense strategies

- **Dynamic analysis: directed fuzzing**

```
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```



Seeds

① x, y, z = (-5, -1, -5)

② x, y, z = (5, -1, -5)

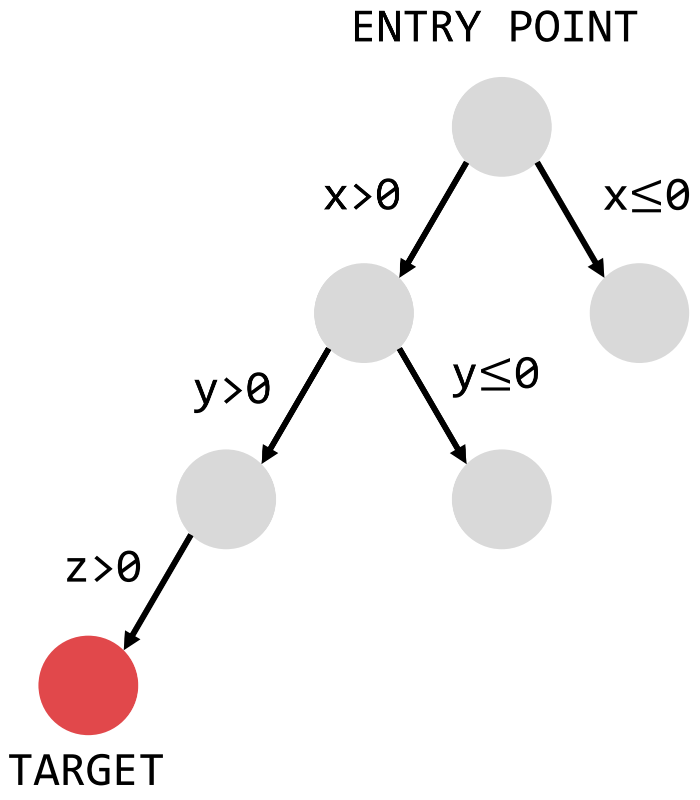
② win!

> Mutating the following testing input from ②

Defense strategies

- **Dynamic analysis: directed fuzzing**

```
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```



Seeds

① x, y, z = (-5, -1, -5)

② x, y, z = (5, -1, -5)

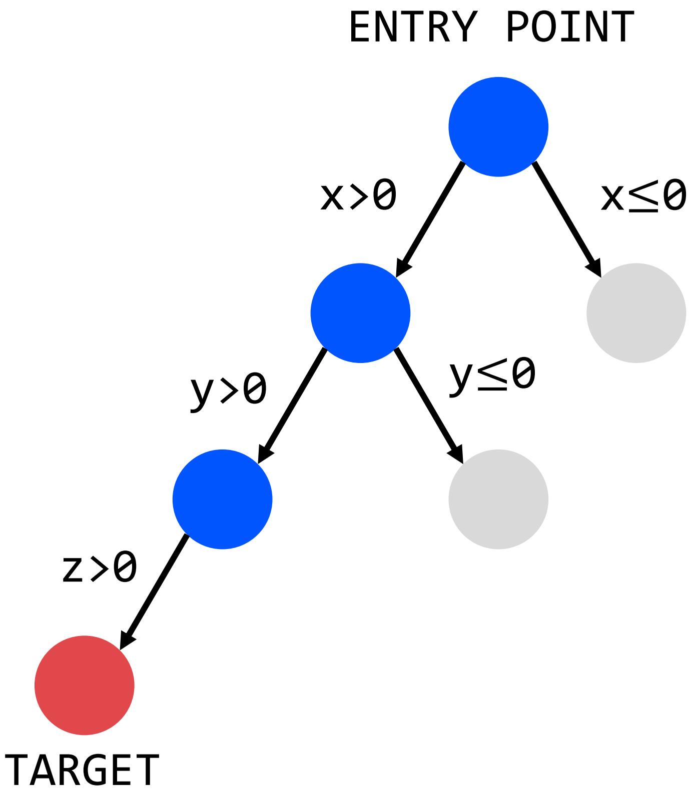
Input

x, y, z = (5, **100**, -5)

Defense strategies

- **Dynamic analysis: directed fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```



Seeds

① x, y, z = (-5, -1, -5)

② x, y, z = (5, -1, -5)

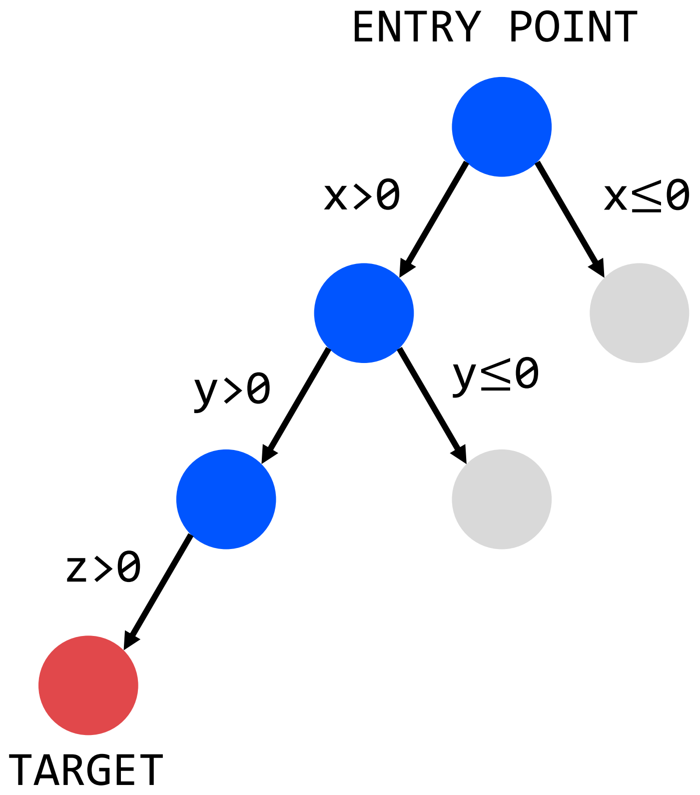
Input

x, y, z = (5, 100, -5)

Defense strategies

- **Dynamic analysis: directed fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```



Seeds

① x, y, z = (-5, -1, -5)

② x, y, z = (5, -1, -5)

Input

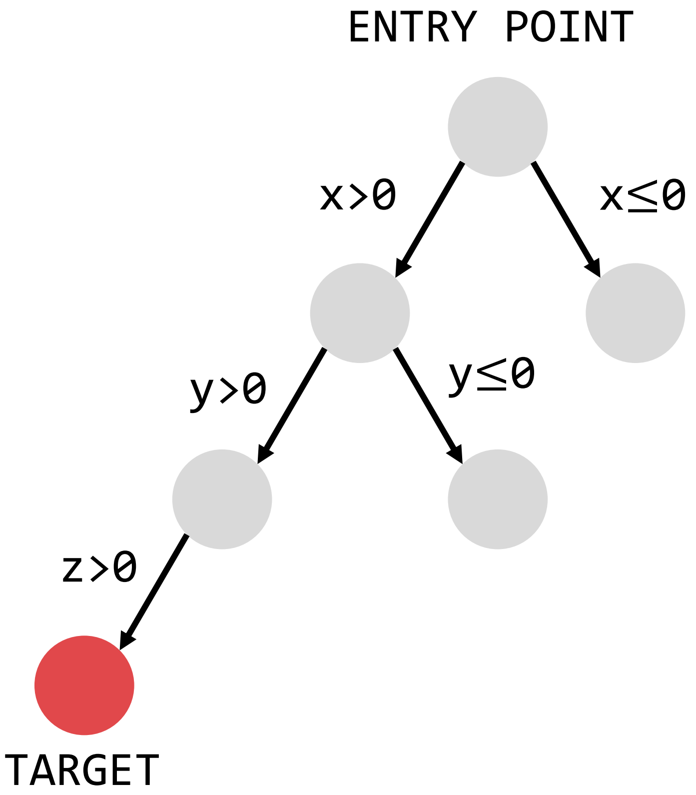
x, y, z = (5, 100, -5)

> Add to seed and mutating the following testing input from this input

Defense strategies

- **Dynamic analysis: directed fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```



Seeds

① x, y, z = (-5, -1, -5)
② x, y, z = (5, -1, -5)
③ x, y, z = (5, 100, -5)

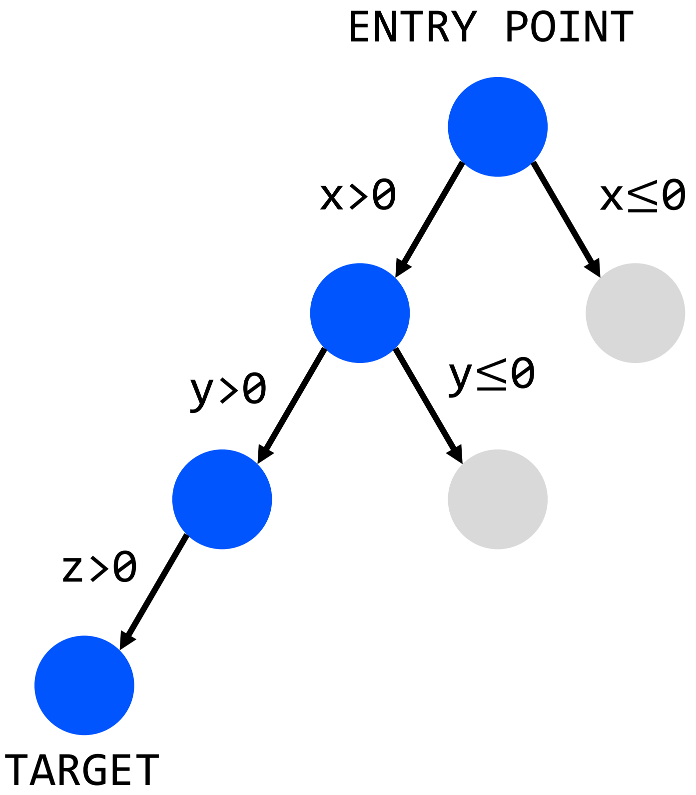
Input

x, y, z = (5, 100, **5**)

Defense strategies

- **Dynamic analysis: directed fuzzing**

```c
void test(int x, int y, int z){
    printf("Test start!!\n");
    if (x > 0){
        printf("X: positive integer");
        if (y > 0){
            printf("Y: positive integer");
            if (z > 0)
                assert();
        }
        else{
            printf("Y: negative integer");
        }
    }
    else{
        printf("X: negative integer");
    }
}
```



Seeds

① x, y, z = (-5, -1, -5)
② x, y, z = (5, -1, -5)
③ x, y, z = (5, 100, -5)

Input

x, y, z = (5, 100, 5)

Crash detect!

Defense strategies

- **Dynamic analysis: taint analysis**
  - A technique for analyzing information flow
  - Tracking how private information flows through the program and if it is leaked to public observers
  - Terms
    - Sources: private data of interest
    - Sinks: locations of interest
      - Check taints of incoming information
      - Determines if there is a leak in the program

Defense strategies

- **Dynamic analysis: taint analysis**

```
x = source();
y = x;
if (y == 0)
  z = x+y;
else
  z = 1;
sink(z);
```

Defense strategies

- **Dynamic analysis: taint analysis**

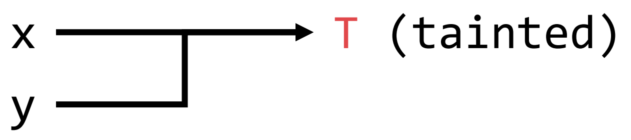
```
x = source();
y = x;
if (y == 0)
  z = x+y;
else
  z = 1;
sink(z);
```

x ⟶ T (tainted)

Defense strategies

- **Dynamic analysis: taint analysis**

```
x = source();
y = x;
if (y == 0)
  z = x+y;
else
  z = 1;
sink(z);
```

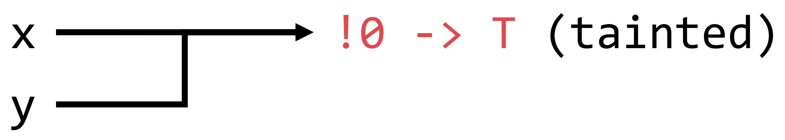


Defense strategies

- **Dynamic analysis: taint analysis**

```
x = source();
y = x;
if (y == 0)
  z = x+y;
else
  z = 1;
sink(z);
```

Case 1) if $x \neq 0$

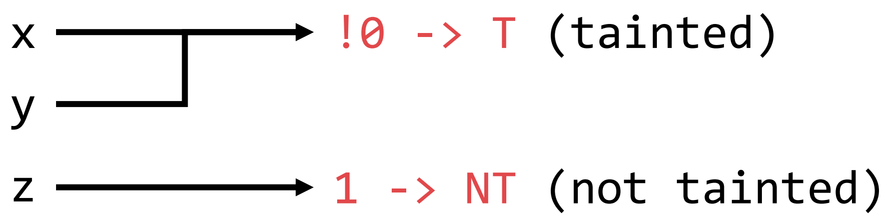


Defense strategies

- **Dynamic analysis: taint analysis**

```
x = source();
y = x;
if (y == 0)
  z = x+y;
else
  z = 1;
sink(z);
```

Case 1) if x ≠ 0



SAFE

Defense strategies

- **Dynamic analysis: taint analysis**

```
x = source();
y = x;
if (y == 0)
  z = x+y;
else
  z = 1;
sink(z);
```

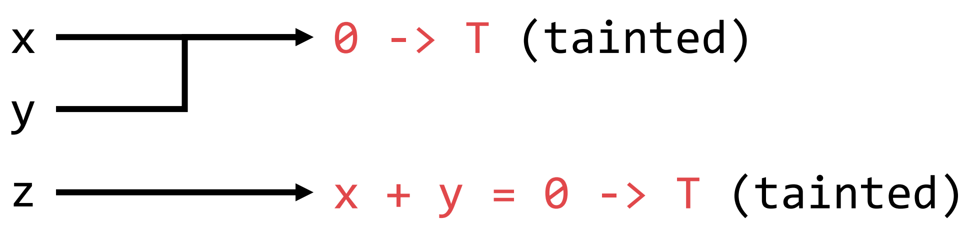
Case 2) if x = 0

Defense strategies

- **Dynamic analysis: taint analysis**

```
x = source();
y = x;
if (y == 0)
  z = x+y;
else
  z = 1;
sink(z);
```

Case 2) if x = 0



LEAK!!

Defense strategies

- **Dynamic analysis: taint analysis**

```
from flask import Flask, request, render_template_string

app = Flask(__name__)

@app.route('/')
def index():
    return '''
        <form action="/greet" method="post">
            <input type="text" name="name">
            <input type="text" name="id">
            <input type="submit" value="Greet">
        </form>
    '''

@app.route('/greet', methods=['POST'])
def greet():
    user_name  = request.form['name']
    user_id    = request.form['id']
    print_name = "Dr." + user_name
    return render_template_string(f'<h1>Hello, {print_name}!</h1>')

if __name__ == '__main__':
    app.run(debug=True)
```

Defense strategies

- **Dynamic analysis: taint analysis**

```
from flask import Flask, request, render_template_string

app = Flask(__name__)

@app.route('/')
def index():
    return '''
        <form action="/greet" method="post">
            <input type="text" name="name">
            <input type="text" name="id">
            <input type="submit" value="Greet">
        </form>
    '''

@app.route('/greet', methods=['POST'])
def greet():
    user_name  = request.form['name']  SOURCE
    user_id    = request.form['id']    SOURCE
    print_name = "Dr." + user_name
    return render_template_string(f'<h1>Hello, {print_name}!</h1>') SINK

if __name__ == '__main__':
    app.run(debug=True)
```

Defense strategies

- **Dynamic analysis: taint analysis**

```python
from flask import Flask, request, render_template_string

app = Flask(__name__)

@app.route('/')
def index():
    return '''
        <form action="/greet" method="post">
            <input type="text" name="name">
            <input type="text" name="id">
            <input type="submit" value="Greet">
        </form>
    '''

@app.route('/greet', methods=['POST'])
def greet():
    user_name  = request.form['name']
    user_id    = request.form['id']
    print_name = "Dr." + user_name
    return render_template_string(f'<h1>Hello, {print_name}!</h1>')

if __name__ == '__main__':
    app.run(debug=True)
```

user_name → T (tainted)

user_id → T (tainted)

Defense strategies

- **Dynamic analysis: taint analysis**

```python
from flask import Flask, request, render_template_string

app = Flask(__name__)

@app.route('/')
def index():
    return '''
        <form action="/greet" method="post">
            <input type="text" name="name">
            <input type="text" name="id">
            <input type="submit" value="Greet">
        </form>
    '''

@app.route('/greet', methods=['POST'])
def greet():
    user_name  = request.form['name']
    user_id    = request.form['id']
    print_name = "Dr." + user_name
    return render_template_string(f'<h1>Hello, {print_name}!</h1>')

if __name__ == '__main__':
    app.run(debug=True)
```

user_name → T (tainted)

user_id → T (tainted)

print_name → T (tainted)

XSS!!

Defense strategies

- **Dynamic analysis: taint analysis**

```python
import os

def process_input(user_input):
    sanitized_input = sanitize(user_input)
    execute_command(sanitized_input)

def sanitize(input_str):
    return input_str.replace(";", "")

def execute_command(command):
    os.system(command)

user_input = input("Enter a command: ")
process_input(user_input)
```

Defense strategies

- **Dynamic analysis: taint analysis**

```
         import os

         def process_input(user_input):
             sanitized_input = sanitize(user_input)
             execute_command(sanitized_input)

         def sanitize(input_str):
             return input_str.replace(";", "")

         def execute_command(command):
SINK ➡       os.system(command)

SOURCE ➡ user_input = input("Enter a command: ")
         process_input(user_input)
```

Defense strategies

- **Dynamic analysis: taint analysis**

```python
import os

def process_input(user_input):
    sanitized_input = sanitize(user_input)
    execute_command(sanitized_input)

def sanitize(input_str):
    return input_str.replace(";", "")

def execute_command(command):
    os.system(command)

user_input = input("Enter a command: ")
process_input(user_input)
```

`user_input` → `T (tainted)`

Defense strategies

- **Dynamic analysis: taint analysis**

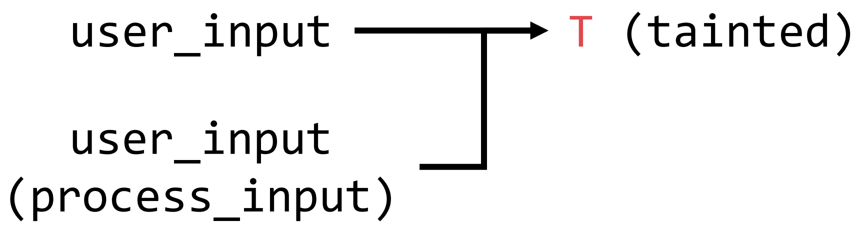
```python
import os

def process_input(user_input):
    sanitized_input = sanitize(user_input)
    execute_command(sanitized_input)

def sanitize(input_str):
    return input_str.replace(";", "")

def execute_command(command):
    os.system(command)

user_input = input("Enter a command: ")
process_input(user_input)
```



Defense strategies

- **Dynamic analysis: taint analysis**

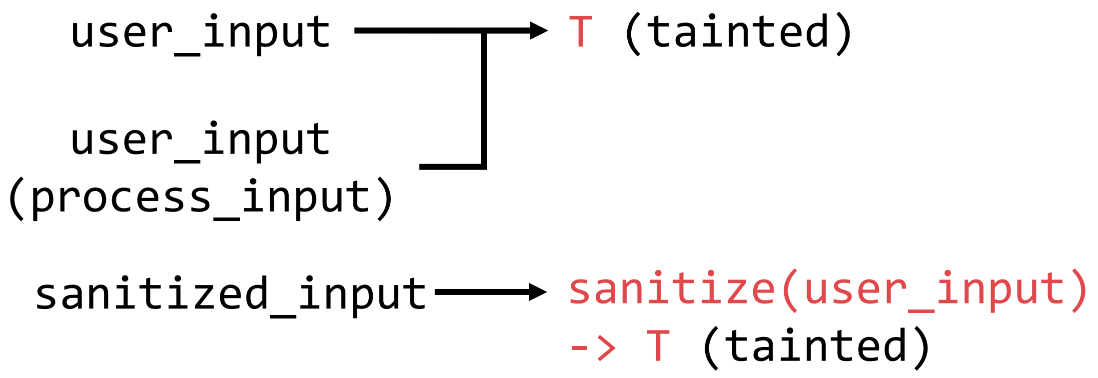
```python
import os

def process_input(user_input):
    sanitized_input = sanitize(user_input)
    execute_command(sanitized_input)

def sanitize(input_str):
    return input_str.replace(";", "")

def execute_command(command):
    os.system(command)

user_input = input("Enter a command: ")
process_input(user_input)
```



Defense strategies

- **Dynamic analysis: taint analysis**

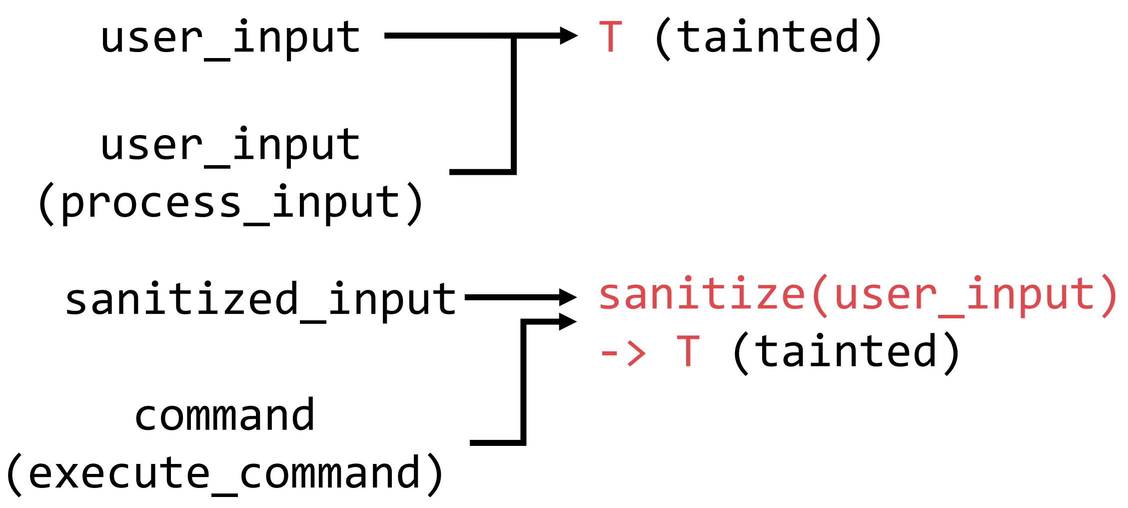
```python
import os

def process_input(user_input):
    sanitized_input = sanitize(user_input)
    execute_command(sanitized_input)

def sanitize(input_str):
    return input_str.replace(";", "")

def execute_command(command):
    os.system(command)

user_input = input("Enter a command: ")
process_input(user_input)
```



Defense strategies

- **Dynamic analysis: taint analysis**

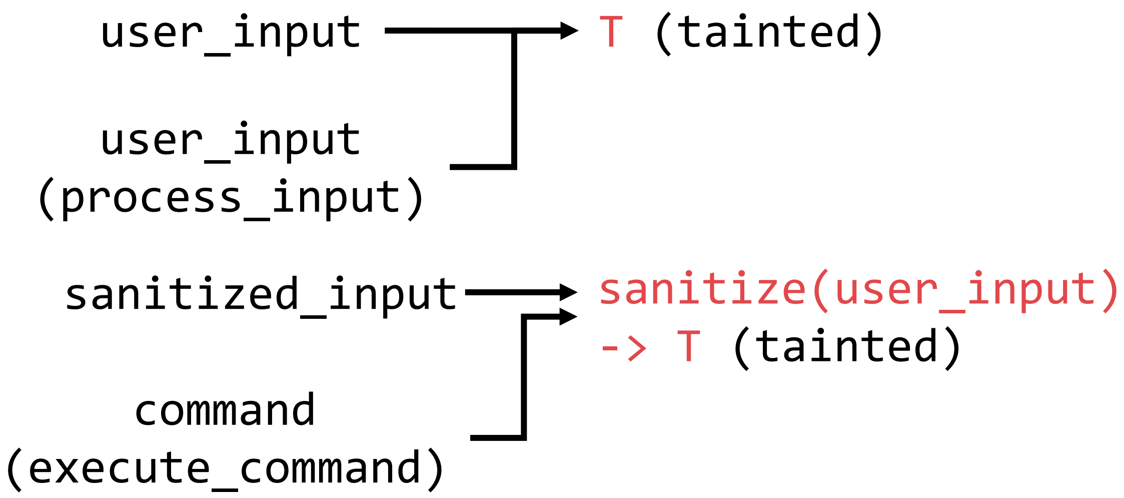
```python
import os

def process_input(user_input):
    sanitized_input = sanitize(user_input)
    execute_command(sanitized_input)

def sanitize(input_str):
    return input_str.replace(";", "")

def execute_command(command):
    os.system(command)

user_input = input("Enter a command: ")
process_input(user_input)
```



Command Injection!!

Defense strategies

- **Dynamic analysis: taint analysis**
  - For convenience, I explained it using the source code, but...
    - In actual dynamic taint analysis, memory and registers are used to track tainted values
    - E.g., if a tainted variable stored in memory is loaded and used to update the value of another variable, then the latter variable is also considered tainted
    - This is a rather deep topic, so I encourage you to understand taint analysis at the source code level!

Defense strategies

- **Dynamic analysis: behavior analysis**
  - A technique to observe and analyze the behavior of malicious code in real time to determine its characteristics and intent
  - Using sandbox environment
    - An isolated execution environment for behavioral analysis
    - Protect the program being analyzed from affecting the actual system

Defense strategies

- **Dynamic analysis: behavior analysis**
  - Example steps
    1. Prepare the environment: Set up an isolated environment for analysis
    2. Program execution: Run the program being analyzed and observe its behavior
    3. Data collection: A variety of data is collected in real time, including file system access, network activity, and process creation
    4. Behavioral Analysis: Analyzes the behavior of a program based on collected data

Next Lecture

- **Special lecture!**