# Web Application Security

Lecture 5

Tamara Rezk

# HAPPY NEW YEAR!

# Web Application Security

- 19/11: Historical introduction to technologies and vulnerabilities that accompany them
- 26/11: Defenses XSS and DOM-XSS: sanitizers, Content Security Policy
- 10/12: Same Origin Policy
- 17/12: JavaScript vulnerabilities and defences
- 07/01: Workshop and Pysa
- 14/01: Workshop and Trusted Types (Google)
- 28/01: examen (9h -12h30), required: mandatory TP exercises

# Workshops

- Workshop presentations 7/1 and 14/1
  - 15’ presentation
  - 5’ questions : choose one group and prepare 2 questions.
  - write your topic in the excel sheet and also the questions group
  - workshop is mandatory so please justify absence
- Google presentation on Trusted types:  14/1 1hour (topics included in exam questions, cf. DOM-XSS)

# Pysa Tutorial

**https://github.com/facebook/pyre-check/tree/main/documentation/pysa_tutorial**

Machine Virtual:

# PYSA= PYthon Static Analyzer

- Goal:  identify potential security issues, flows from sources to sinks. Examples: remote code execution,  SSRF, XSS, broken access control.
- How:  it allows us to annotate sources and taints, define information flow rules to link them, and analyze Python code to see if rules are violated .
- Static Analyzer from META Based on Abstract interpretation.

# PYSA

It scales across millions of line of code:  over 40% Instagram vulnerabilities found with it.

# PYSA= PYthon Static Analyzer

- **Static** Analyzer Based on Abstract interpretation:
  - Advantages: looking at code without running it (no performance overhead), no false negatives (if no violation reported, then no vulnerability w.r.t. the input configuration)
  - Disadvantages: specific for a programming language, false positives (if violation reported, it might be that there is no vulnerability)

# Security

Many **attacks** are flows from

**sources**

to

**sinks**

Security is the absence of attacks for a give attacker model

# Security: confidentiality

Many **attacks** are flows from

**confidential sources**

to

**attacker-controlled sinks**

Example: a gps location flows to an attacker controlled web server

# Security: integrity

Many **attacks** are flows from

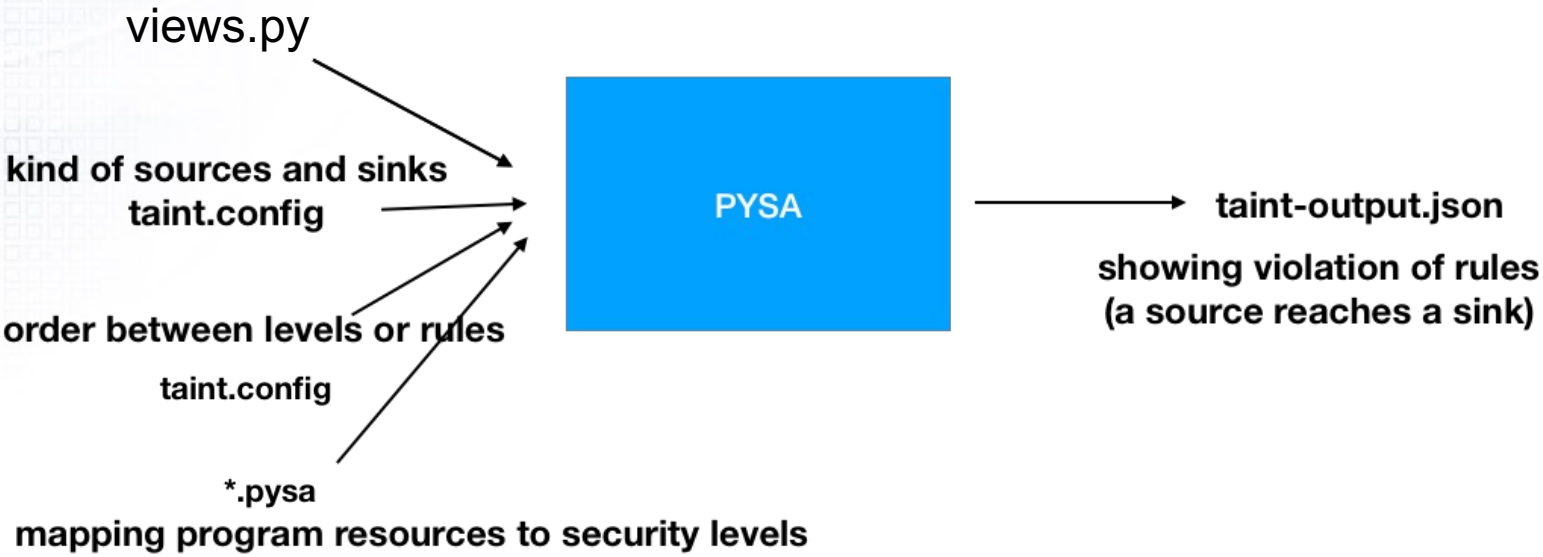
**untrusted sources**

to

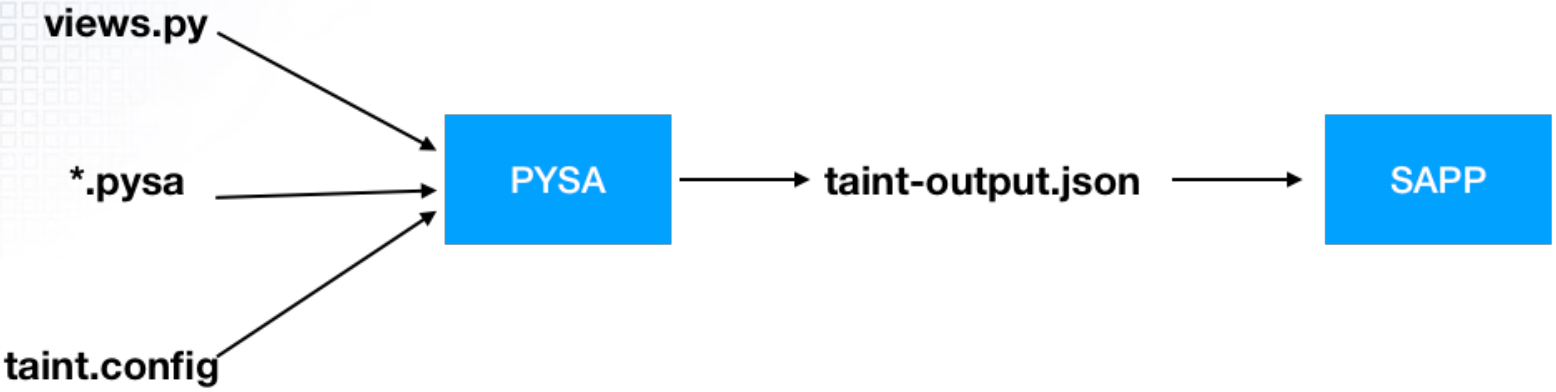
**integrity-sensitive sinks**

Example:
an input coming from the network is used to decide how to change the quantity of a substance used to regulate the water quality in a water plant.

# PYSA= PYthon Static Analyzer

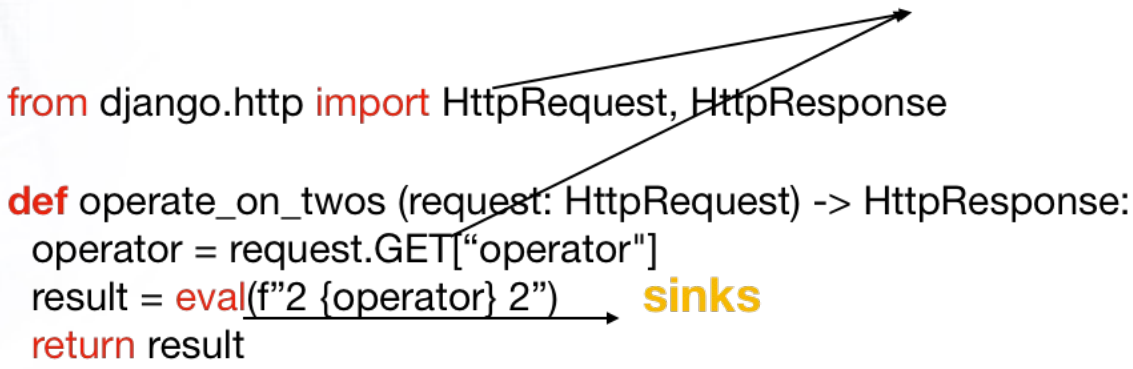


# PYSA= PYthon Static Analyzer



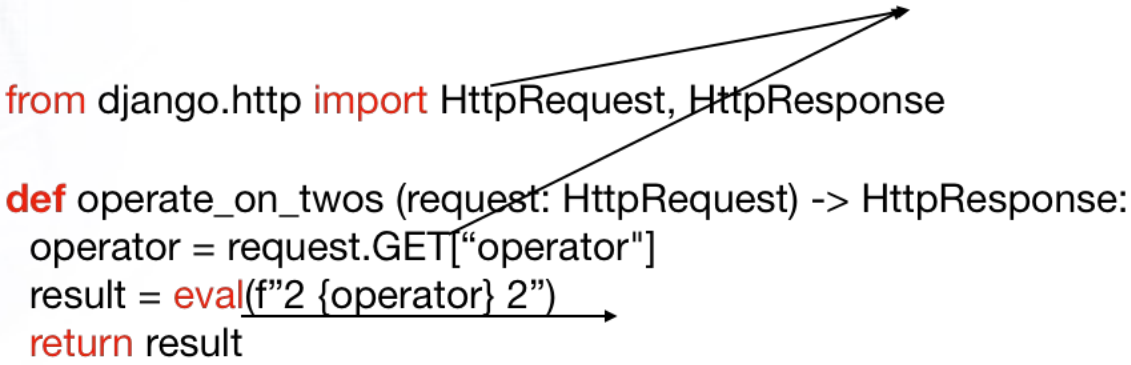
Example 1 from Pysa tutorial
https://github.com/facebook/pyre-check/tree/main/documentation/pysa_tutorial

# Example of Python vulnerable code (Arbitrary code execution)



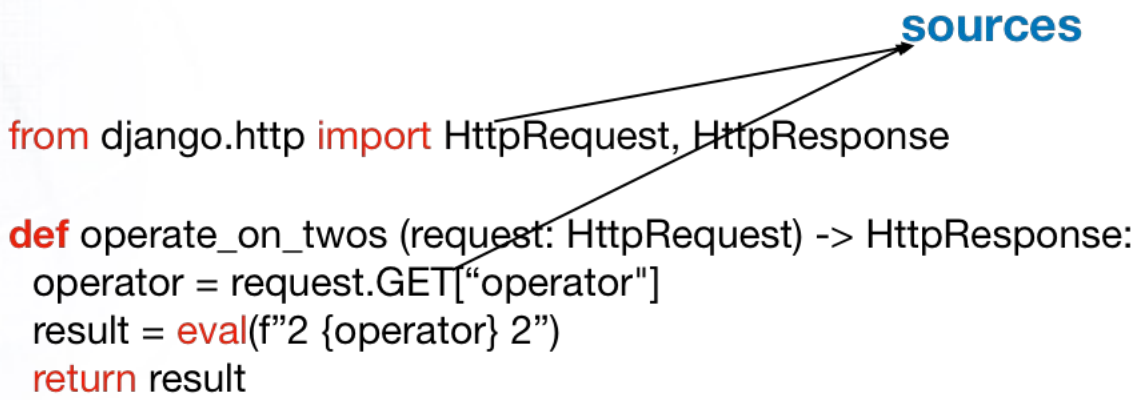

views.py

# Example of Python vulnerable code (Arbitrary code execution)




sources_sinks.pysa

# Example of Python vulnerable code (Arbitrary code execution)




```
django.http.request.HttpRequest.GET: TaintSource[CustomUserControlled] = ...

def eval(__source: TaintSink[CodeExecution], __globals, __locals): ...
```

# Example of Python vulnerable code (Arbitrary code execution)

```
django.http.request.HttpRequest.GET: TaintSource[CustomUserControlled] = ...

def eval(__source: TaintSink[CodeExecution], __globals, __locals): ...
```

**should follow same annotations as typing declarations in Python3**
**See documentation of the typing module**

**Example eval: https://docs.python.org/3/library/functions.html#eval**

**Example django: https://github.com/typeddjango/django-stubs/blob/master/django-stubs/http/request.pyi**

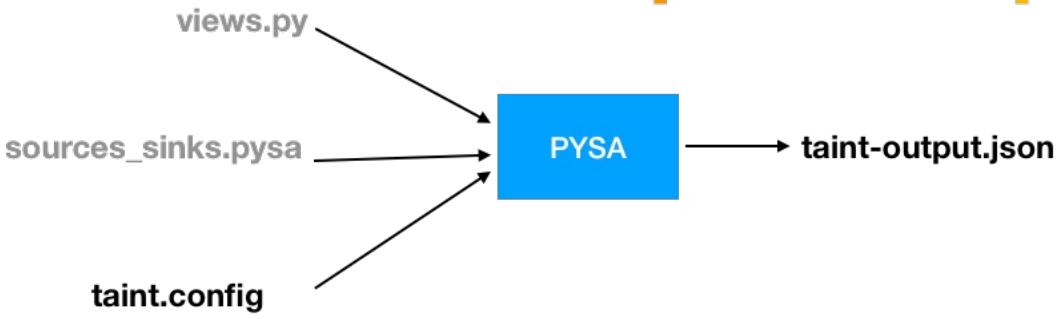
# Example of Python vulnerable code (Arbitrary code execution)

TaintSource[CustomUserControlled]

```
from django.http import HttpRequest, HttpResponse

def operate_on_twos (request: HttpRequest) -> HttpResponse:
  operator = request.GET[“operator"]
  result = eval(f”2 {operator} 2”)
  return result
```

TaintSink[CodeExecution]



Let’s work with the virtual machine and the tutorial: do the first 3 exercises for the TP and analyze the path traversal exercise from lesson 1.

# EXERCISES 1,2,3