SENG 360 - Security Engineering
Web Security - SSRF and Fuzzing

Jens Weber

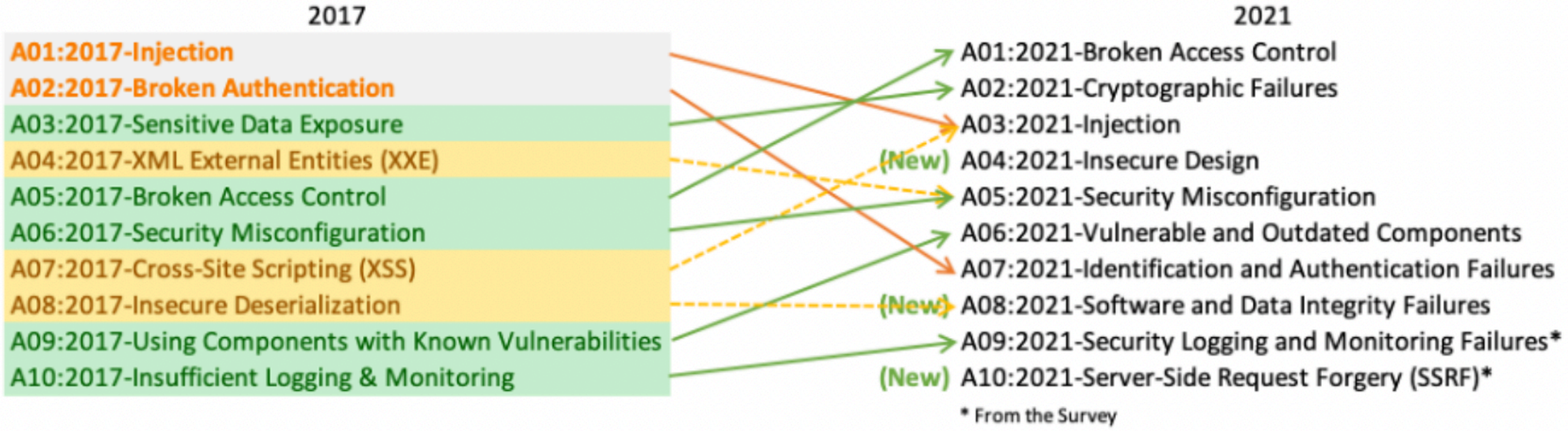
Fall 2021

University of Victoria

Recall from last classes

Top 10 Web Application Security Risks

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.



Learning Objectives

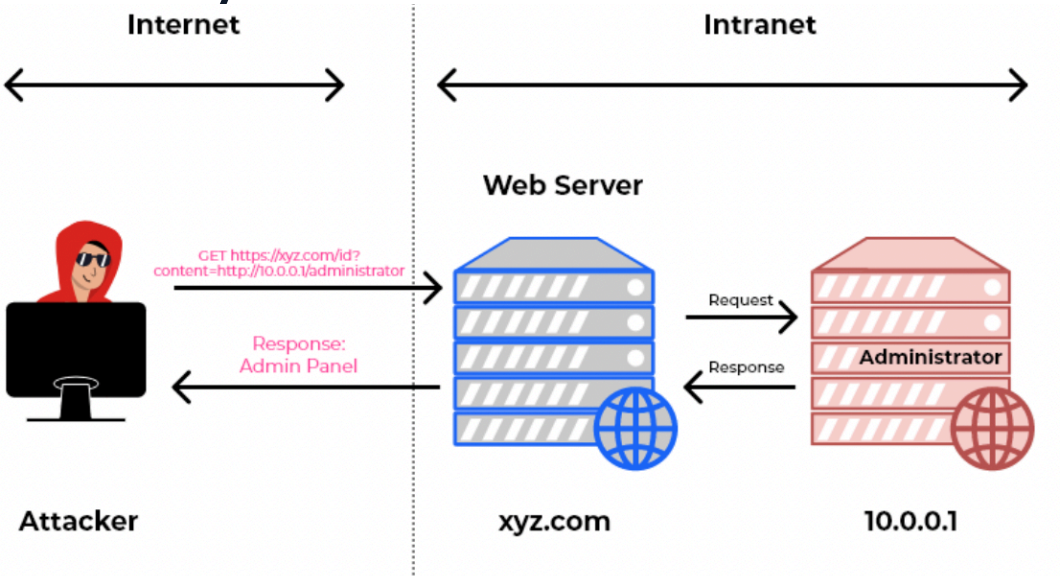
At the end of this class you will be able to

- Describe server-site request forgery vulnerabilities and mitigations
- Describe ways to find security bugs in software, including static analysis and fuzzing

University of Victoria

What is Server-Site Request Forgery (SSRF)?

SSRF is a type of exploit where an attacker abuses the functionality of a server causing it to access or manipulate information in the realm of that server that would otherwise not be directly accessible to the attacker [Wikipedia]



Vulnerable code example: change profile picture

```php
<?php
if (isset($_GET['url'])){
        #Get the new picture
        $url = $_GET['url'];
        $content = file_get_contents($url);
        header("Content-Type: image/png");
        echo $content;

        #
        # user_change_picture($content)
        # ...
        #

        # Redirect to OK
        header("HTTP/1.1 301 Moved Permanently");
        header("Location: /ok.php");
} else {
        echo "Set an URL to change your picture";
}
```

Requests (examples)

As intended:

```
http://myapp.test/change.php?url= https://example.cs.uvic.ca/img/avatar-icon.png
```

Access to an apache functionality of the server (unexposed)

```
http://myapp.test/change.php?url=http://localhost/server-status
```

Access to a web service of the server (unexposed):

```
http://myapp.test/change.php?url=http://localhost:8080
```

Access to a local file:

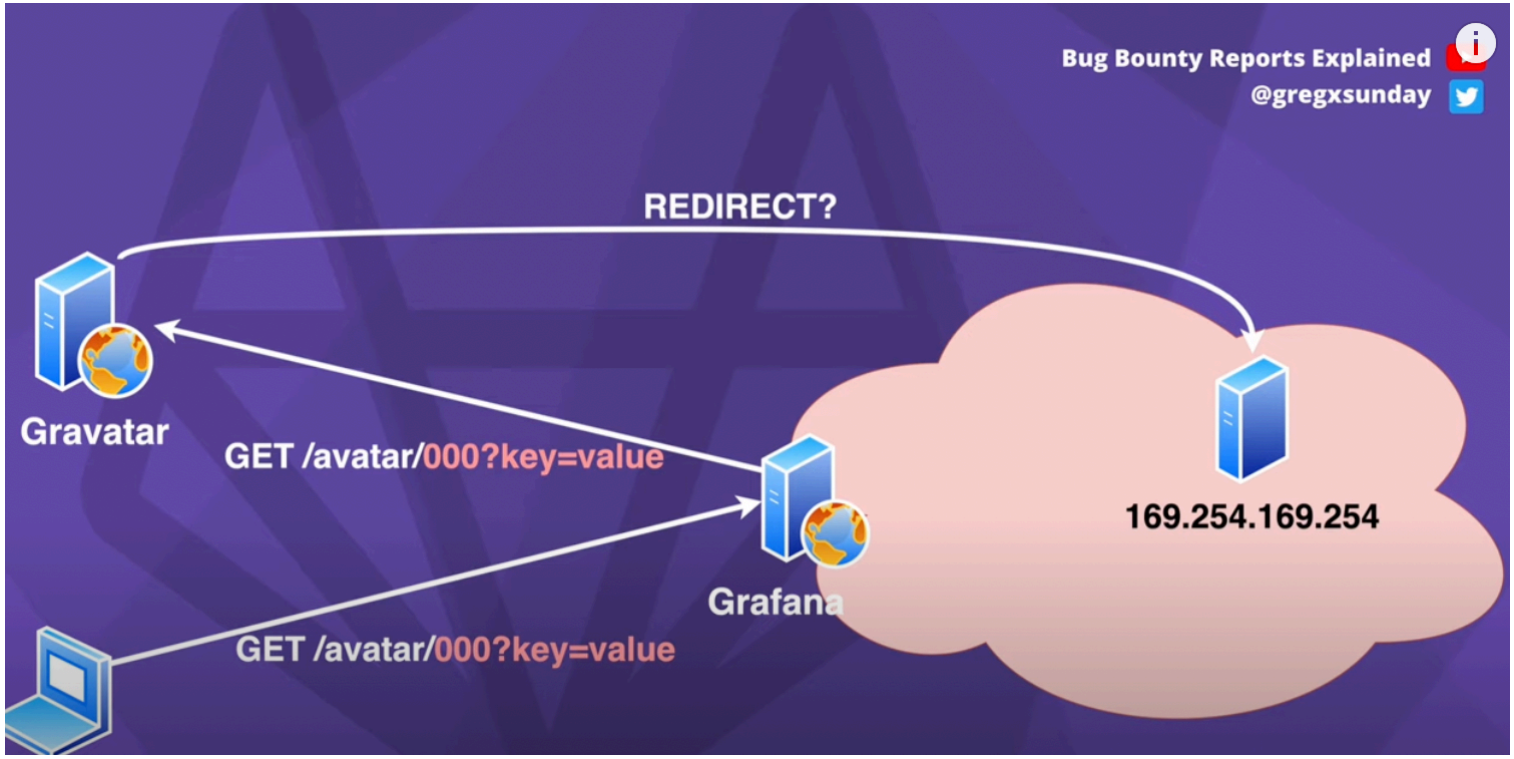
```
http://myapp.test/change.php?url=file:///etc/passwd
```

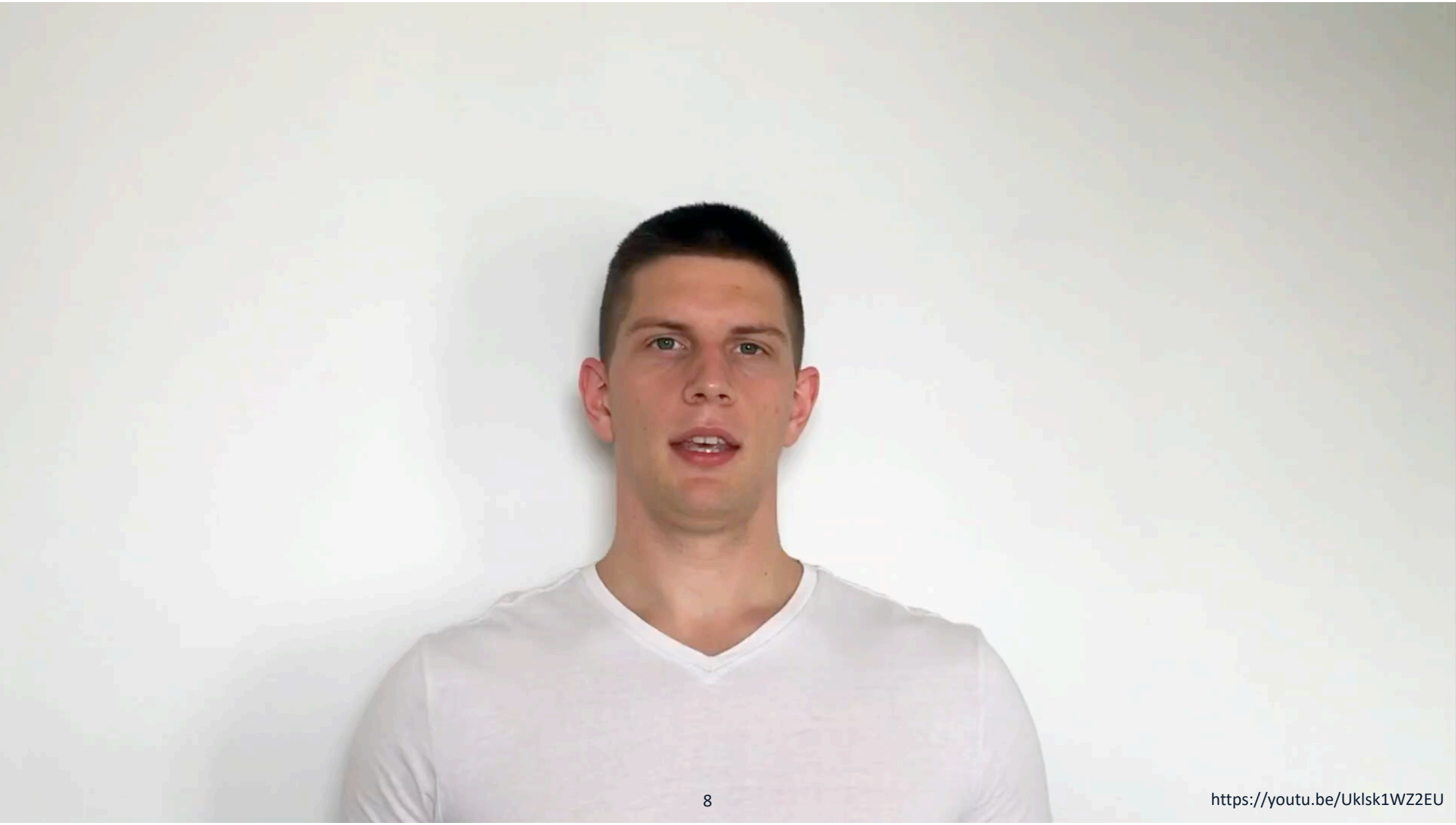
Access to a local file:

```
http://myapp.test/change.php?url=http://10.0.0.1/
```

Real world: SSRF in Gitlab/Grafana

Uses an indirection





https://youtu.be/Uklsk1WZ2EU

Types of SSRF

Content based

- returns content directly

Boolean based

- the answer is different whether resource exists or not

Error based

- error (e.g., HTTP 404) indicates existence

Time based

- timing varies based on existence

University of Victoria


https://youtu.be/ashSoc59z1Y

Mitigations against SSRF

- Validate and sanitize user input
- Whitelist approved domains and protocols
- Validate and sanitize server response
- Enable authentication for services (even if they are available only on local network)
  Example: MongoDB, Redis, ElasticSearch, Memcached

University of Victoria

How to find these Vulnerabilities?

CSRF, XSS, SSRF, SQL Injection, XXE, etc….

- Static analysis

- Dynamic analysis

University of Victoria

Static analysis

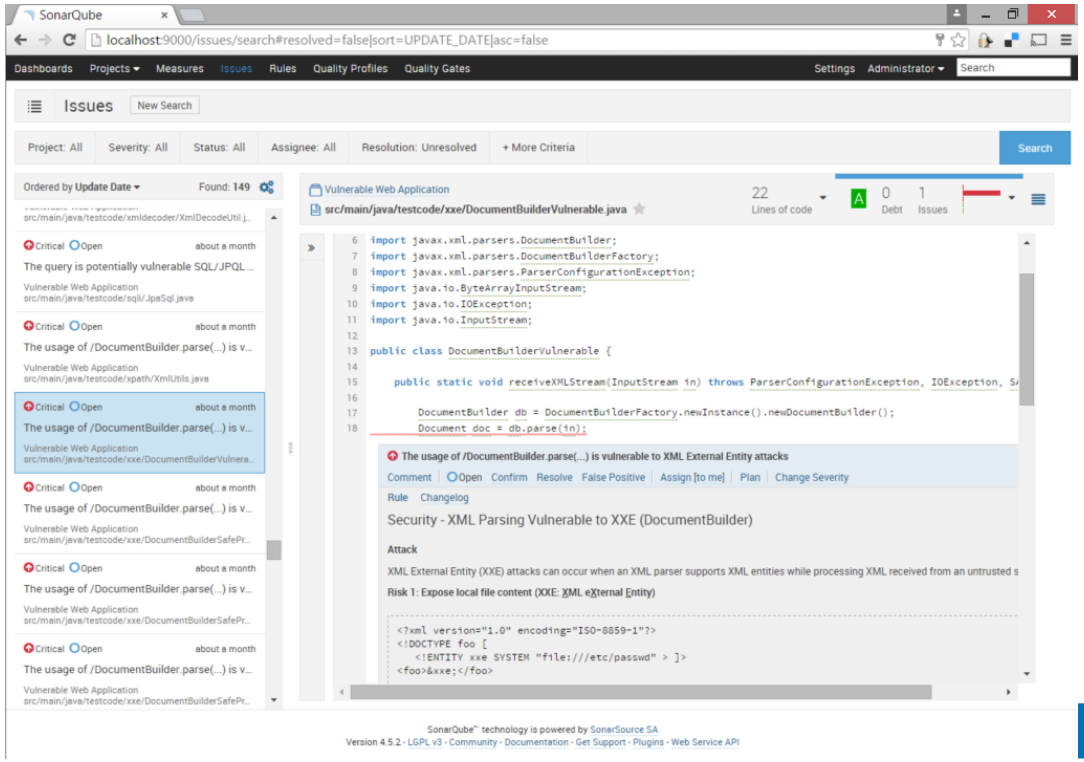
Code analysis without running to program

{} Find Security Bugs

The SpotBugs plugin for security audits of Java web applications.

Download version 1.11.0 | View release notes

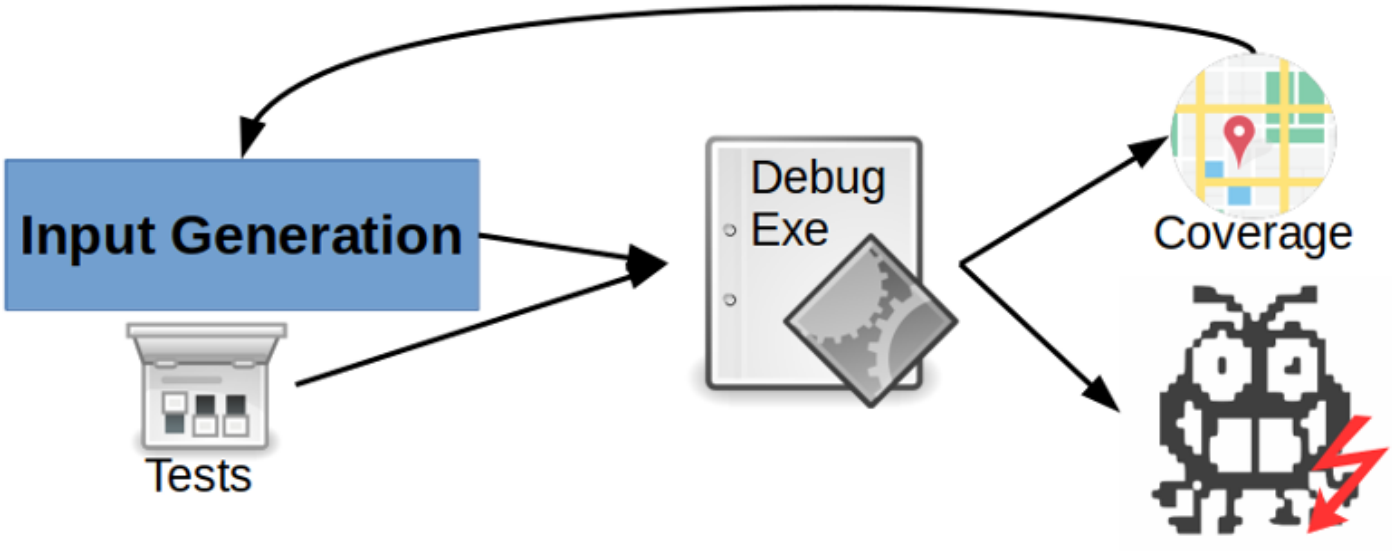
(Last updated: October 29th, 2020)



Challenges:

- a lost of “false positives”
- requires access to source code

Dynamic Analysis: Fuzzing



University of Victoria

https://www.youtube.com/watch?v=rW1iVIKhGj8

Fuzz Testing:
Runtime Bug Hunting

SYNOPSYS®

Fuzzing

- 1st gen tools: mutation fuzzing
- 2nd gen tools: protocol fuzzing
- 3rd gen tools: directed fuzzing (e.g., code coverage)

University of Victoria

Summary and Outlook

- SSRF vulnerabilities expose (unexposed) resources to attackers
- content-based, boolean-based, error-based, time-based
- Find security bugs with static and dynamic code analysis
  - Fuzzers find bugs in running code

Next week: certification and licensing (with guest!)

University of Victoria

Questions?

University of Victoria