Dependency scanning / SAST

Jan Masarik

Agenda

- Dependency scanning (45 min)
  - Intro
  - python safety - hands-on
  - triage of issues found

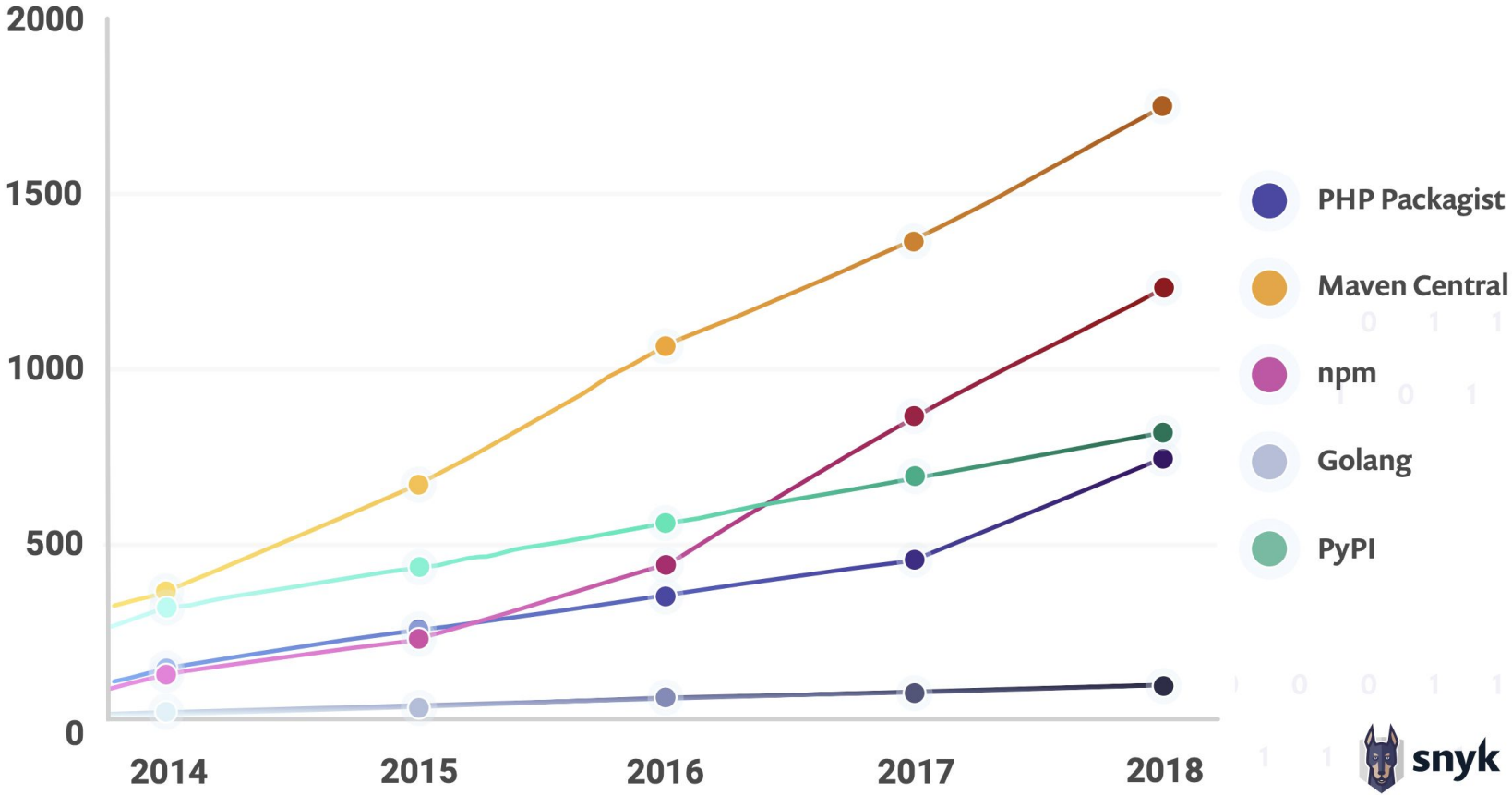
- SAST (35 min)
  - Intro
  - python bandit - hands-on
  - triage of issues found
- HW & Questions (10 min)

Table 3: Characterization of package dependency graphs (without disconnected nodes)

	npm	PyPI
#Nodes	577943	84188
Avg node outdegree	4.27	2.95
Avg dependency tree size	86.55	7.33
Avg dependency tree depth	4.39	1.71

https://www.researchgate.net/publication/331587729_Security_Issues_in_Language-based_Sofware_Ecosystems

New vulnerabilities each year by ecosystem



https://res.cloudinary.com/snyk/image/upload/v1551172581/The-State-Of-Open-Source-Security-Report-2019-Snyk.pdf

Dependency scanning - hands-on

```
git clone https://gitlab.com/janmasarik/kiwi-xssable && cd kiwi-xssable
```

Run python dependency scan (called safety) and sort findings to 3 priority buckets (high priority, medium priority, low/zero priority)

- Via Docker

```
$ cat requirements.txt | \
  docker run -i --rm s14ve/safety safety check --stdin --full-report
```

- Via python package:

```
$ pip install safety
$ safety check --full-report -r requirements.txt
```

Dependency scanning - hands-on - part 2

- Run **safety,** or register on **snyk.io** and run it on a project of your choice.
- Try to understand impact of the vulnerabilities

- Might not work in case requirements are not in pinned (expected) format:

```
bcrypt==3.1.6
cffi==1.12.1
click==7.0
```

- How to pin dependencies in python:

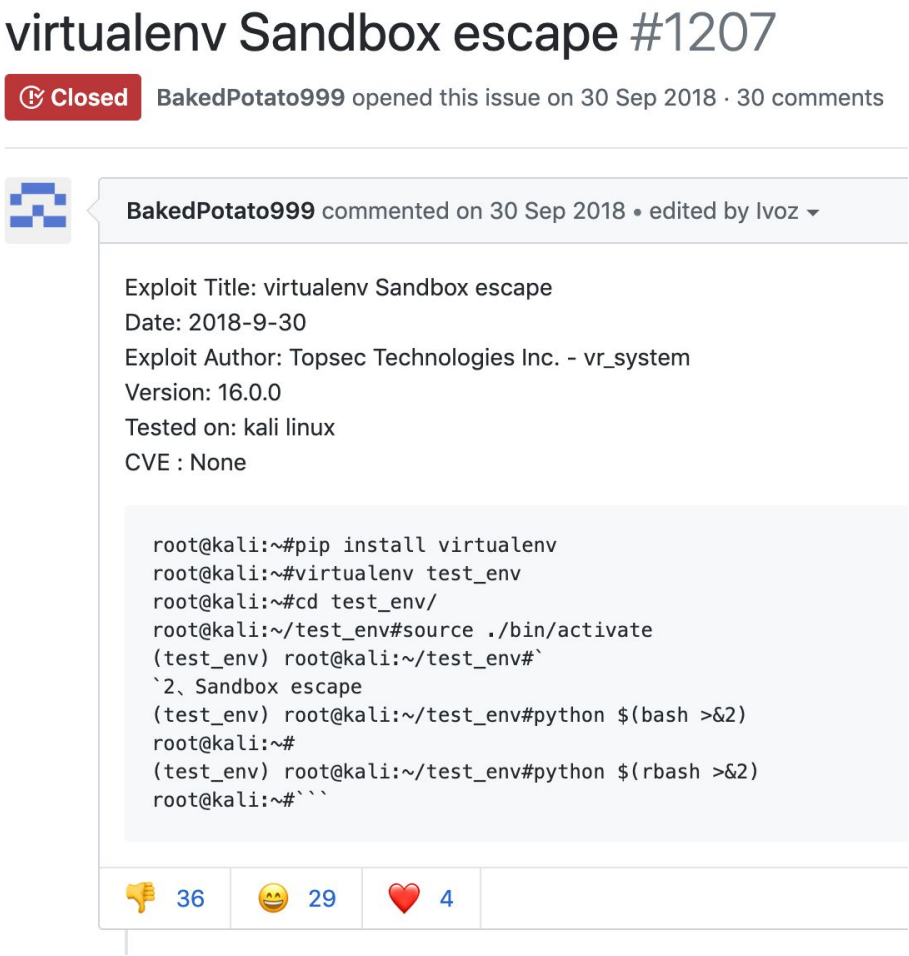
```
$ pip install pip-tools
$ pip-compile --output-file=requirements.txt requirements.in
```

CVE != valid vulnerability

- Can be disputed (CVE-2018-17793)
- CVSS Score: **10.0**

¯_(ツ)_/¯ ¯_(ツ)_/¯
¯_(ツ)_/¯

- virtualenv is a tool to create **isolated Python environments**. virtualenv creates a folder which contains all the necessary executables tc use the packages that a Python project would need.



virtualenv Sandbox escape #1207

Closed BakedPotato999 opened this issue on 30 Sep 2018 · 30 comments

BakedPotato999 commented on 30 Sep 2018 • edited by Ivoz

Exploit Title: virtualenv Sandbox escape
Date: 2018-9-30
Exploit Author: Topsec Technologies Inc. - vr_system
Version: 16.0.0
Tested on: kali linux
CVE : None

```
root@kali:~#pip install virtualenv
root@kali:~#virtualenv test_env
root@kali:~#cd test_env/
root@kali:~/test_env#source ./bin/activate
(test_env) root@kali:~/test_env#`
`2、Sandbox escape
(test_env) root@kali:~/test_env#python $(bash >&2)
root@kali:~#
(test_env) root@kali:~/test_env#python $(rbash >&2)
root@kali:~#```
```

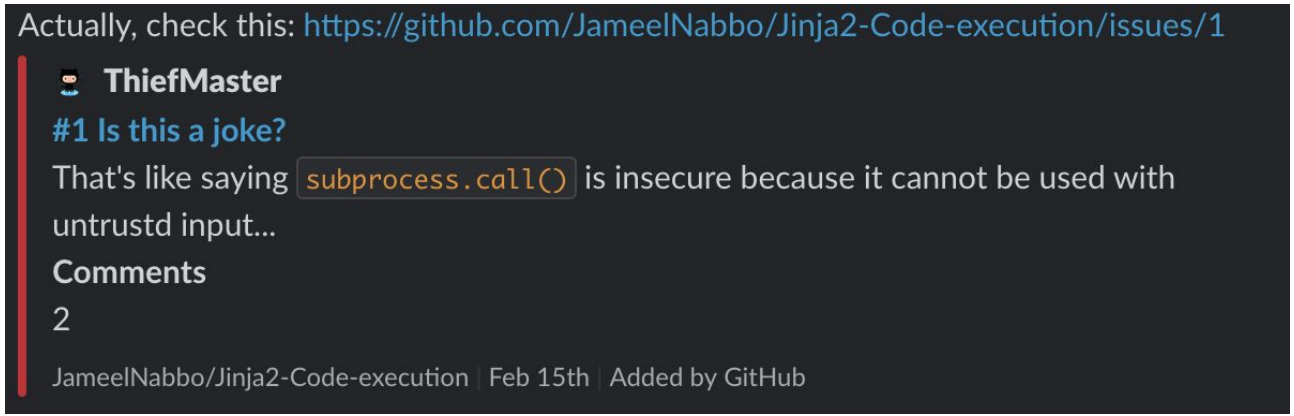
👎 36 😄 29 ❤️ 4

CVE != valid vulnerability vol. 2

- CVE-2019-8341

** DISPUTED ** An issue was discovered in Jinja2 2.10. The from_string function is prone to Server Side Template Injection (SSTI) where it takes the "source" parameter as a template object, renders it, and then returns it. The attacker can exploit it with {{INJECTION COMMANDS}} in a URI. NOTE: The maintainer and multiple third parties believe that this vulnerability isn't valid because users shouldn't use untrusted templates without sandboxing.

- “Exploit” repo got deleted shortly after the guy realized his mistake. Thumbnail:



Dependency scanning - lessons learned

- Always check the reliability of the dependency vulnerability database
  - Every tool might have different DB
- Don’t blindly trust the findings
  - Triage them and check if those issues are valid for your implementation/stack
  - Try to understand the root cause of the issue
- Existing CVE does not always imply that a vulnerability is valid
  - Check for disputed CVEs and understand the impact

SAST - hands-on

```
git clone https://gitlab.com/janmasarik/kiwi-xssable && cd kiwi-xssable
```

Run python SAST

- Semgrep via docker (**recommended** - has more rules and is language agnostic)

  ```
  $ docker run -v "/path/with/kiwi-xssable/:/src" returntocorp/semgrep
  --config "p/r2c-security-audit"
  ```

  - Replace the </path/to/...> according to your machine
  - v mounts the volume inside the docker container
  - Feel free to install via https://semgrep.dev/docs/getting-started/#run-semgrep-locally
- Bandit via Docker:

  ```
  $ docker run -v </path/to/directory/with/kiwi-xssable>:/src/ s14ve/bandit
  ```
- Bandit via python package:

  ```
  $ pip install bandit && bandit -r /path/to/directory/with/kiwi-xssable /
  ```

SAST - hands-on

1. Triage bandit results and split issues to 3 categories:
   - Fix ASAP
   - Fix *some day*
   - False positive / best practice / accepted risk

2. Run bandit/semgrep on a project of your choice and check results

SAST - lessons learned

- Run SAST in the same version of language as the project
  - You may not be able to run python 3.7 code in python 2.7 and vice versa

- Think while triaging issues
  - If you have trusted input (e.g. source is directly from repository), even *unsafe* functions might be safe

Homework - setup

- Register to hackerone - https://hackerone.com/users/sign_up
  - Afterwards, go to https://ctf.hacker101.com -> Login

- Join MUNI group by clicking on invite link: **https://ctf.hacker101.com/group/join?invite=54ff0f11211d6aaf9c06182ac192d366e87c30ae225a5b354d22ea21a993adb8** (if the link is expired, please ping me at 433634@mail.muni.cz)

- Invite link expires in 48 hours. Register and click on it **now**.

Homework - task

- Get 26 points (1 invite to private program) - 50% of HW points
  - Choose challenges based on your skill level
  - *Easy* difficulty is relative, so don't get discouraged and try harder
  - You can use hints, but at least *try* without them
  - Check out videos from Hackerone 101 University in case you have no idea where to start

- Write a short report (1 page) - 50% of HW points
  - Format can be txt/md/pdf
  - Put as many images as you want
  - You can write even about failed paths
  - **Include your hackerone username in report**

- Deadline as usual
  - Bonus points possible in case of 52+ points or exceptional report