Security Certification of Third-Parties Applications

Stanislav Dashevskyi
dashevskyi@fbk.eu

Advisors: Fabio Massacci, Antonino Sabetta

SECENTIS
A European Industrial Doctorate on Security and Trust

Agenda

- Introduction
  - Third-party code in web applications
    - Third-party JavaScript
- The problem
- What was done so far
- Conclusions

Third-party components in modern software

- Software is getting more complex, and developers tend to reuse the work of their colleagues

- This helps to build a strong community over a technology and save development resources

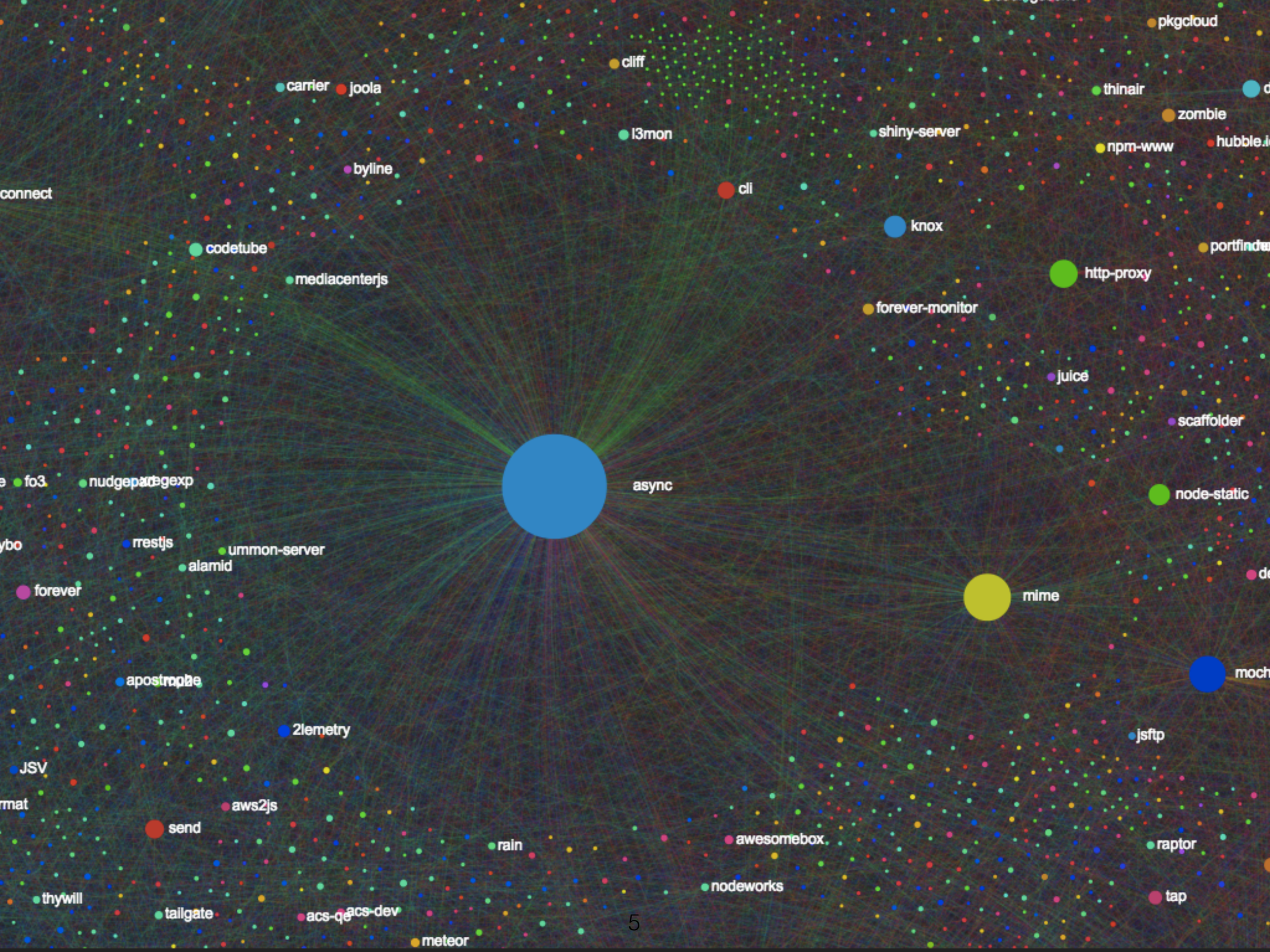
- It makes it possible to create complex software that is able to solve real-world problems

- Every software module used can have bugs of security vulnerabilities that influence the whole product

- Quality of the product becomes a shared value

Third-party components in web applications

- **We consider web applications that are powered by JavaScript**
  - Dynamic and interpreted language
  - Allows performance benefits and homogeneous programming experience
  - Enables reflection and metaprogramming at ease
- **JavaScript web applications use numerous third-party libraries both on client and server**
  - Third-party client script used on the owner's website, but served from a remote source
  - A third-party library for server-side JavaScript, such as those used from Node.js official package registry


pkgcloud
cliff
carrier
joola
thinair
zombie
l3mon
shiny-server
npm-www
byline
cli
connect
knox
codetube
portfinder
http-proxy
mediacenterjs
forever-monitor
juice
scaffolder
async
node-static
fo3
nudgepad
restjs
ummon-server
alamid
mime
forever
apostrophe
2lemetry
jsftp
JSV
aws2js
send
awesomebox
raptor
rain
nodeworks
thywill
tap
tailgate
acs-qe
acs-dev
meteor

Agenda

- Introduction
  - Third-party code in web applications
    - Third-party JavaScript
- The problem
- What was done so far
- Conclusions

The problem: third-party JavaScript

- **Developers often do not know neither the full set of the libraries they use, nor their exact versions**
  - **Their products can be vulnerable for years!**
  - **If one module is vulnerable → the product becomes vulnerable**
- **JavaScript is hard to get right and to analyze automatically**
  - **Dynamic code generation and execution**
  - **Variable/Function Aliasing, Scoping**
  - **Dynamic type systems and various Inheritance mechanisms**
  - **Obfuscation mechanisms → evasion techniques**

The problem: third-party JavaScript (EXAMPLE II)

```
function execute(code) {
    return eval(code)
}
```

```
eval(function(p,a,c,k,e,d){e=function(c){return c};if(!''.replace(/^/,String))
{while(c--){d[c]=k[c]||c}k=[function(e){return d[e]}];e=function(){return'\\w+'};c=1};
while(c--){if(k[c]){p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}
('1 2(0){3 4(0)}',5,5,'code|function|execute|return|eval'.split('|'),0,{}))
```

The problem: third-party JavaScript (CONTINUED)

- **Classical static analysis approaches are insufficient to for finding all possible classes of vulnerabilities in large codebases**
  - **Static analysis must be guided by code annotations, runtime information, or other mechanisms**
  - **Mostly static approaches are able to detect a limited set of vulnerability classes**
  - **Soundness is often sacrificed for the sake of not overwhelming an analyst with false alarms**
- **However, dynamic analysis is too expensive for large codebases**

Agenda

- Introduction
  - Third-party code in web applications
    - Third-party JavaScript
- The problem
- What was done so far
- Conclusions

What was done so far

- **Software developers need a tool support for scanning full codebase of their applications**
  - **All of the “problematic” JavaScript features must be considered**
  - **It must be able to analyze large codebases**
  - **All classes of vulnerabilities must be handled (write rules for finding instances of a certain class)**
- **We have created TestREx*****
  - **To understand the behaviour of vulnerable JavaScript code**
  - **To have a reliable environment for collecting benchmarks and assessing JavaScript analysis approaches**

What is TestREx?

- **Management system for software environments**
  - Provides an isolated playground for every application and its corresponding software environment
- **Testbed for performing web application vulnerability experiments**
  - Run scripted exploits automatically
  - Give testers the access to a sandboxed application and let them play
- **Test suite for managing and running scripted exploits against the corresponding applications**

Motivation for TestREx

- **Systematic collection of exploits into a knowledge base**
  - Exploit DB, WebGoat, etc.
- **Advantages for developers of exploited software**
  - Provide evidence on actual risks of vulnerabilities
  - Study explicit/implicit causes of vulnerabilities, their connections
  - Insight for software analysis tools and verification approaches
- **What about developers using that software?**

Third-party developers' perspective

- **A vulnerability was reported…**
- **How do I actually "repeat" an exploit in my operational environment?**
  - Applications run on different platforms → SQL injection for MySQL will not work on MongoDB
  - Software changes → exploit works only if run in a certain software environment
  - Essentially, it is a "non constructive existence proof"

Exploits (TestREx view)

- **A sequence of [automated] actions required to subvert a vulnerability in and application and verify that subversion was successful**

- Self-contained unit test + metadata

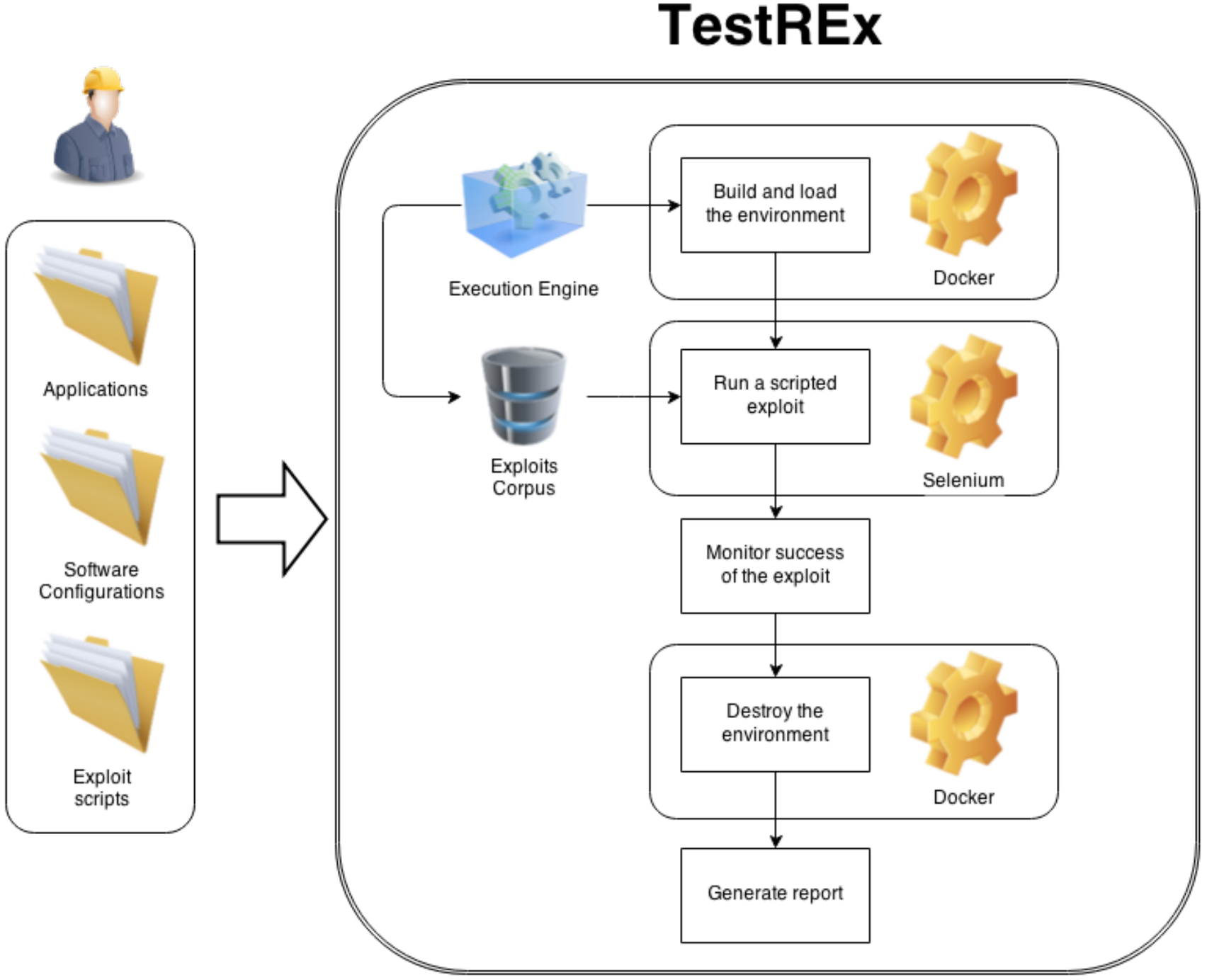
- Python scripts that use Selenium to automate browser and simulate attacker’s actions

- Scripts are controlled by Execution Engine of TestREx

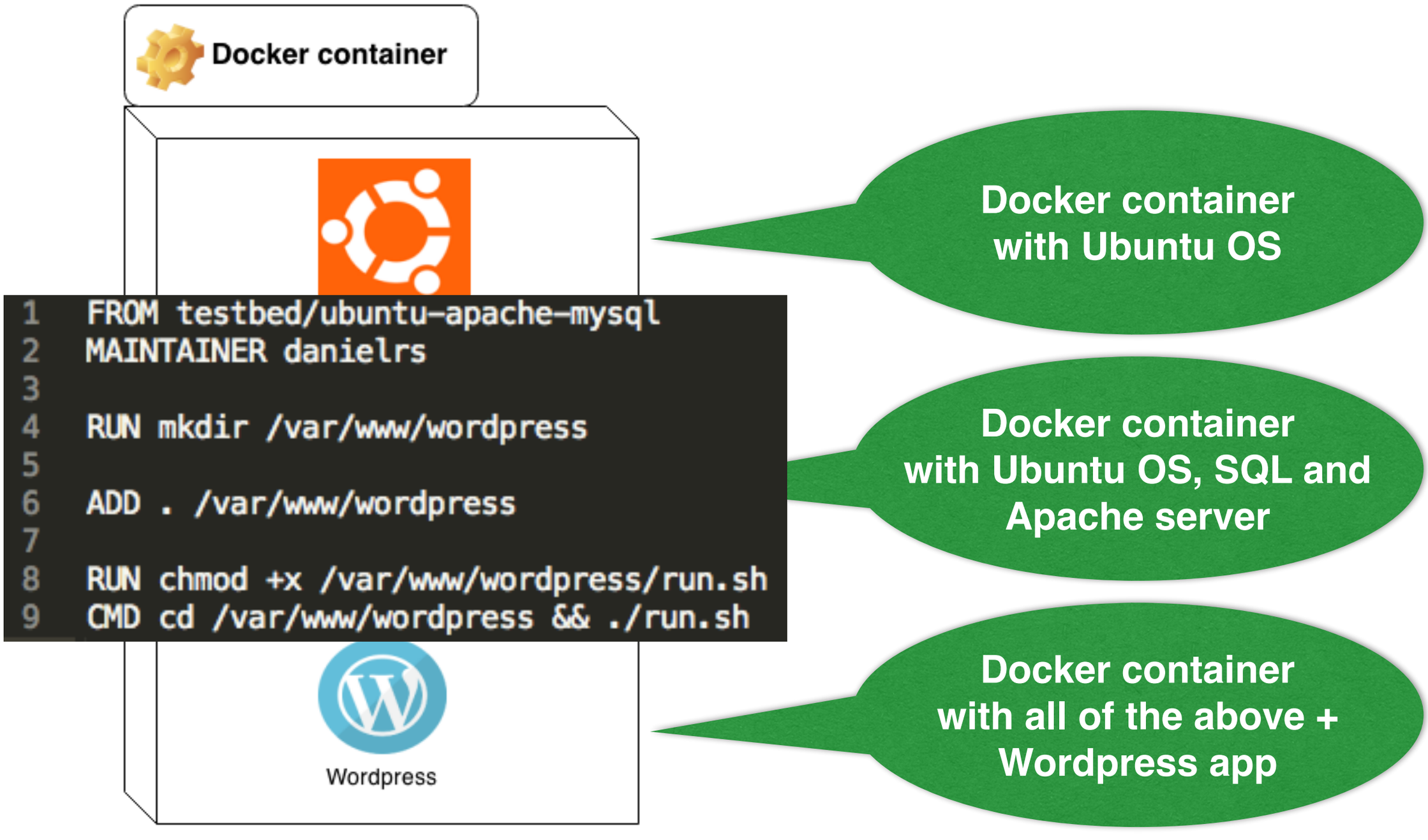
TestREx usage model

- **Executable documentation for software companies**
  - "Document an exploit" == "write a TestREx script"
  - Automated security + configuration + version testing
  - Automated regression testing suite
  - Penetration testing support
- **Aid for security-unaware developers**
  - Part of a training toolkit for studying web application security
  - Benchmark/supporting tool for code analysis approaches evaluation

TestREx: workflow



How sandboxes are implemented?




```
1   FROM testbed/ubuntu-apache-mysql
2   MAINTAINER danielrs
3
4   RUN mkdir /var/www/wordpress
5
6   ADD . /var/www/wordpress
7
8   RUN chmod +x /var/www/wordpress/run.sh
9   CMD cd /var/www/wordpress && ./run.sh
```

Running an experiment

- **Modular ways to run exploits and applications**
  - All exploits are independent and can be supplied by anyone
  - An application can be started either in “clean” or “infected” state
- **Sample scenarios → regression testing and configuration testing**
  - Deploy multiple versions of an application and understand what was fixed through the version history
  - Deploy an application in different platforms and see the correlation between third-party software and vulnerabilities
- **Report generation**
  - A .csv file with exploit run results and exploit metadata

Agenda

- Introduction
  - Third-party code in web applications
    - Third-party JavaScript
- The problem
- What was done so far
- Conclusions

Current work on TestREx

- **We have engaged students in the "Offensive Technologies" course at UNITN**
  - Extension of the exploits/vulnerabilities corpus
  - Implementation of a number of attack scenarios and countermeasures for server-side JavaScript
  - Usage of TestREx as a part of a toolchain for scanning Node.js
  - Semi-automatic exploit generation with MITM proxy

Getting TestREx

- **GitHub:**
  - https://github.com/standash/TestREx
- **DISI Security Lab:**
  - https://securitylab.disi.unitn.it/doku.php?id=malware_analysis
- **Corresponding publication:**
  - ***Dashevskyi, S., Dos Santos, D. R., Massacci, F., & Sabetta, A. (2014, August). TESTREX: a testbed for repeatable exploits. In Proceedings of the 7th USENIX conference on Cyber Security Experimentation and Test (pp. 1-1). USENIX Association.
- **Note: it's free for experiments, however it is under the pending patent from SAP Labs**

Thank you!