Stanislav Dashevskyi, Daniel Ricardo Dos Santos, Fabio Massacci, Antonino Sabbat

TestREx: A Testbed for Repeatable Exploits

UNIVERSITY OF TRENTO - Italy

FONDAZIONE BRUNO KESSLER

Agenda

1. The motivation
2. Empirical security research and software development
3. Getting more information out of the vulnerability corpus
4. What is TestREx?
5. The Exploits
6. Demo of a run
7. TestREx applications
8. Lessons learned
9. Conclusion

The motivation

- Our current research is focused on JavaScript code analysis
- We need the all-purpose framework that will enable us to quickly deploy all kinds of web applications
  - *Observe dynamic behavior of the code*
  - *Get better understanding on how security vulnerabilities are mapped to the code*
  - *Find a better way to assess the amount of false positives and false negatives in our code analysis tool*

Empirical security research and software development

- Systematic collection of exploits into a knowledge base
  - *Study explicit/implicit causes of vulnerabilities, their connections*
  - *Collect evidence on risks that vulnerabilities might pose*
  - *Get insight for software analysis tools and testing approaches*
  - *Lower the probability of making the same mistakes*
- But... having yet another corpus of exploits doesn't scale
  - *Software is evolving → certain exploits work for certain versions*
  - *Sofware configuration does matter → applications often support multiple platforms → one can't execute SQL injection on MongoDB*

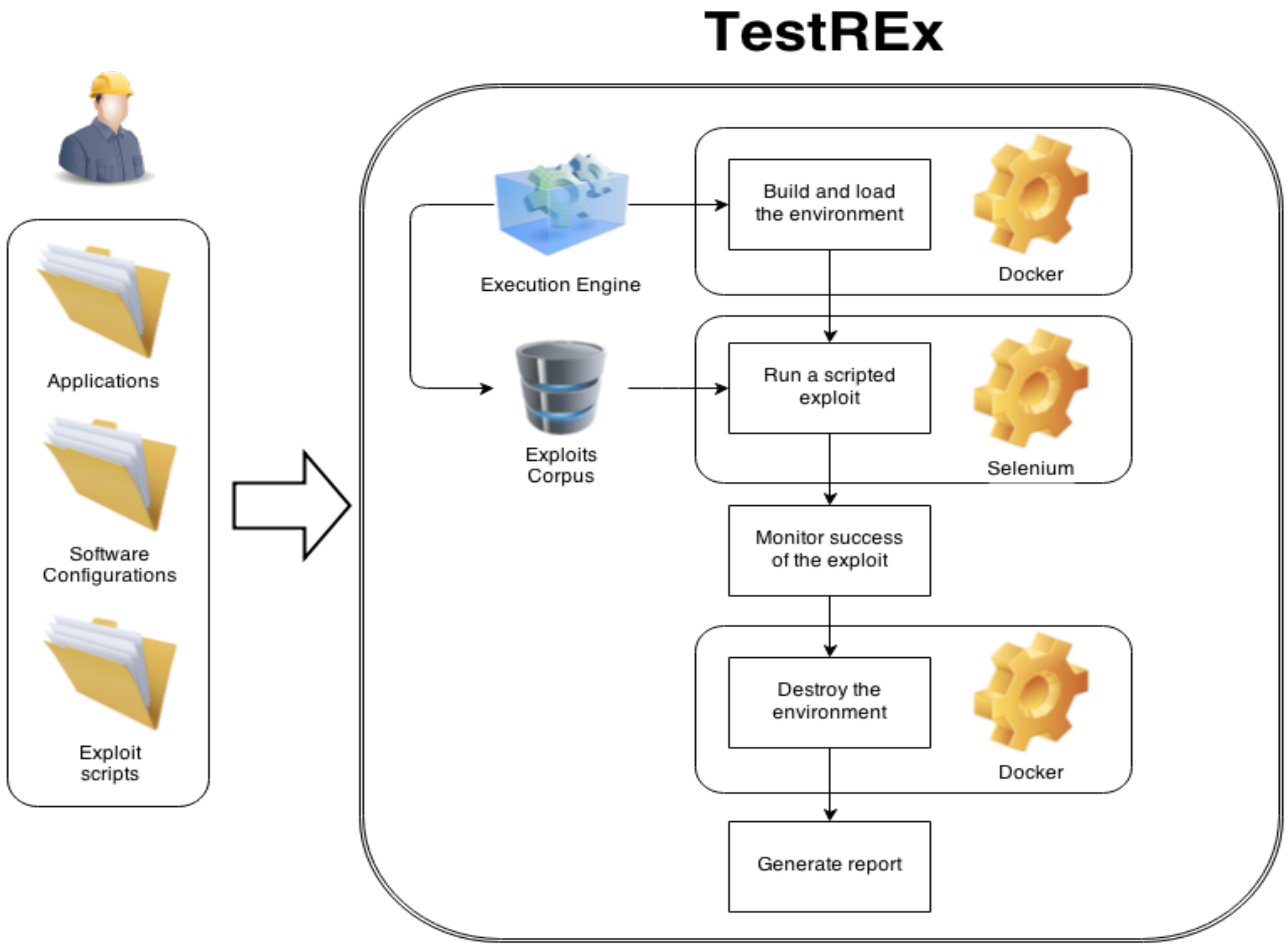
Getting more information out of the corpus

- Apart from "documenting" an exploit, what other information can be inferred?
- Given an environment E, and an exploit X that successfully subverts an application A that is running on E
  - *Will X be successful on the application A running on a new environment E'?*
  - *Will X be successful on a new version of A, A', running on E?*
  - *Will X be successful on a new version of A, A', running on E'?*
- Deploying and matching all possible software configurations and application versions might be very hard and tedious

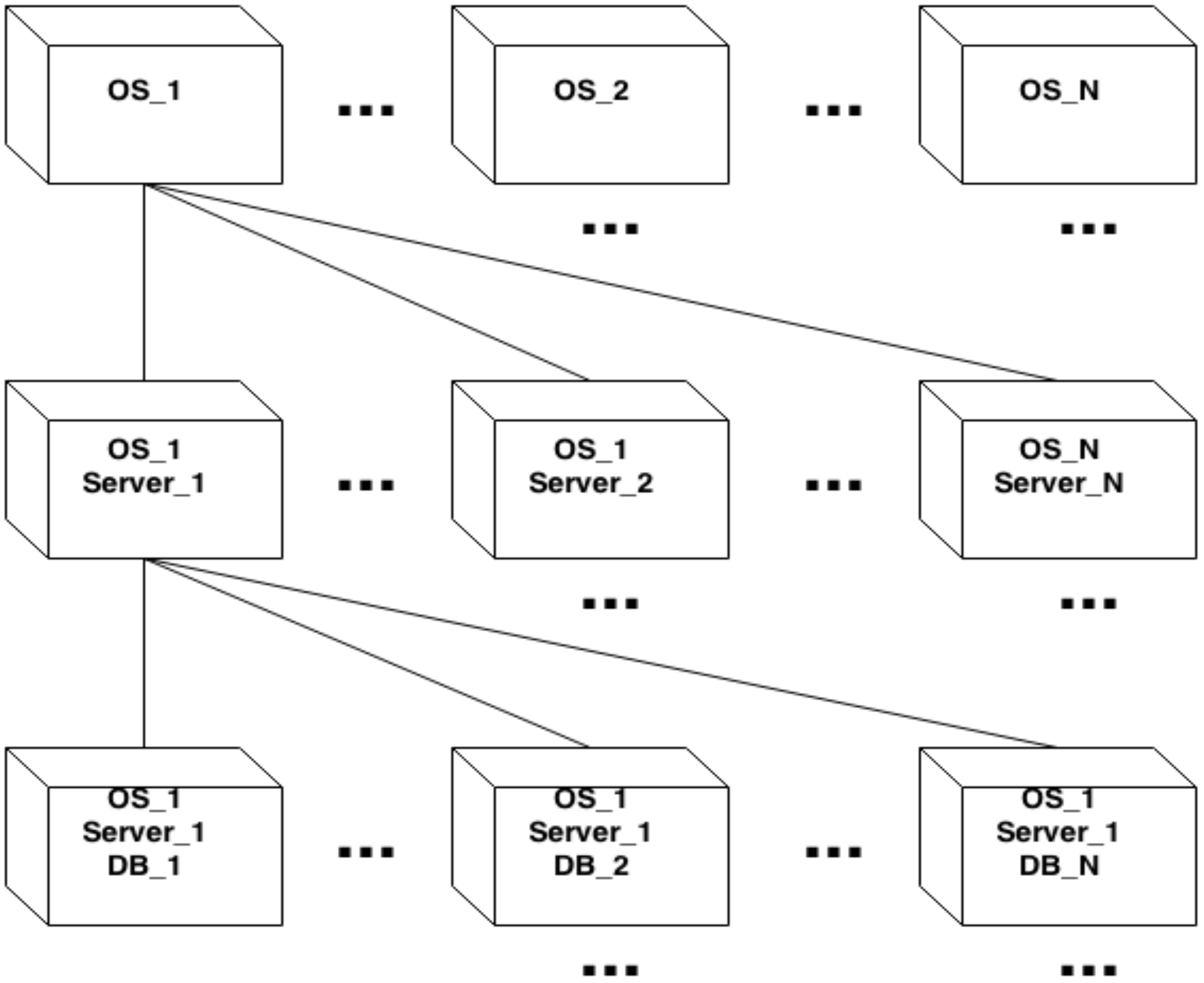
What is TestREx?

- A management system for software environments
  - *We are able to provide an isolated sandbox per every application version and its corresponding software environment*
- A testbed for performing web application vulnerability experimentations
  - *Automatically, via scripted exploits*
  - *Manually, by giving testers the access to the requested application from within its sandbox*
  - *An application can be started in either "clean" or "spoiled" state*
- A test suite for managing and running scripted exploits against corresponding applications

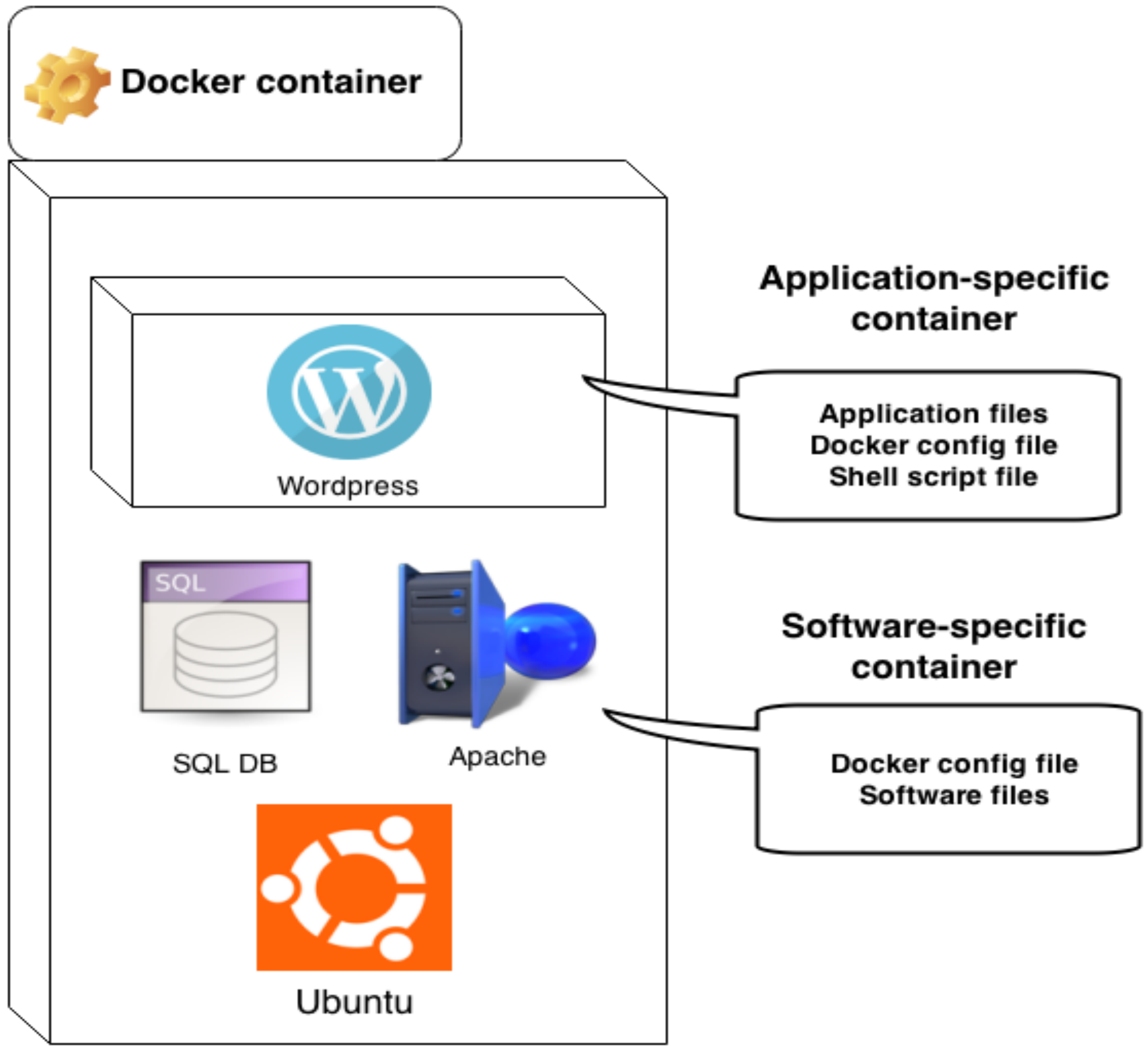
TestREx: The workflow



TestREx: Software Containers hierarchy



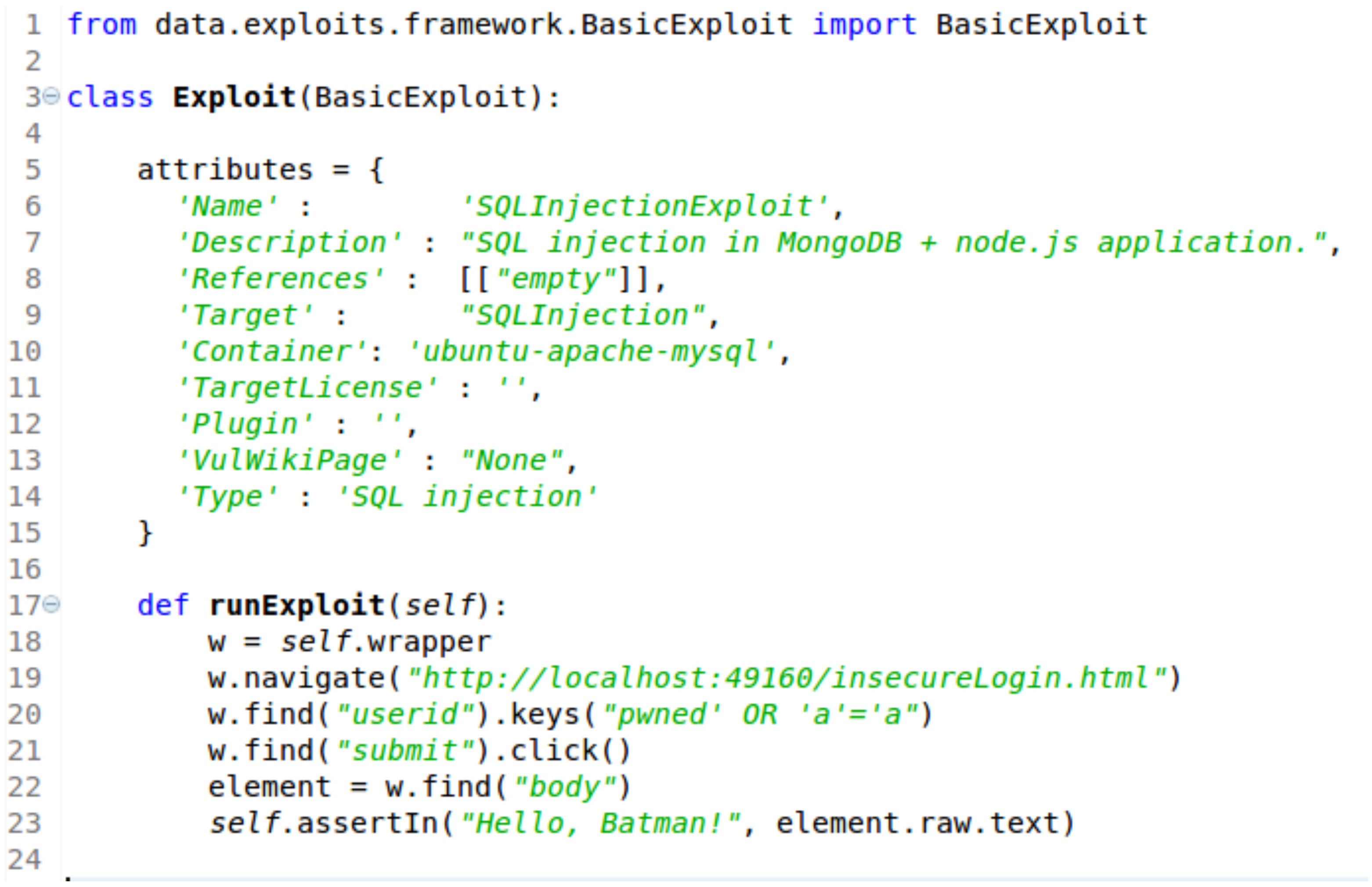
TestREx: Application Container example



The Exploits

- **By exploits we mean sets of [automated] actions required to subvert a vulnerability in an application and verify the success**
- **What is an exploit technically**
  - ***A self-contained unit test that has description metadata***
  - ***A Python script that uses Selenium driver to automate the browser***
  - ***The script passes the results of its run to the Execution Engine***
- **Why Selenium?**
  - ***So far we are dealing only with browser-based attacks***
  - ***We are able to simulate attacker's behavior with a browser***
  - ***Native JavaScript support***

Exploit example

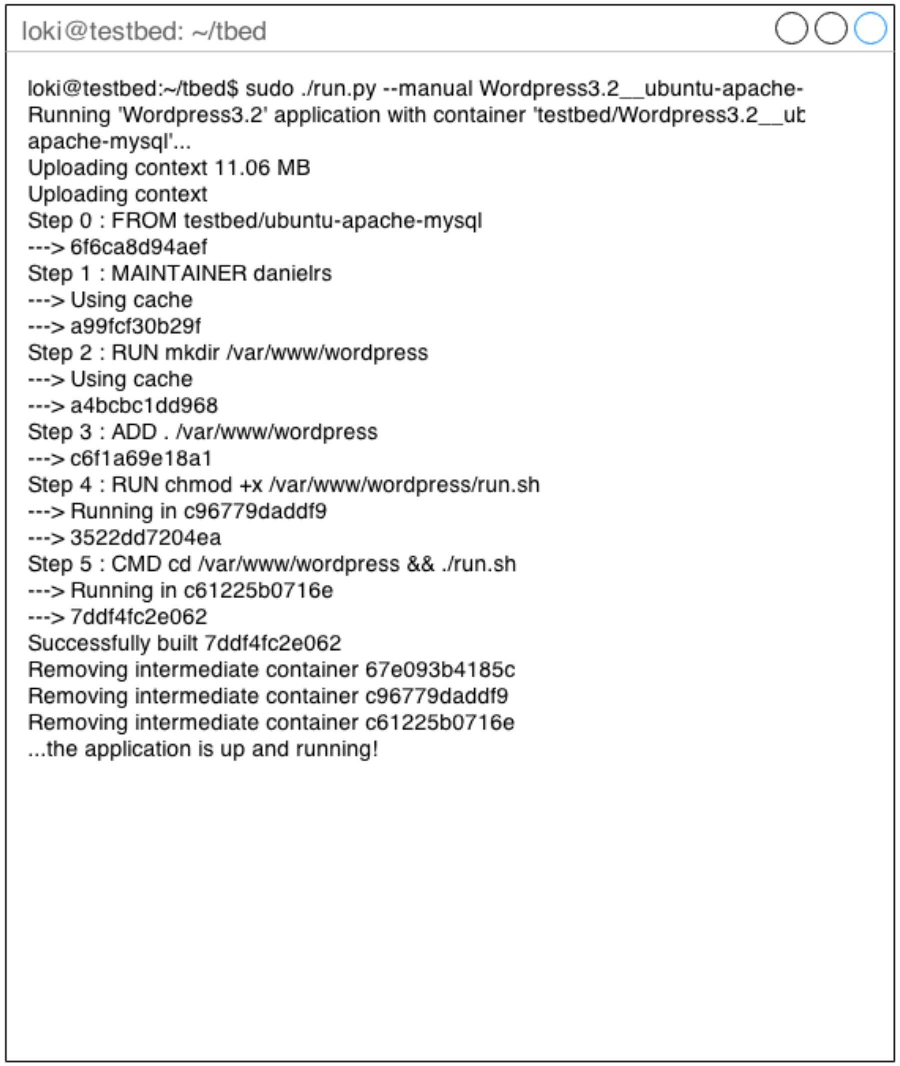


```python
from data.exploits.framework.BasicExploit import BasicExploit

class Exploit(BasicExploit):

    attributes = {
       'Name' :          'SQLInjectionExploit',
       'Description' : "SQL injection in MongoDB + node.js application.",
       'References' :  [["empty"]],
       'Target' :       "SQLInjection",
       'Container': 'ubuntu-apache-mysql',
       'TargetLicense' : '',
       'Plugin' : '',
       'VulWikiPage' : "None",
       'Type' : 'SQL injection'
    }

    def runExploit(self):
        w = self.wrapper
        w.navigate("http://localhost:49160/insecureLogin.html")
        w.find("userid").keys("pwned' OR 'a'='a")
        w.find("submit").click()
        element = w.find("body")
        self.assertIn("Hello, Batman!", element.raw.text)
```

Experimentation capabilities

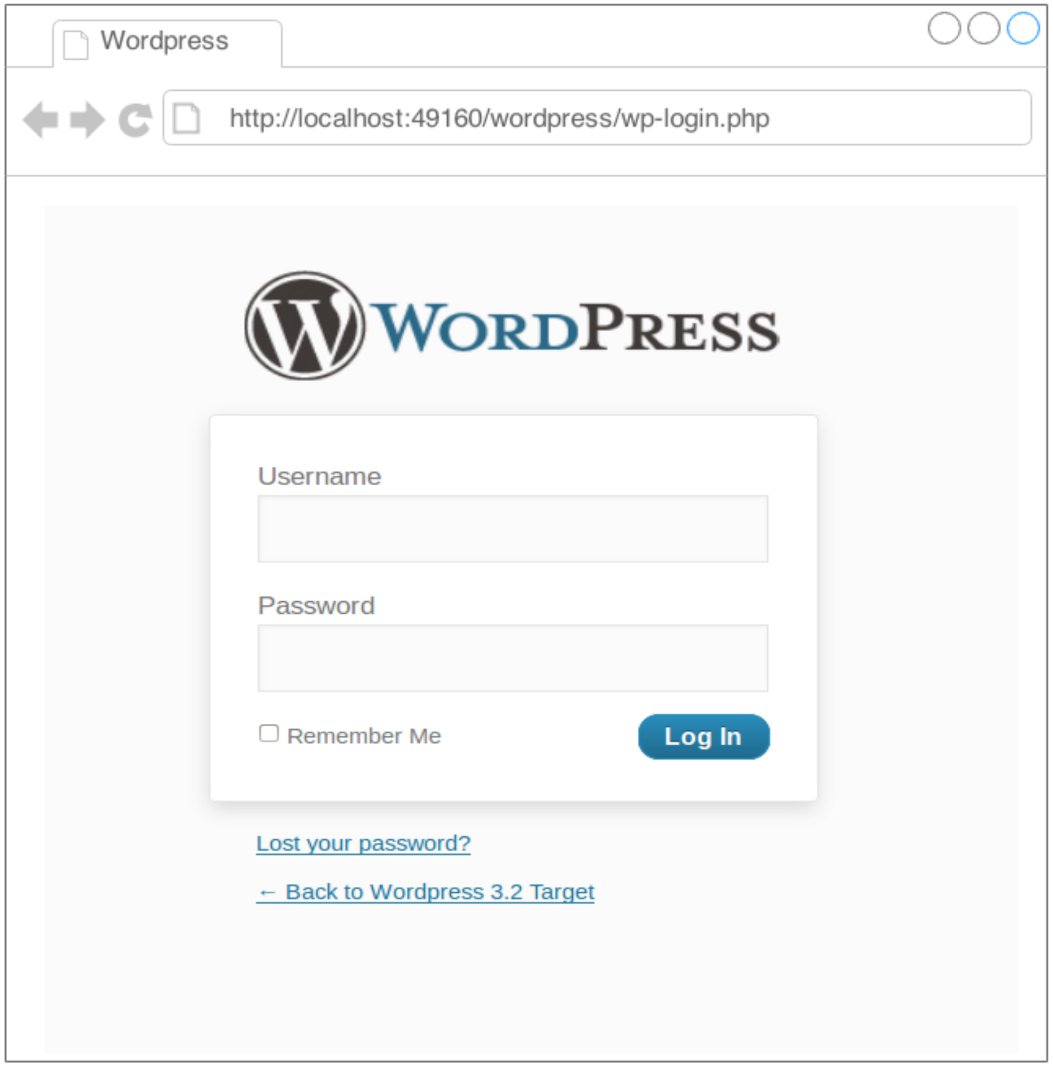
- Flexible way to run exploits and applications
  - *Manual, single and batch runs*
  - *All exploits are independent scripts that can be supplied at any time by anyone*
- Report generation
  - *A .csv file with exploit run results and exploit metadata*
- Regression testing and configuration testing
  - *Deploy multiple versions of an application and understand what was fixed though the whole version history*
  - *Deploy an application on different platforms and see the corellation between third-party software and vulnerabilities*

Demo: Manual run



loki@testbed: ~/tbed

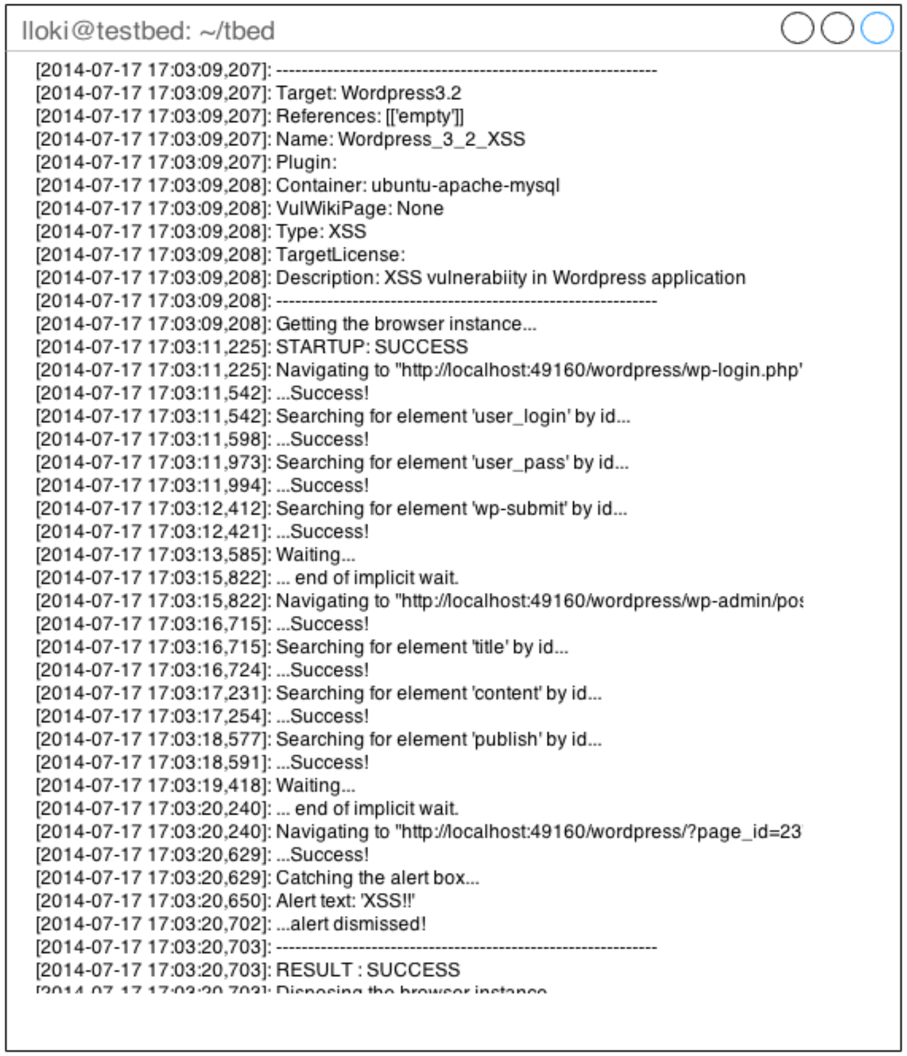
```
loki@testbed:~/tbed$ sudo ./run.py --manual Wordpress3.2__ubuntu-apache-
Running 'Wordpress3.2' application with container 'testbed/Wordpress3.2__ub
apache-mysql'...
Uploading context 11.06 MB
Uploading context
Step 0 : FROM testbed/ubuntu-apache-mysql
---> 6f6ca8d94aef
Step 1 : MAINTAINER danielrs
---> Using cache
---> a99fcf30b29f
Step 2 : RUN mkdir /var/www/wordpress
---> Using cache
---> a4bcbc1dd968
Step 3 : ADD . /var/www/wordpress
---> c6f1a69e18a1
Step 4 : RUN chmod +x /var/www/wordpress/run.sh
---> Running in c96779daddf9
---> 3522dd7204ea
Step 5 : CMD cd /var/www/wordpress && ./run.sh
---> Running in c61225b0716e
---> 7ddf4fc2e062
Successfully built 7ddf4fc2e062
Removing intermediate container 67e093b4185c
Removing intermediate container c96779daddf9
Removing intermediate container c61225b0716e
...the application is up and running!
```



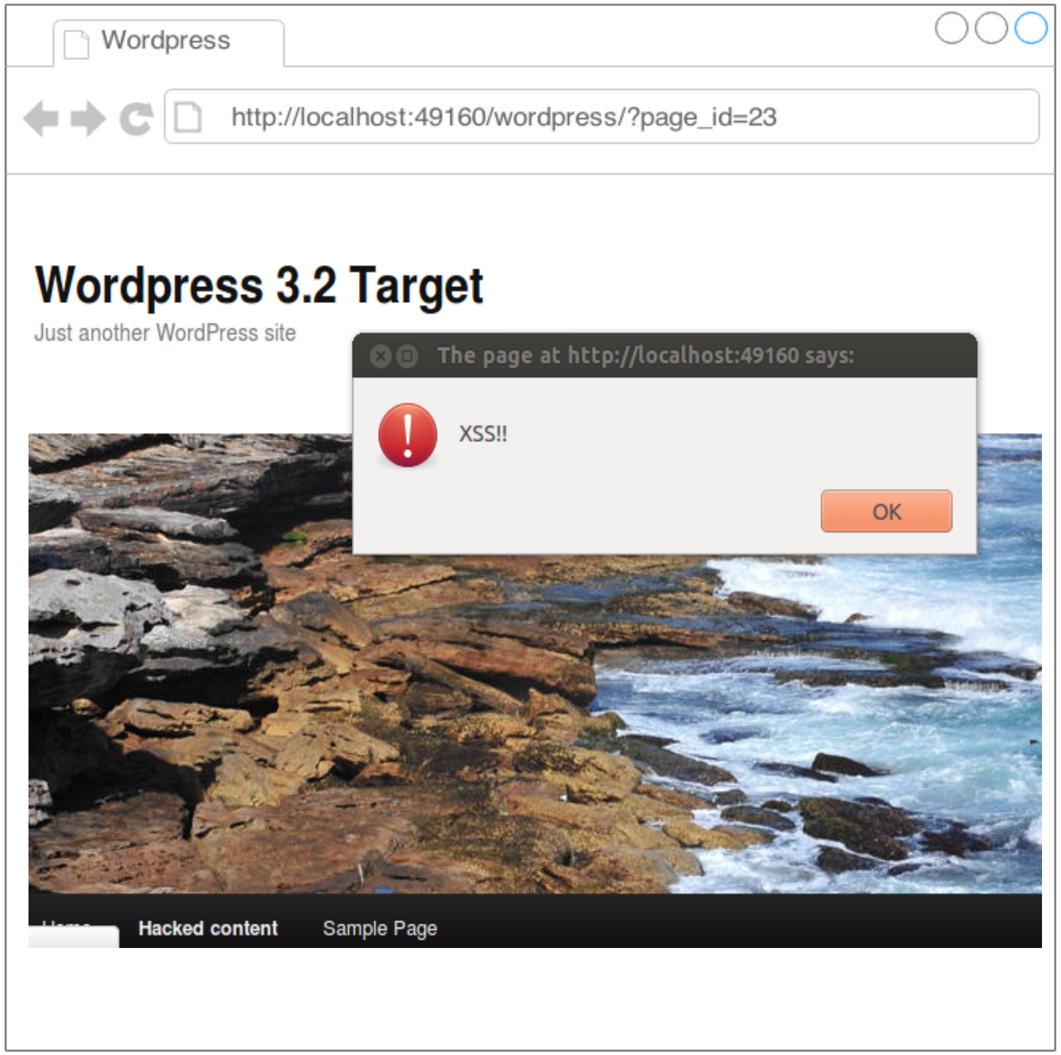
Wordpress

http://localhost:49160/wordpress/wp-login.php

Demo: Single exploit run



```
lloki@testbed: ~/tbed
[2014-07-17 17:03:09,207]: ------------------------------------------------------------
[2014-07-17 17:03:09,207]: Target: Wordpress3.2
[2014-07-17 17:03:09,207]: References: [['empty']]
[2014-07-17 17:03:09,207]: Name: Wordpress_3_2_XSS
[2014-07-17 17:03:09,207]: Plugin:
[2014-07-17 17:03:09,208]: Container: ubuntu-apache-mysql
[2014-07-17 17:03:09,208]: VulWikiPage: None
[2014-07-17 17:03:09,208]: Type: XSS
[2014-07-17 17:03:09,208]: TargetLicense:
[2014-07-17 17:03:09,208]: Description: XSS vulnerabiity in Wordpress application
[2014-07-17 17:03:09,208]: ------------------------------------------------------------
[2014-07-17 17:03:09,208]: Getting the browser instance...
[2014-07-17 17:03:11,225]: STARTUP: SUCCESS
[2014-07-17 17:03:11,225]: Navigating to "http://localhost:49160/wordpress/wp-login.php'
[2014-07-17 17:03:11,542]: ...Success!
[2014-07-17 17:03:11,542]: Searching for element 'user_login' by id...
[2014-07-17 17:03:11,598]: ...Success!
[2014-07-17 17:03:11,973]: Searching for element 'user_pass' by id...
[2014-07-17 17:03:11,994]: ...Success!
[2014-07-17 17:03:12,412]: Searching for element 'wp-submit' by id...
[2014-07-17 17:03:12,421]: ...Success!
[2014-07-17 17:03:13,585]: Waiting...
[2014-07-17 17:03:15,822]: ... end of implicit wait.
[2014-07-17 17:03:15,822]: Navigating to "http://localhost:49160/wordpress/wp-admin/pos
[2014-07-17 17:03:16,715]: ...Success!
[2014-07-17 17:03:16,715]: Searching for element 'title' by id...
[2014-07-17 17:03:16,724]: ...Success!
[2014-07-17 17:03:17,231]: Searching for element 'content' by id...
[2014-07-17 17:03:17,254]: ...Success!
[2014-07-17 17:03:18,577]: Searching for element 'publish' by id...
[2014-07-17 17:03:18,591]: ...Success!
[2014-07-17 17:03:19,418]: Waiting...
[2014-07-17 17:03:20,240]: ... end of implicit wait.
[2014-07-17 17:03:20,240]: Navigating to "http://localhost:49160/wordpress/?page_id=23
[2014-07-17 17:03:20,629]: ...Success!
[2014-07-17 17:03:20,629]: Catching the alert box...
[2014-07-17 17:03:20,650]: Alert text: 'XSS!!'
[2014-07-17 17:03:20,702]: ...alert dismissed!
[2014-07-17 17:03:20,703]: ------------------------------------------------------------
[2014-07-17 17:03:20,703]: RESULT : SUCCESS
[2014-07-17 17:03:20,703]: Disposing the browser instance
```


Wordpress
http://localhost:49160/wordpress/?page_id=23
Wordpress 3.2 Target
Just another WordPress site
The page at http://localhost:49160 says:
XSS!!
OK
Home
Hacked content
Sample Page

TestREx applications

- Executable documentation for software companies
- Penetration testing support tool
- Part of a training toolkit for studying web application security
- Manual/discovery security testing
- Automated regression testing suite
- Automated security + configuration testing
- Benchmark for code analysis tools evaluation
- Aid for security-unaware developers

Future work

- Engage UNITN students
  - *Extension of the exploit/vulnerability corpus*
  - *Implement a number of attack scenarios and countermeasures for JavaScript*
  - *Use TestREx as a part of a toolchain for scanning Node.js*
- Build a hierarchy of exploits similarly to what we did with containers
- Use TestREx for JavaScript static analysis tools evaluation
- Semi-automatic generation of test cases for security vulnerabilities

Conclusions

- Created a small set of our example exploits (17) with WebGoat and server-side JavaScript
- Adapted the corpus of exploits taken from the BugBox to TestREx
- It's possible to quickly switch between execution environments and do effective version/configuration testing
- We envision the scripted exploits as the runnable documentation that can facilitate testing and bug fixing in software development

Lessons learned

- A true value of building on top of the existing approaches
  - *BugBox by Nilson et al. [1]*
  - *MalwareLab by Allodi et al. [2]*
- The importance of simple and modular architecture
- The necessity of reliable information on applications, existing exploits, software and execution environments

References

- **[1]** Nilson G.; Wills K.; Stuckman J.; Purtilo J. “***BugBox: A Vulnerability Corpus for PHP Web Applications***” Presented as part of the 6th Workshop on Cyber Security Experimentation and Test, USENIX, 2013

- **[2]** Allodi L.; Kotov V.; Massacci F. ***“Malwarelab: Experimentation with cybercrime attacks”*** Presented as part of the 6th Workshop on Cyber Security Experimentation and Test, USENIX, 2013

Questions?

- TestREx is available at [https://github.com/standash/TestREx]
- Authors:
  - Stanislav Dashevskyi: dashevskyi@fbk.eu
  - Daniel Ricardo Dos Santos: dossantos@fbk.eu
  - Fabio Massacci: fabio.massacci@unitn.it
  - Antonino Sabetta: antonino.sabetta@sap.com