TestREx:
A Testbed for Repeatable Exploits

Stanislav Dashevskyi\textsuperscript{(1,2)}, Daniel Ricardo Dos Santos\textsuperscript{(1,2)}
Fabio Massacci\textsuperscript{(1)}, Antonino Sabetta\textsuperscript{(3)}

\textsuperscript{(1)} University of Trento, (2) Fondazione Bruno Kessler. (3) SAP Labs

http://securitylab.disi.unitn.it
Exploits Collections

• **Systematic collection of exploits into a knowledge base**
  • Exploit DB, OVSDB, 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 testing approaches

• **What about developers using that software?**
The 3° Party Developer Perspective

“Exploits, exploits every where. Nor a single script to run”
- T.S. Coleridge - The Rime of the Ancyent Marinere
- (Free adaptation by Fabio Massacci)

- How to actually “repeat” the exploit in my operational environment?
  - Applications use different platforms → SQL injection for mySQL may not work in MongoDB
  - Software changes → different exploits work for different versions
  - Software configuration does matter → exploit only works if run in a particular OS
  - Essentially it is a “non-constructive existence proof”
Getting more value out of the corpus!

- Apart from “documenting” an exploit, what other information do we want?
- **Baseline Information**
  - Exploit X successfully subverts a application A that is running in environment E
- **What 3° party developers really want to know is**
  - Does X work on same A in updated E'?
  - Does X work on updated A’ in same E?
  - Does X work on updated A', in updated E'?
- **Deploying and matching all possible software configurations and application versions…**
  - .. as automatically as possible…
TestRex Baseline

- Focus on Web-facing code (Java/JavaScript)
- Building on top of the existing approaches
  - BugBox by Nilson et al.
  - MalwareLab by Allodi et al.
- Objectives
  - Simple and modular architecture to deploy all kind of web-based applications
  - “Actionable” information on applications, exploits, software and execution environments
  - Report successful and unsuccessful exploits
What is TestREx

• **10,000 feet’s view → Management system for software environments**
  - Provide an isolated “playground” per every application version and its corresponding software environment

• **Bird’s eye view → 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

• **Low-level view → Test suite for managing and running scripted exploits against corresponding applications**
TestREx: typical workflow

1. Build and load the environment
2. Run a scripted exploit
3. Monitor success of the exploit
4. Destroy the environment
5. Generate report

TestREx

Execution Engine

Applications
Software Configurations
Exploits Corpus

Docker
Selenium
Docker
TestREx: Application Container example
TestREx: Software Containers hierarchy
TestREx: Exploits

- **Exploit – “TestREx Definition”**
  - sequence of [automated] actions required to subvert a vulnerability in an application and verify its success

- **Low Level Technicality**
  - Self-contained unit test + description metadata
  - Python script + Selenium driver (automate browser)
  - Script passes results of its run to Execution Engine

- **Which exploits are present?**
  - Adapted corpus of exploits taken from BugBox
  - Created own example exploits (17) with WebGoat and server-side JavaScript
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)
Running an Experiment

- **Modular way to run exploits and applications**
  - All exploits are independent scripts that can be supplied by anyone
  - An application can be started in either “clean” or “infected” state

- **Sample scenarios → regression testing and configuration testing**
  - Deploy multiple versions of an application and understand what was fixed though the version history
  - Deploy an application on different platforms and see the correlation between third-party software and vulnerabilities

- **Report generation**
  - A .csv file with exploit run results and exploit metadata
TestREx Business Applications

• **Executable documentation for software companies**
  - “document an exploit” = “write a TestREx script”
  - Automated security + configuration testing
  - Automated regression testing suite
  - Penetration testing support tool

• **Aid for security-unaware developers**
  - Part of training toolkit for studying web app security
  - Benchmark for code analysis tools evaluation

• **Patent Pending for SAP Labs**
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
- Semi-automatic generation of test cases for security vulnerabilities
  - Use TestREx for JavaScript static analysis tools evaluation (to eliminate false positives)
Conclusions

• **We envision a scripted exploit is an executable documentation that can facilitate testing and bug fixing in software development**

• **Getting TestREx?**
  - [https://github.com/standash/TestREx](https://github.com/standash/TestREx)
  - Use for research is free but commercially there is a patent pending for SAP Labs

• **Finally**

  “Farewell, farewell! but this I tell
  To thee, thou Usenix-Guest!
  He codeth well, who exploith well
  Both app, environment and test”

  T.S. Coleridge - The Rime of the Ancyent Marinere
  (Free adaptation by Fabio Massacci)