OWASP

Open Web Application Security Project

AppSec DevSummit

Agile Testing with OWASP ZAP

Contents

- Intro
- ZAP + web Api
- Jenkins
- Blackbox testing on demand
- Proxying Unit Tests
- The ZAP Jenkins Plugin
- Configuring unit tests to run with the Jenkins plugin.
- Presenting scanning results in Jenkins
- ZAP scripting API
- Scripting ZAP
- ZAP script Challenge

Intro

- What are you working on?
- What would you like to take away from this session?
- Developer/DevOps/Pentester/other?
- What's your experience with Jenkins, ZAP?Have you contributed to ZAP or Jenkins yet?

$Context

- Improvement suggestions are welcome
- Also, not affiliated with ZAP dev team or Jenkins

The VM

- Xubuntu
- php + packages, python, virtualenv, pip, ZAP, Jenkins
- Other goodies: IDEs, Editors
- All usernames and passwords are: test

Conventions

- Zap runs on port 8090
- Jenkins runs on 8080
- All the python builds are virtualenv
- Servers run on demand on port 7070 either with php -S or python <path to main application>
- Logs are either on the Jenkins output or $PROJECT_ROOT/logs

ZAP +REST API

- ZAP has a REST API
- It allows control of ZAP without the UI.
- Lives in http://zap.
- It can be accessed from any proxied browser.
- Also: There's a Python API
- pip package: python-owasp-zap-v2.4

Step 0

- Before anything, activate virtual environment now
- cd ~/ZapWorkshop
- source ./env/bin/activate
- This activates the virtual environment for this terminal so use it to run the python scripts

Exercise 1

1. Open ZAP
2. Change the Port it binds to 8090
3. Proxy Firefox through it
4. Configure the API access
5. Spider http://localhost/ through the api (don't use the GUI)

Solution

1. Double click ~/Zap/zap.sh or execute same file from terminal (useful for logging and troubleshooting).
2. ZAP:Tools->Options->Local Proxy (or Ctrl-Alt-O ->Local Proxy) set listener to 127.0.0.1 and port 8090
3. Firefox: Edit->preferences->advanced->network->settings, Set proxy to 127.0.0.1 and port 8090
4. ZAP: Open Options-> API
   - Set an easy to type API key or disable it (Never on prod)
   - Uncheck Secure Only (Never on prod)
5. Firefox: visit http://zap ->api->spider ->scan ->fill URL -> click submit -> you should see a JSON document with id:0 <-- This is the spider id, you need it to see results.

Jenkins

- Continuous Integration project
- Java
- Ubuntu package: Jenkins
- Installed and runs on localhost:8080
- Allows users to build and manage a chain of tools to run on a codebase.
- Extensive customization through plugins.

Exercise 2

- Make a test local git repository and add it as a new project on Jenkins
- Add build step that prints Job name to a log file

Solution

- open terminal->mkdir ~/Desktop/JenkinsFoo && git init ~/Desktop/JenkinsFoo
- Firefox: Jenkins-> new item -> set any name, select freestyle project
- Source Code Management :"Git", Repository URL: file:///home/test/Desktop/JenkinsFoo/
- Add shell build step, the variable JOB_NAME contains the name of the project you can echo $JOB_NAME>/tmp/JenkinsFoo.log
- Save and click 'build now'

Blackbox Testing

- ~/ZapWorkshop/Call_Zap_In_Blackbox_Mode
- /app contains a php application
- A temporary php web server can be setup with

php -S localhost:7070

- What's wrong with it?

PoC

- zapExample.py orchestrates a basic test.
  - Runs ZAP in daemon mode, sets up php Server
  - Keep logs of everything (useful for debugging)
  - ZAP REST API orders a spider and aggressive scan
  - Print results
  - Exit
- Execute from terminal, last result should be XSS or Cross Site Scripting on the parameter 'query'.

ZAP + Jenkins

- Project is called "BlackBox Testing"
- Jenkins config of the previous exercise
  - Instead of the echo a python command launches the script

Exercise 3

- Add an identical page named login.php
- Make the python script also scan this page

Solution

- Add another target or switch the target variable to point to the new page

Problem

- The BlackBox test can only attack what it can spider.
- Hidden parameters, REST APIs, authentication,multistep logic are unreachable (Sort of)

Solutions

- Code special tests for multistep
- Add all the unique application paths to target
- The API offers support for authentication via provided context
- It also allows loading of ZEST scripts and recorded actions.
- But what about...

Unit tests

- The pros
  - Proxy + Tests = improved logic, less spidering, access to unspiderable areas.
  - Easy to set, just proxy any HTTP tests.
  - Results from both the tools and the tests can be correlated with static code analysis and other tools in one nice report (for management)
- The cons
  - Not everything supports proxying
  - Results from both the tools and the tests can be correlated with static code analysis and other tools in one nice report (for management)
  - Suddenly your output is all red.

New project: Access_To_Unit_Tests/app

- Flask application with unit tests in app_test.py
- Selenium webdriver.
- Proxies through localhost:8090
- Orchestration script now also launches unit tests.
- The hidden parameter that would otherwise remain unnoticed gets scanned.

Plug into Jenkins

- No difference, the new script handles unit tests + xvfb

Exercise 4

- Because of the tests, the issues get reported multiple times, can you make them unique?

Solution

- You can set a dictionary with keys every parameter and values a dictionary of the issues identified.
- For general issues just set the server as a parameter.
- You can pprint that.

The ZAP Jenkins plugin

- ZAP introduced an Official Zap Plugin
- Extensive documentation
- Supports blackbox, unit tests, authentication.
- Check project: zapPlugin <-- already contains unit tests configuration

Exercise 5

- Clone this project and make it run in Blackbox mode

Solution

- Untick the prebuilt step
- Modify the python step that sets up the web server Before the ZAP plugin

ZAP Jenkins Plugin Settings

- Prebuilt step -- Setup ZAP before the built but don't do anything else
- Then run whatever comes in the build chain before the plugin.
- Then run the plugin
- By running the proxied unit tests before the plugin you can proxy them.

Presenting Scanning Results

- Jenkins Plugin: HTML Publisher
- Configure:
  - Where ZAP dumps the HTML report
  - Where publisher should find the HTML report
  - Then each build should have it's own report
- Project: Reporting

Exercise 6

- Make a dedicated folder for reports in $Project_Root/reports/ and have zap and publisher publish there.
- Bonus: Make all reports appear in /tmp

Solution

- change the names of the report directories in the project settings.

ZAP Scripting API

- ZAP is extendable
- Anything it doesn't support natively can be scripted
- There is also an interactive scripting console.
- With scripting templates
- In Javascript
- (no code completion yet)

ZAP Scripting API

- There is already a lot of scripts: https://github.com/zaproxy/community-scripts
- Scripts implement callbacks under the classes named after the type of the script.
- It's handy to have ZAP source open if you're trying to develop one.
- Getting started: https://github.com/zaproxy/zaproxy/wiki/InternalScripting

Scripting ZAP

- Scenario: Need to submit form but CSRF token.
- (Who knows what CSRF is?)
- Problem: ZAP can't automagically recognize CSRF tokens and add to next POST.
- Solution: Test CSRF with unit tests
  - Messy
  - Misses a number of tests that ZAP might perform
  - Could be more elegant
  - Better leave security to the security toolchain
- Solution: Get valid CSRF token, break, scan, pause, get valid CSRF token, break, scan, pause,.....
  - We're getting somewhere
  - Race condition, if the request to pause reaches the API after the next attack request has already left we just missed a payload.
- Solution: Implement same functionality in a script
  - ZAP offers the HTTP Sender facility.
  - Callbacks for every request/response no matter where it comes from.
  - Also, context aware, knows which tool the request/response came from.

CSRF Script

```
if request.comes_from(zap.scanner){
token = zap.get(url_responding_with_csrf_token).extract(token);
request.body = request.body.toString().sub('csrf=\w','csrf='+token);
}
```

Script already exists

You can find it under ZapScript/csrf/ or

https://github.com/zaproxy/community-scripts/blob/master/httpsender/Capture%20and%20Replace%20Anti%20CSRF%20Token.js

Exercise 7

- Load the script
- Load the application under
- Point it to $csrf_example_app
- Try to make a request, what's happening?

Exercise 8

- Edit the script to recognize the csrf variable of our page
- Scan

To recap

- Jenkins
- ZAP
- Blackbox test
- Proxied unit tests
- Jenkins plugin
- Present results
- Extending ZAP through scripts

Credits/Links/Resources

- ZAP Github:
  - https://github.com/zaproxy/zap-core-help/wiki/HelpAddonsScriptsScripts
  - https://github.com/zaproxy/zaproxy/wiki/InternalScripting
  - https://github.com/zaproxy/zap-extensions/tree/alpha/src/org/zaproxy/zap/extension/simpleExample
- ZAP Jenkins Plugin
  - https://wiki.jenkins-ci.org/display/JENKINS/zap+plugin
- Others have blogged
  - https://www.google.co.uk/?gfe_rd=cr&ei=C44QWYDLOrHA8ge2za_AAQ#safe=strict&q=PrakhashS+zap+jenkins
  - https://www.google.co.uk/?gfe_rd=cr&ei=C44QWYDLOrHA8ge2za_AAQ#safe=strict&q=burak+zap+jenkins

Post credits Slide

- What about the ZAP scripting challenge?
- Right...

ZAP Scripting Challenge!

- In /challenge you will find an Angular application
- It emulates a login form
- with XSS
- Problem: it's via POST
  - So what?
  - POST is a web standard
  - All proxies should understand it, right? RIGHT?!

Unless you're Angular

- Old-timey POST has a content-type of application/form-url-encoded.
- But that's old in the age of apis.
- Angular POSTs with a JSON content type.
- Body is not parsable anymore.

Comments Slide:

- you can POST by setting a custom content type.
- But it's not the default and we're going for seamless security testing.

Challenge

- Write a script that allows ZAP to translate the body, inject payloads and translate back into json.

Thank you

- Questions?
- Feedback?