N different strategies to automate OWASP ZAP

The OWASP Zed Attack Proxy

Marudhamaran Gunasekaran
Zap Contributor
@gmaran23
Software Security Consultant at DevOn / Prowareness
Agenda

• Application Security Program Challenges
• Lightning Introduction to ZAP
• The ZAP API
• The N ways of Automating ZAP
• Scripting for ZAP
• Tips for CI / CD and Case Studies
The problems

- Most developers know very little about security
- Most companies have very few application security folks
- Security testing is done late in the application development lifecycle (it is done at all is done)
Part of the Solution

• Use a security tool like ZAP in development
• In addition to security training, secure development lifecycle, threat modelling, static source code analysis, secure code reviews, professional pentesting...
What can you get out of this?

• A way to quickly evaluate your apps
• Options for more thorough scanning
• An introduction to the ZAP API
Why ZAP?

• An **easy to use** webapp pentest tool
• Completely free and **open source**
• Source code **updated** multiple **times a day**
• One of the OWASP **Flagship projects**
• Ideal for beginners, But also used by professionals

• **Powerful API** - for automated security tests
The app sec tool foundations

• **Spider or Crawler**
  – Gather information about what to attack

• **Passive Scan**
  – Static analysis on the gathered information (HTTP requests and responses)

• **Active Scan**
  – Send attack (potentially harmful) payloads to exploit / confirm weakness
ZAP API demo

Headless attack!
ZAP API demo

Demo Flow:

1. Open the ZAP GUI on the right of the screen
2. Browse the API from the left portion of the screen
3. As we trigger a spider scan, it would be visible in the UI
4. Poll the Spider Status API
5. Get results from passive scan
6. Trigger an Active Scan from the API, the scanning would start and it would be evident on the ZAP UI
7. Demonstrate a Shutdown
ZAP Baseline scan

1. Quick and fast
2. No prior ZAP experience required
3. Docker is the only dependency
4. Configurable with Command line Options
5. Quickly baseline the security controls of an application or many applications (just passive scanning)
ZAP Baseline scan

Finds issues like:

• Missing / incorrect security headers
• Cookie problems
• Information / error disclosure
• Missing CSRF tokens
ZAP Baseline scan - Demo

Demo flow:

1. Pull the zap docker image
2. `docker run -t owasp/zap2docker-stable zap-baseline.py -t http://www.renthoughtsweb.com:8020`
2. Interpreting the results of the baseline scan
3. Generating and Using a scan configuration file
4. Mass baseline scan
The available API Clients

1. Java
2. Python
3. DotNet
4. PHP
5. Node JS
6. GO
7. 
8. 

Automating Quick Scan - via python API client

Demo flow:
1. Start ZAP programmatically
2. Wait for ZAP to initialize
3. `zap.spider.scan(targeturl)`
4. Wait till `zap.spider.status(scanid)` is 100
5. `zap.ascan.scan(target)`
6. Wait till `zap.ascan.status(scanid)` is 100
7. `zap.core.alerts()`
8. `zap.core.htmlreport(target)`
Automating authenticated scans

1. *Create a context* in the name of the application
2. Choose the mode of *authentication* (for instance Forms Authentication)
3. Provide *Authentication information*
4. Spider
5. Scan
6. Extract Results
Automating Authenticated Scan
- Demo via Dot Net API Client
Authenticated Scan Demo

Demo flow:
1. Start ZAP programmatically
2. Wait for ZAP to initialize
3. api.context.newContext
4. api.context.includeInContext
5. api.users.newUser
6. api.forcedUser.setForcedUser
7. api.forcedUser.setForcedUserModeEnabled
8. api.spider.scan
9. api.ascan.scan
10. api.core.htmlreport
Security Regression Testing

Well, let me watch you here!
Integrating with Selenium

Test cases

- Demo via Java API Client
Selenium Integration Demo

Demo flow:
1. Start ZAP programmatically
2. Wait for ZAP to initialize
3. Set up Selenium web driver with proxy settings
4. Run the selenium test cases
5. api.spider.scan
6. api.ascan.scan
7. api.core.htmlreport

A recorded quick demo - https://vimeo.com/222238217
Official Jenkins plugin
Tips from the field for CI / CD Integration

1. Tune the scan policies for faster scans
2. Option to fail the on Critical Security Control failure
3. Secure HTTP headers check is trivial yet highly useful
4. Timed passive scans (baseline scan) on Continuous Integration
5. Deep Scan on nightly builds
Scripting for ZAP

Script things that are not supported out of the box
Script for automating regular VAPT activities
Script to modify request and responses
.. And much more
Quick Demo – ZEST scripting

Demo flow:
1. Add a new ZEST Script
2. Add a ZEST Replace to add must-validate to the Cache-Control HTTP Response Header
Quick Demo – ZEST
Security Regression Scripting

Demo flow:
1. Demonstrate an Open Redirect Flaw
2. Add a ZEST Script
3. Add an Assert to ensure the Application doesn’t redirect to other domains
Quick Demo – Python scripting

1. Find insecure HTTP verbs on server
Useful cmdline options

• Turn off db recovery (speeds things up)
  -config database.recoverylog=false

• Update all add-ons
  -addonupdate

• Run without the UI
  -daemon

• Listen on a specified host and port
  -host 127.0.0.1 -port 7070

• Setting the API key
  -config api.key=j8Wd0Eq8dhwWE24VGDsreP

• Disable API key in a safe environment
  -config api.disablekey=true
ZAP – Need Help?

ZAP user group -
https://groups.google.com/forum/#!forum/zaproxy-users

ZAP Evangelists -
https://github.com/zaproxy/zaproxy/wiki/ZapEvangelists

ZAP Developers group -
https://groups.google.com/forum/#!forum/zaproxy-develop
ZAP - Get Involved

Use the tool
Recommend
Write Add-ons
Write Scanners / Scripts
Report bugs
Conclusion

• Consider security at all stages of development cycle
• OWASP ZAP is ideal for automating security tests
• It is also a great way to learn about security

“Man is a tool-using animal. Without tools he is nothing, with “right set of” tools he is all”
Any Questions?

http://www.owasp.org/index.php/ZAP