we45

Security Testing your Apps with OWASP ZAP

Sudarshan Narayanan

we45

Agenda

- Brief overview of OWASP Zed Attack Proxy
  - Introduction to the ZAP UI and API.
  - Key features of ZAP and how to use them (Spidering, Intercepting Proxy, Active vs. Passive Scan)
- ZAP API and its capabilities
- ZAP Clients
- Introduction to Parameterized DAST Scanning using ZAP
- ZAP Scripting

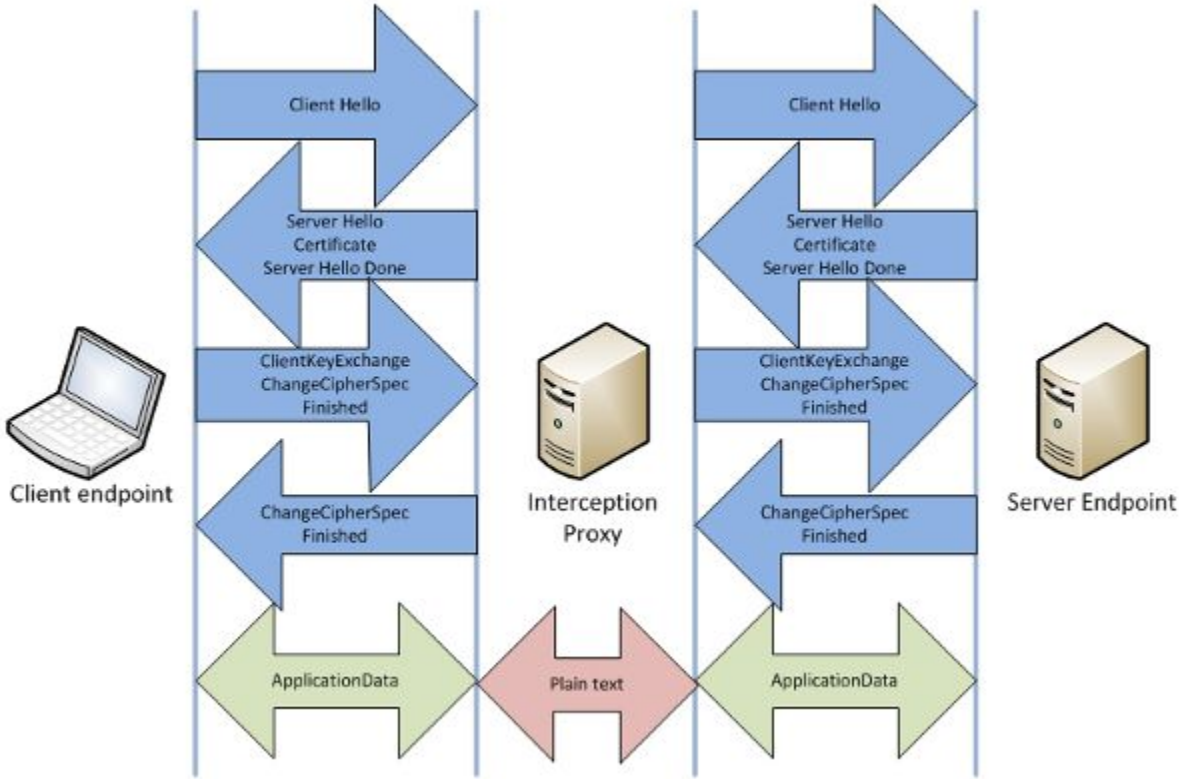
Why ZAP?

- Developer's entry point into AppSec.
- Evidence of no vulnerability != no evidence of vulnerability
- Automated pre-defined checks for potential vulnerabilities on target applications.
- Acts as a “sanity check” for security testing, before handing over to SecOps/AppSec team.
- Integrates with CI to “bake-in” security rather than “bolt on”

OWASP ZAP - An Overview

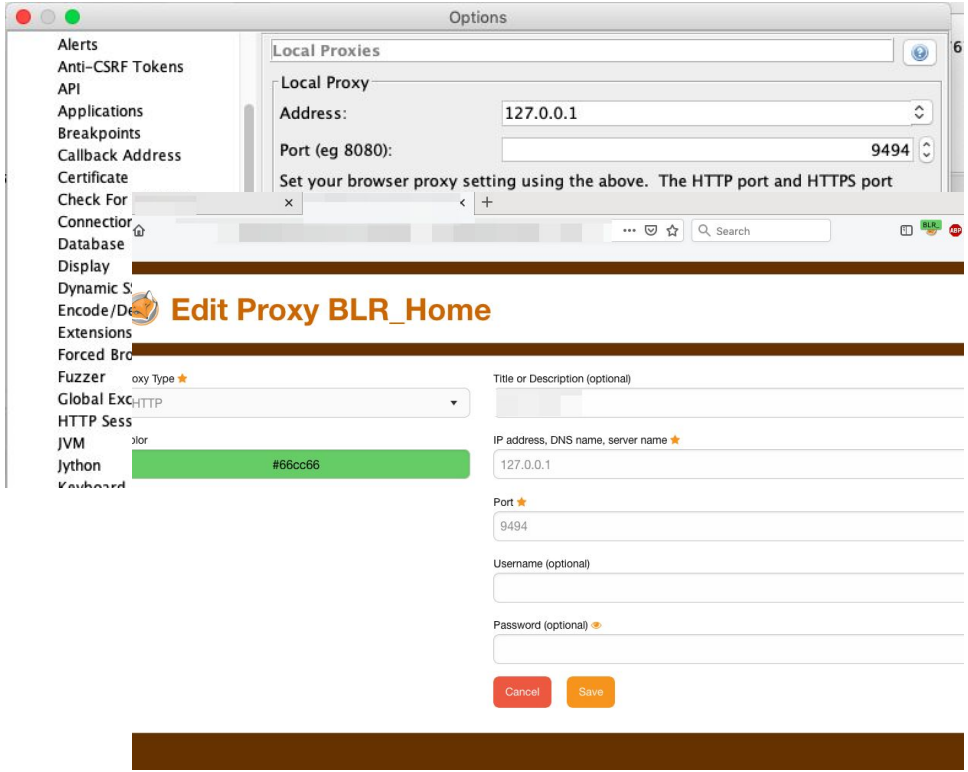
- Flagship OWASP Project with wide user base
- Free and Open source Web Application Scanner
- Community Support - Wide variety of Extensions, Scripts and Plugins
- Robust API and provides scripting frameworks for integration with CI

What is a proxy?



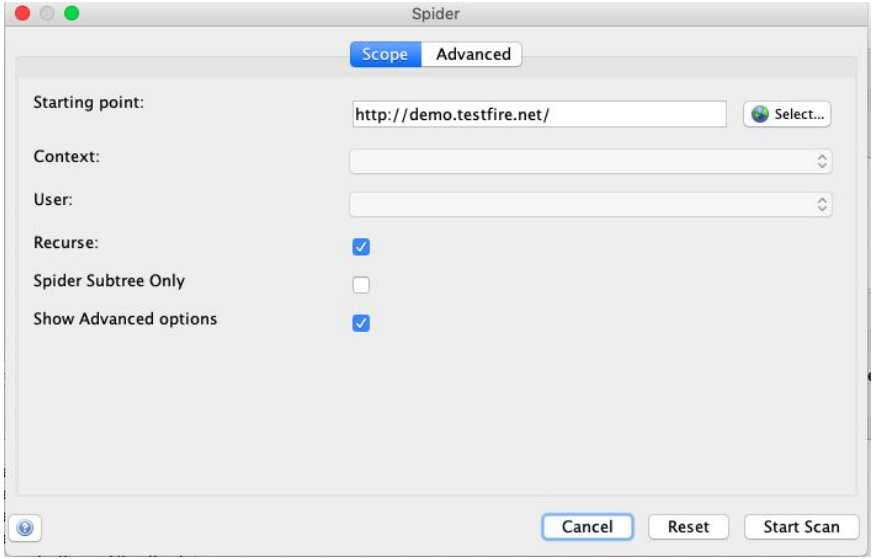
ZAP Features - Intercepting Proxy

- “Man-in-the-middle” proxy that captures requests and responses before being sent to the Web Application server.
- Can be configured in ZAP to listen on a select IP and Port.
- Set break points in ZAP to alter requests before being sent to the Web app



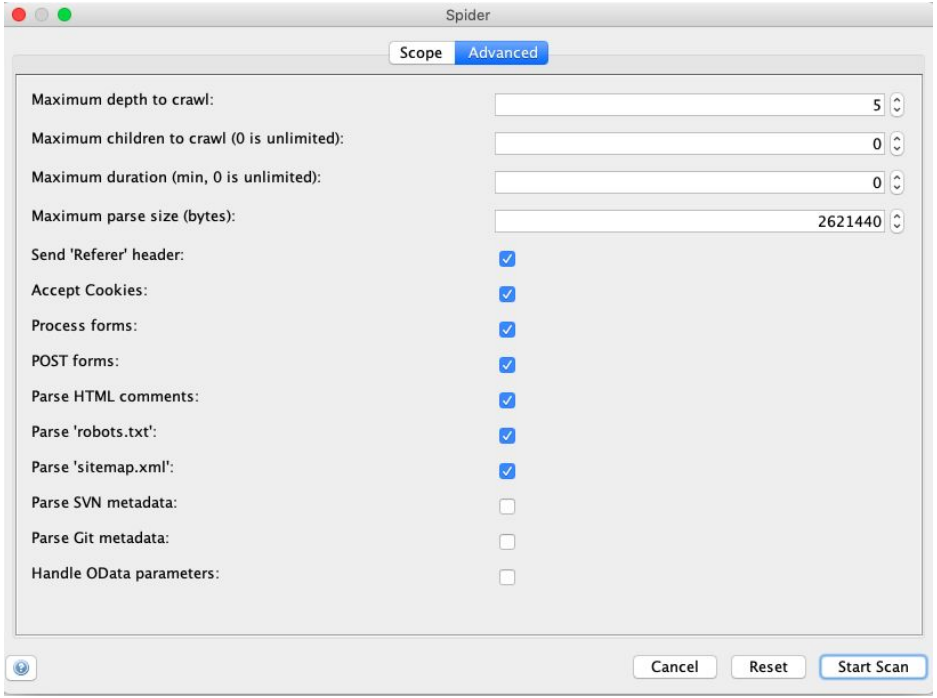
ZAP Features - Spidering

- Tool within ZAP that fetches the various links within a target application.
- Depending on the config, recursively traverses links within each page of the application.
- Checks for various types of resources
  - HTML - Tags such as Base, Href,
  - OData Atom format
  - Robots.txt



ZAP Features - Spidering Parameters

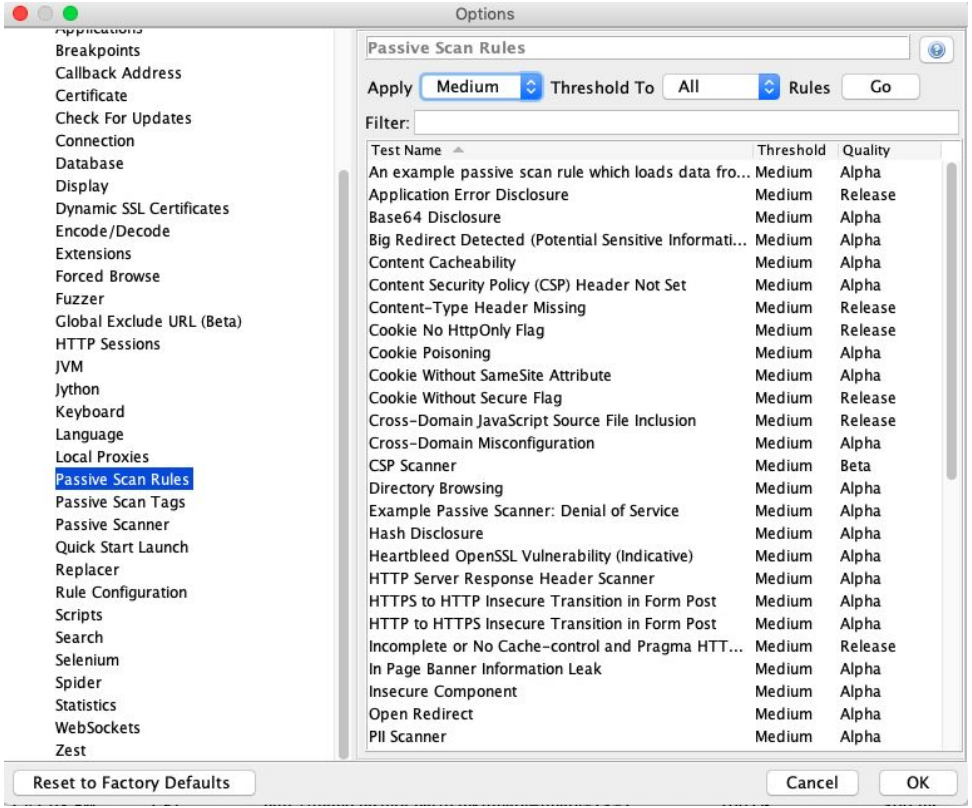
- Under Advanced settings, the following parameters can be set under the Spider Scan
  - Max Depth to crawl -
  - Max threads
  - Max duration
  - Max parse size
  - Parse ‘robots.txt’
  - Parse HTML Comments
  - Parse ‘sitemap.xml’
  - Accept Cookies
  - POST forms



ZAP - Passive Scans

- What is the need for passive scan
- How it works in ZAP?
- Rules/Configs pertaining to Passive scans

OWASP ZAP - Passive Scan Rules



ZAP - Active Scans

- What is the need for Active scan
- How it works in ZAP?
- Rules/Configs pertaining to Active scans

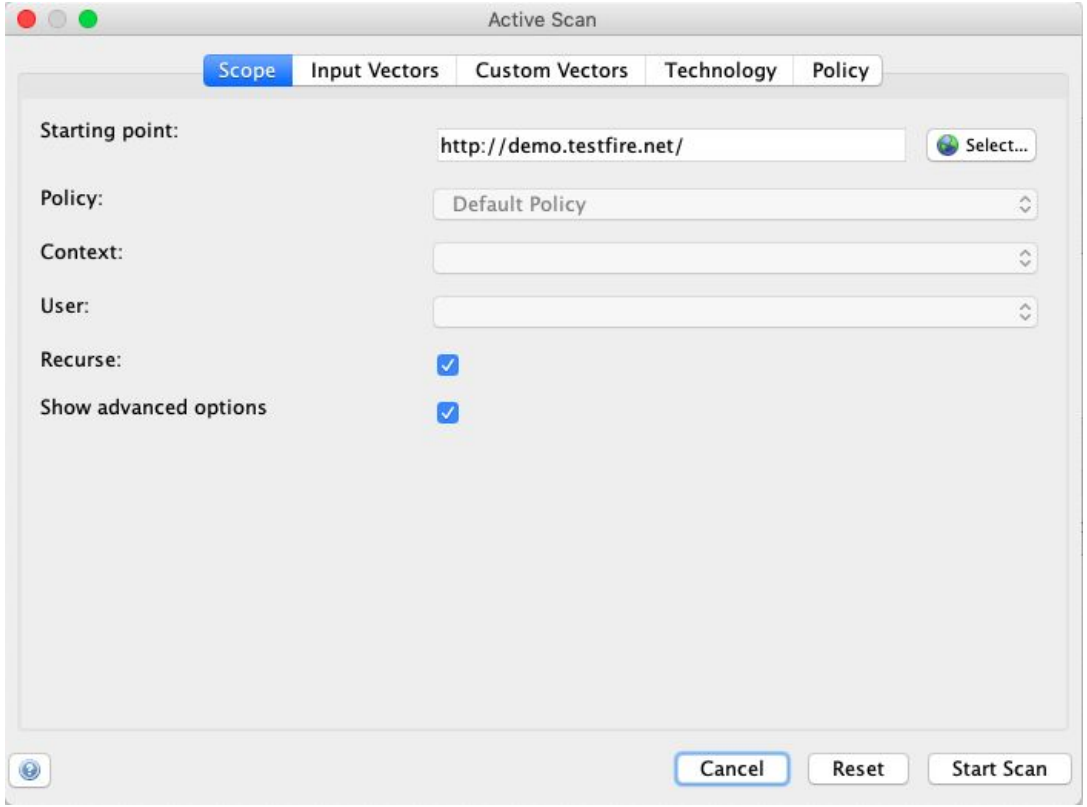
ZAP Features - Active Scans

- Performs vulnerability scans against target web application.
- Scans based on scan policy and configuration settings prior to commencing the scan
- DO NOT USE IT ON WEB APPS WITHOUT PROPER AUTHORIZATION
- Used to find common web application vulnerabilities like
  - SQL Injection
  - Cross Site Scripting
  - Misconfigured HTTP headers
  - CSRF
- Cannot be used to find
  - Business logic flaws
  - Broken Access Control (Authentication/Authorization flaws)

Active Scan Policies and Rules

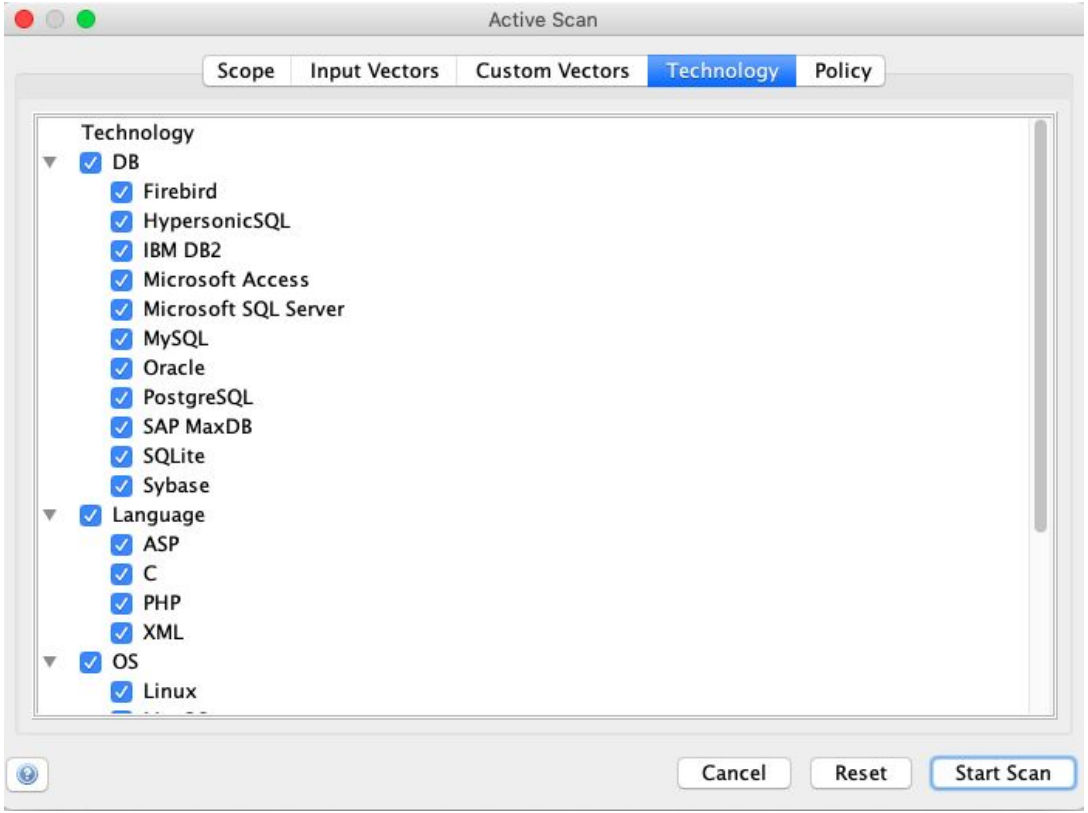
- Performs vulnerability scans against target web application.
- Scans based on scan policy and configuration settings prior to commencing the scan
- Used to find common web application vulnerabilities like
  - SQL Injection
  - Cross Site Scripting
  - Misconfigured HTTP headers
  - CSRF
- Cannot be used to find
  - Business logic flaws
  - Broken Access Control (Authentication/Authorization flaws)

Active Scan Configs and Policies



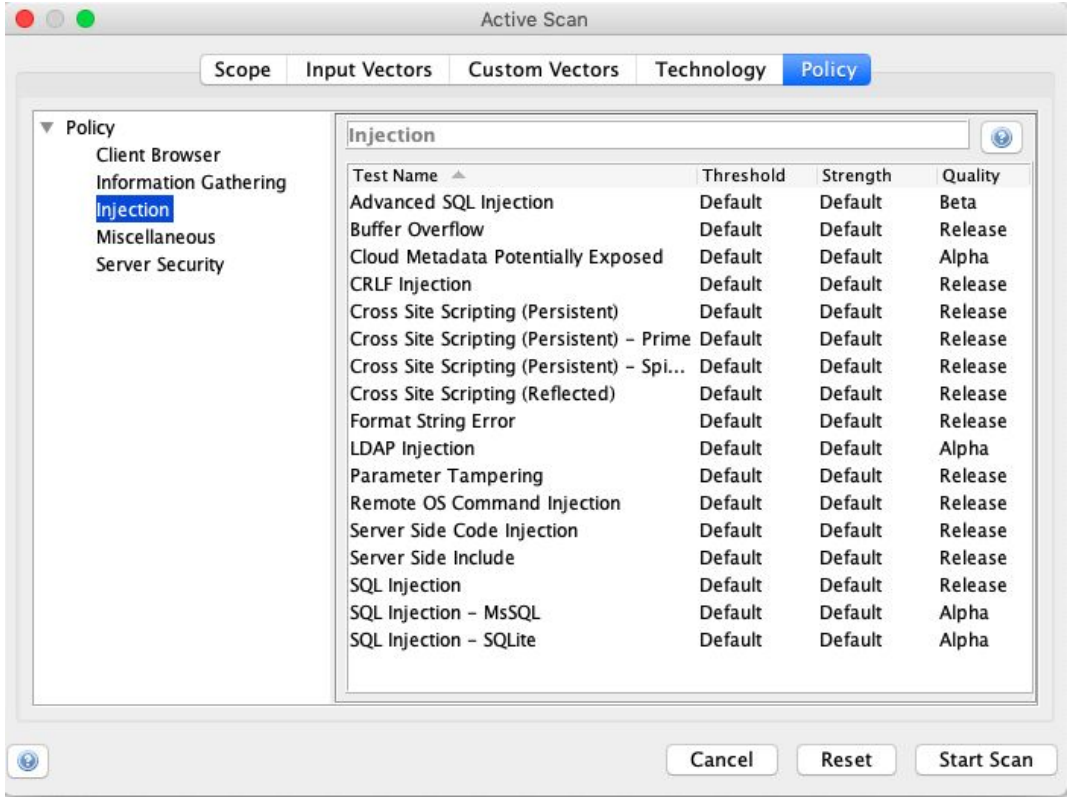
we45

Active Scan Configs and Policies



we45

Active Scan Configs and Policies



we45

ZAP Features - Active vs Passive Scans

we45

Active Scans

- Sends malicious requests to the target web app server
- Actively looks for vulnerabilities by running various payloads on the target application
- Can potentially impact the application in terms of load, based on payloads configured.
- Takes longer to complete.

Passive Scans

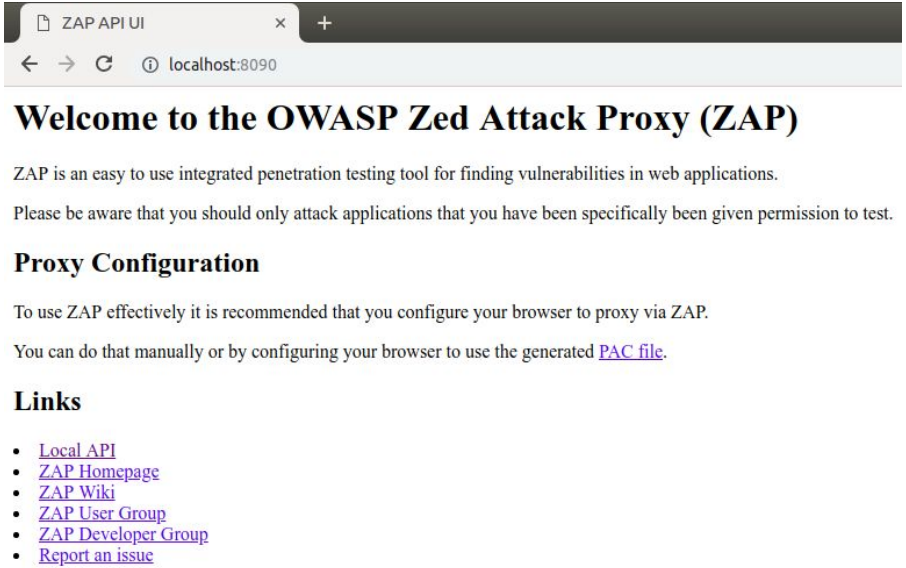
- Based on intercepted traffic, performs static checks on the target web app.
- Runs in the background by default.
- Studies requests and responses without changing them as done in Active Scans.
- Completes faster than Active scans

ZAP Reports

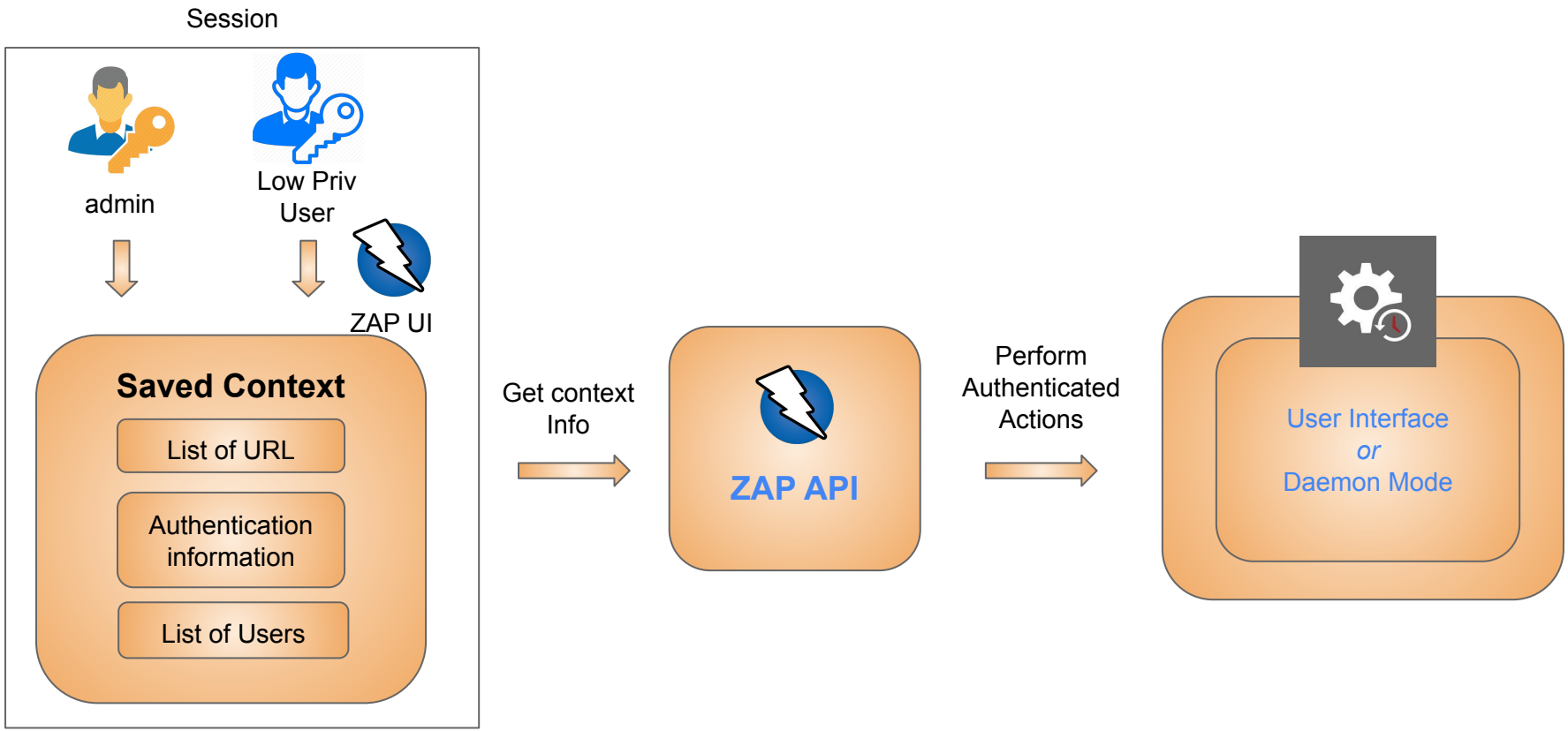
- Generates HTML/JSON/XML/Markdown reports of Active Scan results.
- Contains details such as CWE ID, Severity, Confidence, Vulnerability Description,
- Useful for developers to look into results to fix issues on their applications.

The ZAP API

- Well Defined and Documented REST API
  - https://github.com/zaproxy/zaproxy/wiki/ApiDetails
- API can be accessed at:
  - http://zap
  - http://localhost:<proxy port>
- API can also be accessed through the client implementations.



Authenticated Scan Through API



ZAP API Clients

ApiDetails · zaproxy/zapr

GitHub, Inc. [US] | https://github.com/zaproxy/zaproxy/w...

The ZAP API

ZAP provides a REST Application Programming Interface (API) which allows you to interact with ZAP programmatically.

The REST API can be accessed directly or via one of the client implementations detailed below.

It is documented briefly in the ZAP user guide, but there is more information here on the wiki.

A set of wiki pages which lists all of the available functionality is generated by the code and is available here: API details

In order to be able to use the API when using the ZAP UI you have to first enable it. You can do this via the Options API screen:

- Tools / Options... / API

If you run ZAP in 'headless' or 'daemon' mode (by starting ZAP via the command line and using the -daemon flag) then the API will be automatically enabled.

The ZAP API is particularly useful for Security Regression Tests.

A summary of the clients available:

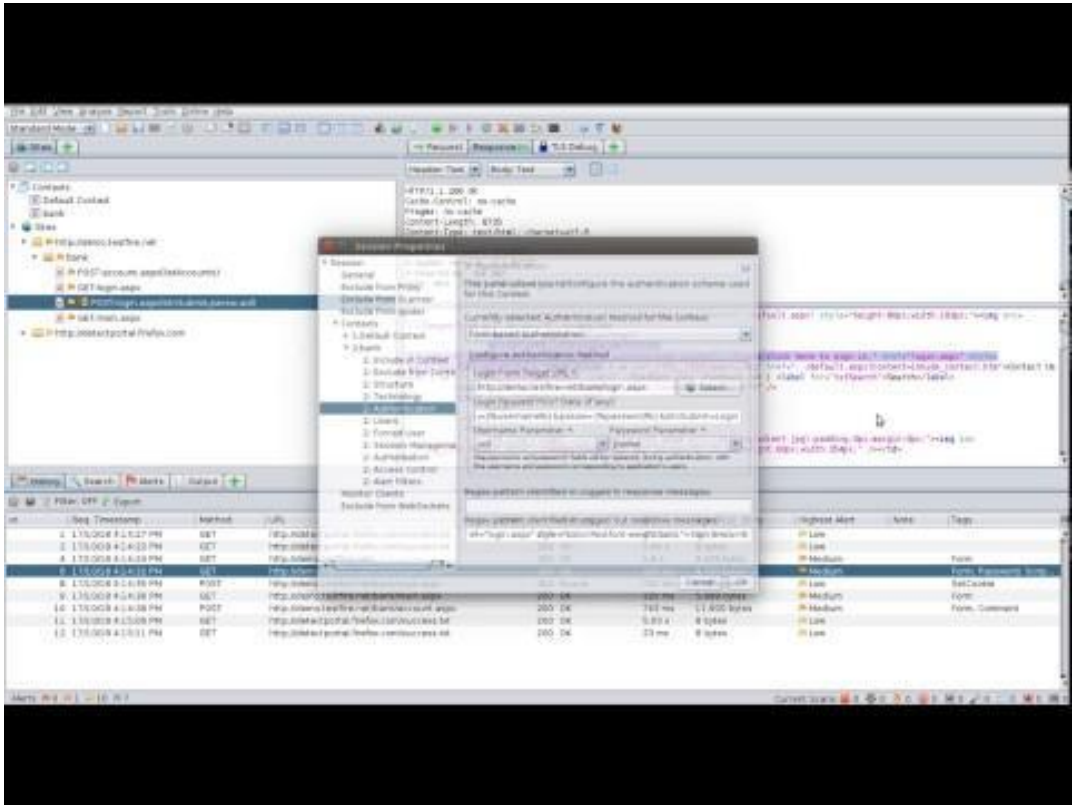
Language	Download links	Notes
Java	GitHub	Official API
Python	PyPI	Official API
Node.js	NPM	In process of becoming an official API
PHP	GitHub Packagist	In process of becoming an official API
Ruby	GitHub	
Ruby	GitHub	

ZAP API Client Python - DEMO

https://pypi.org/project/python-owasp-zap-v2.4/

pip install python-owasp-zap-v2.4

Authenticated Scan Through API - Demo



ZAP Scripting

- Changes to the way ZAP works
- Develop Scripts Inside ZAP
- Access to all internal aspects

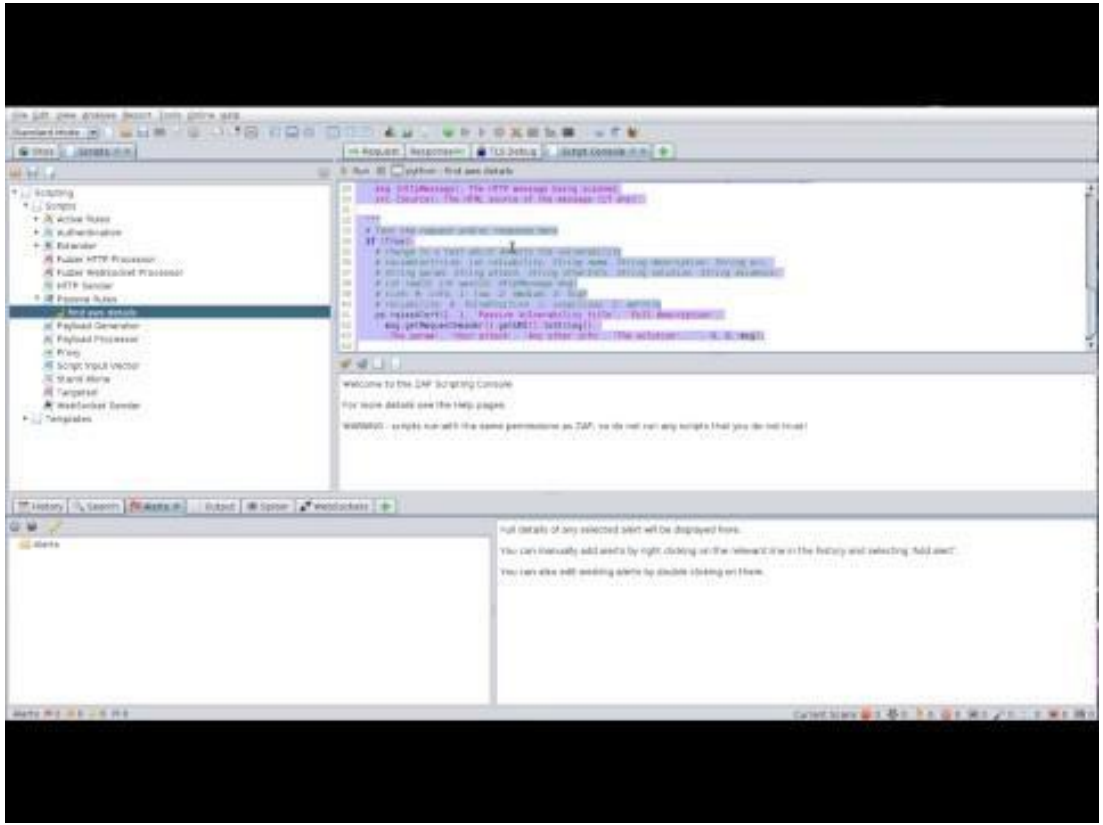
ZAP Script Types

- **Stand alone**
  - Independent scripts to run manually
- **Targetted**
  - Independent script that can be run on a specific target
- **Proxy**
  - Changing Request and Response at proxy
- **HTTP sender**
  - Running on all requests and response.
- **Passive Scan Rule**
  - Rules tested as part of Passive scan
- **Active Scan Rule**
  - Rules tested as part of Active scan
- **Authentication**
  - To perform authentication for context

ZAP Scripting Modules in Python

```
msg
    #the message object that is acted upon to parse/manipulate

msg.getRequestHeader()
    #Request Header Object

msg.getRequestHeader().getURI()
    #fetches the URI from the request header

msg.getRequestBody()
    #Fetches the request body from the request

msg.getResponseBody()
    #Fetches the request body from the request

msg.setRequestBody()
    #Sets a different request body from the one in the original request
```

ZAP Passive Scan Rule - Demo



ZAP Active Scan Rule - Demo

- Application allowing “none” algorithm in JWT tokens
- Attacker can create own token with any payload to exploit this vulnerability

Encoded PASTE A TOKEN HERE


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJ
SMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```


Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

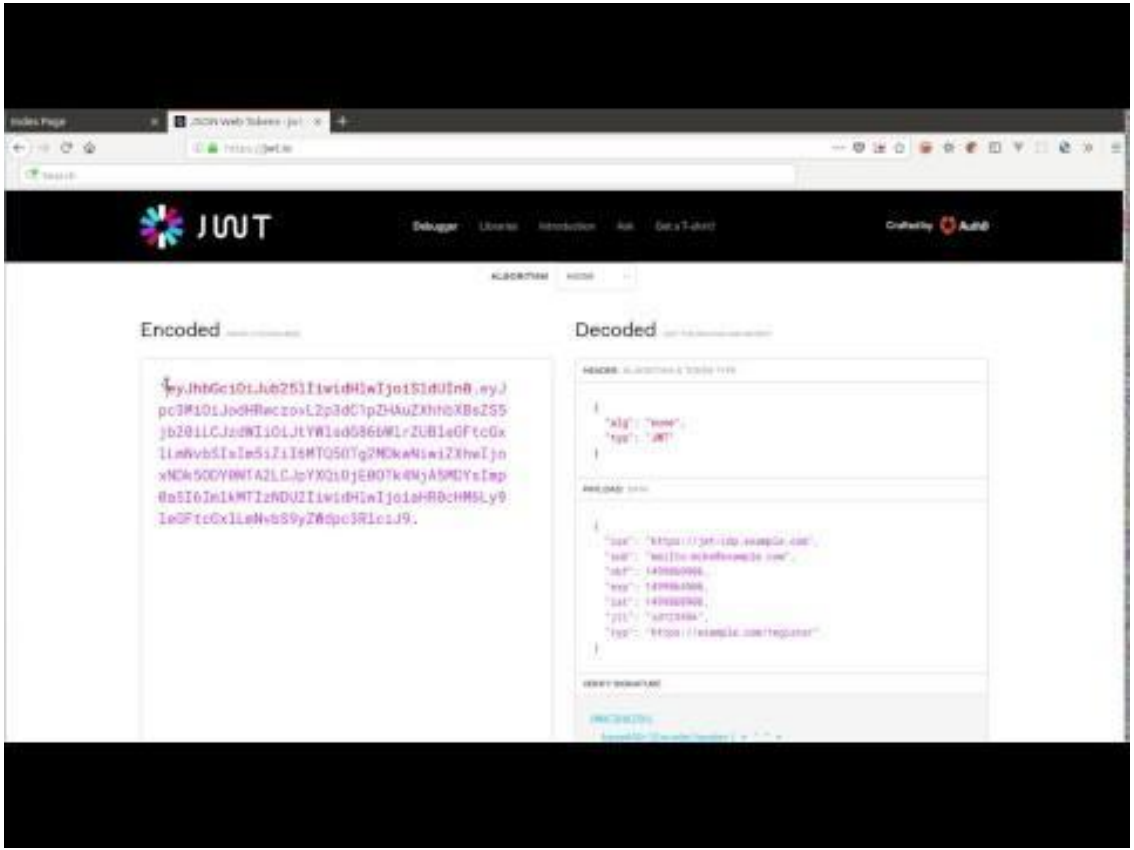
PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

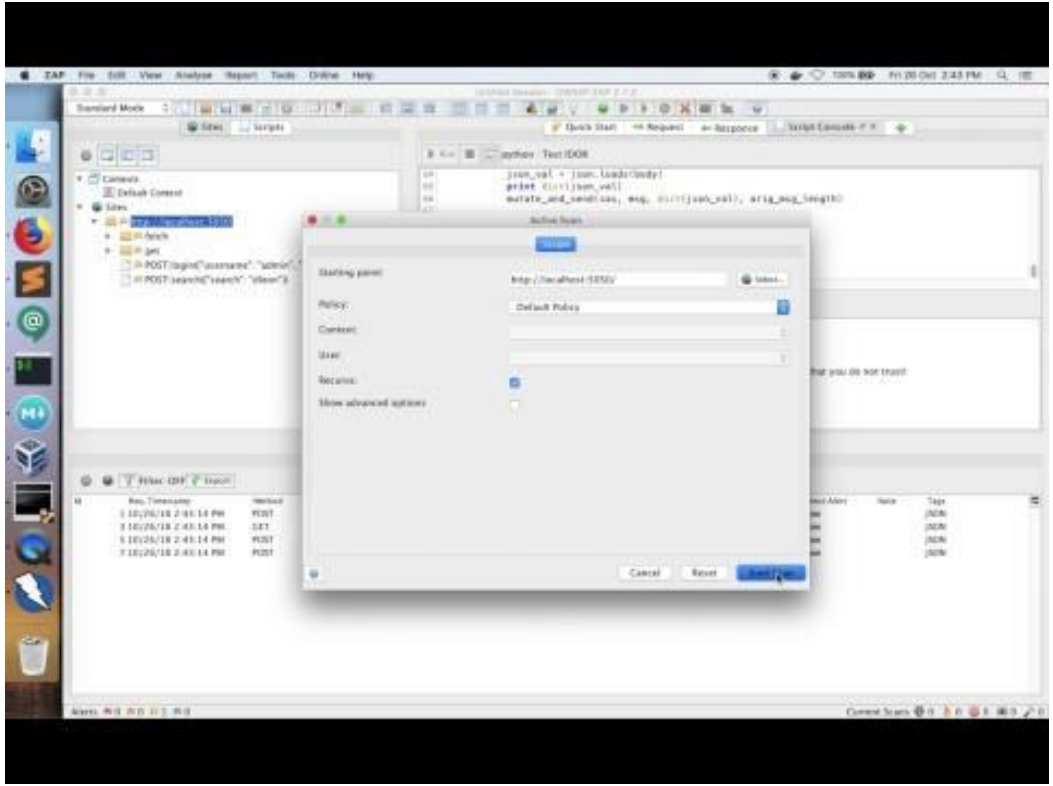
VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

ZAP Active Scan Rule - Demo



ZAP Scripting - Custom Exploits



Parameterized DAST scans with ZAP

- Leverage functional automation scripts to “walkthrough” the application.
- Session and Context help ZAP in understanding the target application
- Achieve this through
  - Selenium Walkthrough scripts used by Automation/QA Teams
  - Robot scripts to perform walkthrough and trigger ZAP active scans.

ZAP Automation using PySelenium

Demo

ZAP Automation using Robot Framework

Demo

ZEST Automation Framework.

References

Download ZAP : https://github.com/zaproxy/zaproxy/wiki/Downloads

ZAP Help : https://github.com/zaproxy/zap-core-help/wiki

Community Scripts: https://github.com/zaproxy/community-scripts