Project document

Most popular web application attacks

SQL Injection

Parametized statements

The key difference is the data being passed to the executeQuery(...) method. In the first case, the parameterized string and the parameters are passed to the database separately, which allows the driver to correctly interpret them. In the second case, the full SQL statement is constructed before the driver is invoked, meaning we are vulnerable to maliciously crafted parameters.

Sanitizing inputs -

Broken Authentication

-Using old session id from computer

-automated attacks with default passwords

-automated attacks with most popular users + passwords called credential stuffing

Solution

Multi factor auth – fingerprint, eye scanner etc..

Limit login attempts to stop automated attacks/credential stuffing

Password complexity

What I am doing

Automating web testing

Why automated over manual testing?

Increases test coverage –

Test automation can easily execute thousands of different complex test cases during every test run which is impossible with manual tests.

Improved accuracy –

When doing manual tests there will always be mistakes at some stage. Automated takes out those mistakes.

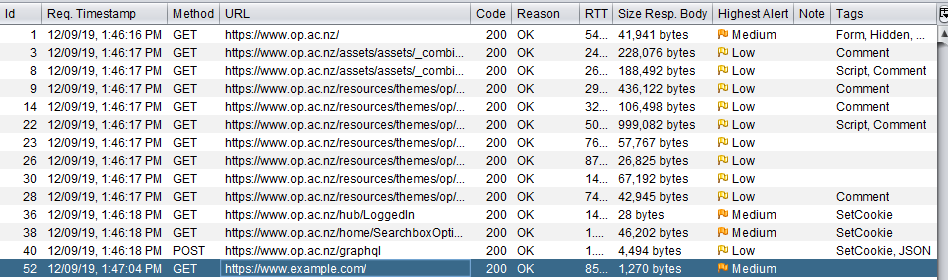
Helps developers and testers –

Tests can run automatically whenever source code changes are checked in and notify the team or developer if they fail.

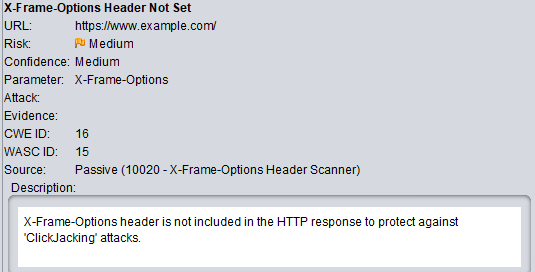
Setting up Zap with browser as proxy

Configuring ZAP to act as a proxy when using a browser.

* Anything clicked on the browser will automatically be run against a passive scan and output will be recorded in the history tab on ZAP.



* Clicking on the Alerts tab will show any immediate problems with the scanned URL.



* Example error shown for my example website.

docker pull owasp/zap2docker-weekly

docker pull owasp/zap2docker-stable

docker run -t owasp/zap2docker-stable zap-baseline.py -t <https://www.example.com>

-t parameter is that target website

docker run -u zap -p 8080:8080 -i owasp/zap2docker-stable zap.sh -daemon -host 0.0.0.0 -port 8080 -config api.addrs.addr.name=.\* -config api.addrs.addr.regex=true -config api.key=<api-key>

-u URL of config file

-daemon starts ZAP in daemon mode without a UI

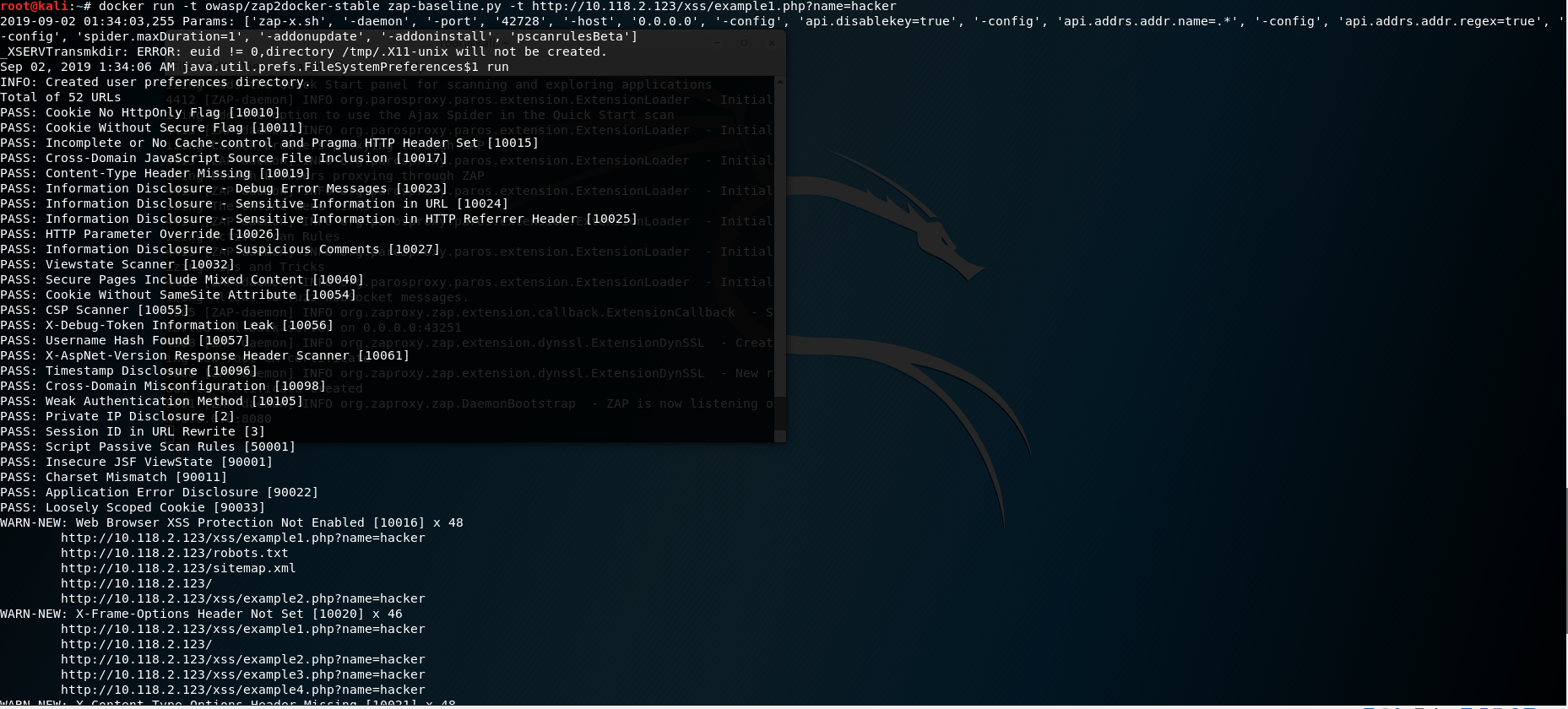
-config Overrides the specified key=value pair in the config file.

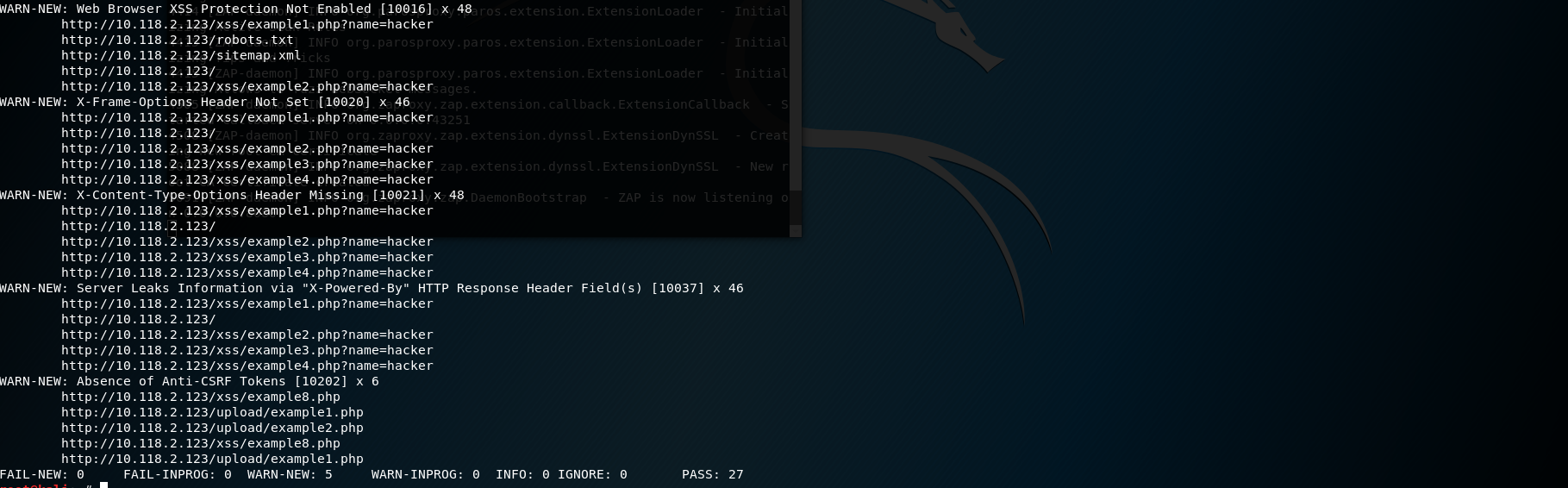
-host overrides the host used for proxying the specified in the config file.

-port overrides the port specified in the config file.

Depending on the pulled document replace “stable” with weekly.

Below is the baseline stable output on the example website.





Important Output to note

Cookie no secure flag

The secure flag is an option that can be set by the application server when sending a new cookie to the user within an HTTP Response. The purpose of the secure flag is to prevent cookies from being observed by unauthorized parties due to the transmission of a the cookie in clear text.

To accomplish this goal, browsers which support the secure flag will only send cookies with the secure flag when the request is going to a HTTPS page. Said in another way, the browser will not send a cookie with the secure flag set over an unencrypted HTTP request.

By setting the secure flag, the browser will prevent the transmission of a cookie over an unencrypted channel.

XSS protection not enabled

XSS protection is designed to enable the XSS scripting filter built into modern web browsers. Usually enabled by default but using it will enforce it. Will enable the browser to block the response in the event that a malicious script has been inserted from user input instead of sanitizing.

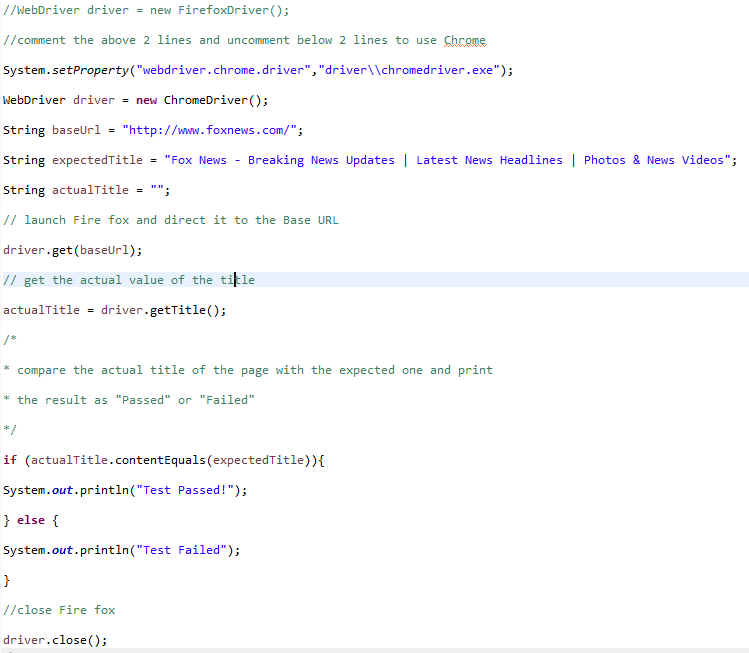
HTTP only enabled

The cookie cannot be accessed through client side script ( if the browser supports the flag ). Browser will not reveal the cookie to a third party if a user accidently accesses a link that exploits the flaw.

Absence of anti-csrf tokens

Sets a token value that the attacker will not know, without this the attacker will not be able to use an attack, for example when the web browser sends a POST request the attacker might know the syntax of this request, the attacker could you code to publish an advertisement on the users profile. Token sets a token in the session cookie of your web browser right after you log in. All form submissions then include a hidden field containing the token.

Using Selenium to run automated scans



Running this code with automatically open and run <http://foxnews.com/>”

It is also possible to code ZAP to open and run automatically and start scanning anything that is ran through the following code.