

SiteScan

A Linux-based web app basic security
scanner scripting project



Abertay
University

Thomas Sinclair

CMP320 (Advanced Ethical Hacking)

BSc Ethical Hacking – Year 3

2022/23

Abstract

Web applications are the inescapable interfaces of the Internet for billions of people around the world, facilitating interaction between millions of organisations, individuals, and other entities every day. This makes them incredibly lucrative targets for aspiring cyber-criminals. Losses from cyber security attacks can be devastating for any victim, with repercussions being especially severe for organisations which manage sensitive information. Combined, these factors illustrate why the security of said applications is so paramount.

The rapid growth in the number of web applications and demand for cyber-security professionals to thoroughly test them is outweighing the current supply, however. An interim load-balancing approach is required to address this, utilising software tools to automate standard menial tasks in the testing process; more efficiently distributing workload and increasing productivity.

This project investigates automating many of these more ‘basic’ tests by developing a Python script to perform these actions. This report is a documentation and critical review of this. The background of, and justification for, the project is first discussed, as well as the tools used by the script. The tests carried out by the script are described alongside a relevant professional procedure manual and their relation to it. The code of the script is then explained, divided by function (subroutine). The testing platform is described, with demonstrative examples of operation given. Finally, the script is critically evaluated, concerning its technical (code) and design (usability) aspects, and opportunities for improvement and future work are considered.

The resulting product is a small suite of data-gathering tools, with 10 modules testing for cookies usage, code comments, default web server files, server filesystem directories, encryption usage, data entry points, HTTP header information, general server vulnerabilities, web technologies, and web application structure. These tests are in fulfilment of some steps outlined in the OWASP Web Security Testing Guide. Findings are collated into an output report document.

The script is command-line-based, with intuitive options and informative user communication, and utilises thorough input validation and error handling at all stages. It follows good coding practices overall and features highly flexible, adaptable design wherever possible.

Table of Contents

1 INTRODUCTION	5
1.1 BACKGROUND	5
1.2 AIMS.....	7
1.3 DESIGN OVERVIEW	8
1.3.1 THE IDEA	8
1.3.2 METHODOLOGY STEPS COVERED	9
1.3.3 TESTING TOOLS FEATURED	10
2 DEVELOPMENT	11
2.1 PROCEDURE.....	11
2.1.1 GENERAL NOTES.....	11
2.1.2 MAIN FUNCTION	11
2.1.3 PARSE ARGUMENTS (GETARGS)	12
2.1.4 INITIALISATION CHECKS (INITCHECKS).....	13
2.1.4.1 STARTUP ARGS SUPPLIED	13
2.1.4.2 ALL LOGIN CREDENTIALS SUPPLIED	14
2.1.4.3 INITIALISE WORKING DIRECTORY	14
2.1.4.4 MODIFY NORMAL SCRIPT FUNCTIONALITY	14
2.1.4.5 DEPENDANT TESTS.....	14
2.1.4.6 ARE REQUIRED TOOLS INSTALLED	15
2.1.4.7 SET TEST RUNNING ORDER.....	15
2.1.4.8 TARGET TYPE IDENTIFICATION	15
2.1.4.9 URL TARGET CLEANING	16
2.1.4.10 TARGET PORT IDENTIFICATION	16
2.1.4.11 URL TYPE MISIDENTIFICATION CHECK.....	16
2.1.4.12 CREATE FULL (URL-LIKE) TARGET.....	16
2.1.4.13 ENSURE TARGET IS ONLINE.....	16
2.1.4.14 ENSURE TARGET PORT IS OPEN.....	17
2.1.4.15 DIRECTORY CREATION	17
2.1.5 COOKIES USAGE TEST (TEST_COOKIES).....	17
2.1.6 CODE COMMENTS SEARCH (TEST_COMMENTS)	17
2.1.7 DEFAULT FILES SEARCH (TEST_DEFAULTS)	19
2.1.8 DIRB DIRECTORY BRUTEFORCE SCAN (TEST_DIRB)	19

2.1.9 ENCRYPTION USAGE TEST (TEST_ENCRYPTION).....	20
2.1.10 ENTRY POINT SEARCH (TEST_ENTRY).....	21
2.1.11 HTTP HEADER TEST (TEST_HEADER)	21
2.1.12 NIKTO WEBSERVER SCAN (TEST_NIKTO)	22
2.1.13 WHATWEB WEBSERVER SCAN (TEST_WHATWEB).....	22
2.1.14 WGET SPIDER & FILE DOWNLOADER (TEST_WGET)	22
2.1.15 VARIOUS NON-PROCEDURAL FUNCTIONS	23
2.1.15.1 CHECK IF A COMMAND EXISTS (COMMANDEXISTS)	23
2.1.15.2 CHECK IF A VALUE IS VALID (ISVALIDVALUE)	23
2.1.15.3 WRITE TEXT TO A FILE (WRITETOFILE).....	24
2.1.15.4 LIST FILES IN A DIRECTORY (LISTFILES).....	24
2.2 TESTING AND RESULTS	26
2.2.1 TESTING	26
2.2.2 RESULTS.....	26
2.2.2.1 EXAMPLE 1	26
2.2.2.2 EXAMPLE 2.....	28
3 DISCUSSION.....	30
3.1 CRITICAL ANALYSIS.....	30
3.1.1 THE CODE	30
3.1.1.1 POSITIVES.....	30
3.1.1.2 NEGATIVES	31
3.1.2 THE USER EXPERIENCE.....	31
3.1.2.1 POSITIVES.....	31
3.1.2.2 NEGATIVES	31
3.2 FUTURE WORK.....	32
3.3 FULFILMENT OF AIMS.....	34
4 REFERENCES	35
5 APPENDICES	36
5.1 APPENDIX A – RESULTS EXAMPLE 1 REPORT	36

1 INTRODUCTION

1.1 BACKGROUND

In today's world, the use of the Internet is practically unavoidable for those in society. This is the case for both individuals and organisations for varying applicable reasons, not to mention the universal rapid acceleratory impact of the COVID-19 global pandemic. One of the main interfaces of the internet is that of websites/web apps, commonly those which handle sensitive information, making them lucrative targets for cyber-criminals. In their 'Data Breach Investigations Report 2022' Verizon noted that over 60% of the 18,000 security incidents reported to them were associated with hacks of basic web applications (Thomas, 2022).

Organisations that are victims of such cyber-attacks, can – and often do – face serious consequences as a result of things such as personal user data leaks; names, email addresses, passwords, payment information, etc. Not only does this come with reputational damage, but remediation costs (both financial and temporal), legal action, and fines are common. In particularly severe cases it can result in the halting of activities – potentially permanently. These impacts – particularly financial – have only been worsening over the years, and are expected to worsen further (Figure 1).

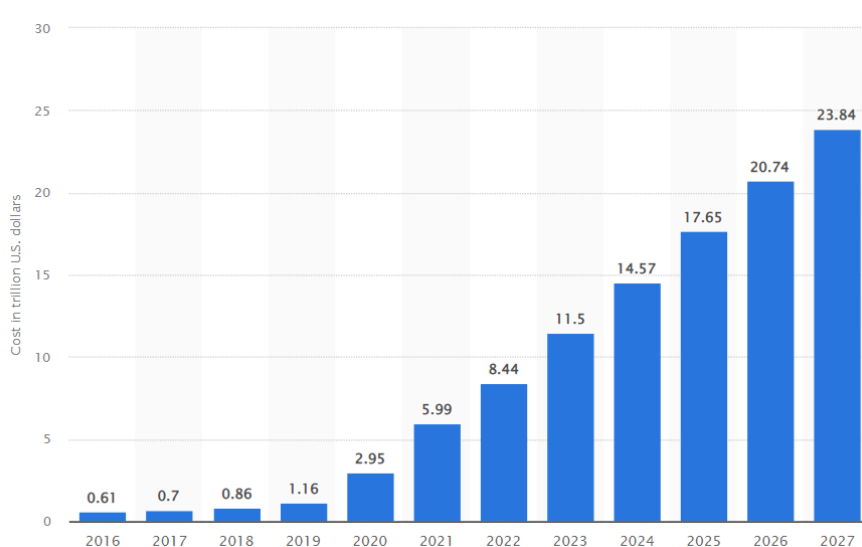


Figure 1 | Estimated cost of cybercrime worldwide from 2016 to 2027 (Petrosyan, 2022)

Understandably, these organisations are attempting to thwart criminals by developing robust cyber-security defences for their web applications. One of the best ways to test that these defences are fit for purpose is to perform a “penetration test” – an authorised, simulated, practical test which aims to exploit vulnerabilities and discover the degree to which a malicious attacker could gain unauthorised access to the system.

As a result of application architecture complexity, attack sophistication, and the substantial scale of potential consequences, modern-day penetration tests are very comprehensive. This, of course, takes time and security professionals to conduct. As the number of web applications rapidly increases (*Figure 2*), so too does the demand for security professionals to perform these tests.

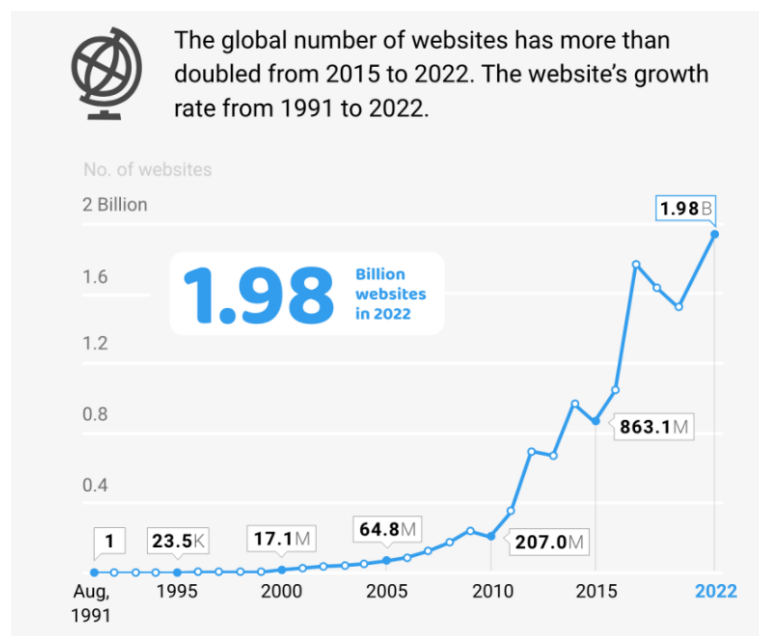


Figure 2 | Estimated number of websites in the world from 1991 to 2022 (Djuraskovic, 2023)

This demand, however, is greatly outweighing supply, so any efforts to reduce the workload for these security professionals – at least in the interim – would be greatly effective at improving overall productivity and the number of clients served. Whilst software suites exist which automate much of this to an extent, there are often many steps in various testing methodologies that are typically completed without automation. Some of these are due to their complexity, but many are simply standard or routine checks and are relatively menial.

This project investigates automating many of these more ‘basic’ tests.

1.2 AIMS

This project aims to develop and document a web app basic security scanning script. This overall aim encompasses several sub-aims, respective to the practical script development and written documentation work portions of the project. The former of these concerns the technical requirements that the script must meet.

- Develop a command-line-based Python script that will:
 - Conduct appropriate preliminary investigative security tests, as defined in the OWASP Web Security Testing Guide
 - Output results in a readable format, and through an appropriate medium
 - Automate all tasks, where possible and appropriate
- Describe the project background and justify its undertaking
- Describe the OWASP Web Security Testing Guide and explain its relation to the project
- Explain the script's code, demonstrating where appropriate
- Describe the testing procedure and demonstrate the results
- Critically evaluate the script's technical and usability aspects
- Consider future work to be investigated, pertaining to the script

1.3 DESIGN OVERVIEW

1.3.1 The Idea

For their security scripting project, the developer (writer) decided to investigate the ability to automate the routine, often tedious tasks associated with web app (application) security testing. The hope is that such a tool could be used by applicable security professionals to improve their efficiency and workflow in a world increasingly desperate to enlist their skills.

It is best practice to follow a specified testing methodology to carry out penetration tests, one example of this being the OWASP Web Application Security Testing methodology, produced as part of the OWASP Web Security Testing Guide (*OWASP, 2020*). This methodology is comprehensive and produced by the OWASP foundation – a highly reputable and accredited institution specialising in web application security – and was consequently chosen as the most appropriate to base this tool upon. The stable version 4.2 of this was chosen in place of the in-development version 5.0, due to the potential for change.

The functionality of this tool is not limited solely to this methodology though, as the tests it performs are relatively standard and applicable to other many use-cases. The OWASP methodology steps covered are discussed further in section 1.3.2. The required tools used by the script are discussed further in section 1.3.3.

The script was designed to be operated from the command-line without a GUI, similar to many penetration-testing tools, making it lightweight and easier to run. Each test can be run by use of individual options/switches to enable them and will be run in-sequence automatically. The specified target to investigate can be either a URL, IP address, or Domain. The results of the tests are formatted in a generated report.

The feature list is as follows, including the identification and investigation of:

- Code comments
- Cookie usage
- Data entry points
- Default files
- Encryption usage
- HTTP Header information
- Information leakage
- Server technologies
- Storage directory information
- Website structure

Aside from the tools themselves – discussed in section 1.3.3 – the only requirement is the Python programming language, which is required to execute the script (written in Python v3.10). The tool is primarily targeted for use on the Kali Linux operating system (which has all the required tools pre-installed) however will work for all Linux versions.

1.3.2 Methodology Steps Covered

The OWASP methodology consists of 12 steps (“sections”), however only some contained material applicable to this tool. With more advanced tools, other elements may be automatable, however, this is out-of-scope for this scripting project. The sections and sub-sections covered are listed below, along with their corresponding documental numeric identifier:

4.1 | Information Gathering

- 4.1.2 | Fingerprint Web Server
- 4.1.3 | Review Webserver Metafiles for Information Leakage
- 4.1.5 | Review Webpage Content for Information Leakage
- 4.1.6 | Identify Application Entry Points
- 4.1.7 | Map Execution Paths Through Application
- 4.1.8 | Fingerprint Web Application Framework

4.2 | Configuration and Deployment Management Testing

- 4.2.2 | Test Application Platform Configuration
- 4.2.3 | Test File Extensions Handling for Sensitive Information
- 4.2.5 | Enumerate Infrastructure and Application Admin Interfaces
- 4.2.6 | Test HTTP Methods
- 4.2.7 | Test HTTP Strict Transport Security

4.6 | Session Management Testing

- 4.6.1 | Testing for Session Management Schema
- 4.6.2 | Testing for Cookies Attributes

4.9 | Testing for Weak Cryptography

- 4.9.1 | Testing for Weak Transport Layer Security

1.3.3 Testing Tools Featured

Various software tools are used by the script to provide the functionality listed in section 1.3.1 prior. These tools are not included with the script in order to keep it lightweight and provide flexibility – a test can only be performed if the corresponding tool is detected as being installed on the machine. Operating systems like Kali Linux – a specialised operating system for and used by security professionals – come with all these required tools pre-installed.

cURL – Command-line tool and library for transferring data with URL syntax.

Dirb – Directory and filename brute-force searcher.

Nikto – Web server (web application) vulnerability scanner.

SSLScan – SSL service cipher enumeration (detection) tool.

wget – A tool that retrieves (downloads) content from web servers. Can also perform a web spidering function.

WhatWeb – Web application scanner. Identifies the web technologies running on the target site.

2 DEVELOPMENT

2.1 PROCEDURE

2.1.1 General Notes

The script is written in Python 3 and comprises a singular file. This file contains many functions – some referred to as “procedural” (core tasks that run as part of normal operation), and others as “functional” (supplemental tasks that perform a specific operation; generally used to reduce code repetition).

This means that the script uses the ‘functional programming’ paradigm, as opposed to the ‘object-oriented programming’ paradigm – this is discussed more in the discussion.

All code makes thorough use of commenting to explain code, and all functions use Python’s try-except coding technique to manage errors. In most cases, if an unresolvable error is encountered, it will simply call `sys.exit()` to stop the script.

2.1.2 Main Function

This is the main controlling function that calls each of the procedural/core functions in turn. This begins by taking note of the current time (script launch time), then calling `getArgs()` to parse the command line arguments, and `initChecks()` to perform various initialisation tasks.

Once these have been executed successfully, a 15-second countdown initiates to give users a chance to review the `initChecks()` output messages and abort execution if anything needs to be changed. This timer essentially updates the same command line for a better-looking output.

It then uses the `writeToFile()` function to create an opening header for the output report file, containing information about the current scan, such as the target and the scans being run.

Each target is then run in turn by calling their respective functions, according to a running order created by `initChecks()`.

Finally, the output report is ended, and its location is printed to the console before the system calls `sys.exit()` to end the execution of the script.

2.1.3 Parse Arguments (getArgs)

This function uses Python's standard 'argparse' library to detect and parse the command-line arguments supplied when the script is run. Whilst this contains 'groups' for required and optional arguments, due to the complex nature of what was needed for the script, all but one of the arguments are classed as optional – even if sometimes they are required for other components to function. Any discrepancies get picked up in `initChecks()` afterwards though.

All arguments are stored in respective variables, with some (such as `runTest`) grouped using dictionaries for better usability. See the Discussion for more comments on this. Descriptions of each option also creates a help menu ("`-h`", see *Figure 3*). All arguments except 'target' require an option/switch to enter data; 'target' can go anywhere.

```
(kali@kali)-[~]
$ python3.10 test.py -h
usage: [-h] [-p {80,443}] [-login LOGINPATH] [-user USERNAME] [-pass PASSWORD] [-M] [-all] [-ck] [-cmt] [-df] [-dir] [-enc] [-ent] [-hd] [-nk] [-ww] [-wg] target

options:
  -h, --help            show this help message and exit

required arguments:
  target                The target IPv4 address, HTTP(S) URL, or Domain

optional arguments:
  -p {80,443}, --port {80,443}
                        Specific application port/protocol to target (takes precedence!)
  -login LOGINPATH, --loginpath LOGINPATH
                        Relative path to login page (e.g. '/login/')
  -user USERNAME, --username USERNAME
                        Username to use for login
  -pass PASSWORD, --password PASSWORD
                        Password to use for login
  -M, --modify           Modify/Edit normal script behaviour
                        ...
  -all, --runall         Run ALL tests (takes precedence!)
  -ck, --cookies         Run Cookie usage test
  -cmt, --comments       Run Code Comments search
  -df, --defaults        Run Default Files search
  -dir, --dirb           Run Dirb scan
  -enc, --encryption     Run Encryption analysis
  -ent, --entry          Run Entry Point search
  -hd, --header          Run HTTP Header analysis
  -nk, --nikto           Run Nikto scan
  -ww, --whatweb         Run WhatWeb scan
  -wg, --wget            Run Wget spider scan and file download (all)
```

Figure 3 | Help menu, viewable by using the "-h" option/argument

Input validation is difficult at this stage, so is mostly left to `initChecks()` afterwards. Some arguments (`target`, `port`, `login{}`) are stored as their supplied string values, others (`modifyBehaviour`, `runTest{}`) as boolean values.

If an argument is not supplied, the value will become `None`, or in the case of booleans will be set to `False` (`action="store_true"` – see the code comments).

All elements of `runTest{}` are set to `True` by default so that “-all” can be used, this value gets updated accordingly – if the argument isn’t supplied, the `runTest[test] = False`.

2.1.4 Initialisation Checks (`initChecks`)

This function contains various input validation checks and data initialisation tasks, which are detailed below.

Later in the function, a series of tasks check that the supplied target information is correct. Due to this taking several stages, and being an uncommon thing to get wrong (at least in comparison to something like not knowing that test X needs test Y run before it), these checks were placed towards the end.

Together, these checks allow for a great range of target input formats, whilst still validating that only the appropriate inputs get through. For example, “`http://1.1.1.1`”, “`1.1.1.1`”, “`http://google.com`”, “`google.com`” etc. The only input it cannot currently handle is having the port number at the end of the string (e.g., “`1.1.1.1:80`”) – this needs to be input separately if not already specified using the leading protocol. See the Discussion for comments on this.

2.1.4.1 *Startup Args Supplied*

Because of how the argument parser works, it is possible that no test/module has been specified at all – perform a **check to ensure that at least one test has been specified in the startup args**. Check if none of the values in the `runTest{}` dictionary are `True`.

2.1.4.2 All Login Credentials Supplied

Because of how the argument parser works, it is possible that no test/module has been specified at all – perform a **check to ensure that at least one test has been specified in the startup args**. Check if none of the values in the `runTest{}` dictionary are True.

2.1.4.3 Initialise Working Directory

Initialise the script's working directory, where it will store data and output files. Set it to the current working directory (where the script is) + the `"/SiteScan/"` directory.

2.1.4.4 Modify Normal Script Functionality

Check if the user wants to modify normal functionality. If yes, this uses the 'pyinputplus' library to get yes/no/string inputs – this handles applicable input validation and repeats until it gets a satisfactory response. For ease, this has been configured to allow a blank response as a "no" response.

This contains controls for whether all tests should be output in the report, regardless of whether they have been specified in the startup args (`outputAllTests` – some tests require others run before them, which is done automatically), as well as if the working directory and output report filename should be changed.

This library had some quirks with how it operates (such as returning the user-entered value for a yes/no input, and not having a way to change this to return as just a boolean) but its other options and code-simplification still made it a more elegant solution than creating an input validation system from scratch.

2.1.4.5 Dependant Tests

Check for any test dependencies – meaning, do any specific tools require another tool to be run before it? This code has been designed to be easily copyable for future use, with lists for the dependant tests and their corresponding dependencies.

It loops through all dependant tests, if that test has been specified to run, it loops through the dependency tests and ensures they will run (which may be the case already anyway). It also sets the dependency tests to the front of the running order (in

the order they were defined in that local list) so that they will be run first. This code also considers the `outputAllTests` modification option.

This code is such that it can be re-used, even with the ability to manage dependencies of dependencies – the block of code just needs to be placed before another and the running order will manage itself accordingly.

2.1.4.6 Are Required Tools Installed

Now that it is known what tests need to be run, it's worth checking whether they can be run at all – **are the tools required for each test installed and executable through Path?** This just runs through any tests that require a tool to function and calls the `commandExists()` function to check if the corresponding command is available. An optional argument makes this function raise an exception and stop execution if not.

2.1.4.7 Set Test Running Order

Now, **set the running order for the tests**. Loop through all tests, if it's specified to run and isn't already in the running order, insert it into the order. This insert runs on the order that the `runTest` dictionary keys were defined in `getArgs()`.

2.1.4.8 Target Type Identification

The first check **ensures that the given target is a valid IPv4 address, URL, or Domain**. This uses the `isValidValue()` function, which itself uses the 'validators' library – which does exactly what's needed. Using a specific logical order for the `if/elif` statements, this is checked.

A flag is set to indicate if 'validators' thinks it is a URL – it does this solely based on whether there is a protocol (`://`) in it, so has the potential to misidentify an IP address entered as "`http://1.1.1.1`" as a URL. This is dealt with later.

The `isValidValue()` function contains a condition to check whether the supplied URL is a HTTP(S) web address, so will not allow any others such as FTP through.

2.1.4.9 URL Target Cleaning

The **URL target is next “cleaned”**, using string searching to specifically look for patterns and remove unwanted parts. Details are in the code comments, but this includes removing the HTTP protocol, suspected port identifier, query parameters, and JavaScript IDs. It stores this as `target_full`, then removes the protocol identifier to turn it into a domain (or just misidentified IP).

2.1.4.10 Target Port Identification

The **port to be used is identified** using either the protocol in the URL string, arg-specified port, or just defaulting to port 80 (HTTP) otherwise. The prior validation means that only ports 80 or 443 (HTTPS) will be selected here.

2.1.4.11 URL Type Misidentification Check

The **URL is then checked for type misidentification** by running the “cleaned” target (now just a domain or IPv4) through the `isValidValue()` function again for IPv4. Since it might still just be a domain, the `isValidValue()` domain check is performed if needed, which catches all errors and correctly identifies the type.

2.1.4.12 Create Full (URL-like) Target

By this stage, the target has been confirmed as a valid IP address or domain, so **create a full target (target with protocol) based on the port**, which will have been set by one mechanism or another by now. This is primarily here in case only a domain was supplied, with no protocol or arg-port.

2.1.4.13 Ensure Target Is Online

Finally, the target validation checks are finished. Next is it worth **ensuring that the target is online**. This runs a simple `ping` command to send 4 ICMP probes to the target, with a subprocess timeout of 10 seconds. Any errors with the subprocess (which runs the command) are handled, with error messages returned or explained.

2.1.4.14 Ensure Target Port Is Open

Now **ensure that the target port is open** by utilising an in-built Python method of socket connection. This time, timeout after 5 seconds as it is just one connection.

2.1.4.15 Directory Creation

All initialisation checks are complete, it is now appropriate to **create the required directories** to store data and files in. All tasks are now complete for this function.

2.1.5 Cookies Usage Test (test_cookies)

This test is designed to extract the cookies in use on the site as well as their attributes, using Python's 'requests' library. It creates a 'requests' 'session' object to store session data in, and sends a standard GET request to the main web app page, storing the response.

If login information has been specified in the startup arguments, this is used to retrieve any cookies that only appear once logged in. Without knowing the exact login form details, it has to effectively guess what they are – the standard/most common ones are "username" and "email", so both of these fields are used to hopefully get a match.

A POST request is made to the arg-supplied login page, and hopefully, a session is created (login successful), in which case any cookies are stored in the session object. This is all done before the standard GET request mentioned prior.

If any were returned, all cookies are looped through, and data such as their name, value, and attributes are output to the report file. The list of cookie attributes is defined as a list for ease and better overall coding practice.

2.1.6 Code Comments Search (test_comments)

This test is designed to search through the downloaded files for occurrences of common web-related comment strings (such as "`//`", "`<!--`", and "`/*`"). This requires

the Dirb and Wget tests to be run before it, the latter of which generates a list of the locally-downloaded file paths.

A list of 'strings to find' is defined, along with a list of 'strings to skip' – this is intended to ignore any sub-string matches that are known to be something else (such as `“//”` being detected in `“://”` – which is a URL protocol identifier, not a comment).

The list of downloaded file paths is opened and iterated through. Each line is a file path, the file at that path is opened and each line of that is searched for the 'string to find'. If found, it checks that none of the 'strings to skip' match – if they do it just moves on to the next 'string to find'; if not it will write the finding to the output report, then move on to the next line – this is to stop duplicate results (e.g., `“/*”` and `“*/”` appear in the same line as they are complements). It then opens the next file and so on...

Line output is limited to 150 characters from the identified string (comment identifier), as the output is only meant to serve as an introductory tool to discover things of note, which can then be investigated in-depth afterwards. This also helps in case a file contains a very long string that is not broken up by new lines.

Libraries for this purpose do exist, such as the one originally tried by the script developer called 'comment-parser'. Unfortunately, this was not fit for purpose. It worked by using Linux MIME types to automatically identify what type of file it was, in order to know what language comments to search for (it could not search for all languages regardless of type). If it did not recognise the MIME type, it would simply crash outright due to an unsupported type.

Amazingly, this meant it could not search plain text files. For example, CSS files are classified as the same MIME type as TXT, so it would not be able to search for CSS comments (it does not specifically have this ability anyway). One way around this was to construct some solution where if you knew the comment format, and there happened to be an overlap between languages, you could tell it that the file was a specific MIME type – for example, CSS and JavaScript have similar comments so you could say the `.css` file is actually an `“application/javascript”` MIME and it would work.

This could be automated to search for all MIME types, but that would largely waste resources by searching for C++ comments in a HTML file, for example. Not to mention

if there is an overlap in comments it would repeat itself (also, it treats HTML and XML comments as different for some reason).

With all that considered, it was much more appropriate to use a relatively simple self-made solution, than overcomplicate things with a poor library.

2.1.7 Default Files Search (test_defaults)

This test is designed to search through the list of downloaded files (not the files themselves) for occurrences of certain file names/paths – in this case, common or default web server resources. This requires the Dirb and Wget tests to be run prior.

A list of exact strings to find is defined alongside a list of general (non-exact/sub-string) strings to find. The idea behind this being someone could specify an exact path where they know a file always is (“/robots.txt”) and search for other files which may exist anywhere (not just the root directory, for example).

The list of downloaded files is opened, and each line is searched for exact matches of each ‘exact string’ and sub-string matches for each ‘general string’. If either is found, the finding is output to the report file.

2.1.8 Dirb Directory Brute-force Scan (test_dirb)

This test is designed to perform a ‘Dirb’ directory brute-forcing scan to enumerate any directories on the webserver. This uses the Dirb ‘common.txt’ file string list, instead of pure brute-forcing – this could be changed in the code though. If login credentials have been supplied, Dirb will use them in case any directories are protected – though this does not necessarily create a persistent session like `test_cookies()`.

Dirb seemingly doesn’t have a clean method of outputting what directories it found, instead outputting the logs of what it’s found, which contains a lot of unwanted data for this purpose. Even this output contains ‘duplicates’, so that needs handling as well.

The output of the `dirb` command is saved to `./data/raw/dirb_raw-output.txt`. This is opened and each line is searched in turn. If the line doesn't contain the target address, or if it will just be a duplicate, ignore it. Otherwise, a complex block of code extracts the **relative path** directories and filenames from each line, allowing for spaces in names, multiple dots in filenames and paths, and other various Dirb output formatting.

Exact details are explained in the code comments, however, it is highly recommended to look at the 'dirb raw' file when you do so, so that it makes sense. For example, in the screenshot below (*Figure 4*), you can see examples of the ignored strings.

```
==> DIRECTORY: http://192.168.1.10/admin/includes/
+ http://192.168.1.10/admin/index.php (CODE:200|SIZE:1133)
==> DIRECTORY: http://192.168.1.10/admin/layout/
==> DIRECTORY: http://192.168.1.10/admin/uploads/

---- Entering directory: http://192.168.1.10/includes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
      (Use mode '-w' if you want to scan it anyway)
+ http://192.168.1.10/includes/admin.cgi (CODE:403|SIZE:975)
+ http://192.168.1.10/includes/admin.pl (CODE:403|SIZE:975)
```

Figure 4 | Example of Dirb scan raw output

The complex string searching essentially attempts to get everything from the end of the target to wherever the file truly ends, taking spaces and multiple dots in filenames into account, as well as whether it might just be a directory. Since the Dirb output only really has the 2 different outputs (ignoring certain ones remember), it is easier.

It does this by saving everything from the end of the target to the end of the line as "subLine", and then searching that for dots (to identify a file) or for slashes (to identify a directory). Once it thinks it has identified correctly, it will work out whether the string is on a line with a space or just the end of the line. It will also consider whether the path points to a file with no extension, and act accordingly.

Finally, it will only write to the output `dirb.txt` if the path is not just the root directory.

2.1.9 Encryption Usage Test (test_encryption)

This test is designed to investigate what encryption methods are used on the web app.

It firstly searches for the HTTP Strict Transport Security (HSTS) policy by using a `cURL` command to send a standard GET request to the target address and then using `grep` to case-insensitively extract the term “Strict-Transport-Security”. Whether this returns anything determines whether it is in use. This information is output to the report.

It then runs an `SSLScan` command to find out what security certificates are in use. The output of this command is sent to the report as well.

Whether the web app uses HTTPS is determined if a connection to port 443 can be made, which can be tested trivially and doesn’t really need to be checked here.

2.1.10 Entry Point Search (test_entry)

This test is designed to search through the downloaded files for occurrences of HTML form input tags and search through the list of downloaded files for occurrences of query parameters. This requires the Dirb and Wget tests to be run prior.

The list of downloaded files is opened, each line contains a file path, this file is opened, and each line is read in turn. If “<input “ is found in the line, it is written to the report output file along with its file path and line number within it.

The downloaded files list is opened again, this time searching each line (file path) for “?”. If that is found, output the file path to the output report file.

Like `test_comments()`, the output of the line is limited to 150 characters from the identified string, for the same reasons around report formatting and function.

2.1.11 HTTP Header Test (test_header)

This test is designed to display the HTTP Header returned by the home page of the target, using a standard GET request from the Python ‘requests’ library. Unlike `test_cookies()`, this does not require session objects as it is just a singular request. All returned headers are written to the report file on a new line.

2.1.12 Nikto Webserver Scan (test_nikto)

This test is designed to run a Nikto web server security scan. This was much like all the other commands run – use subprocess, capture the output like how it would be displayed in a console window, and write the output to the report file. This can take a minute(s), so a message is displayed informing the user, like was done in `test_dirb()`.

2.1.13 WhatWeb Webserver Scan (test_whatweb)

This test is designed to run a WhatWeb web server technologies enumeration scan. This is also like the other `subprocess.run(command)` tests, with the output here being split into new lines instead of commas for better readability.

2.1.14 Wget Spider & File Downloader (test_wget)

This test is designed to act as a web spider and file downloader, using the ‘Wget’ tool.

When run, it will [first] perform a recursive (follows links) scan and download against the target, saving the files to `“./data/<target>”`. The line `“2>&1”` in the command redirects all error messages to the output. Unfortunately, this causes the output to be sizeable, however without it, nothing is output at all.

The recursiveness behaviour causes Wget to use query parameters where it can, and one of those places is on index.html files on Apache web servers. These support a feature called “fancy indexing” (basically just directory sorting of index.html files), which Wget has a field day with and saves every different combination as a separate file, so one index.html turns into ten, which is unnecessary and wasteful. The line `“-R ‘*\?C+*’”` is used to detect any of these files and stop Wget from downloading them.

It will then check if a Dirb scan has been run (this should happen before it, so checking the `runTest[“dirb”]` variable is fine here). If it has, it will access the directories discovered by Dirb (listed in `dirb.txt`), performing a recursive (spider-like) download of each file. The `“-N”` option ensures that if the file already exists locally, it will only download a new copy if it is an updated version.

It will also check if a `robots.txt` file has been downloaded, and does the same download of files within that, for a comprehensive record of site content. Although this file's existence is checked later in `test_defaults()`, its content is too valuable to ignore here. Similar future developments, such as for `sitemap.xml`, can be found in the Discussion.

Once it has all files downloaded, it will call the `listFiles()` function to generate a record of all the files it has downloaded and stored. This is the file that gets used by Dirb/Wget's dependent tests. By default, this will download all file types, however adding a command specified in the code comments, Wget can be configured to reject specific files, such as multimedia (which could be large). Although only text-readable files are required for this, it was deemed that downloading all files by default would be more beneficial and equally as appropriate given the scenario/use case.

2.1.15 Various Non-Procedural Functions

2.1.15.1 Check If A Command Exists (`commandExists`)

This function uses the Python 'shutil' standard library to check whether a command is executable through Path. This takes 2 arguments – the command to check, and an optional boolean value (default to True) which determines whether the function raises an exception or returns 'False' in the event of a failure condition.

2.1.15.2 Check If A Value Is Valid (`isValidValue`)

This function uses the 'validators' library to check whether a supplied value matches the supplied type. This works with 3 'types': "ipv4", "url", and "domain" – each of which calls the applicable 'validators' function.

The URL checker also contains code to check whether the URL starts with the string "http" (checking that the protocol is HTTP(S)). The function will return the outcome status (True/False).

2.1.15.3 Write Text To A File (*writeToFile*)

This function is used to write specified 'text' to a specified 'file path'. It uses `with open` to automatically close the file after use. It takes 2 other (optional) arguments: `'textStyle'` and `'openMode'` – these default to `1`, and `"a"` respectively.

The former is used to add custom styling, which is used in this script for the output report file, with different styles for plain, capitals, a report section header, and a report section sub-header. This is the primary reason this is a function at all.

For ease of use, the value of this is converted to an integer to give leeway in case, for example, string `"2"` was accidentally passed in, instead of integer `2`.

The latter is used to determine what mode to open the file with – `"a"` means append, so the text will be added to the end of the file if it exists, and create the file if it doesn't. This is also used throughout the script in `"w"` mode, which will overwrite the file if it exists, and create it if it doesn't – this is used here to create files when first using them.

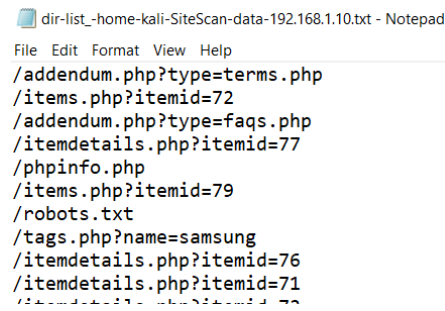
2.1.15.4 List Files In A Directory (*listFiles*)

This function is used to list the files (not directories) within a given directory (`'startPath'`) and generate an output text file. This takes 3 other (optional) arguments: `recursive` – which defines whether it lists recursively (or lists the supplied directory content only), `relative` – which defines whether it lists full or relative paths, and `mimeTypeStartsWith` – which defines what files it will list.

These are set to `True`, `True`, and `["text"]` (a Python list), respectively. For the latter, the default value causes it to only list files that are ASCII/UTF-8 etc encoded (those that could be read by a text editor) – for example `"text/plain"`, `"text/xml"`. This means that it will skip multimedia (`"image"`, `"video"`) and archive files (`"application"`) which are text-editor-unreadable. The input takes the form of a list, so multiple strings can be supplied and it will work like an OR statement in the `startswith()`.

The function works by first checking that the supplied directory points to an existent directory (not a file, or non-existent). If that is ok, it will begin working through a sorted list of all directories and files in that file path (or only files, if `recursive` is `False`). For each file, if it is of the correct type (`startswith()` MIME type), write the path (full or relative) to the output file.

This text file has the name “dir-list_” + the starting path (with slashes replaced by dashes). An example of how this output looks can be found in below *Figure 5*.

A screenshot of a Notepad window titled "dir-list_-home-kali-SiteScan-data-192.168.1.10.txt - Notepad". The window contains a list of file paths, each on a new line. The paths are: /addendum.php?type=terms.php, /items.php?itemid=72, /addendum.php?type=faqs.php, /itemdetails.php?itemid=77, /phpinfo.php, /items.php?itemid=79, /robots.txt, /tags.php?name=samsung, /itemdetails.php?itemid=76, /itemdetails.php?itemid=71, and /itemdetails.php?itemid=70. The text is in a monospaced font.

```
dir-list_-home-kali-SiteScan-data-192.168.1.10.txt - Notepad
File Edit Format View Help
/addendum.php?type=terms.php
/items.php?itemid=72
/addendum.php?type=faqs.php
/itemdetails.php?itemid=77
/phpinfo.php
/items.php?itemid=79
/robots.txt
/tags.php?name=samsung
/itemdetails.php?itemid=76
/itemdetails.php?itemid=71
/itemdetails.php?itemid=70
```

Figure 5 | Example of the contents of the generated directory file list txt

2.2 TESTING AND RESULTS

2.2.1 Testing

The script was tested throughout development, and again once the script was finished to identify any bugs/errors. Kali Linux (v2022.3) was used as this is specifically designed for penetration testing and contains all required tools pre-installed. The tools used, and the version that was used are as follows: cURL (7.84), dirb (2.22), Nikto (2.1.6), SSLScan (2.0.15), WhatWeb (0.5.5), and Wget (1.21.3). Python 3.10 was used.

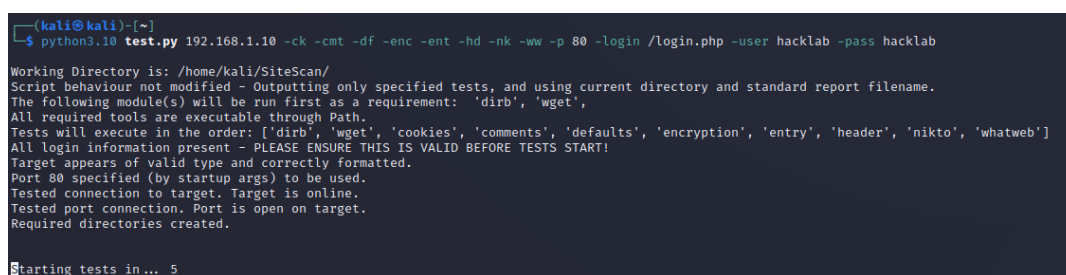
The developer used a pre-configured web server and vulnerable web application, sourced from their organisation (Abertay University). The exact details are not relevant, but the application was a simulated online market, with all the requisite features to test the script, hosted on 192.168.1.10 port 80. The web server used the Apache HTTP server and ran the Tiny Core Linux operating system.

Both the web server and Kali Linux machine were VMWare virtual machines hosted locally on and connected through the developer's PC.

2.2.2 Results

2.2.2.1 Example 1

Figure 6 below shows an example case of script usage, using the temporary development filename "test.py". The command line is shown to be targeting "192.168.1.10" on port 80, running modules: cookies, comments, defaults, encryption, entry points, HTTP header, nikto, and whatweb. Also shown are the supplied arguments for login information: the login page relative path, username, and password.



```
(kali@kali)~$ python3.10 test.py 192.168.1.10 -ck -cmt -df -enc -ent -hd -nk -ww -p 80 -login /login.php -user hacklab -pass hacklab
Working Directory is: /home/kali/SiteScan/
Script behaviour not modified - Outputting only specified tests, and using current directory and standard report filename.
The following module(s) will be run first as a requirement: 'dirb', 'wget',
All required tools are executable through Path.
Tests will execute in the order: ['dirb', 'wget', 'cookies', 'comments', 'defaults', 'encryption', 'entry', 'header', 'nikto', 'whatweb']
All login information present - PLEASE ENSURE THIS IS VALID BEFORE TESTS START!
Target appears of valid type and correctly formatted.
Port 80 specified (by startup args) to be used.
Tested connection to target. Target is online.
Tested port connection. Port is open on target.
Required directories created.
Starting tests in... 5
```

Figure 6 | Script command and initial output

This demonstrates the various initialisation messages that are printed, as well as how the countdown timer is displayed. It also shows how the script has identified the test dependencies and activated them automatically, as well as sorted the running order.

```
Required directories created.

Dirb scan starting...
(Rrecursive search is on, this may take a short while)
List of relative files and directories found in './data/dirb.txt'
Dirb scan complete.

Wget scan starting...
Dirb scan run previously. Scanning directories in ./data/dirb.txt...
Robots.txt file downloaded. Scanning those directories as well...
'/home/kali/SiteScan/data/192.168.1.10/' directory files listed in dir-list txt file in './data/'
Wget scan(s) complete.

Cookies test starting...
Cookies test complete.

Code Comments search starting...
Code Comments search complete.

Default Files search starting...
Default Files search complete.

Encryption analysis starting...
Encryption analysis complete.

Entry Points search starting...
Entry Points search complete.

HTTP Header analysis starting...
HTTP Header analysis complete.

Nikto scan starting...
(This may take a short while)
Nikto scan complete.

WhatWeb test starting...
WhatWeb test complete.

OUTPUT REPORT FOUND AT: /home/kali/SiteScan/Findings-Report.txt
```

Figure 7 | The rest of the script output, showing tests being run

Figure 7 above shows the following output when each test is run, including any informative messages that get displayed in the terminal window. A copy of the output report text can be found in *Appendix A*. As per standard operation, the non-arg-specified tests are omitted from the report.

Figures 8 and 9 show some of the directories and files created by the script. Part of the “dir-list” file can be seen previously in Figure 5 (section 2.1.15.4).

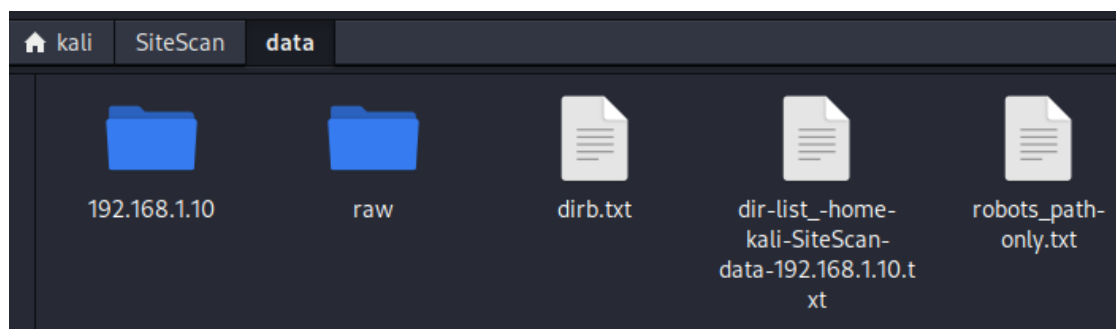


Figure 8 | SiteScan/data/ directory, showing directories and files created

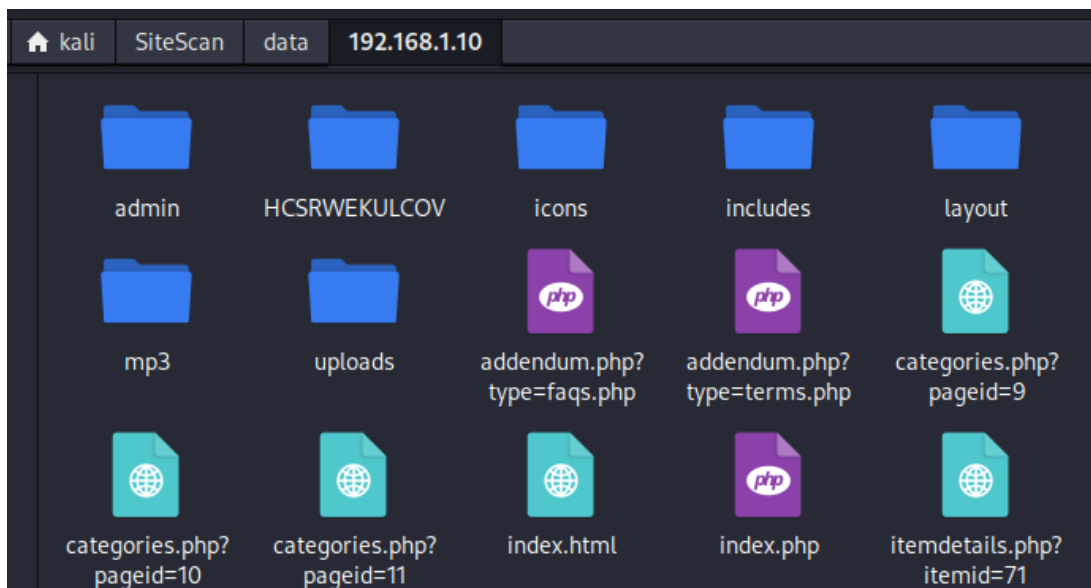


Figure 9 | Contents of the <target> directory, containing the files downloaded by Wget

2.2.2.2 Example 2

Figure 10 shows two commands. The first is an attempt to run the script with only the “-M” modify option, demonstrating how the first task in `initChecks()` (2.1.4.1) ensures that at least one test argument is supplied.

The second command demonstrates the script’s ability to handle the differently-formatted target – not only correctly identifying the target’s true type and correcting itself but also extracting the port to use from the URL protocol.

```
(kali@kali)-[~]
$ python3.10 test.py http://192.168.1.10 -M
An error occurred: At least one test must be run. Use -all/--runall to run all modules.

(kali@kali)-[~]
$ python3.10 test.py http://192.168.1.10 -M --whatweb
Working Directory is: /home/kali/SiteScan/
(You have chosen to modify script behaviour. Answer the following with yes or no... )
(A blank answer for any of the following indicates 'No' ... )

Should all modules run be output in the report, regardless of whether they have been specified in the initial args?: y
Should files be saved into a different directory than current?:
Should the output report file name be changed?: y
> Enter new filename ONLY to save to the working directory (no spaces): my_report.txt

All required tools are executable through Path.
Tests will execute in the order: ['whatweb']
Target appears of valid type and correctly formatted.
URL target cleaned and identified domain stored.
No port specified, inferred 80 from HTTP protocol.
(Target IP misidentified as URL, but IP is valid. Continuing... )
Tested connection to target. Target is online.
Tested port connection. Port is open on target.
Required directories created.

WhatWeb test starting...
WhatWeb test complete.

OUTPUT REPORT FOUND AT: /home/kali/SiteScan/my_report.txt
```

Figure 10 | Two script commands, showing different outputs

It also shows the “modify normal operation” functionality of the script. In this case, all tests run were to be output to the report, and the output report file name was changed to “my_report.txt” (confirmed at the bottom). In this instance, the WhatWeb option (shown specified using the alternative “--” notation) did not have any dependencies, so only that module was run.

3 DISCUSSION

3.1 CRITICAL ANALYSIS

3.1.1 The Code

3.1.1.1 *Positives*

The code makes use of error handling (“try...catch” statements) in every function to properly handle any errors that may occur during execution, allowing for appropriate error messages to be displayed. Extensive commenting in the code allows for debugging/troubleshooting of processes if required. Logical and intuitive function names contribute towards this, and overall code maintenance as well. Editable variable values for things such as file names do the same.

Flexible functions designed for re-use across multiple different areas of the script have been implemented, along with default function parameters where appropriate to make re-use easier. For example, `listFiles()` features options to disable recursive directory listing, give full file paths instead of relative, and specify what file types to list (such as text, image, and video).

Most of the libraries used by the program are standard Python libraries, and any non-standard libraries contain the corresponding “pip install” command in the code comments so that it will be shown in the default error message if it’s missing.

Ensuring that the required tools are installed before executing the corresponding commands helps to prevent error messages for the user, and follows best practice.

Practices such as handling common programmer errors help to prevent relatively minor error messages from appearing for the user – for example `writeToFile()`, which converts the `textStyle` optional parameter to an integer, in case it gets passed in as a string by mistakes. Python uses ‘duck typing’ (where functions support passing in all data types), and all the other parameters for this function are strings, so this would be an easy mistake to make – and it also opens the ability for all parameters to be passed as strings if that is easier for whatever reason.

3.1.1.2 Negatives

Opportunities for improvements and future work are discussed in the following *section* 3.2. Overall, this comprises refactorisation to follow object-oriented code design, more appropriate variables, port detection based on semicolons, and changes to the Wget module to add more functionality.

3.1.2 The User Experience

3.1.2.1 Positives

The script makes use of appropriate formatting in the output report, with acceptable spacing between test sections and the use of section headings. The script also uses informative error, procedural, and instructional messages to make actions clear to the user.

Input validation on all entered data is used to ensure that only acceptable values are allowed to be processed by the application, and are stopped beforehand if incorrect. Input options for the target provide support for a varied selection of formats, including URL addresses, IP addresses, and Web Domains, with handling for protocols and unnecessary information.

Intuitive command-line switch/option names are used for activating tests, along with logical file names and directory names and structures, for saved data. Widely used and documented tools are used for conducting tests where needed.

3.1.2.2 Negatives

Opportunities for improvement and future work are discussed in the following *section* 3.2. Overall, this comprises improving the output report formatting, extending the number of tests conducted, and upgrading the script to include analysis functionality and Windows OS support.

3.2 FUTURE WORK

The developer has identified several areas of improvement with the script, that could help to bolster its usability and add to its usefulness, as well as optimise the code.

One major area of future work would be the refactorisation of the code to follow an object-oriented programming paradigm, as opposed to the current procedural design. Particularly for more complex projects, this becomes exponentially more important for modularity/flexibility, troubleshooting, efficiency, and readability purposes.

In its current form, the script would likely benefit from the use of a class system for storing information about each test such as its name, dependencies, and command(s) used.

Some variables, notably those containing target information, could be changed to use a dictionary structure, similar to the `runTest{}` and `login{}` variables. This would help “group” these variables, improving code clarity, and would be relatively simple to change.

As noted prior, the code cannot detect the port based on the semicolon (;) separator. Given the current code, this would be very easy to implement. Likewise, because the `-port` argument is often used, this could simply be made a requirement, and the script only needs to take in the domain/IP address. This would simplify the script but would reduce the flexibility of inputs allowed, so might be a detriment instead.

Similarly, the `startswith()` check in the `isValidValue()` function is case-sensitive and will require changing to let it detect capitalised protocols – though these are relatively uncommon in this context.

The `Wget` test module could be improved to detect and analyse files such as “`sitemap.xml`”, which would help create a more complete view of the web app structure. It could also download these files, and other default/common files such as “`phpinfo.php`”. This opens a further review of how the ‘default files scan’ module is used throughout the script.

This module would also benefit from having the option to download all file types or not, which would be easily programmable thanks to the “modify normal operation” functionality already implemented.

The output report format is not the easiest to read, so finding a better way of displaying the gathered information would be greatly beneficial to usability. Using graphical/visual elements in the output would be very helpful.

The script functions more like a small suite of data-gathering tools, and, whilst relatively extensive, could still be furthered to cover more areas important to web application penetration tests. Some ideas include:

- Website structure analysis
- Nmap server scans
 - Port scanning
 - Software detection
 - Banner grabbing
 - OS detection
- Vulnerable function analysis
- CVSS detection for discovered vulnerabilities
- Testing injection techniques on discovered entry points

The aforementioned point also leads into a discussion of future work expanding the tool into more of a smart analysis script, than an information-gathering one as it is currently. This should be considered only after other changes discussed here though. The same goes for adding Windows operating system support. This would likely not be too difficult though as Python is a flexible programming language.

3.3 FULFILMENT OF AIMS

Overall, the aims set out for both the practical script development and written documentation work are considered as being met. These are discussed in detail below:

Completion of the script meets that core aim, with each sub-aim contributing toward this:

“Conduct appropriate preliminary investigative security tests, as defined in the OWASP Web Security Testing Guide” was met by using a variety of security tests in the script, each working fully or in-part towards supporting the methodology steps included in the OWASP WSTG, all laid out in *section 1.3*.

“Output results in a readable format, and through an appropriate medium” was met through outputting to a universally-readable ‘.txt’ (text) file, and by using section divisions and headings for text formatting.

“Automate all tasks, where possible and appropriate” was met by requiring user input only initially (at the beginning of the script) and running all operations sequentially and independently.

“Describe the project background and justify its undertaking” and **“Describe the OWASP Security Testing Guide and explain its relation to the project”** were met by the related explanatory paragraphs found within *sections 1.1 and 1.3*.

“Explain the script’s code, demonstrating where appropriate” was met by explaining what the code does by breaking it down by respective code functions and giving a logical overview of what it does. Most of this is self-explanatory, however, screenshots are included for explanations that would benefit particularly from visual aids.

“Describe the testing procedure and demonstrate the results” was met by describing how the tests were conducted and the platforms on which testing was conducted. Two examples of script operation were provided, showing the variety of features.

“Critically evaluate the script’s technical and usability aspects” and **“Consider future work to be investigated, pertaining to the script”** were met in the related explanatory paragraphs found within *sections 3.1 and 3.2*.

4 REFERENCES

Thomas, D. (2022) What are web-based attacks, and which industries are most vulnerable?, SC Magazine. Available at: <https://www.scmagazine.com/resource/application-security/what-are-web-based-attacks-and-which-industries-are-most-vulnerable> (Accessed: March 31, 2023).

Petrosyan, A. (2022) Estimated cost of cybercrime worldwide from 2016 to 2027, Statista. Available at: <https://www.statista.com/statistics/1280009/cost-cybercrime-worldwide/> (Accessed: March 31, 2023).

Djuraskovic, O. (2023) How many websites are there? - web stats 2023, FirstSiteGuide. Available at: <https://firstsiteguide.com/how-many-websites/> (Accessed: March 31, 2023).

OWASP (2020) Web Security Testing Guide v4.2, WSTG - v4.2. The OWASP Foundation. Available at: <https://owasp.org/www-project-web-security-testing-guide/v42/> (Accessed: April 8, 2023).

5 APPENDICES

5.1 APPENDIX A – RESULTS EXAMPLE 1 REPORT

REFERENCED IN – 2.2.2.1

~~~~~ SiteScan Report ~~~~~

Started at: 2023-05-16 22:36:55

Target: 192.168.1.10

Port: 80

Modules run, in order: ['dirb', 'wget', 'cookies', 'comments', 'defaults', 'encryption', 'entry', 'header', 'nikto', 'whatweb']

Login path: /login.php

Login username: hacklab

Login password: hacklab

Only modules specified in the startup arguments are displayed in this report. '-all' emulates that behaviour.

~~~~~

==== Cookies Used ====

-= (Using Python 'requests' library) =-

(SUPPLIED LOGIN DATA USED FOR LOGIN ATTEMPT)

Name: PHPSESSID

Value: c4g3g6jqqt5krlvac1a3bob8n3

Domain: 192.168.1.10

Path: /

Name: SecretCookie

Value: 756e7078796e6f3a756e7078796e6f3a31363834323931303230

Domain: 192.168.1.10

Path: /

--=== Code Comments ===--

-= Text search of downloaded files for: ['/', '<!--', '-->', '/', '*'] =-

-= Ignoring strings with: ['./', '-/'] =-

/items.php?itemid=74 (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/index.html (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/items.php?itemid=78 (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/items.php?itemid=73 (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/items.php?itemid=72 (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/phpinfo.php (Line 93) >>> /* </td></tr>

/phpinfo.php (Line 126) >>> /* </td></tr>

/phpinfo.php (Line 994) >>> /*</td></tr>

/items.php?itemid=79 (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/items.php?itemid=76 (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/items.php?itemid=77 (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/items.php?itemid=71 (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/index.php (Line 1) >>> <!-- *** Note to self: Door entry number is 1846 -->

/login.php (Line 62) >>> <!-- Start Login Form -->

/login.php (Line 84) >>> <!-- End Login Form -->

/login.php (Line 85) >>> <!-- Start Signup Form -->

/login.php (Line 127) >>> <!-- End Signup Form -->

/admin/layout/css/backend.css (Line 1) >>> /* Start Main Rulez */

/admin/layout/css/backend.css (Line 23) >>> /* End Main Rulez */

/admin/layout/css/backend.css (Line 25) >>> /* Start Login Form */

```

/admin/layout/css/backend.css (Line 48) >>> /* End Login Form */
/admin/layout/css/backend.css (Line 50) >>> /* Start Bootstrap Edits */
/admin/layout/css/backend.css (Line 96) >>> /* End Bootstrap Edits */
/admin/layout/css/backend.css (Line 98) >>> /* Start Dashboard Page */
/admin/layout/css/backend.css (Line 221) >>> /* End Dashboard Page */
/admin/layout/css/backend.css (Line 223) >>> /* Start Members Page */
/admin/layout/css/backend.css (Line 254) >>> /* You Can Use Thead */
/admin/layout/css/backend.css (Line 267) >>> /* End Members Page */
/admin/layout/css/backend.css (Line 269) >>> /* Start Category Page */
/admin/layout/css/backend.css (Line 395) >>> /* End Category Page */

/admin/layout/css/bootstrap.css (Line 1) >>> /*!
/admin/layout/css/bootstrap.css (Line 5) >>> */
/admin/layout/css/bootstrap.css (Line 6) >>> /*! normalize.css v3.0.3 | MIT License |
github.com/necolas/normalize.css */
/admin/layout/css/bootstrap.css (Line 6760) >>> /*# sourceMappingURL=bootstrap.css.map */

/admin/layout/css/font-awesome.min.css (Line 1) >>> /*!
/admin/layout/css/font-awesome.min.css (Line 4) >>> /*@font-face{font-
family:'FontAwesome';src:url('../fonts/fontawesome-
webfont.eot?v=4.5.0');src:url('../fonts/fontawesome-webfont.eot?#iefix&v=4.5.0')

/admin/layout/css/bootstrap.min.css (Line 1) >>> /*!
/admin/layout/css/bootstrap.min.css (Line 6) >>> /*# sourceMappingURL=bootstrap.min.css.map */

/admin/layout/css/jquery-ui.css (Line 1) >>> /*! jQuery UI - v1.11.4 - 2016-04-23
/admin/layout/css/jquery-ui.css (Line 5) >>> */
/admin/layout/css/jquery-ui.css (Line 7) >>> /* Layout helpers
/admin/layout/css/jquery-ui.css (Line 8) >>> */
/admin/layout/css/jquery-ui.css (Line 42) >>> /* support: IE7 */
/admin/layout/css/jquery-ui.css (Line 51) >>> /* support: IE8 */
/admin/layout/css/jquery-ui.css (Line 59) >>> /* Interaction Cues
/admin/layout/css/jquery-ui.css (Line 60) >>> */
/admin/layout/css/jquery-ui.css (Line 66) >>> /* Icons
/admin/layout/css/jquery-ui.css (Line 67) >>> */
/admin/layout/css/jquery-ui.css (Line 69) >>> /* states and images */
/admin/layout/css/jquery-ui.css (Line 78) >>> /* Misc visuals
/admin/layout/css/jquery-ui.css (Line 79) >>> */
/admin/layout/css/jquery-ui.css (Line 81) >>> /* Overlays */
/admin/layout/css/jquery-ui.css (Line 182) >>> /* support: IE7 */
/admin/layout/css/jquery-ui.css (Line 217) >>> /* removes extra width in IE */
/admin/layout/css/jquery-ui.css (Line 226) >>> /* to make room for the icon, a width needs to be set
here */
/admin/layout/css/jquery-ui.css (Line 230) >>> /* button elements seem to need a little more width
*/

```

```

/admin/layout/css/jquery-ui.css (Line 241) >>> /* button text element */
/admin/layout/css/jquery-ui.css (Line 266) >>> /* no icon support for input elements, provide
padding by default */
/admin/layout/css/jquery-ui.css (Line 271) >>> /* button icon element(s) */
/admin/layout/css/jquery-ui.css (Line 296) >>> /* button sets */
/admin/layout/css/jquery-ui.css (Line 305) >>> /* workarounds */
/admin/layout/css/jquery-ui.css (Line 306) >>> /* reset extra padding in Firefox, see h5bp.com/l */
/admin/layout/css/jquery-ui.css (Line 409) >>> /* with multiple calendars */
/admin/layout/css/jquery-ui.css (Line 442) >>> /* RTL support */
/admin/layout/css/jquery-ui.css (Line 552) >>> /* support: IE7 */
/admin/layout/css/jquery-ui.css (Line 553) >>> /* support: IE10, see #8844 */
/admin/layout/css/jquery-ui.css (Line 554) >>> ///yH5BAEAAAAALAAAAABAAEAAABRAA7");
/admin/layout/css/jquery-ui.css (Line 568) >>> /* icon support */
/admin/layout/css/jquery-ui.css (Line 576) >>> /* left-aligned */
/admin/layout/css/jquery-ui.css (Line 585) >>> /* right-aligned */
/admin/layout/css/jquery-ui.css (Line 600) >>>
///yH/C05FVFNQVBFM4wAwEAAAh+QQJABACwAAAAKAAoAAACkYwNqXrdC52DS06a7MFZI
+4FHBCoDeWkXqymPqGqxvJrXZbMx7Ttc+w9XgU2FB3IOyQRWET2IFGiU9m1frDVpxZZc6bfHwv
/admin/layout/css/jquery-ui.css (Line 602) >>> /* support: IE8 */
/admin/layout/css/jquery-ui.css (Line 618) >>> /* Support: IE7 */
/admin/layout/css/jquery-ui.css (Line 679) >>> /* support: IE8 - See #6727 */
/admin/layout/css/jquery-ui.css (Line 752) >>> /* more specificity required here to override default
borders */
/admin/layout/css/jquery-ui.css (Line 758) >>> /* vertically center icon */
/admin/layout/css/jquery-ui.css (Line 772) >>> /* TR overrides */
/admin/layout/css/jquery-ui.css (Line 774) >>> /* need to fix icons sprite */
/admin/layout/css/jquery-ui.css (Line 778) >>> /* position: relative prevents IE scroll bug (element
with position: relative inside container with overflow: auto appear as "fixed") */
/admin/layout/css/jquery-ui.css (Line 830) >>> /* Component containers
/admin/layout/css/jquery-ui.css (Line 831) >>> */
/admin/layout/css/jquery-ui.css (Line 864) >>> /* Interaction states
/admin/layout/css/jquery-ui.css (Line 865) >>> */
/admin/layout/css/jquery-ui.css (Line 917) >>> /* Interaction Cues
/admin/layout/css/jquery-ui.css (Line 918) >>> */
/admin/layout/css/jquery-ui.css (Line 957) >>> /* support: IE8 */
/admin/layout/css/jquery-ui.css (Line 964) >>> /* support: IE8 */
/admin/layout/css/jquery-ui.css (Line 968) >>> /* support: IE8 - See #6059 */
/admin/layout/css/jquery-ui.css (Line 971) >>> /* Icons
/admin/layout/css/jquery-ui.css (Line 972) >>> */
/admin/layout/css/jquery-ui.css (Line 974) >>> /* states and images */
/admin/layout/css/jquery-ui.css (Line 1004) >>> /* positioning */
/admin/layout/css/jquery-ui.css (Line 1142) >>> /* ui-icon-seek-first is deprecated, use ui-icon-seek-
start instead */
/admin/layout/css/jquery-ui.css (Line 1183) >>> /* Misc visuals
/admin/layout/css/jquery-ui.css (Line 1184) >>> */

```

```

/admin/layout/css/jquery-ui.css (Line 1186) >>> /* Corner radius */
/admin/layout/css/jquery-ui.css (Line 1212) >>> /* Overlays */
/admin/layout/css/jquery-ui.css (Line 1216) >>> /* support: IE8 */
/admin/layout/css/jquery-ui.css (Line 1223) >>> /* support: IE8 */

/admin/layout/css/jquery.selectBoxIt.css (Line 1) >>> /*
/admin/layout/css/jquery.selectBoxIt.css (Line 4) >>> */
/admin/layout/css/jquery.selectBoxIt.css (Line 6) >>> /*
/admin/layout/css/jquery.selectBoxIt.css (Line 10) >>> */
/admin/layout/css/jquery.selectBoxIt.css (Line 12) >>> /* SelectBoxIt container */
/admin/layout/css/jquery.selectBoxIt.css (Line 20) >>> /* Styles that apply to all SelectBoxIt elements
*/
/admin/layout/css/jquery.selectBoxIt.css (Line 23) >>> /* Prevents text selection */
/admin/layout/css/jquery.selectBoxIt.css (Line 35) >>> /* Button */
/admin/layout/css/jquery.selectBoxIt.css (Line 37) >>> /* Width of the dropdown button */
/admin/layout/css/jquery.selectBoxIt.css (Line 47) >>> /* Height and Vertical Alignment of Text */
/admin/layout/css/jquery.selectBoxIt.css (Line 49) >>> /* Height of the drop down */
/admin/layout/css/jquery.selectBoxIt.css (Line 50) >>> /* Vertically positions the drop down text */
/admin/layout/css/jquery.selectBoxIt.css (Line 54) >>> /* Focus pseudo selector */
/admin/layout/css/jquery.selectBoxIt.css (Line 59) >>> /* Disabled Mouse Interaction */
/admin/layout/css/jquery.selectBoxIt.css (Line 69) >>> /* Button Text */
/admin/layout/css/jquery.selectBoxIt.css (Line 81) >>> /* Options List */
/admin/layout/css/jquery.selectBoxIt.css (Line 85) >>> /* Minimum Width of the dropdown list box
options */
/admin/layout/css/jquery.selectBoxIt.css (Line 103) >>> /* Individual options */
/admin/layout/css/jquery.selectBoxIt.css (Line 108) >>> /* Individual Option Hover Action */
/admin/layout/css/jquery.selectBoxIt.css (Line 113) >>> /* Individual Option Optgroup Header */
/admin/layout/css/jquery.selectBoxIt.css (Line 115) >>> /* Horizontal Positioning of the select box
option text */
/admin/layout/css/jquery.selectBoxIt.css (Line 120) >>> /* The first Drop Down option */
/admin/layout/css/jquery.selectBoxIt.css (Line 126) >>> /* The first Drop Down option optgroup */
/admin/layout/css/jquery.selectBoxIt.css (Line 132) >>> /* The last Drop Down option */
/admin/layout/css/jquery.selectBoxIt.css (Line 138) >>> /* Drop Down optgroup headers */
/admin/layout/css/jquery.selectBoxIt.css (Line 143) >>> /* Drop Down optgroup header hover
psuedo class */
/admin/layout/css/jquery.selectBoxIt.css (Line 148) >>> /* Drop Down down arrow container */
/admin/layout/css/jquery.selectBoxIt.css (Line 150) >>> /* Positions the down arrow */
/admin/layout/css/jquery.selectBoxIt.css (Line 156) >>> /* Drop Down down arrow */
/admin/layout/css/jquery.selectBoxIt.css (Line 158) >>> /* Horizontally centers the down arrow */
/admin/layout/css/jquery.selectBoxIt.css (Line 166) >>> /* Drop Down down arrow for jQueryUI and
jQuery Mobile */
/admin/layout/css/jquery.selectBoxIt.css (Line 171) >>> /* Drop Down individual option icon
positioning */
/admin/layout/css/jquery.selectBoxIt.css (Line 182) >>> /* Drop Down individual option icon
positioning */

```



```

/admin/layout/css/jquery.selectBoxIt.css (Line 202) >>> /* jQueryUI and jQuery Mobile compatability
fix - Feel free to remove this style if you are not using jQuery Mobile */
/admin/layout/css/jquery.selectBoxIt.css (Line 207) >>> /* Another jQueryUI and jQuery Mobile
compatability fix - Feel free to remove this style if you are not using jQuery Mobile */
/admin/layout/css/jquery.selectBoxIt.css (Line 212) >>> /*
/admin/layout/css/jquery.selectBoxIt.css (Line 216) >>> */

/admin/layout/js/backend.js (Line 5) >>> // Dashboard
/admin/layout/js/backend.js (Line 23) >>> // Trigger The Selectboxit
/admin/layout/js/backend.js (Line 31) >>> // Hide Placeholder On Form Focus
/admin/layout/js/backend.js (Line 45) >>> // Add Asterisk On Required Field
/admin/layout/js/backend.js (Line 57) >>> // Convert Password Field To Text Field On Hover
/admin/layout/js/backend.js (Line 71) >>> // Confirmation Message On Button
/admin/layout/js/backend.js (Line 79) >>> // Category View Option
/admin/layout/js/backend.js (Line 103) >>> // Show Delete Button On Child Cats

/admin/layout/js/bootstrap.min.js (Line 1) >>> /*!
/admin/layout/js/bootstrap.min.js (Line 5) >>> */

/admin/layout/js/jquery-ui.min.js (Line 1) >>> /*! jQuery UI - v1.11.4 - 2016-04-21
/admin/layout/js/jquery-ui.min.js (Line 4) >>> */
/admin/layout/js/jquery-ui.min.js (Line 10) >>> /*)?i+parseInt(e.substring(1),10):e.match(/[+\\-
.]+)/)?c+parseInt(e,10):parseInt(e,10);return
isNaN(t)?c:t,f=p(d[0]),m=Math.max(f,p(d[1]| | "")),f=s?Math

/admin/layout/js/jquery.selectBoxIt.min.js (Line 1) >>> /*! jquery.selectBoxIt - v3.8.1 - 2013-11-17
/admin/layout/js/jquery.selectBoxIt.min.js (Line 3) >>> */

/admin/layout/js/jquery-1.12.1.min.js (Line 1) >>> /*! jQuery v1.12.1 | (c) jQuery Foundation |
jquery.org/license */
/admin/layout/js/jquery-1.12.1.min.js (Line 4) >>> //,Mb=/^([\w.+-
]+:)(?:\V(?:[^\V?#]*@|)([^\V?#]*)?(?:\d+))|),Nb={},Ob={},Pb="*/".concat(""),Qb=Cb.href,Rb=Mb
.exec(Qb.toLowerCase())| |[]];function

/layout/css/bootstrap.css (Line 1) >>> /*!
/layout/css/bootstrap.css (Line 5) >>> */
/layout/css/bootstrap.css (Line 6) >>> /*! normalize.css v3.0.3 | MIT License |
github.com/necolas/normalize.css */
/layout/css/bootstrap.css (Line 6760) >>> /*# sourceMappingURL=bootstrap.css.map */

/layout/css/font-awesome.min.css (Line 1) >>> /*!
/layout/css/font-awesome.min.css (Line 4) >>> */@font-face{font-
family:'FontAwesome';src:url('../fonts/fontawesome-
webfont.eot?v=4.5.0');src:url('../fonts/fontawesome-webfont.eot?#iefix&v=4.5.0')

```

```

/layout/css/bootstrap.min.css (Line 1) >>> /*!
/layout/css/bootstrap.min.css (Line 6) >>> /*# sourceMappingURL=bootstrap.min.css.map */

/layout/css/jquery-ui.css (Line 1) >>> /*! jQuery UI - v1.11.4 - 2016-04-23
/layout/css/jquery-ui.css (Line 5) >>> */
/layout/css/jquery-ui.css (Line 7) >>> /* Layout helpers
/layout/css/jquery-ui.css (Line 8) >>> */
/layout/css/jquery-ui.css (Line 42) >>> /* support: IE7 */
/layout/css/jquery-ui.css (Line 51) >>> /* support: IE8 */
/layout/css/jquery-ui.css (Line 59) >>> /* Interaction Cues
/layout/css/jquery-ui.css (Line 60) >>> */
/layout/css/jquery-ui.css (Line 66) >>> /* Icons
/layout/css/jquery-ui.css (Line 67) >>> */
/layout/css/jquery-ui.css (Line 69) >>> /* states and images */
/layout/css/jquery-ui.css (Line 78) >>> /* Misc visuals
/layout/css/jquery-ui.css (Line 79) >>> */
/layout/css/jquery-ui.css (Line 81) >>> /* Overlays */
/layout/css/jquery-ui.css (Line 182) >>> /* support: IE7 */
/layout/css/jquery-ui.css (Line 217) >>> /* removes extra width in IE */
/layout/css/jquery-ui.css (Line 226) >>> /* to make room for the icon, a width needs to be set here */
/layout/css/jquery-ui.css (Line 230) >>> /* button elements seem to need a little more width */
/layout/css/jquery-ui.css (Line 241) >>> /* button text element */
/layout/css/jquery-ui.css (Line 266) >>> /* no icon support for input elements, provide padding by
default */
/layout/css/jquery-ui.css (Line 271) >>> /* button icon element(s) */
/layout/css/jquery-ui.css (Line 296) >>> /* button sets */
/layout/css/jquery-ui.css (Line 305) >>> /* workarounds */
/layout/css/jquery-ui.css (Line 306) >>> /* reset extra padding in Firefox, see h5bp.com/l */
/layout/css/jquery-ui.css (Line 409) >>> /* with multiple calendars */
/layout/css/jquery-ui.css (Line 442) >>> /* RTL support */
/layout/css/jquery-ui.css (Line 552) >>> /* support: IE7 */
/layout/css/jquery-ui.css (Line 553) >>> /* support: IE10, see #8844 */
/layout/css/jquery-ui.css (Line 554) >>> ///yH5BAEAAAAALAAAAABAAEAAAIIBRAA7");
/layout/css/jquery-ui.css (Line 568) >>> /* icon support */
/layout/css/jquery-ui.css (Line 576) >>> /* left-aligned */
/layout/css/jquery-ui.css (Line 585) >>> /* right-aligned */
/layout/css/jquery-ui.css (Line 600) >>>
///yH/C05FVFNDQVBFMi4wAwEAAAAh+QJJAQABACwAAAAAKAAoAAACkYwNqXrdC52DS06a7MFZI
+4FHBCoDeWKXqymPqGqxvJrXZbMx7Ttc+w9XgU2FB3IOyQRWET2IFGiU9m1frDVpxZZc6bfHwv
/layout/css/jquery-ui.css (Line 602) >>> /* support: IE8 */
/layout/css/jquery-ui.css (Line 618) >>> /* Support: IE7 */
/layout/css/jquery-ui.css (Line 679) >>> /* support: IE8 - See #6727 */
/layout/css/jquery-ui.css (Line 752) >>> /* more specificity required here to override default borders
*/
/layout/css/jquery-ui.css (Line 758) >>> /* vertically center icon */

```

```

/layout/css/jquery-ui.css (Line 772) >>> /* TR overrides */
/layout/css/jquery-ui.css (Line 774) >>> /* need to fix icons sprite */
/layout/css/jquery-ui.css (Line 778) >>> /* position: relative prevents IE scroll bug (element with
position: relative inside container with overflow: auto appear as "fixed") */
/layout/css/jquery-ui.css (Line 830) >>> /* Component containers
/layout/css/jquery-ui.css (Line 831) >>> */
/layout/css/jquery-ui.css (Line 864) >>> /* Interaction states
/layout/css/jquery-ui.css (Line 865) >>> */
/layout/css/jquery-ui.css (Line 917) >>> /* Interaction Cues
/layout/css/jquery-ui.css (Line 918) >>> */
/layout/css/jquery-ui.css (Line 957) >>> /* support: IE8 */
/layout/css/jquery-ui.css (Line 964) >>> /* support: IE8 */
/layout/css/jquery-ui.css (Line 968) >>> /* support: IE8 - See #6059 */
/layout/css/jquery-ui.css (Line 971) >>> /* Icons
/layout/css/jquery-ui.css (Line 972) >>> */
/layout/css/jquery-ui.css (Line 974) >>> /* states and images */
/layout/css/jquery-ui.css (Line 1004) >>> /* positioning */
/layout/css/jquery-ui.css (Line 1142) >>> /* ui-icon-seek-first is deprecated, use ui-icon-seek-start
instead */
/layout/css/jquery-ui.css (Line 1183) >>> /* Misc visuals
/layout/css/jquery-ui.css (Line 1184) >>> */
/layout/css/jquery-ui.css (Line 1186) >>> /* Corner radius */
/layout/css/jquery-ui.css (Line 1212) >>> /* Overlays */
/layout/css/jquery-ui.css (Line 1216) >>> /* support: IE8 */
/layout/css/jquery-ui.css (Line 1223) >>> /* support: IE8 */

/layout/css/front.css (Line 1) >>> /* Start Main Rulez */
/layout/css/front.css (Line 42) >>> /* End Main Rulez */
/layout/css/front.css (Line 44) >>> /* Start Bootstrap Edits */
/layout/css/front.css (Line 92) >>> /* End Bootstrap Edits */
/layout/css/front.css (Line 94) >>> /* Start Header */
/layout/css/front.css (Line 106) >>> /* End Header */
/layout/css/front.css (Line 108) >>> /* Start Login Page */
/layout/css/front.css (Line 155) >>> /* End Login Page */
/layout/css/front.css (Line 157) >>> /* Start Categories Page */
/layout/css/front.css (Line 187) >>> /* End Categories Page */
/layout/css/front.css (Line 189) >>> /* Start Profile Page */
/layout/css/front.css (Line 224) >>> /* End Profile Page */
/layout/css/front.css (Line 226) >>> /* Start Show Item Page */
/layout/css/front.css (Line 298) >>> /* End Show Item Page */
/layout/css/front.css (Line 300) >>> /* Start Our Custom */
/layout/css/front.css (Line 306) >>> /* End Our Custom */

/layout/css/jquery.selectBoxIt.css (Line 1) >>> /*
/layout/css/jquery.selectBoxIt.css (Line 4) >>> */

```

```

/layout/css/jquery.selectBoxIt.css (Line 6) >>> /*
/layout/css/jquery.selectBoxIt.css (Line 10) >>> */
/layout/css/jquery.selectBoxIt.css (Line 12) >>> /* SelectBoxIt container */
/layout/css/jquery.selectBoxIt.css (Line 20) >>> /* Styles that apply to all SelectBoxIt elements */
/layout/css/jquery.selectBoxIt.css (Line 23) >>> /* Prevents text selection */
/layout/css/jquery.selectBoxIt.css (Line 35) >>> /* Button */
/layout/css/jquery.selectBoxIt.css (Line 37) >>> /* Width of the dropdown button */
/layout/css/jquery.selectBoxIt.css (Line 47) >>> /* Height and Vertical Alignment of Text */
/layout/css/jquery.selectBoxIt.css (Line 49) >>> /* Height of the drop down */
/layout/css/jquery.selectBoxIt.css (Line 50) >>> /* Vertically positions the drop down text */
/layout/css/jquery.selectBoxIt.css (Line 54) >>> /* Focus pseudo selector */
/layout/css/jquery.selectBoxIt.css (Line 59) >>> /* Disabled Mouse Interaction */
/layout/css/jquery.selectBoxIt.css (Line 69) >>> /* Button Text */
/layout/css/jquery.selectBoxIt.css (Line 81) >>> /* Options List */
/layout/css/jquery.selectBoxIt.css (Line 85) >>> /* Minimum Width of the dropdown list box options
*/
/layout/css/jquery.selectBoxIt.css (Line 103) >>> /* Individual options */
/layout/css/jquery.selectBoxIt.css (Line 108) >>> /* Individual Option Hover Action */
/layout/css/jquery.selectBoxIt.css (Line 113) >>> /* Individual Option Optgroup Header */
/layout/css/jquery.selectBoxIt.css (Line 115) >>> /* Horizontal Positioning of the select box option
text */
/layout/css/jquery.selectBoxIt.css (Line 120) >>> /* The first Drop Down option */
/layout/css/jquery.selectBoxIt.css (Line 126) >>> /* The first Drop Down option optgroup */
/layout/css/jquery.selectBoxIt.css (Line 132) >>> /* The last Drop Down option */
/layout/css/jquery.selectBoxIt.css (Line 138) >>> /* Drop Down optgroup headers */
/layout/css/jquery.selectBoxIt.css (Line 143) >>> /* Drop Down optgroup header hover psuedo class
*/
/layout/css/jquery.selectBoxIt.css (Line 148) >>> /* Drop Down down arrow container */
/layout/css/jquery.selectBoxIt.css (Line 150) >>> /* Positions the down arrow */
/layout/css/jquery.selectBoxIt.css (Line 156) >>> /* Drop Down down arrow */
/layout/css/jquery.selectBoxIt.css (Line 158) >>> /* Horizontally centers the down arrow */
/layout/css/jquery.selectBoxIt.css (Line 166) >>> /* Drop Down down arrow for jQueryUI and jQuery
Mobile */
/layout/css/jquery.selectBoxIt.css (Line 171) >>> /* Drop Down individual option icon positioning */
/layout/css/jquery.selectBoxIt.css (Line 182) >>> /* Drop Down individual option icon positioning */
/layout/css/jquery.selectBoxIt.css (Line 202) >>> /* jQueryUI and jQuery Mobile compatability fix -
Feel free to remove this style if you are not using jQuery Mobile */
/layout/css/jquery.selectBoxIt.css (Line 207) >>> /* Another jQueryUI and jQuery Mobile
compatability fix - Feel free to remove this style if you are not using jQuery Mobile */
/layout/css/jquery.selectBoxIt.css (Line 212) >>> /*
/layout/css/jquery.selectBoxIt.css (Line 216) >>> */

/layout/js/front.js (Line 5) >>> // Switch Between Login & Signup
/layout/js/front.js (Line 17) >>> // Trigger The Selectboxit
/layout/js/front.js (Line 25) >>> // Hide Placeholder On Form Focus

```

```
/layout/js/front.js (Line 39) >>> // Add Asterisk On Required Field
/layout/js/front.js (Line 51) >>> // Confirmation Message On Button
```

```
/layout/js/bootstrap.min.js (Line 1) >>> /*!
/layout/js/bootstrap.min.js (Line 5) >>> */
```

```
/layout/js/jquery-ui.min.js (Line 1) >>> /*! jQuery UI - v1.11.4 - 2016-04-21
/layout/js/jquery-ui.min.js (Line 4) >>> */
/layout/js/jquery-ui.min.js (Line 10) >>> */?i+parseInt(e.substring(1),10):e.match(/[+\\-
].*/)?c+parseInt(e,10):parseInt(e,10);return
isNaN(t)?c:t,f=p(d[0]),m=Math.max(f,p(d[1]| | "")),f=s?Math
```

```
/layout/js/jquery.selectBoxIt.min.js (Line 1) >>> /*! jquery.selectBoxIt - v3.8.1 - 2013-11-17
/layout/js/jquery.selectBoxIt.min.js (Line 3) >>> */
```

```
/layout/js/jquery-1.12.1.min.js (Line 1) >>> /*! jQuery v1.12.1 | (c) jQuery Foundation |
jquery.org/license */
/layout/js/jquery-1.12.1.min.js (Line 4) >>> //,Mb=/^([\w.+-
]+:)(?:\V(?:[^\V?#]*@|)([^\V?#:]*)?(?:\d+)))/,Nb={},Ob={},Pb="*/".concat(""),Qb=Cb.href,Rb=Mb
.exec(Qb.toLowerCase())| |[];function
```

--=== Default Files ===--

-- Searching downloaded file list for exact: ['/robots.txt', '/sitemap.xml', '/humans.txt', '/license.txt',
'/readme.html', '/.htaccess', '/.phpinfo.php'] ==

-- Also searching loosely for: ['web.config', 'crossdomain.xml'] ==

'/robots.txt' found.

--=== Encryption Usage ===--

-- HSTS (HTTP Strict Transport Security) Policy ==

-- curl http://192.168.1.10 -s | grep -i Strict-Transport-Security ==

! HSTS not detected !

-- SSL/TLS Certificates --

-- sslscan --verbose --no-color 192.168.1.10:80 --

Version: 2.0.15-static

OpenSSL 1.1.1q-dev xx XXX xxxx

Connected to 192.168.1.10

Some servers will fail to response to SSLv3 ciphers over STARTTLS

If your scan hangs, try using the --tlsall option

Testing SSL server 192.168.1.10 on port 80 using SNI name 192.168.1.10

SSL/TLS Protocols:

SSLv2 disabled

SSLv3 disabled

TLSv1.0 disabled

TLSv1.1 disabled

TLSv1.2 disabled

TLSv1.3 disabled

TLS Fallback SCSV:

OpenSSL OpenSSL 1.1.1q-dev xx XXX xxxx looks like version 0.9.8m or later; I will try SSL_OP to enable renegotiation

Connection failed - unable to determine TLS Fallback SCSV support

TLS renegotiation:

OpenSSL OpenSSL 1.1.1q-dev xx XXX xxxx looks like version 0.9.8m or later; I will try SSL_OP to enable renegotiation

use_unsafe_renegotiation_op

Session renegotiation not supported

TLS Compression:

OpenSSL OpenSSL 1.1.1q-dev xx XXX xxxx looks like version 0.9.8m or later; I will try SSL_OP to enable renegotiation

Compression disabled

Heartbleed:

Supported Server Cipher(s):

Unable to parse certificate
Unable to parse certificate
Unable to parse certificate
Unable to parse certificate
Certificate information cannot be retrieved.

--==== Entry Points - HTML Form Inputs ===--
-= Text search of downloaded files =-

```
/login.php (Line 65) >>> t  
/login.php (Line 74) >>> t  
/login.php (Line 82) >>> <input class="btn btn-primary btn-block" name="login" type="submit"  
value="Login" />  
/login.php (Line 88) >>> t  
/login.php (Line 99) >>> t  
/login.php (Line 109) >>> t  
/login.php (Line 119) >>> t  
/login.php (Line 125) >>> <input class="btn btn-success btn-block" name="signup" type="submit"  
value="Signup" />
```

```
/admin/index.html (Line 15) >>> <input class="form-control" type="text" name="user"  
placeholder="Username" autocomplete="off" />  
/admin/index.html (Line 16) >>> <input class="form-control" type="password" name="pass"  
placeholder="Password" autocomplete="new-password" />  
/admin/index.html (Line 17) >>> <input class="btn btn-primary btn-block" type="submit"  
value="Login" />
```

```
/admin/index.php (Line 15) >>> <input class="form-control" type="text" name="user"  
placeholder="Username" autocomplete="off" />  
/admin/index.php (Line 16) >>> <input class="form-control" type="password" name="pass"  
placeholder="Password" autocomplete="new-password" />  
/admin/index.php (Line 17) >>> <input class="btn btn-primary btn-block" type="submit"  
value="Login" />
```

```
/admin/layout/js/jquery-ui.min.js (Line 9) >>> <input type='text' id='"+r+"' style='position: absolute;  
top: -100px; width: 0px;'/>"),this._dialogInput.keydown(this._doKeyDown),e("body").append(thi
```

```
/admin/layout/js/jquery-1.12.1.min.js (Line 3) >>> <input
type='checkbox'/>",l.leadingWhitespace=3===a.firstChild.nodeType,l.tbody=!a.getElementsByTagName("tbody").length,l.htmlSerialize=!a.getElement
/admin/layout/js/jquery-1.12.1.min.js (Line 4) >>> <input
type='checkbox'/>",a=c.getElementsByTagName("a")[0],b.setAttribute("type","checkbox"),c.appendC
hild(b),a=c.getElementsByTagName("a")[0],a.style
```

```
/layout/js/jquery-ui.min.js (Line 9) >>> <input type='text' id="" +r"" style='position: absolute; top: -
100px; width: 0px;/>"),this._dialogInput.keydown(this._doKeyDown),e("body").append(thi
```

```
/layout/js/jquery-1.12.1.min.js (Line 3) >>> <input
type='checkbox'/>",l.leadingWhitespace=3===a.firstChild.nodeType,l.tbody=!a.getElementsByTagName("tbody").length,l.htmlSerialize=!a.getElement
/layout/js/jquery-1.12.1.min.js (Line 4) >>> <input
type='checkbox'/>",a=c.getElementsByTagName("a")[0],b.setAttribute("type","checkbox"),c.appendC
hild(b),a=c.getElementsByTagName("a")[0],a.style
```

--=== Entry Points - Web Page Query Parameters ===--

-= Text search of downloaded files list for filenames containing '?' =-

THE FOLLOWING FILES HAVE BEEN OBSERVED TO USE QUERY PARAMETERS:

```
/items.php?itemid=74
/items.php?itemid=78
/itemdetails.php?itemid=73
/items.php?itemid=73
/categories.php?pageid=9
/addendum.php?type=terms.php
/items.php?itemid=72
/addendum.php?type=faqs.php
/itemdetails.php?itemid=77
/items.php?itemid=79
/tags.php?name=samsung
/itemdetails.php?itemid=76
/itemdetails.php?itemid=71
/itemdetails.php?itemid=72
/items.php?itemid=76
/items.php?itemid=77
/items.php?itemid=71
/itemdetails.php?itemid=79
```


/itemdetails.php?itemid=74
/tags.php?name=phone
/categories.php?pageid=11
/tags.php?name=redmi
/itemdetails.php?itemid=78
/tags.php?name=apple
/tags.php?name=bed
/categories.php?pageid=10

--=== HTTP Header ===--

-= (Using Python 'requests' library .get() function =-

Date: Wed, 17 May 2023 02:37:03 GMT
Server: Apache/2.4.3 (Unix) PHP/5.4.7
X-Powered-By: PHP/5.4.7
Set-Cookie: PHPSESSID=4j9soe507jso42htdj37if0ld0; path=/'
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 6792
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

--=== Nikto ===--

-= nikto -h http://192.168.1.10 =-

- Nikto v2.1.6

+ Target IP: 192.168.1.10
+ Target Hostname: 192.168.1.10
+ Target Port: 80
+ Start Time: 2023-05-16 22:37:20 (GMT-4)

+ Server: Apache/2.4.3 (Unix) PHP/5.4.7

- + Retrieved x-powered-by header: PHP/5.4.7
- + The anti-clickjacking X-Frame-Options header is not present.
- + The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
- + The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
- + Cookie PHPSESSID created without the httponly flag
- + "robots.txt" contains 1 entry which should be manually viewed.
- + Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See <http://www.wisec.it/sectou.php?id=4698ebdc59d15>. The following alternatives for 'index' were found: HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var
- + Apache/2.4.3 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
- + PHP/5.4.7 appears to be outdated (current is at least 7.2.12). PHP 5.6.33, 7.0.27, 7.1.13, 7.2.1 may also current release for each branch.
- + OSVDB-112004: /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability (<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271>).
- + OSVDB-112004: /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability (<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278>).
- + Web Server returns a valid response with junk HTTP methods, this may cause false positives.
- + OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
- + /phpinfo.php: Output from the phpinfo() function was found.
- + OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
- + OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
- + OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
- + OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
- + OSVDB-3092: /admin/: This might be interesting...
- + OSVDB-3268: /includes/: Directory indexing found.
- + OSVDB-3092: /includes/: This might be interesting...
- + OSVDB-3268: /mp3/: Directory indexing found.
- + OSVDB-3092: /mp3/: This might be interesting...
- + OSVDB-3093: /admin/index.php: This might be interesting... has been seen in web logs from an unknown scanner.
- + OSVDB-3233: /cgi-bin/printenv: Apache 2.0 default script is executable and gives server environment variables. All default scripts should be removed. It may also allow XSS types of attacks. <http://www.securityfocus.com/bid/4431>.

+ OSVDB-3233: /cgi-bin/test-cgi: Apache 2.0 default script is executable and reveals system information. All default scripts should be removed.
+ OSVDB-3233: /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ /login.php: Admin login page/section found.
+ 9687 requests: 0 error(s) and 30 item(s) reported on remote host
+ End Time: 2023-05-16 22:38:28 (GMT-4) (68 seconds)

+ 1 host(s) tested

--=== WhatWeb ===--

-= whatweb http://192.168.1.10 --color=never =-

http://192.168.1.10 [200 OK] Apache[2.4.3]
Bootstrap
Cookies[PHPSESSID]
Country[RESERVED][ZZ]
HTML5
HTTPServer[Unix][Apache/2.4.3 (Unix) PHP/5.4.7]
IP[192.168.1.10]
jQuery[1.12.1]
PHP[5.4.7]
Script
Title[Homepage]
X-Powered-By[PHP/5.4.7]

~~~~~