National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Andrew Snook
Student ID: x23283301

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Andrew Snook……

Student ID: …… x23283301……..……

Programme: MSc Cybersecurity …… **Year:** ……2025……..

Module: **……** MSc (Research) Practicum Part 2 ……

Lecturer: ……Michael Pantridge……..……

Submission Due Date: ……11 December 2025……..……

Project Title: Applying threat modelling to a Model Context Protocol Implementation

Word Count: …… **Page Count:** ……..…….……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Andrew Snook……

Date: ……8th December 2025……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

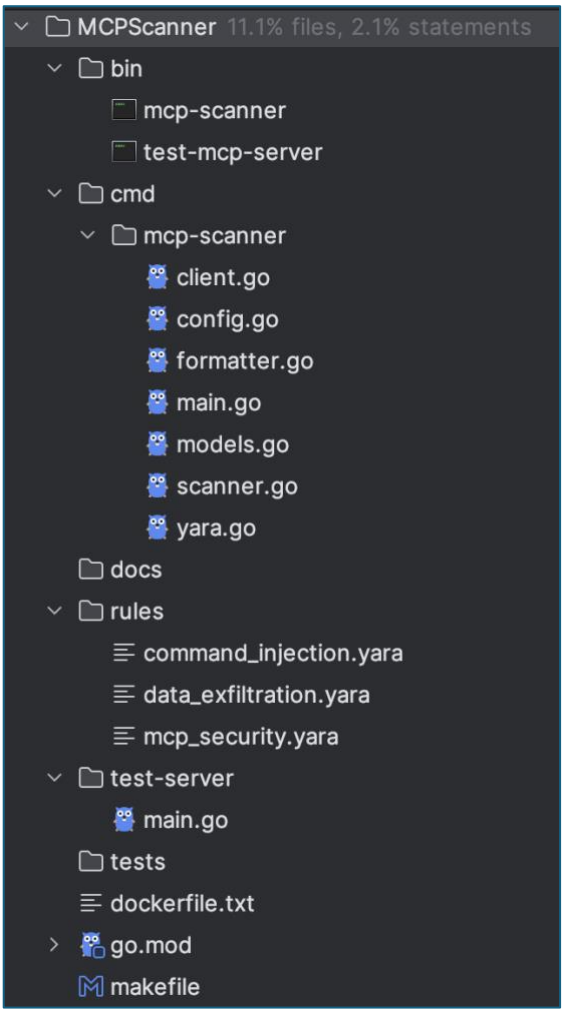
Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Andrew Snook
Student ID: x23283301

1. File Layout



test-mcp-server – this the test vulnerability server that needs to be running for the mcp-scanner to scan

```
./bin/test-mcp-server
```

mcp-scanner – the binary file that is created when the application is compiled with the following command:

```
./bin/mcp-scanner scan --server-url http://localhost:8000 --analyzers yara
```

client.go – this is the http MCP client which acts as the communication bridge between the scanner and the MCP servers. It serialises the Go structs to JSON-RPC requests which is that standard for RPC over HTTP. The main methods are listed below:

```
Connect()           Initialize connection & handshake
ListTools()         Fetch all tools from server
ListPrompts()       Fetch all prompts
ListResources()     Fetch all resources
sendRequest()       Low-level JSON-RPC communication
```

config.go – this file manages the setup of the different scanners. For the purposes of this it is just the YARA rules.

formatter.go – this manages the formatting out of output, displaying it into a readable view.

main.go – this is main entry point for the application. It sets up the commands that run the scanner, creates client and the scanner itself.

models.go – the data models that are used in the other files

scanner.go – this is the core scanner logic.

yara.go – this file categorises the types on vulnerabilities as part of the YARA analyser.

/test-server/main.go – test MCP server with the vulnerabilities that will be scanned and reported on.

go.mod – set the version of GoLang and the libraries that are used in the project.

2. Install instructions

2.1 Requirements

GoLang 1.21 or higher

2.2 From Source (the files in the MCPScanner.zip)

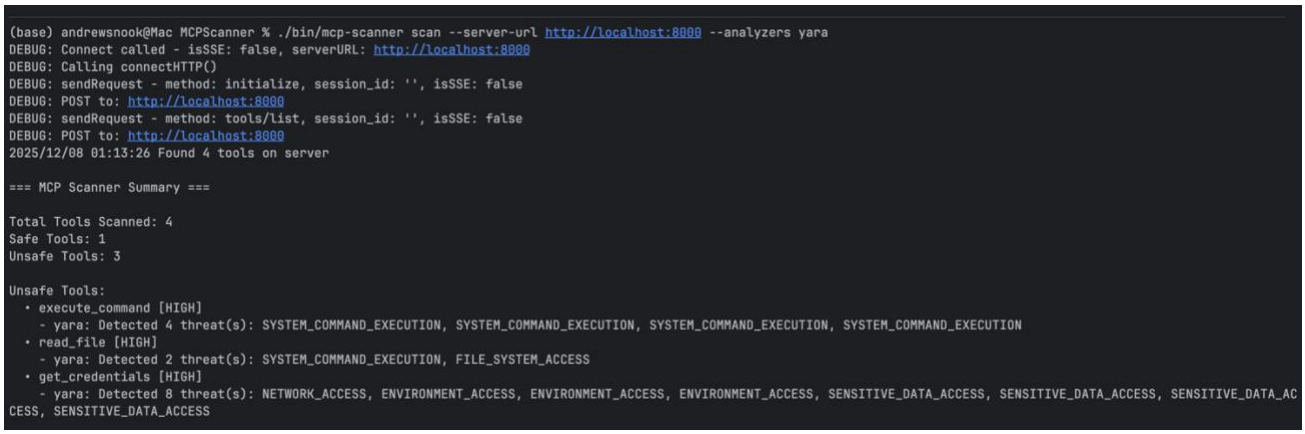
1. Open the solution in an IDE (GoLand (GoLand was used in this project), VS Code).
2. From the terminal, navigate to the location where the source files are located
3. Run the following commands in the terminal:
    a. `go mod download`
    b. `go build -o bin/mcp-scanner`
    c. `export MCP_SCANNER_YARA_RULES="./rules"`

2.3 Running the scanner

1. Start the server by running: ./`bin/test-mcp-server`
2. Start the scanner by running:

```
./bin/mcp-scanner scan --server-url http://localhost:8000 --analyzers yara
```

2.4 Expected output



(base) andrewsnook@Mac MCPScanner % ./bin/mcp-scanner scan --server-url http://localhost:8000 --analyzers yara
DEBUG: Connect called - isSSE: false, serverURL: http://localhost:8000
DEBUG: Calling connectHTTP()
DEBUG: sendRequest - method: initialize, session_id: '', isSSE: false
DEBUG: POST to: http://localhost:8000
DEBUG: sendRequest - method: tools/list, session_id: '', isSSE: false
DEBUG: POST to: http://localhost:8000
2025/12/08 01:13:26 Found 4 tools on server

=== MCP Scanner Summary ===

Total Tools Scanned: 4
Safe Tools: 1
Unsafe Tools: 3

Unsafe Tools:
• execute_command [HIGH]
- yara: Detected 4 threat(s): SYSTEM_COMMAND_EXECUTION, SYSTEM_COMMAND_EXECUTION, SYSTEM_COMMAND_EXECUTION, SYSTEM_COMMAND_EXECUTION
• read_file [HIGH]
- yara: Detected 2 threat(s): SYSTEM_COMMAND_EXECUTION, FILE_SYSTEM_ACCESS
• get_credentials [HIGH]
- yara: Detected 8 threat(s): NETWORK_ACCESS, ENVIRONMENT_ACCESS, ENVIRONMENT_ACCESS, ENVIRONMENT_ACCESS, SENSITIVE_DATA_ACCESS, SENSITIVE_DATA_ACCESS, SENSITIVE_DATA_ACCESS, SENSITIVE_DATA_ACCESS

3. Acknowledgements

3.1 Use of AI

Claude AI was used in this project to assist with debugging and best practices. GoLang is not my main language, in the past I have used other programming languages such as C#.

As I was getting to grips with it I needed some assistance here and there. I also relied on opensource MCP libraries such as https://github.com/invariantlabs-ai/mcp-scan - this uses Python though.

A common approach I used was to refer to https://mcp-go.dev and if I hit a blocker, I would ask Claude AI to debug.