RE Preprocessor Test Plan

Overall Test Plan

Testing will be conducted with unit and integration tests. I plan to use PyTest for unit testing and to use simulated data for standard and non-standard conditions for integration tests. Many tests will be run via a Github workflow which will catalog results and notify developers of issues before they are merged into the main branch.

Test Case Descriptions

PL1.0 Plugin Loading

PL1.1 System builds packaged plugins

PL1.1.1 During the build process, unit tests are run (see PL2.0)

PL1.2 System installs built plugins

PL1.3 System verifies expected plugins are discovered

PL2.0 Plugin Unit Tests

PL2.1 During the build process unit tests are conducted

PL2.2 Unit tests will use pytest to verify the functionality of core plugin components such as file detection

FI1.0 File Identification Test

FI1.1 System uses packaged plugin to identify a known-good test file

FI1.2 Results from identification tested against expected results

FE1.0 File Extraction

FE1.1 System uses a packaged plugin to identify a known-good test file

FE1.2 System uses a packaged plugin to extract a known-good test file

FE1.3 Resulting file is compared with expected results

RE1.0 Recursive/Nested File Extraction

RE1.1 System uses a known-good test file to identify and extract a file

RE1.2 System detects another compressed file within the original test file and extracts that file

RE1.3 System decompresses as many files are as contained in the known-good tester

RE1.4 Resulting files are compared with expected results

FIN1.0 File Information

FIN1.1 System detects an informational plugin being installed

FIN1.2 System uses an informational plugin to display additional details about a known-bad file (a file which the system cannot identify)

FIN1.3 Results are compared against expected

GH1.0 Ghidra Plugin Compatibility

GH1.1 A Ghidra plugin containing the preprocessor can be successfully installed into the latest public release of Ghidra (v12.0.2)

GH1.2 The plugin can be used to identify and extract a file which can then be loaded into Ghidra

GH1.3 Ghidra is able to successfully decompile the program

BN1.0 Binary Ninja Plugin Compatibility

BN1.1 A Binary Ninja plugin containing the preprocessor can be successfully installed into the latest public release of Binary Ninja (v5.2 ‘IO’)

BN1.2 The plugin can be used to identify and extract a file which can be loaded into Binary Ninja

BN1.3 Binary Ninja is able to decompile the compressed program

IDA1.0 IDA Pro Plugin Compatibility

IDA1.1 An IDA Pro plugin containing the preprocessor can be successfully installed into the latest public release of IDA Pro (v9.0)

IDA1.2 The plugin can be used to identify and extract a file which can then be loaded into IDA

IDA1.3 IDA is able to decompile the compressed program

UN1.0 Uncompressed File

UN1.1 Provide the system with a normal, uncompressed file

UN1.2 Verify the system recognizes that this file is not an error condition

CR1 Corrupt Archive

UN1.1 Provide the system with a corrupted archive

UN1.2 Verify the system recognizes that it cannot extract the file and registers it as an error

Test Case Matrix

Test Case	Normal/Abnormal	Blackbox/Whitebox	Functional/Performance	Unit/Integration
PL1	Normal	Blackbox	Functional	Integration
PL2	Normal	Whitebox	Functional	Unit
FI1	Normal	Blackbox	Functional	Integration
FE1	Normal	Blackbox	Functional	Integration
RE1	Normal	Blackbox	Functional	Integration
FIN1	Normal	Blackbox	Functional	Integration
GH1	Normal	Blackbox	Functional	Integration
BN1	Normal	Blackbox	Functional	Integration
IDA1	Normal	Blackbox	Functional	Integration
UN1	Abnormal	Blackbox	Functional	Integration
CR1	Abnormal	Blackbox	Functional	Integration