Malware Analysis Sandbox Report

Project: Malware Analysis Sandbox (Python)
Author / Analyst: Scofield
Date:1/09/2025
Environment: Windows, Python 3.13, pip, PowerShell (editable install)

1. Executive summary

This document summarizes the Malware Analysis Sandbox project implementation, lists the primary files and their functions, explains the errors encountered while installing and running the tool, shows how each error was fixed, and provides places to paste screenshots for a report or presentation. The sandbox is a lightweight Python CLI tool that runs a target executable in a monitored subprocess and writes a `report.json` (and optionally an HTML report).

2. How the tool works (high-level flow)

1. User calls the CLI:
   `malwaresandbox --file C:\path\to\suspicious.exe`

2. CLI (`main.py`) parses arguments and calls the sandbox runner.

3. Runner (`runner.py`) launches the target in a subprocess and calls monitoring modules.

4. Monitoring modules collect:

    - Processes (`monitor_process.py`)

    - File activity snapshot (`monitor_files.py`)

    - Network connections (`monitor_network.py`)

5. `report.py` collects the logs and writes `report.json` (and can be extended to HTML/PDF).

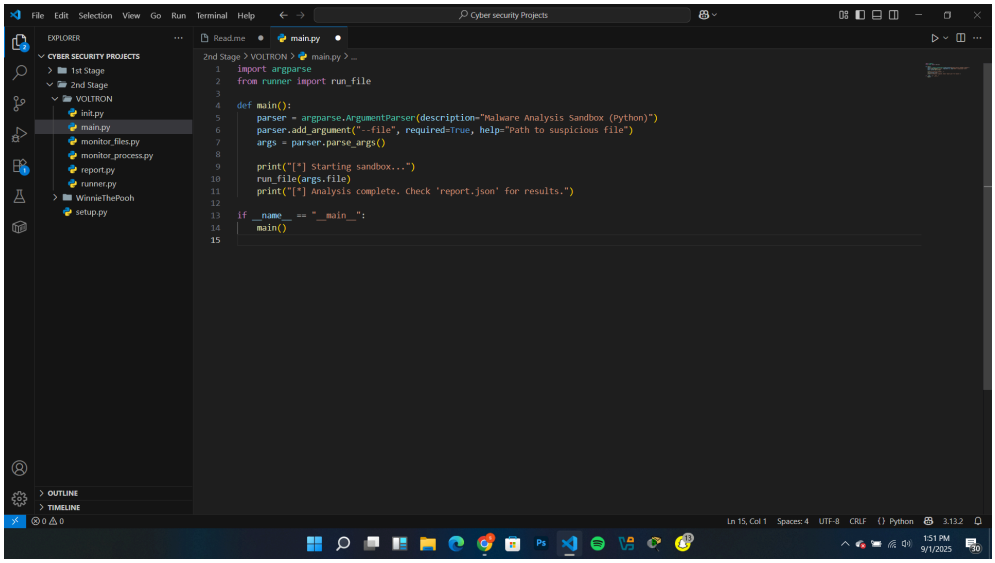
6. Runner terminates the target (best-effort) and completes.

3. Files & function descriptions

Use this section to explain each file so non-developers understand what they do.

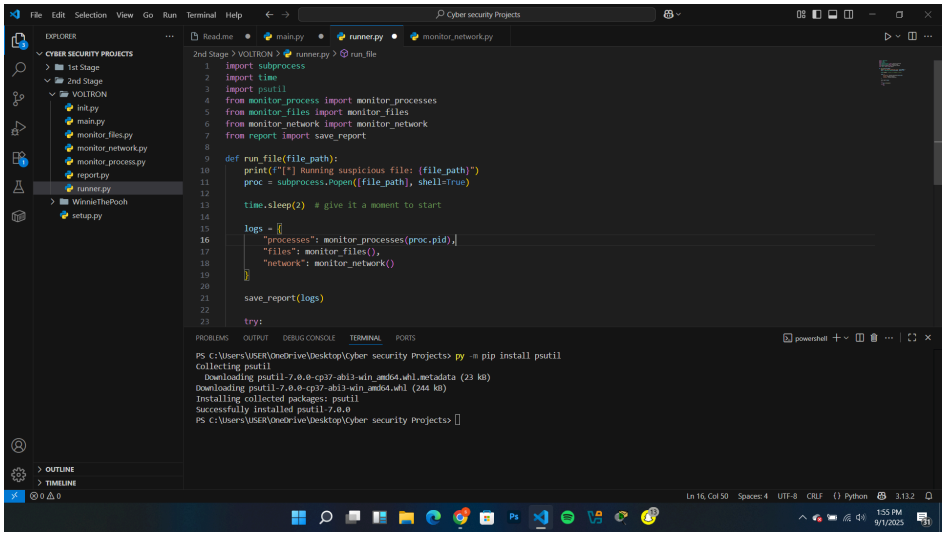
`malwaresandbox/main.py`

- **Role:** CLI entrypoint. Uses `argparse` to accept `--file`. Calls the runner.
- **Key behavior:** Validates input, prints start/complete messages, and hands off path to `runner.run_file()`.



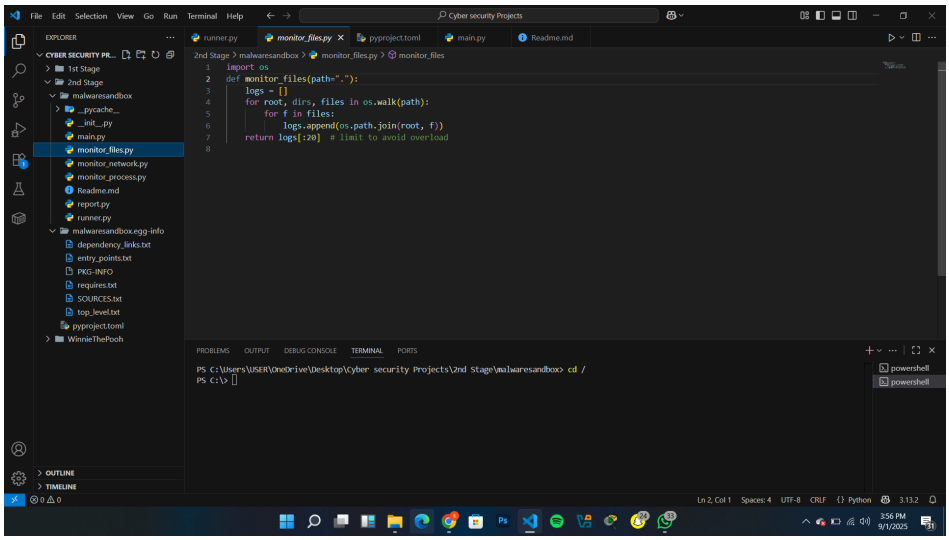
`malwaresandbox/runner.py`

- **Role:** Orchestrator. Launches the suspect file in a subprocess, calls monitors, and saves the report.
- **Important functions:**
    - `run_file(file_path)`: `subprocess.Popen([file_path], shell=True)` → waits a bit → calls monitors → saves results → attempts to terminate child.
- **Note:** Should use package-relative imports (e.g., `from .monitor_process import monitor_processes`).



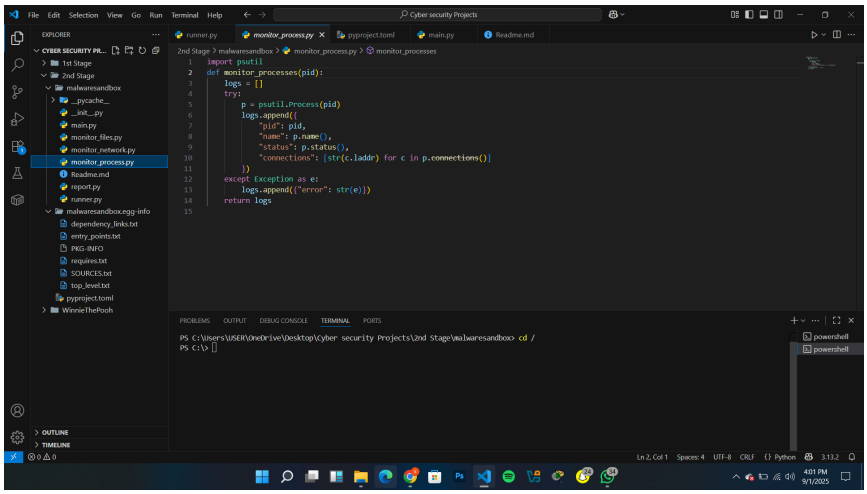
malwaresandbox/monitor_process.py

- **Role:** Uses `psutil` to inspect the spawned process (pid, name, status, connections).
- **Output:** A small dict/list describing the process and connections.



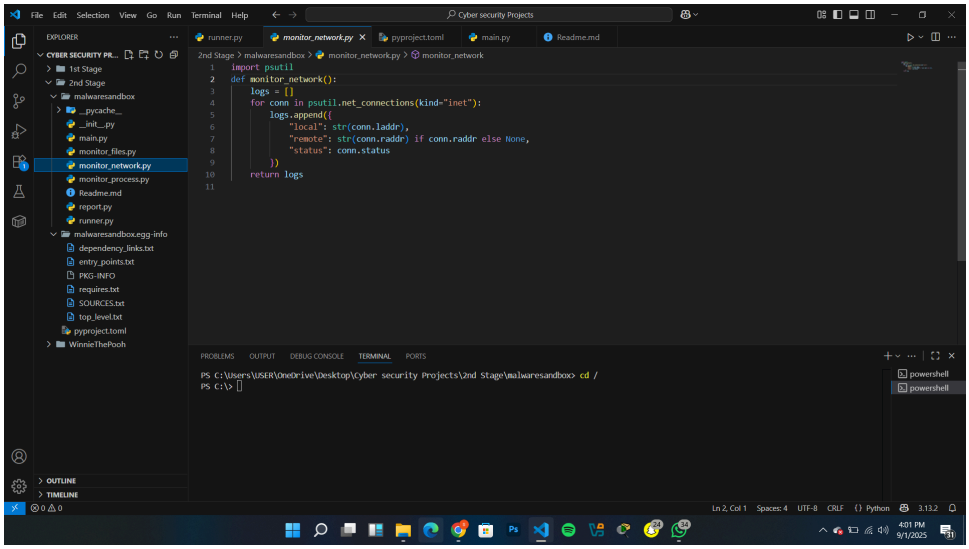
malwaresandbox/monitor_files.py

- **Role:** Walks a specified path (or `%TEMP%`) and records recently seen files. Simple snapshot; can be replaced by `watchdog` for event-based monitoring.
- **Output:** List of file paths (limited to avoid overload).



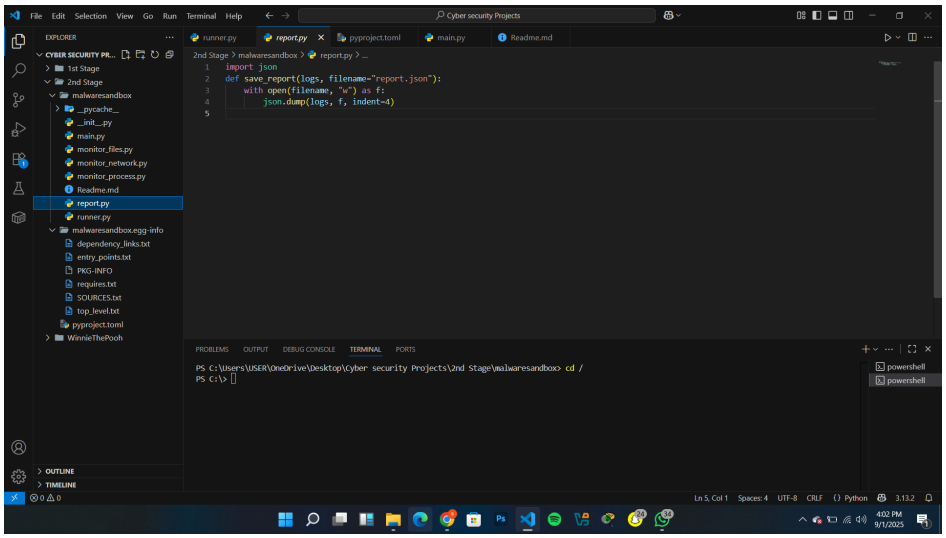
malwaresandbox/monitor_network.py

- **Role:** Uses `psutil.net_connections()` to list current network sockets and their states.
- **Output:** List of local/remote addresses and statuses.



malwaresandbox/report.py

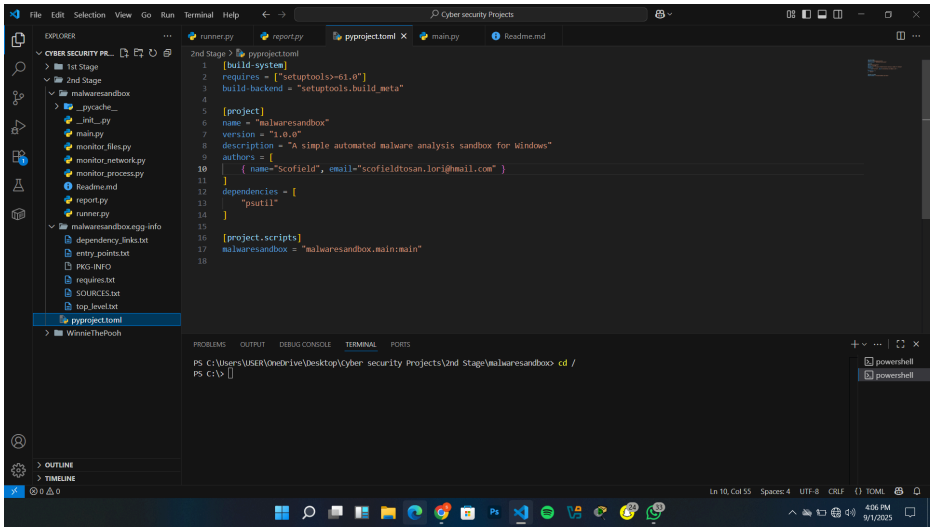
- **Role:** Serializes logs to `report.json` and helper to build HTML (optional). Keeps structure consistent for later parsing.



`pyproject.toml` (packaging file)

- **Role:** Declares package metadata and an entry point mapping `malwaresandbox` → `malwaresandbox.main:main`. Also lists dependencies like `psutil`.

- **Important note:** When using an editable install (`pip install -e .`), `pyproject.toml` should be consistent with package layout.



4. Errors encountered (what you saw) — cause & fix (copy into doc)

Error 1 `does not appear to be a Python project: neither 'setup.py' nor 'pyproject.toml' found`

- **What it means:** pip couldn't find a project configuration in the folder you pointed it at.
- **Cause:** `setup.py` or `pyproject.toml` missing / in wrong folder.
- **Fix:** Ensure `pyproject.toml` or `setup.py` exists at the root of the folder used for installation.

Error 2 : `AssertionError: Multiple .egg-info directories found`

- **What it means:** Multiple leftover build metadata directories were present, confusing setuptools.
- **Cause:** Repeated editable-installs left `.egg-info` directories scattered.
- **Fix:** Delete all `.egg-info`, `build/`, `dist/` directories:

```
Get-ChildItem -Recurse -Filter *.egg-info | Remove-Item -Recurse -Force
# or
Get-ChildItem -Recurse -Include *.egg-info, build, dist | Remove-Item -Recurse -Force
```

Then reinstall.

Error 3: `error: Multiple top-level packages discovered in a flat-layout: ['VOLTRON', 'WinnieThePooh']`

- **What it means:** pip/setuptools found several top-level directories and refused to decide which to package.
- **Cause:** The project root contained multiple unrelated packages/folders.

- **Fix (two options):**

    1. **Recommended:** Move only the sandbox package folder into a fresh directory (isolate the project), then install from there.

    2. **Alternative:** Tell setuptools explicitly which package to include by adding a `tool.setuptools.packages.find` section in `pyproject.toml`:

```
[tool.setuptools.packages.find]
include = ["malwaresandbox"]
```

Error 4: `WARNING: The script malwaresandbox.exe is installed in '...Scripts' which is not on PATH.`

- **What it means:** The console script was installed but the Scripts directory isn't in your PATH, so Windows can't find the command.

- **Fix:** Add the Scripts folder to Windows PATH via Environment Variables (System Properties → Advanced → Environment Variables → Path → New → paste `C:\Users\USER\AppData\Local\Programs\Python\Python313\Scripts`), then open a **new** CMD/PowerShell window.

Error 5: `ModuleNotFoundError: No module named 'malwaresandbox'` (then later `No module named 'runner'`, `No module named 'monitor_process'`)

- **What it means:** Python import resolution failed because imports used were not package-relative, and/or packaging entry point pointed to wrong module.

- **Cause(s):**

    1. CLI entrypoint referenced `malwaresandbox.main` but your package folder name or module layout differed.

    2. Files used absolute sibling imports like `from runner import run_file` instead of relative imports.

- **Fixes applied:**

1. Ensure package folder name matches the import target (rename `VOLTRON` → `malwaresandbox` or update entry point in `pyproject.toml` to `VOLTRON.main:main`).

2. Use package-relative imports inside modules:

    - Change `from runner import run_file` → `from .runner import run_file`

    - Change `from monitor_process import monitor_processes` → `from .monitor_process import monitor_processes`

3. Ensure `__init__.py` exists inside the package folder.

4. Reinstall editable package: `py -m pip install -e .` (or uninstall then reinstall).

6. Test cases performed & results

Test 1 Notepad (harmless executable)

- Command: `malwaresandbox --file "C:\Windows\System32\notepad.exe"`

- Result: Notepad launched; tool created `report.json` with process details. **PASS**

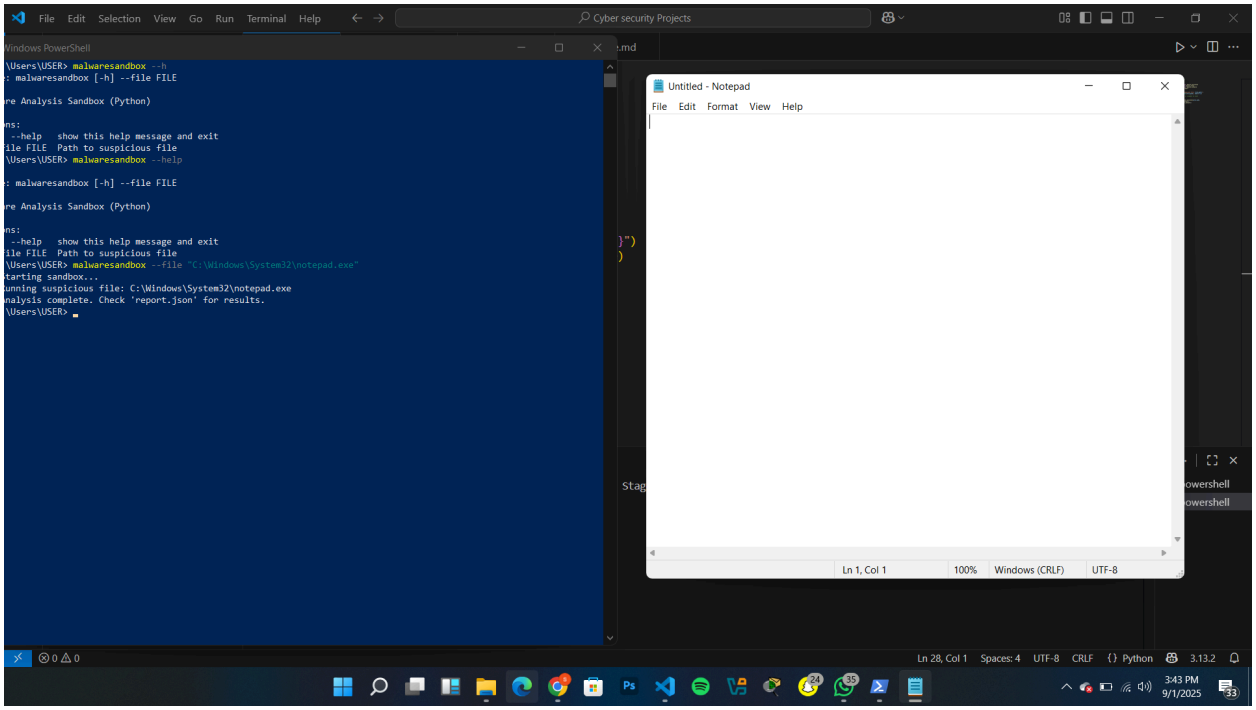
Test 2 — Batch script creating file

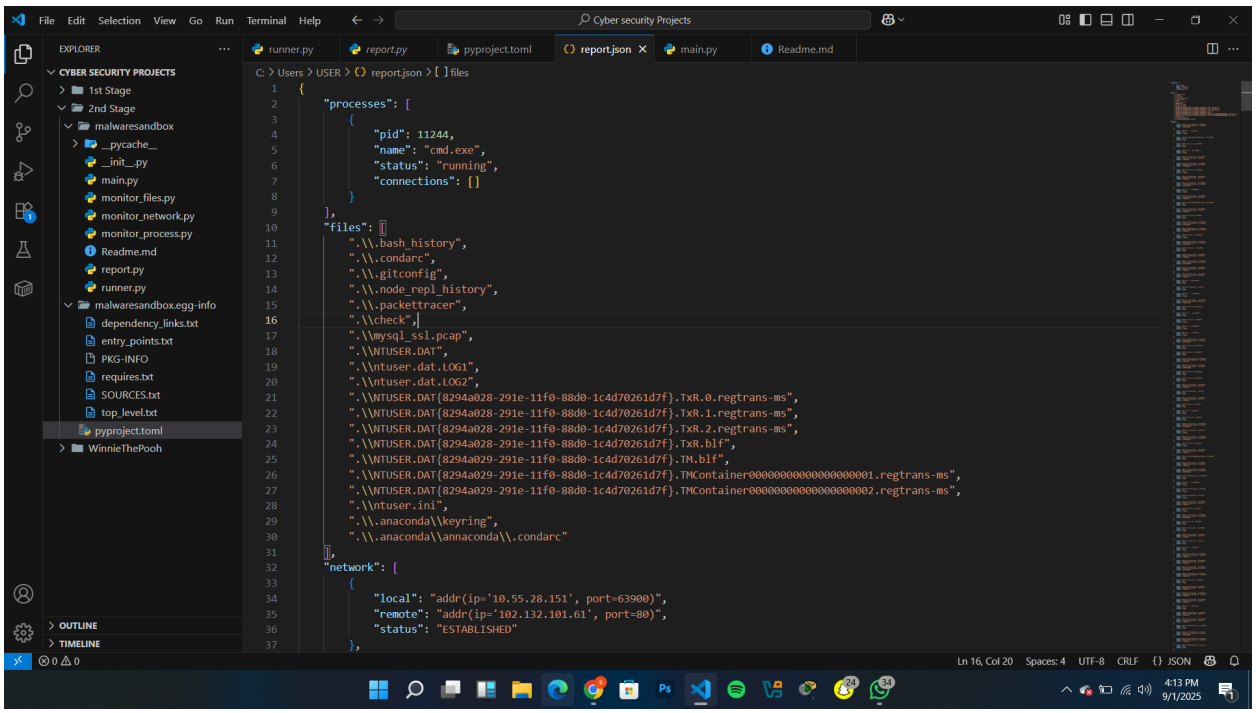
- Created `test.bat`:

```
@echo off
echo sandbox test > "%TEMP%\sandbox_test.txt"
timeout /t 5 > nul
```

- Command: `malwaresandbox --file "C:\Users\USER\Desktop\test.bat"`

- Result: `%TEMP%\sandbox_test.txt` created; `report.json` contained process info. **PASS**


File Edit Selection View Go Run Terminal Help
Cyber security Projects
Windows PowerShell
\Users\USER> malwaresandbox --h
: malwaresandbox [-h] --file FILE
re Analysis Sandbox (Python)
ns:
--help show this help message and exit
ile FILE Path to suspicious file
\Users\USER> malwaresandbox --help
: malwaresandbox [-h] --file FILE
re Analysis Sandbox (Python)
ns:
--help show this help message and exit
ile FILE Path to suspicious file
\Users\USER> malwaresandbox --file "C:\Windows\System32\notepad.exe"
tarting sandbox...
unning suspicious file: C:\Windows\System32\notepad.exe
nalysis complete. Check 'report.json' for results.
\Users\USER>
Untitled - Notepad
File Edit Format View Help
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
Ln 28, Col 1
Spaces: 4
UTF-8
CRLF
Python
3.13.2
3:43 PM
9/1/2025


File Edit Selection View Go Run Terminal Help
Cyber security Projects
EXPLORER
CYBER SECURITY PROJECTS
1st Stage
2nd Stage
malwaresandbox
__pycache__
__init__.py
main.py
monitor_files.py
monitor_network.py
monitor_process.py
Readme.md
report.py
runner.py
malwaresandbox.egg-info
dependency_links.txt
entry_points.txt
PKG-INFO
requires.txt
SOURCES.txt
top_level.txt
pyproject.toml
WinnieThePooh
OUTLINE
TIMELINE
runner.py
report.py
pyproject.toml
report.json
main.py
Readme.md
C: > Users > USER > report.json > [ ] files
{
"processes": [
{
"pid": 11244,
"name": "cmd.exe",
"status": "running",
"connections": []
}
],
"files": [
".\\.bash_history",
".\\.condarc",
".\\.gitconfig",
".\\.node_repl_history",
".\\.packettracer",
".\\check",
".\\mysql_ssl.pcap",
".\\NTUSER.DAT",
".\\ntuser.dat.LOG1",
".\\ntuser.dat.LOG2",
".\\NTUSER.DAT{8294a028-291e-11f0-88d0-1c4d70261d7f}.TxR.0.regtrans-ms",
".\\NTUSER.DAT{8294a028-291e-11f0-88d0-1c4d70261d7f}.TxR.1.regtrans-ms",
".\\NTUSER.DAT{8294a028-291e-11f0-88d0-1c4d70261d7f}.TxR.2.regtrans-ms",
".\\NTUSER.DAT{8294a028-291e-11f0-88d0-1c4d70261d7f}.TxR.blf",
".\\NTUSER.DAT{8294a029-291e-11f0-88d0-1c4d70261d7f}.TM.blf",
".\\NTUSER.DAT{8294a029-291e-11f0-88d0-1c4d70261d7f}.TMContainer00000000000000000001.regtrans-ms",
".\\NTUSER.DAT{8294a029-291e-11f0-88d0-1c4d70261d7f}.TMContainer00000000000000000002.regtrans-ms",
".\\ntuser.ini",
".\\.anaconda\\keyring",
".\\.anaconda\\annaconda\\.condarc"
],
"network": [
{
"local": "addr(ip='10.55.28.151', port=63900)",
"remote": "addr(ip='102.132.101.61', port=80)",
"status": "ESTABLISHED"
},
Ln 16, Col 20
Spaces: 4
UTF-8
CRLF
JSON
4:13 PM
9/1/2025