EYE - Focused Website Vulnerability Scanner

Executive summary

EYE v2 is a safe, permissioned, focused website vulnerability scanner implemented in Python with a PyQt5 GUI. It performs lightweight, non-exploitative probes for high-impact indicators (missing security headers, reflected inputs, SQL error disclosures, directory listing, and common admin paths). Results are presented with severity levels and remediation guidance. The app can export reports to TXT and HTML.

What EYE does

- **Fingerprinting** — reads web response headers (Server, X-Powered-By) for technology hints.
- **Security headers check** — looks for HSTS, Content-Security-Policy, X-Frame-Options, etc.
- **Robots & index check** — fetches `robots.txt` and looks for directory listings (e.g., `Index of /`).
- **Crawling (limited)** — parses internal links and simple forms (non-JS) from the target root.
- **Reflection tests (XSS indicator)** — sends harmless tokens to query parameters and form inputs to detect reflection.
- **SQL error heuristics** — sends a single quote to parameters to spot DB error messages (indicator-only).
- **Common path discovery** — HEAD checks for `/admin`, `/wp-admin`, `/login`, etc.
- **Forms checks** — non-exploitative reflection heuristics on form inputs.
- **Reporting** — findings include severity (CRITICAL / HIGH / MEDIUM / LOW), evidence, remediation and timestamps; exportable to TXT and HTML.

Safety: EYE is intentionally non-exploitative. Always scan only systems you own or where you have **explicit written permission**.

Tools & libraries used

- **Python 3.x** — implementation language.
- **PyQt5** — GUI framework (tabs, progress bar, dialogs).
- **requests** — HTTP(s) requests and headers.
- **beautifulsoup4 (bs4)** — HTML parsing for links & forms.
- **Standard libs** — `urllib.parse`, `re`, `datetime`, `html.escape`, `threading` (via PyQt `QThread`).

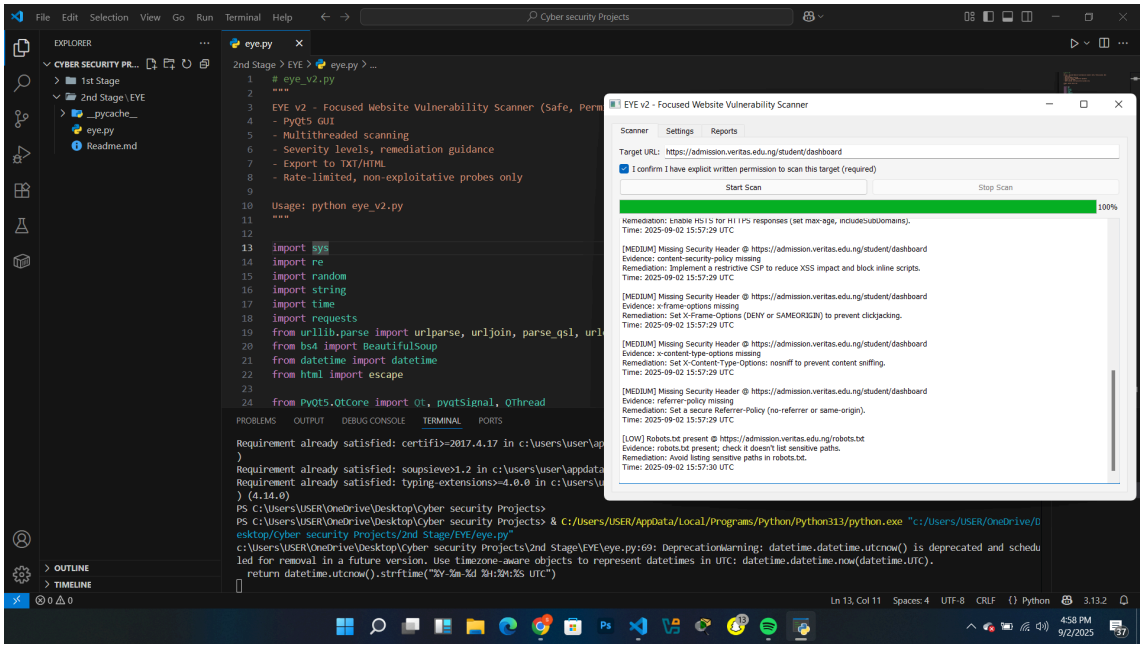
App structure & the tabs (from `eye.py`)

EYE v2 GUI uses 3 tabs — these are exactly what the code creates:

Tab: Scanner

This is the main operational tab.

- **Target URL input** — a QLineEdit where you enter the target (must include `http://` or `https://`).
- **Permission checkbox** — "I confirm I have explicit written permission…" (scan blocks until checked).
- **Start Scan / Stop Scan buttons** — start or stop the QThread scanner.
- **Progress bar** — shows scan progress (%) updated by the background thread.
- **Output area** — a read-only text box where progress messages and findings are printed live.

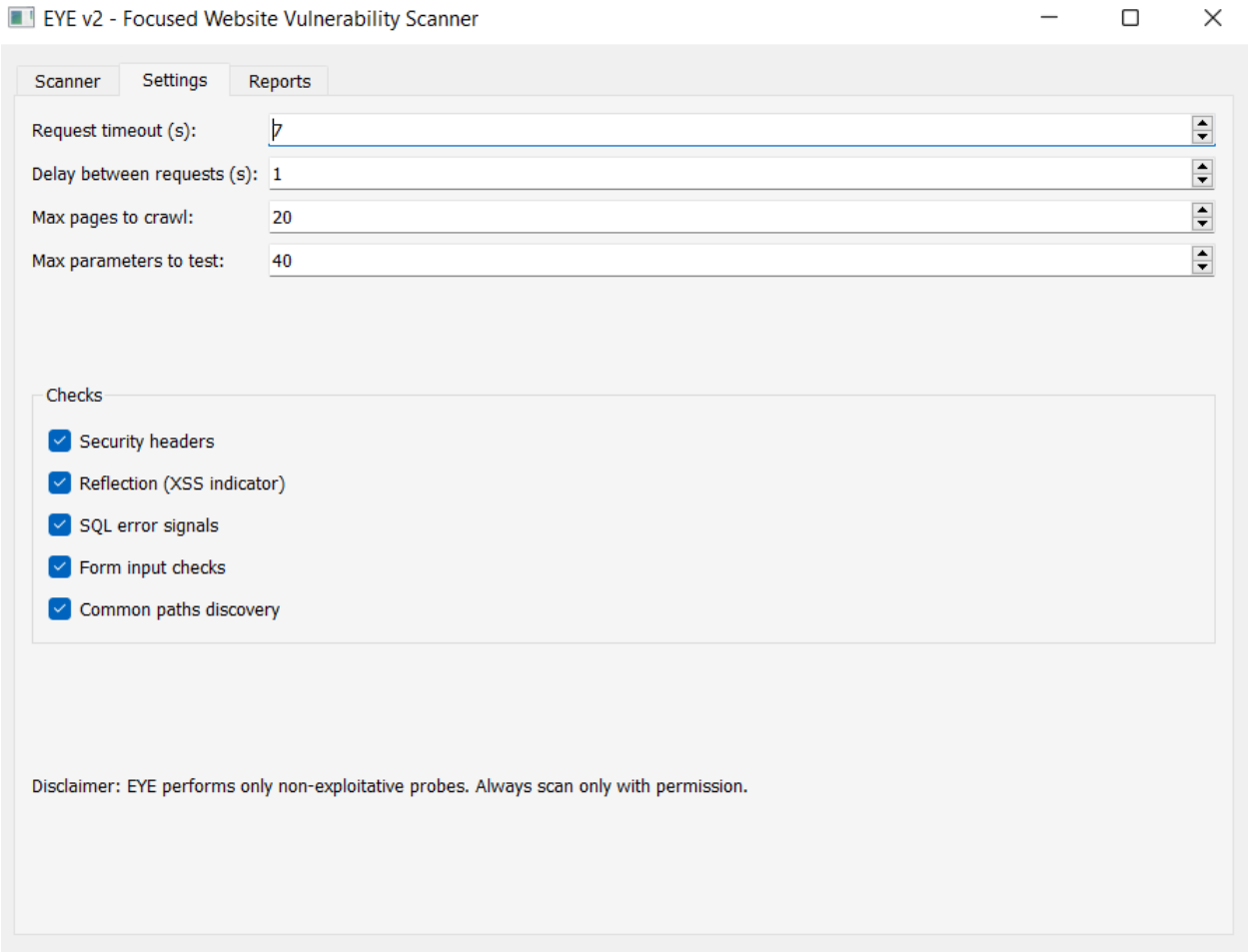


Tab: Settings

Used to tune scan speed and depth.

- **Request timeout (s)** — how long each HTTP request waits before failing.
- **Delay between requests (s)** — rate-limit requests to avoid overloading the target.
- **Max pages to crawl** — limit number of same-origin links crawled.
- **Max parameters to test** — stop after this many parameter tests.
- **Checks group** — checkboxes to enable/disable:
    - Security headers
    - Reflection (XSS indicator)
    - SQL error signals
    - Form input checks
    - Common paths discovery

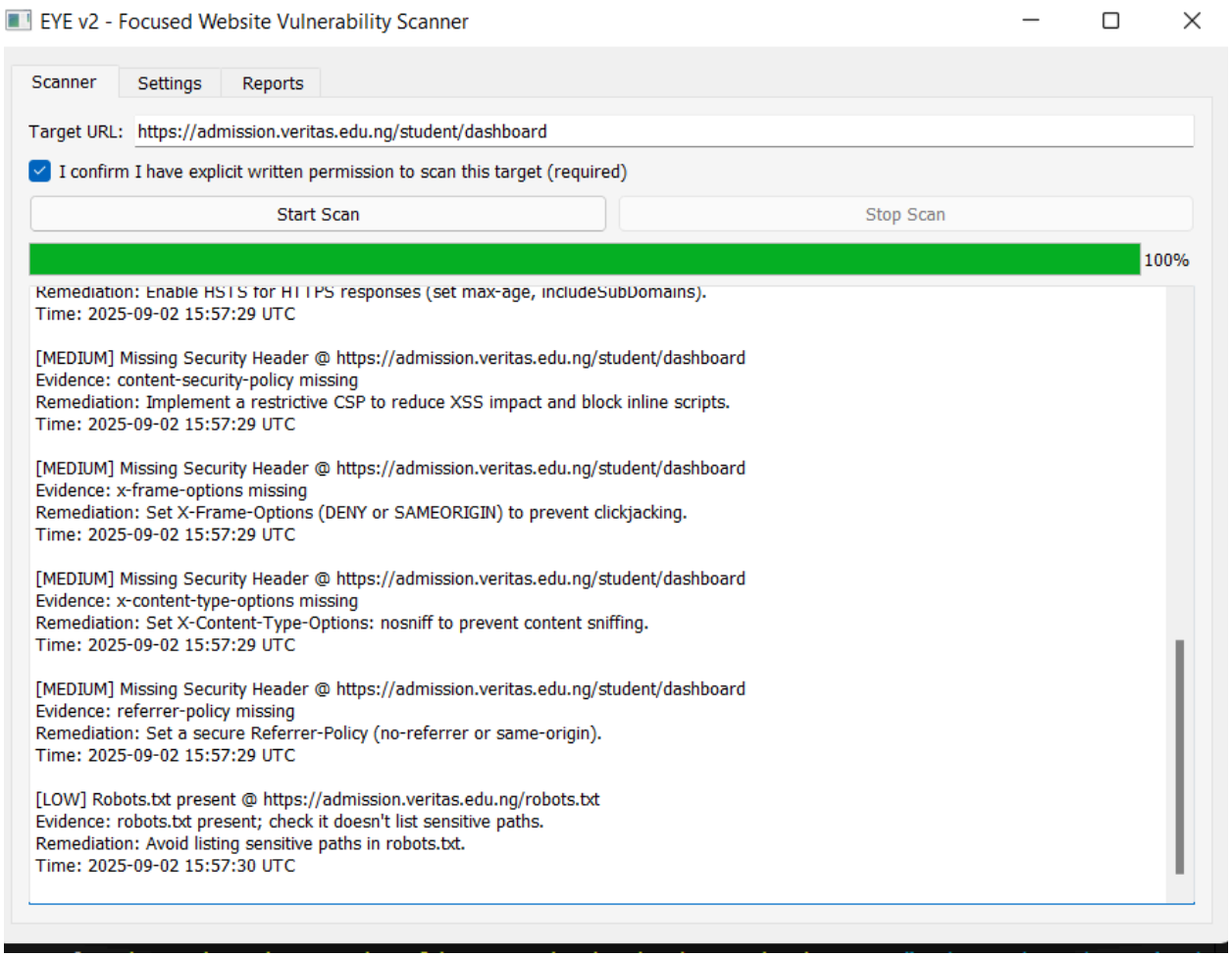
- **Disclaimer** — reminder about permission & legal constraints.



Tab: Reports

Where results are collected and saved.

- **Report viewer** — shows formatted HTML version of the findings after a scan.
- **Save Report (TXT)** — save a plain text report.
- **Save Report (HTML)** — save a styled HTML report for sharing.



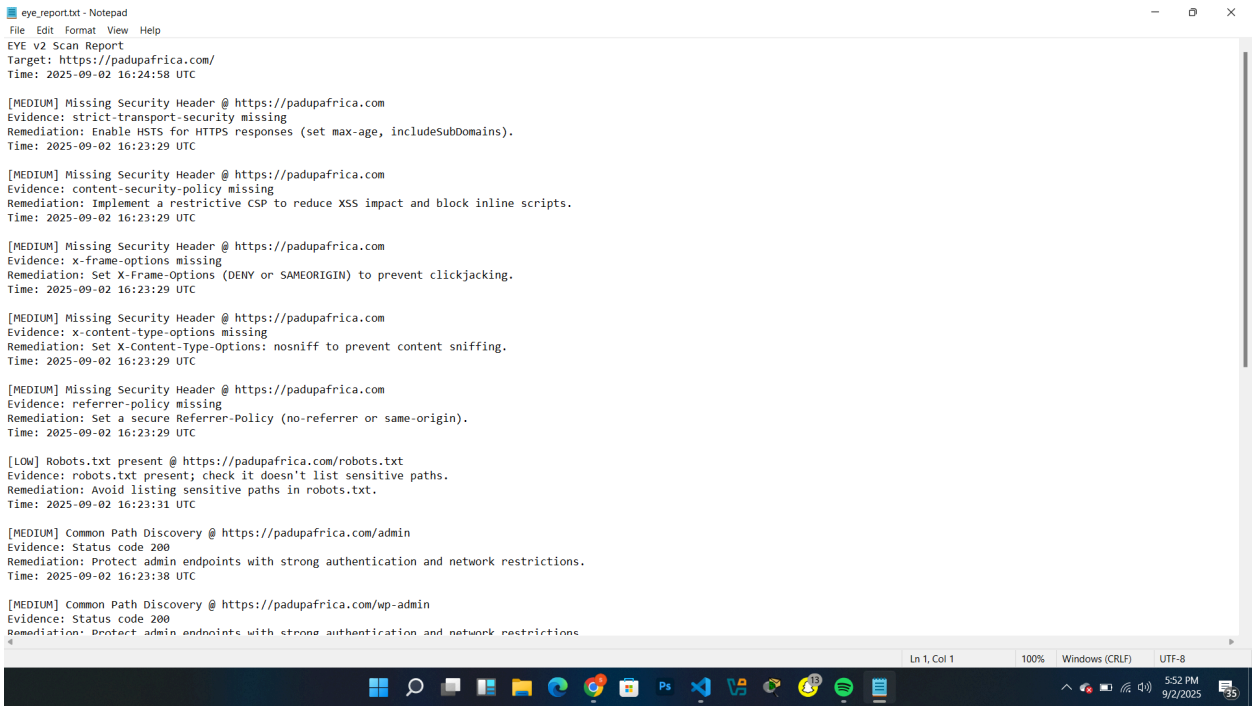
How findings are presented

Each finding is an instance of `Finding` with:

- **Severity**: CRITICAL / HIGH / MEDIUM / LOW
- **Kind**: e.g., “Possible SQL Error Disclosure”, “Missing Security Header”
- **URL**: where it was observed
- **Evidence**: snippet or description
- **Remediation**: short actionable advice

- **Timestamp**

The UI prints `Finding.to_text()` in the output area and `Finding.to_html()` into the Reports tab (for HTML export).



How to call EYE from PowerShell (so `eye --help` or `eye`)

Example usage

Show help (from any directory):

```
eye --help
```

- Output: the help text from the `.bat` (no GUI launched).

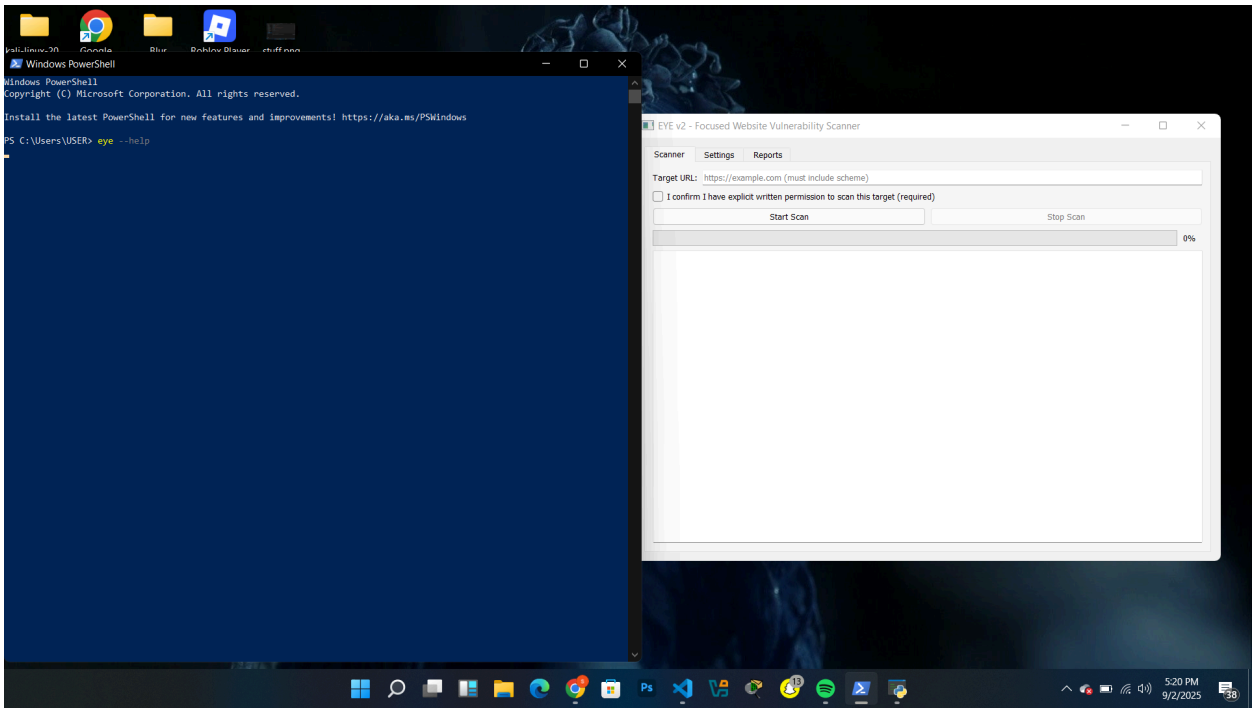
Launch GUI:

```
eye
```

or

```
eye arg1 arg2
```

- (args are forwarded to the Python script but the script currently ignores CLI args and will open the GUI.)



Conclusion

The Eye project demonstrates the potential of integrating multiple security-focused tools into a single, user-friendly application. By combining reconnaissance, scanning, and monitoring features, it provides both beginners and professionals with a central platform for understanding and analyzing network activities.

Through its tabbed interface, users can access different functionalities without relying on separate scripts or command-line tools. Additionally, with proper configuration, it can be called directly from PowerShell or the command prompt, making it versatile for different environments.

While this version focuses on functionality and usability, there is room for future improvements, such as enhanced automation, additional security modules, and improved reporting features. Overall, The Eye (V2) serves as a practical foundation for learning, testing, and applying core cybersecurity concepts in a controlled environment.