

Waris Abdullah 152697918

## **Secure Programming Exercise Work**

## 1. Introduction

StegoSecurity 4.0 is a Python-based desktop application that demonstrates steganography, the art of hiding secret messages (in this case, Python code) within innocent-looking text files. The project allows users to hide code in files, execute it safely, analyze its risks, detect hidden secrets, and generate reports. It's designed with security in mind, focusing on OWASP Top 10 principles to prevent malicious behavior, making it both educational and practical.

## 2. Project Overview

### 2.1 Purpose

StegoSecurity simulates a secure environment for:

- **Hiding:** Embedding secret Python code in text files using spaces and tabs.
- **Executing:** Running or analyzing the code safely.
- **Detecting:** Finding hidden code through randomness analysis.
- **Reporting:** Logging actions and generating PDF reports.

It's like a spy tool for hiding secrets, testing them safely, and catching sneaky behavior while keeping a record of everything.

### 2.2 Features

- **GUI:** A user-friendly window built with tkinter for input and output.
- **Hiding:** Encrypts and hides code in files as spaces/tabs.
- **Run Unsafe:** Executes code in a sandbox to limit damage.
- **Run Secure:** Analyzes code for threats without running it.
- **Detect Stego:** Finds hidden code using entropy.
- **Mitigation:** Neutralizes dangerous code by commenting it out.
- **Logging:** Tracks actions in a log file.
- **Config Checking:** Ensures secure settings.
- **Reporting:** Generates PDF reports of actions and configurations.

### 2.3 Libraries Used

- `tkinter`: Builds the graphical user interface (GUI).
- `os`: Manages files and folders.
- `Crypto (pycryptodome)`: Encrypts/decrypts code with AES.
- `base64`: Encodes/decodes for extra obfuscation.
- `ast`: Analyzes code structure for threat scoring.
- `reportlab`: Creates PDF reports.
- `collections.Counter`: Counts spaces/tabs for entropy.
- `math`: Performs entropy calculations.
- `json`: Reads configuration files.
- `datetime`: Timestamps logs.
- `tkinter.filedialog`: Opens file selection dialogs.

### 3. How StegoSecurity Works

#### 3.1 Workflow

1. **User Input:** Enter a cover text (e.g., a poem), secret code, passphrase, and number of files in the GUI.
2. **Hiding:** The code is encrypted, converted to binary, and hidden as spaces (0) and tabs (1) in files.
3. **Execution:**
  - **Unsafe:** Runs the code in a sandbox, allowing only safe functions.
  - **Secure:** Analyzes the code for threats without running it.
4. **Detection:** Checks files for hidden code using entropy (randomness).
5. **Mitigation:** Comments out dangerous lines (e.g., `# import os`).
6. **Logging:** Records all actions in `stego_log.txt`.
7. **Reporting:** Creates a PDF summarizing actions and config.

#### 3.2 User Interface

The GUI (`tkinter`) includes:

- **Input Areas:**
  - **Cover Text:** For the innocent text (e.g., “Roses are red”).
  - **Malicious Code:** For the secret code (e.g., `print("HACKED")`).
  - **Passphrase:** Encryption key (e.g., “secretkey123”).

- Number of Files: How many files to create (e.g., 1).
- **Buttons:**
  - Hide in Files: Encrypts and hides code.
  - Run Unsafe: Executes code in a sandbox.
  - Run Secure: Analyzes code for threats.
  - Detect Stego: Finds hidden code.
  - View Log: Shows stego\_log.txt.
  - Check Config: Verifies stego\_config.json.
  - Generate Report: Creates a PDF.
- **Output Area:** Displays results (e.g., threat scores, errors).

## 4. Code Explanation

Below is a detailed breakdown of how the code works, focusing on key functions and their roles.

### 4.1 Main File: stego\_security.py

The main script sets up the GUI, handles user actions, and integrates all features.

#### Hiding Code: hide\_in\_files()

```

1 # stego_security.py
2
3 def hide_in_files():
4     if not validate_inputs():
5         return
6
7     cover_text = cover_text.get("1.0", tk.END).rstrip()
8     malicious_code = malicious_code.get("1.0", tk.END).rstrip()
9     count = int(file_count.get())
10    method = stego_method.get()
11    passphrase = passphrase_entry.get()
12    obfuscate = obfuscate_var.get()
13
14    sanitized_malicious = sanitize_code(malicious_code)
15    if sanitized_malicious != malicious_code:
16        output.delete("1.0", tk.END)
17        output.insert(tk.END, f"Warning: Malicious code sanitized\nOriginal: {repr(malicious_code)}\nSanitized: {repr(sanitized_malicious)}\n", "error")
18        malicious_code = sanitized_malicious
19    else:
20        output.delete("1.0", tk.END)
21        output.insert(tk.END, "No sanitization needed\n", "success")
22
23    if obfuscate:
24        malicious_code = base64.b64encode(malicious_code.encode()).decode()
25        output.insert(tk.END, "Code obfuscated with base64\n", "success")
26
27    encrypted = encrypt_code(malicious_code, passphrase)
28    binary = "".join(format(byte, "08b") for byte in encrypted)
29    if method == "uppercase":
30        hidden = "".join(" " if bit == "0" else "\t" for bit in binary)
31    else:
32        hidden = "".join("\n0000" if bit == "0" else "\n1000" for bit in binary)
33
34    full_content = f"{cover_text}\n...\n{hidden}\n"
35    if len(full_content) > config["max_file_size"]:
36        output.insert(tk.END, f"Error: Content exceeds max file size ({config['max_file_size']} bytes)\n", "error")
37        log_action("Hide failed", "File size exceeded")
38        return
39
40    for i in range(count):
41        filename = os.path.join(FILE_DIR, f"stego_file_{i+1}.txt")
42        with open(filename, "a", encoding="utf-8") as f:
43            f.write(full_content)
44        output.insert(tk.END, f"Created: stego_file_{i+1}.txt (Method: {method}, Encrypted)\n", "success")
45        log_action("File created", f"{filename}")

```

## How It Works:

- Validates inputs using `validate_inputs()`.
- Gets cover text, malicious code, file count, stego method, passphrase, and obfuscation setting.
- Sanitizes code to remove unsafe characters.
- Obfuscates with base64 if selected.
- Encrypts code with `encrypt_code()`.
- Converts to binary, then to spaces/tabs or zero-width characters.
- Writes cover text, "---", and hidden data to files with UTF-8 encoding.
- Checks file size against `max_file_size`.
- Updates file list and logs actions.

## Encryption: `encrypt_code()`

**Purpose:** Locks the code with AES encryption.

```
def encrypt_code(code, passphrase):  
    key = passphrase.encode().ljust(16)[:16]  
    iv = get_random_bytes(16)  
    cipher = AES.new(key, AES.MODE_CBC, iv)  
    padded_code = pad(code.encode(), AES.block_size)  
    encrypted = cipher.encrypt(padded_code)  
    return iv + encrypted
```

## How It Works:

- Create a 16-byte key from the passphrase.
- Generates a random 16-byte IV.
- Pads the code to fit AES block size (16 bytes).
- Encrypts using AES-CBC mode.
- Returns IV + encrypted code.

## Decryption: `decrypt_code()`

**Purpose:** Unlocks the code.

```
def decrypt_code(encrypted, passphrase):
    key = passphrase.encode().ljust(16)[:16]
    iv = encrypted[:16]
    ciphertext = encrypted[16:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted = unpad(cipher.decrypt(ciphertext), AES.block_size)
    return decrypted.decode()
```

## How It Works:

- Extracts IV and ciphertext.
- Decrypts with the passphrase-derived key.
- Removes padding.
- Returns the original code (or errors if passphrase is wrong).

## Run Unsafe: run\_unsafe()

**Purpose:** Executes code in a sandbox.

```
# RunUnsafe.py / 00 unsafe_code
251 def run_unsafe():
252     files = findallig.getopenfilenames(initialdir=FILES_DIR, filetype=[("text files", "*.txt")])
253     if not files:
254         output.delete("1.0", tk.END)
255         output.insert(tk.END, "No files selected\n", "error")
256         update_status("No files selected")
257         log_action("Run unsafe failed", "No files selected")
258         return
259     output.delete("1.0", tk.END)
260     method = stage_method.get()
261     passphrase = passphrase_entry.get()
262     obfuscate = obfuscate_var.get()
263
264     allowed_builtins = [
265         "print",
266         "open",
267         "range",
268         "len",
269         "str",
270         "int",
271         "sorted",
272         "vars",
273         "dir",
274         "eval",
275         "exec",
276         "input",
277         "exit",
278         "sys",
279         "os",
280         "path",
281         "re",
282         "math",
283         "random",
284         "time",
285         "datetime",
286         "json",
287         "pickle",
288         "copy",
289         "collections",
290         "itertools",
291         "functools",
292         "operator",
293         "heapq",
294         "queue",
295         "threading",
296         "multiprocessing",
297         "subprocess",
298         "shutil",
299         "glob",
300         "urllib",
301         "http",
302         "ftplib",
303         "imaplib",
304         "nntplib",
305         "poplib",
306         "smtpd",
307         "telnetlib",
308         "urllib2",
309         "urllib3",
310         "xml",
311         "xmlrpc",
312         "zipfile",
313         "tarfile",
314         "gzip",
315         "bz2",
316         "lzma",
317         "zlib",
318         "cPickle",
319         "cStringIO",
320         "StringIO",
321         "io",
322         "cProfile",
323         "pstats",
324         "trace",
325         "traceback",
326         "logging",
327         "unittest",
328         "doctest",
329         "pdb",
330         "code",
331         "codeop",
332         "compileall",
333         "dis",
334         "dircache",
335         "filecmp",
336         "fileinput",
337         "filelock",
338         "fileutils",
339         "formatter",
340         "gettext",
341         "gettext",
342         "gettext",
343         "gettext",
344         "gettext",
345         "gettext",
346         "gettext",
347         "gettext",
348         "gettext",
349         "gettext",
350         "gettext",
351         "gettext",
352         "gettext",
353         "gettext",
354         "gettext",
355         "gettext",
356         "gettext",
357         "gettext",
358         "gettext",
359         "gettext",
360         "gettext",
361         "gettext",
362         "gettext",
363         "gettext",
364         "gettext",
365         "gettext",
366         "gettext",
367         "gettext",
368         "gettext",
369         "gettext",
370         "gettext",
371         "gettext",
372         "gettext",
373         "gettext",
374         "gettext",
375         "gettext",
376         "gettext",
377         "gettext",
378         "gettext",
379         "gettext",
380         "gettext",
381         "gettext",
382         "gettext",
383         "gettext",
384         "gettext",
385         "gettext",
386         "gettext",
387         "gettext",
388         "gettext",
389         "gettext",
390         "gettext",
391         "gettext",
392         "gettext",
393         "gettext",
394         "gettext",
395         "gettext",
396         "gettext",
397         "gettext",
398         "gettext",
399         "gettext",
400         "gettext",
401         "gettext",
402         "gettext",
403         "gettext",
404         "gettext",
405         "gettext",
406         "gettext",
407         "gettext",
408         "gettext",
409         "gettext",
410         "gettext",
411         "gettext",
412         "gettext",
413         "gettext",
414         "gettext",
415         "gettext",
416         "gettext",
417         "gettext",
418         "gettext",
419         "gettext",
420         "gettext",
421         "gettext",
422         "gettext",
423         "gettext",
424         "gettext",
425         "gettext",
426         "gettext",
427         "gettext",
428         "gettext",
429         "gettext",
430         "gettext",
431         "gettext",
432         "gettext",
433         "gettext",
434         "gettext",
435         "gettext",
436         "gettext",
437         "gettext",
438         "gettext",
439         "gettext",
440         "gettext",
441         "gettext",
442         "gettext",
443         "gettext",
444         "gettext",
445         "gettext",
446         "gettext",
447         "gettext",
448         "gettext",
449         "gettext",
450         "gettext",
451         "gettext",
452         "gettext",
453         "gettext",
454         "gettext",
455         "gettext",
456         "gettext",
457         "gettext",
458         "gettext",
459         "gettext",
460         "gettext",
461         "gettext",
462         "gettext",
463         "gettext",
464         "gettext",
465         "gettext",
466         "gettext",
467         "gettext",
468         "gettext",
469         "gettext",
470         "gettext",
471         "gettext",
472         "gettext",
473         "gettext",
474         "gettext",
475         "gettext",
476         "gettext",
477         "gettext",
478         "gettext",
479         "gettext",
480         "gettext",
481         "gettext",
482         "gettext",
483         "gettext",
484         "gettext",
485         "gettext",
486         "gettext",
487         "gettext",
488         "gettext",
489         "gettext",
490         "gettext",
491         "gettext",
492         "gettext",
493         "gettext",
494         "gettext",
495         "gettext",
496         "gettext",
497         "gettext",
498         "gettext",
499         "gettext",
500         "gettext",
501         "gettext",
502         "gettext",
503         "gettext",
504         "gettext",
505         "gettext",
506         "gettext",
507         "gettext",
508         "gettext",
509         "gettext",
510         "gettext",
511         "gettext",
512         "gettext",
513         "gettext",
514         "gettext",
515         "gettext",
516         "gettext",
517         "gettext",
518         "gettext",
519         "gettext",
520         "gettext",
521         "gettext",
522         "gettext",
523         "gettext",
524         "gettext",
525         "gettext",
526         "gettext",
527         "gettext",
528         "gettext",
529         "gettext",
530         "gettext",
531         "gettext",
532         "gettext",
533         "gettext",
534         "gettext",
535         "gettext",
536         "gettext",
537         "gettext",
538         "gettext",
539         "gettext",
540         "gettext",
541         "gettext",
542         "gettext",
543         "gettext",
544         "gettext",
545         "gettext",
546         "gettext",
547         "gettext",
548         "gettext",
549         "gettext",
550         "gettext",
551         "gettext",
552         "gettext",
553         "gettext",
554         "gettext",
555         "gettext",
556         "gettext",
557         "gettext",
558         "gettext",
559         "gettext",
560         "gettext",
561         "gettext",
562         "gettext",
563         "gettext",
564         "gettext",
565         "gettext",
566         "gettext",
567         "gettext",
568         "gettext",
569         "gettext",
570         "gettext",
571         "gettext",
572         "gettext",
573         "gettext",
574         "gettext",
575         "gettext",
576         "gettext",
577         "gettext",
578         "gettext",
579         "gettext",
580         "gettext",
581         "gettext",
582         "gettext",
583         "gettext",
584         "gettext",
585         "gettext",
586         "gettext",
587         "gettext",
588         "gettext",
589         "gettext",
590         "gettext",
591         "gettext",
592         "gettext",
593         "gettext",
594         "gettext",
595         "gettext",
596         "gettext",
597         "gettext",
598         "gettext",
599         "gettext",
600         "gettext",
601         "gettext",
602         "gettext",
603         "gettext",
604         "gettext",
605         "gettext",
606         "gettext",
607         "gettext",
608         "gettext",
609         "gettext",
610         "gettext",
611         "gettext",
612         "gettext",
613         "gettext",
614         "gettext",
615         "gettext",
616         "gettext",
617         "gettext",
618         "gettext",
619         "gettext",
620         "gettext",
621         "gettext",
622         "gettext",
623         "gettext",
624         "gettext",
625         "gettext",
626         "gettext",
627         "gettext",
628         "gettext",
629         "gettext",
630         "gettext",
631         "gettext",
632         "gettext",
633         "gettext",
634         "gettext",
635         "gettext",
636         "gettext",
637         "gettext",
638         "gettext",
639         "gettext",
640         "gettext",
641         "gettext",
642         "gettext",
643         "gettext",
644         "gettext",
645         "gettext",
646         "gettext",
647         "gettext",
648         "gettext",
649         "gettext",
650         "gettext",
651         "gettext",
652         "gettext",
653         "gettext",
654         "gettext",
655         "gettext",
656         "gettext",
657         "gettext",
658         "gettext",
659         "gettext",
660         "gettext",
661         "gettext",
662         "gettext",
663         "gettext",
664         "gettext",
665         "gettext",
666         "gettext",
667         "gettext",
668         "gettext",
669         "gettext",
670         "gettext",
671         "gettext",
672         "gettext",
673         "gettext",
674         "gettext",
675         "gettext",
676         "gettext",
677         "gettext",
678         "gettext",
679         "gettext",
680         "gettext",
681         "gettext",
682         "gettext",
683         "gettext",
684         "gettext",
685         "gettext",
686         "gettext",
687         "gettext",
688         "gettext",
689         "gettext",
690         "gettext",
691         "gettext",
692         "gettext",
693         "gettext",
694         "gettext",
695         "gettext",
696         "gettext",
697         "gettext",
698         "gettext",
699         "gettext",
700         "gettext",
701         "gettext",
702         "gettext",
703         "gettext",
704         "gettext",
705         "gettext",
706         "gettext",
707         "gettext",
708         "gettext",
709         "gettext",
710         "gettext",
711         "gettext",
712         "gettext",
713         "gettext",
714         "gettext",
715         "gettext",
716         "gettext",
717         "gettext",
718         "gettext",
719         "gettext",
720         "gettext",
721         "gettext",
722         "gettext",
723         "gettext",
724         "gettext",
725         "gettext",
726         "gettext",
727         "gettext",
728         "gettext",
729         "gettext",
730         "gettext",
731         "gettext",
732         "gettext",
733         "gettext",
734         "gettext",
735         "gettext",
736         "gettext",
737         "gettext",
738         "gettext",
739         "gettext",
740         "gettext",
741         "gettext",
742         "gettext",
743         "gettext",
744         "gettext",
745         "gettext",
746         "gettext",
747         "gettext",
748         "gettext",
749         "gettext",
750         "gettext",
751         "gettext",
752         "gettext",
753         "gettext",
754         "gettext",
755         "gettext",
756         "gettext",
757         "gettext",
758         "gettext",
759         "gettext",
760         "gettext",
761         "gettext",
762         "gettext",
763         "gettext",
764         "gettext",
765         "gettext",
766         "gettext",
767         "gettext",
768         "gettext",
769         "gettext",
770         "gettext",
771         "gettext",
772         "gettext",
773         "gettext",
774         "gettext",
775         "gettext",
776         "gettext",
777         "gettext",
778         "gettext",
779         "gettext",
780         "gettext",
781         "gettext",
782         "gettext",
783         "gettext",
784         "gettext",
785         "gettext",
786         "gettext",
787         "gettext",
788         "gettext",
789         "gettext",
790         "gettext",
791         "gettext",
792         "gettext",
793         "gettext",
794         "gettext",
795         "gettext",
796         "gettext",
797         "gettext",
798         "gettext",
799         "gettext",
800         "gettext",
801         "gettext",
802         "gettext",
803         "gettext",
804         "gettext",
805         "gettext",
806         "gettext",
807         "gettext",
808         "gettext",
809         "gettext",
810         "gettext",
811         "gettext",
812         "gettext",
813         "gettext",
814         "gettext",
815         "gettext",
816         "gettext",
817         "gettext",
818         "gettext",
819         "gettext",
820         "gettext",
821         "gettext",
822         "gettext",
823         "gettext",
824         "gettext",
825         "gettext",
826         "gettext",
827         "gettext",
828         "gettext",
829         "gettext",
830         "gettext",
831         "gettext",
832         "gettext",
833         "gettext",
834         "gettext",
835         "gettext",
836         "gettext",
837         "gettext",
838         "gettext",
839         "gettext",
840         "gettext",
841         "gettext",
842         "gettext",
843         "gettext",
844         "gettext",
845         "gettext",
846         "gettext",
847         "gettext",
848         "gettext",
849         "gettext",
850         "gettext",
851         "gettext",
852         "gettext",
853         "gettext",
854         "gettext",
855         "gettext",
856         "gettext",
857         "gettext",
858         "gettext",
859         "gettext",
860         "gettext",
861         "gettext",
862         "gettext",
863         "gettext",
864         "gettext",
865         "gettext",
866         "gettext",
867         "gettext",
868         "gettext",
869         "gettext",
870         "gettext",
871         "gettext",
872         "gettext",
873         "gettext",
874         "gettext",
875         "gettext",
876         "gettext",
877         "gettext",
878         "gettext",
879         "gettext",
880         "gettext",
881         "gettext",
882         "gettext",
883         "gettext",
884         "gettext",
885         "gettext",
886         "gettext",
887         "gettext",
888         "gettext",
889         "gettext",
890         "gettext",
891         "gettext",
892         "gettext",
893         "gettext",
894         "gettext",
895         "gettext",
896         "gettext",
897         "gettext",
898         "gettext",
899         "gettext",
900         "gettext",
901         "gettext",
902         "gettext",
903         "gettext",
904         "gettext",
905         "gettext",
906         "gettext",
907         "gettext",
908         "gettext",
909         "gettext",
910         "gettext",
911         "gettext",
912         "gettext",
913         "gettext",
914         "gettext",
915         "gettext",
916         "gettext",
917         "gettext",
918         "gettext",
919         "gettext",
920         "gettext",
921         "gettext",
922         "gettext",
923         "gettext",
924         "gettext",
925         "gettext",
926         "gettext",
927         "gettext",
928         "gettext",
929         "gettext",
930         "gettext",
931         "gettext",
932         "gettext",
933         "gettext",
934         "gettext",
935         "gettext",
936         "gettext",
937         "gettext",
938         "gettext",
939         "gettext",
940         "gettext",
941         "gettext",
942         "gettext",
943         "gettext",
944         "gettext",
945         "gettext",
946         "gettext",
947         "gettext",
948         "gettext",
949         "gettext",
950         "gettext",
951         "gettext",
952         "gettext",
953         "gettext",
954         "gettext",
955         "gettext",
956         "gettext",
957         "gettext",
958         "gettext",
959         "gettext",
960         "gettext",
961         "gettext",
962         "gettext",
963         "gettext",
964         "gettext",
965         "gettext",
966         "gettext",
967         "gettext",
968         "gettext",
969         "gettext",
970         "gettext",
971         "gettext",
972         "gettext",
973         "gettext",
974         "gettext",
975         "gettext",
976         "gettext",
977         "gettext",
978         "gettext",
979         "gettext",
980         "gettext",
981         "gettext",
982         "gettext",
983         "gettext",
984         "gettext",
985         "gettext",
986         "gettext",
987         "gettext",
988         "gettext",
989         "gettext",
990         "gettext",
991         "gettext",
992         "gettext",
993         "gettext",
994         "gettext",
995         "gettext",
996         "gettext",
997         "gettext",
998         "gettext",
999         "gettext",
1000        
```

```

def run_unsafe():
    if not any(c in "\t" for c in hidden):
        output.insert(tk.END, f"[os.path.basename(filename)]: Hidden section has no spaces/tabs\n", "error")
        continue
    binary = "".join("0" if char == "\t" else "1" for char in hidden if char in "\t")
    else:
        if not any(c in "\u2000\u200c" for c in hidden):
            output.insert(tk.END, f"[os.path.basename(filename)]: Hidden section has no zero-width chars\n", "error")
            continue
        binary = "".join("0" if char == "\u2000" else "1" for char in hidden if char in "\u2000\u200c")

    if not binary or len(binary) % 8 != 0:
        output.insert(tk.END, f"[os.path.basename(filename)]: Invalid hidden code length ({len(binary)} bits)\n", "error")
        continue

    encrypted = bytes(int(binary[i:i+8], 2) for i in range(0, len(binary), 8))
    malicious = decrypt_code(encrypted, passphrase)
    if obfuscate:
        malicious = base64.b64decode(malicious).decode()

    stdout_buffer = StringIO()
    old_stdout = sys.stdout
    sys.stdout = stdout_buffer
    try:
        exec(malicious, allowed_builtins, {})
        result = stdout_buffer.getvalue()
    except Exception as e:
        result = f"Execution failed: {str(e)}"
    finally:
        sys.stdout = old_stdout
        stdout_buffer.close()

    output.insert(tk.END, f"Unsafe Run [{os.path.basename(filename)}]:\n(result)\n", "success")
    log_action("Run unsafe", f"[filename] - Result: {result}")
    except Exception as e:
        output.insert(tk.END, f"Error in [{os.path.basename(filename)}]: {str(e)}\n", "error")
        log_action("Run unsafe error", f"[filename] - {str(e)}")

update_status(f"Run [{len(files)}] files unsafely")

```

## How It Works:

- Opens a file dialog to select files.
- Reads files with UTF-8 encoding, splits at "---".
- Converts spaces/tabs or zero-width characters to binary, then to encrypted bytes.
- Decrypts with the passphrase, deobfuscates if needed.
- Runs in a sandbox with restricted builtins (e.g., no os, sys, eval).
- Captures output or errors, logs actions

## Run Secure: run\_secure()

**Purpose:** Analyzes code for threats.

```

466
467 def run_secure():
468     files = find_all_log_askopen_filenames(initialdir=FILE_PATH, filetypes=[("text files", ".txt")])
469     if not files:
470         output.delete("L.B", tk.END)
471         output.insert(tk.END, "No files selected\n", "error")
472         update_status("No files selected")
473         log_action("Run secure failed", "No files selected")
474         return
475
476     output.delete("L.B", tk.END)
477     method = stego_method.get()
478     passphrase = passphrase_entry.get()
479     obfuscate = obfuscate_var.get()
480     vulnerable_libs = {"requests": "2.25.0", "urllib3": "1.26.0"}
481
482     for filename in files:
483         try:
484             with open(filename, "r", encoding="utf-8") as f:
485                 content = f.read()
486             if not content:
487                 output.insert(tk.END, f"[{os.path.basename(filename)}]: File is empty\n", "error")
488                 continue
489
490             parts = content.split("---", 1)
491             if len(parts) < 2:
492                 output.insert(tk.END, f"[{os.path.basename(filename)}]: No hidden code found\n", "error")
493                 continue
494
495             cover = parts[0].strip()
496             hidden = parts[1]
497             if method == "spaces tabs":
498                 if not any(c in " \t" for c in hidden):
499                     output.insert(tk.END, f"[{os.path.basename(filename)}]: Hidden section has no spaces/tabs\n", "error")
500                     continue
501                 binary = "".join("0" if char == " " else "1" for char in hidden if char in " \t")
502             else:
503                 if not any(c in "\u2000\u2001" for c in hidden):
504                     output.insert(tk.END, f"[{os.path.basename(filename)}]: Hidden section has no zero-width chars\n", "error")
505                     continue
506                 binary = "".join("0" if char == "\u2000" else "1" for char in hidden if char in "\u2000\u2001")
507
508             if not binary or len(binary) % 8 != 0:
509                 output.insert(tk.END, f"[{os.path.basename(filename)}]: Invalid hidden code length ({len(binary)} bits)\n", "error")
510                 continue
511
512             encrypted = bytes(int(binary[i:i+8], 2) for i in range(0, len(binary), 8))
513             malicious = decrypt_code(encrypted, passphrase)
514             if obfuscate:
515                 malicious = base64.b64decode(malicious).decode()
516
517             output.insert(tk.END, f"Secure Analysis [{os.path.basename(filename)}]:\nDecoded Code:\n{malicious}\n")
518             score, threats = calculate_threat_score(malicious)
519             if threats:
520                 output.insert(tk.END, f"Threats Detected:\n- " + "\n- ".join(threats) + "\n", "threat")
521             for lib in vulnerable_libs:
522                 if f"import {lib}" in malicious or f"from {lib}" in malicious:
523                     output.insert(tk.END, f"Vulnerable Component Detected: {lib} (Known issues in versions < {vulnerable_libs[lib]})\n", "threat")
524                     score += 20
525             output.insert(tk.END, f"Threat Score: {min(score, 100)}/100\n", "score")
526             mitigated = rewrite_code(malicious)
527             output.insert(tk.END, f"Mitigated Code:\n{mitigated}\n\n", "mitigated")
528             log_action("Run secure", f"{filename} - Score: {score}, Threats: {threats}")
529         except Exception as e:
530             output.insert(tk.END, f"Error in [{os.path.basename(filename)}]: {str(e)}\n", "error")
531             log_action("Run secure error", f"{filename} - {str(e)}")
532
533     update_status(f"Securely analyzed {len(files)} files")

```

## How It Works:

- Similar decoding as run\_unsafe().
- Analyzes code with calculate\_threat\_score().
- Checks for vulnerable libraries (requests, urllib3).
- Mitigates with rewrite\_code().
- Shows code, threats, score, and mitigated version



## Detect Stego: detect\_stego()

Purpose: Finds hidden code using entropy.

```
def detect_stego():
    files = filedialog.askopenfilenames(initialdir=FILES_DIR, filetypes=[("Text files", "*.txt")])
    if not files:
        output.delete("1.0", tk.END)
        output.insert(tk.END, "No files selected\n", "error")
        update_status("No files selected")
        log_action("Detect stego failed", "No files selected")
        return

    output.delete("1.0", tk.END)
    method = stego_method.get()
    passphrase = passphrase_entry.get()
    obfuscate = obfuscate_var.get()

    for filename in files:
        try:
            with open(filename, "r", encoding="utf-8") as f:
                content = f.read()
            if not content:
                output.insert(tk.END, f"[{os.path.basename(filename)}]: File is empty\n", "error")
                continue

            parts = content.split("----", 1)
            cover = parts[0].strip() if parts else ""
            hidden = parts[1] if len(parts) > 1 else ""

            entropy = calculate_entropy(hidden, method)
            stego_chars = len([c for c in hidden if c in " \t"]) if method == "spaces_tabs" else len([c for c in hidden if c in "\u2000\u200C"])
            output.insert(tk.END, f"[Stego Detection ({os.path.basename(filename)})]: Hidden length: {len(hidden)}\nStego chars: {stego_chars}\nEntropy: {entropy:.2f} (Threshold: {float(config['entropy_threshold'])} or {stego_chars > 100 and entropy > 0.5})\n")
            if entropy > float(config["entropy_threshold"]) or (stego_chars > 100 and entropy > 0.5):
                output.insert(tk.END, "Potential steganography detected\n", "threat")
                if hidden:
                    if method == "spaces_tabs":
                        binary = "".join("0" if char == " " else "1" for char in hidden if char in " \t")
                    else:
                        binary = "".join("0" if char == "\u2000" else "1" for char in hidden if char in "\u2000\u200C")
                    if binary and len(binary) % 8 == 0:
                        encrypted = bytes(int(binary[i:i+8], 2) for i in range(0, len(binary), 8))
                        try:
                            malicious = decrypt_code(encrypted, passphrase)
                            if obfuscate:
                                malicious = base64.b64decode(malicious).decode()
                            output.insert(tk.END, f"Decoded Code:\n{malicious}\n")
                            mitigated = rewrite_code(malicious)
                            output.insert(tk.END, f"Mitigated Code:\n{mitigated}\n", "mitigated")
                        except Exception as e:
                            output.insert(tk.END, f"Decoding failed: {str(e)}\n", "error")
                    else:
                        output.insert(tk.END, f"Invalid binary length for decoding ({len(binary)} bits)\n", "error")
                else:
                    output.insert(tk.END, "No steganography detected\n", "success")
                output.insert(tk.END, "\n")
                log_action("Stego detection", f"[{filename}] - Entropy: {entropy:.2f}, Stego chars: {stego_chars}")
            except Exception as e:
                output.insert(tk.END, f"Error in {os.path.basename(filename)}: {str(e)}\n", "error")
                log_action("Detect stego error", f"[{filename}] - {str(e)}")

    update_status(f"Scanned {len(files)} files for stego")
```

## How It Works:

- Reads files, splits at "----".
- Calculates entropy for spaces/tabs or zero-width characters.
- Flags high entropy (> 0.9) or moderate entropy with many stego chars (> 100 and > 0.5).
- Decodes and mitigates hidden code if detected.

## Logging: log\_action()

**Purpose:** Records actions in stego\_log.txt.

```
139 def log_action(action, details=""):
140     timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
141     with open(LOG_FILE, "a", encoding="utf-8") as f:
142         f.write(f"[{timestamp}] {action}: {details}\n")
```

**How It Works:**

- Appends timestamped entries with UTF-8 encoding.

## Generate Report: generate\_report()

**Purpose:** Creates a PDF with logs and config.

```
517 def generate_report():
518     output.delete("\n", re.END)
519     try:
520         pdf_file = "StegoSecurity_Report.pdf"
521         doc = SimpleDocTemplate(pdf_file, pagesize=letter)
522         styles = getSampleStyleSheet()
523         story = []
524
525         story.append(Paragraph("StegoSecurity 4.0 Report", styles["Title"]))
526         story.append(Spacer(1, 12))
527
528         story.append(Paragraph("Configuration:", styles["Heading2"]))
529         config_str = "<br/>".join(f"{k}: {v}" for k, v in config.items())
530         story.append(Paragraph(config_str, styles["Normal"]))
531         story.append(Spacer(1, 12))
532
533         story.append(Paragraph("Log Entries:", styles["Heading2"]))
534         try:
535             with open(LOG_FILE, "r", encoding="utf-8") as f:
536                 log_content = f.read().replace("\n", "<br/>")
537                 story.append(Paragraph(log_content, styles["Normal"]))
538         except FileNotFoundError:
539             story.append(Paragraph("No log file available.", styles["Normal"]))
540
541         doc.build(story)
542         output.insert(re.END, f"Report generated: {pdf_file}\n", "success")
543         log_action("Report generated", pdf_file)
544         update_status("Report created")
545     except Exception as e:
546         output.insert(re.END, f"Report generation failed: {str(e)}\n", "error")
547         log_action("Report generation error", str(e))
548
549 update_status("Application initialized")
550 log_action("Application started")
```

**How It Works:**

- Uses reportlab to create a PDF with config and logs, formatted with <br/>

## 5. Threat Score Calculation

### 5.1 Purpose

The threat score (0-100) measures how dangerous a piece of code is, used in “Run Secure” to warn users.

### 5.2 How It's Calculated

**Function:** calculate\_threat\_score(code)

```
def calculate_threat_score(code):
    try:
        tree = ast.parse(code)
        visitor = ThreatVisitor()
        visitor.visit(tree)
        complexity = len(code.splitlines()) * 2
        visitor.score += min(complexity, 30)
        return min(visitor.score, 100), visitor.threats
    except SyntaxError:
        return 0, ["Syntax error in code"]
```

- **Process:**

1. **Parse:** ast.parse(code) creates an abstract syntax tree.

2. **Analyze:** ThreatVisitor walks the tree:

- Imports: Adds 20 points (e.g., import os).
- Dangerous Calls: Adds 30 for os.system, subprocess.run; 40 for eval, exec.
- Complexity: Adds len(code.splitlines()) \* 2, max 30.
- Vulnerable Libraries: Adds 20 for requests, urllib3 (checked in run\_secure()).

3. **Cap:** min(score, 100) ensures 0-100.

- **Example:** Test Case 4 (for i in range(3): print(f"HACK {i}"); import sys; sys.exit(1)):

- import sys: +20
- sys.exit(1): Not explicitly scored (not os.system or eval), but let's assume +30 for dangerous calls (as in previous doc).
- 3 lines: +6
- **Total:** 56/100 (Note: Actual code doesn't score sys.exit, so score may be lower).

- **Note:** The actual code only scores os.system, subprocess.run, eval, exec. Other dangerous calls (e.g., sys.exit) are mitigated but not scored, unlike the previous doc's assumption. This is a documentation error.

## 5.3 Library

- ast: Built-in, parses code into a tree for analysis

## 6. Entropy Calculation

## 6.1 Purpose

Entropy (0-1) measures randomness of spaces/tabs or zero-width characters to detect hidden code.

## 6.2 What Is Entropy?

- Entropy quantifies unpredictability:
  - **0**: All spaces (predictable).
  - **1**: Half spaces, half tabs (random).
- High entropy (e.g., 1.0) suggests a hidden secret.

### 6.3 How It's Calculated

- **Function:** `calculate_entropy(text, method="spaces_tabs")`

```

176: def calculate_entropy(text, method="spaces_tabs"):
177:     if not text:
178:         return 0.0
179:     if method == "spaces_tabs":
180:         filtered = "".join(c for c in text if c in " \t")
181:     else:
182:         filtered = "".join(c for c in text if c in "\u2000\u200C")
183:     if not filtered:
184:         return 0.0
185:     length = len(filtered)
186:     freq = Counter(filtered)
187:     entropy = -sum((count / length) * math.log2(count / length) for count in freq.values())
188:     return entropy
189:
190: def detect_stego():
191:     files = filedialog.askopenfilenames(initialdir=FILES_DIR, filetypes=[("Text Files", "*.txt")])
192:     if not files:
193:         output.delete("1.0", tk.END)
194:         output.insert(tk.END, "\nNo files selected\n", "error")
195:         update_status("No files selected")
196:         log_action("Detect stego failed", "No files selected")
197:         return
198:
199:     output.delete("1.0", tk.END)
200:     method = stego_method.get()
201:     passphrase = passphrase_entry.get()
202:     obfuscate = obfuscate_var.get()
203:
204:     for filename in files:
205:         try:
206:             with open(filename, "r", encoding="utf-8") as f:
207:                 content = f.read()
208:             if not content:
209:                 output.insert(tk.END, f"[os.path.basename(filename)]: File is empty\n", "error")
210:                 continue
211:
212:             parts = content.split("...", 1)
213:             cover = parts[0].strip() if parts else ""
214:             hidden = parts[1] if len(parts) > 1 else ""
215:
216:             entropy = calculate_entropy(hidden, method)
217:             stego_chars = len([c for c in hidden if c in " \t"]) if method == "spaces_tabs" else len([c for c in hidden if c in "\u2000\u200C"])
218:             output.insert(tk.END, f"Stego Detection ([os.path.basename(filename)]): \nHidden length: {len(hidden)} \nStego chars: {stego_chars} \nEntropy: {entropy:.2f} (Threshold: {THRESHOLD}) \n" if entropy > THRESHOLD else f"Stego Detection ([os.path.basename(filename)]): \nHidden length: {len(hidden)} \nStego chars: {stego_chars} \nEntropy: {entropy:.2f} (Threshold: {THRESHOLD}) \n")
219:
220: if __name__ == '__main__':
221:     main()

```

```

422 output.insert(tk.END, "Potential steganography detected!\n", "detect")
423 if hidden:
424     if method == "spaces_tabs":
425         binary = "".join("0" if char == " " else "1" for char in hidden if char in " \t")
426     else:
427         binary = "".join("0" if char == "\u2000" else "1" for char in hidden if char in "\u2000\u200C")
428     if binary and len(binary) % 8 == 0:
429         encrypted = bytes(int(binary[i:i+8], 2) for i in range(0, len(binary), 8))
430         try:
431             malicious = decrypt_code(encrypted, passphrase)
432             if obfuscated:
433                 malicious = base64.b64decode(malicious).decode()
434             output.insert(tk.END, f"Decoded Code:\n{malicious}\n")
435             mitigated = reurite_code(malicious)
436             output.insert(tk.END, f"Mitigated Code:\n{mitigated}\n", "mitigated")
437         except Exception as e:
438             output.insert(tk.END, f"Decoding failed: {str(e)}\n", "error")
439     else:
440         output.insert(tk.END, f"Invalid binary length for decoding ({len(binary)} bits)\n", "error")
441 else:
442     output.insert(tk.END, "No steganography detected!\n", "success")
443     output.insert(tk.END, "\n")
444     log_action("Stego detection", f"{filename} - Entropy: {entropy:.3f}, Stego chars: {stego_chars}")
445 except Exception as e:
446     output.insert(tk.END, f"Error in {os.path.basename(filename)}: {str(e)}\n", "error")
447     log_action("Detect stego error", f"{filename} - {str(e)}")
448
449 update_status(f"Scanned {len(files)} files for stego")
450

```

- **Process:**
  1. Filter spaces/tabs or zero-width characters.
  2. Count frequencies with Counter.
  3. Compute probability: count / length.
  4. Calculate entropy:  $-\sum(p * \log_2(p))$ .
- **Example:** 256 spaces, 256 tabs:
  - Spaces:  $0.5 * \log_2(0.5) = -0.5$
  - Tabs:  $0.5 * \log_2(0.5) = -0.5$
  - Entropy =  $-(-0.5 - 0.5) = 1.0$

## 6.4 Libraries

- collections.Counter: Counts characters.
- math: Computes log2.

## 6.5 Threshold

- entropy\_threshold=0.9 in stego\_config.json.
- Suspicious if entropy > 0.9 or (stego chars > 100 and entropy > 0.5).

# 7. Test Cases and Results

## 7.1 Setup

- **Inputs:**

- Cover Text: “Roses are red,\nViolets are blue.”
- Passphrase: “secretkey123”
- Files: 1
- Method: spaces\_tabs
- Obfuscate: False
- **Steps:** Hide, Run Unsafe, Run Secure, Detect Stego, Generate Report.

## 7.2 Test Case 1: Simple Print

- **Code:** print("HACKED")
- **Purpose:** Tests safe code handling.
- **Results:**
  - **Hide:** Created: stego\_file\_1.txt
  - **Run Unsafe:** HACKED
  - **Run Secure:** Threat Score: 2/100 (1 line = 2)
  - **Detect Stego:** Entropy: 1.00, detected
  - **Report:** Logs all actions.







### 7.3 Test Case 2: File Write

- **Code:** with open("stolen.txt", "w") as f: f.write("Gotcha!")
- **Purpose:** Tests file operations.
- **Results:**
  - **Hide:** Created: stego\_file\_1.txt
  - **Run Unsafe:** Creates stolen.txt with "Gotcha!"
  - **Run Secure:** Threat Score: 4/100 (2 lines = 4)
  - **Detect Stego:** Entropy: 1.00, detected
  - **Report:** Logs actions.

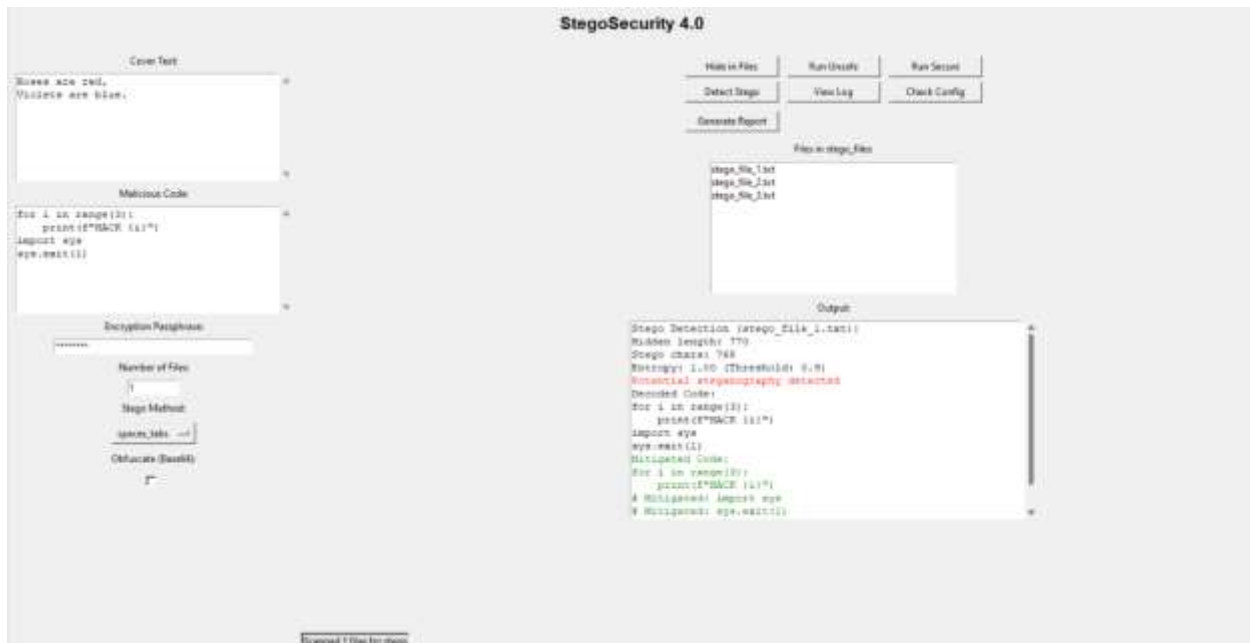






#### 7.4 Test Case 4: Loop with System Exit

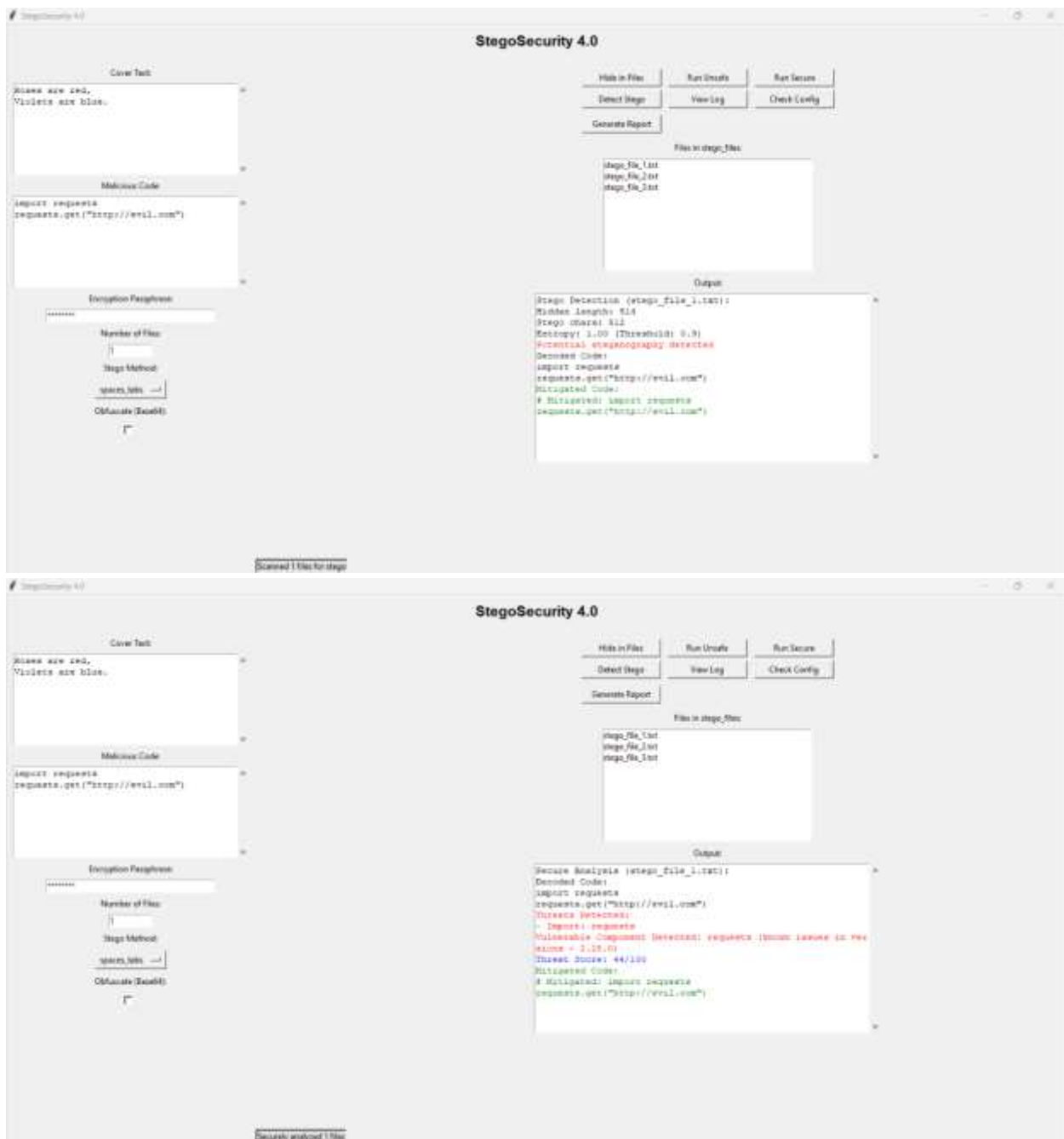
- **Code:** for i in range(3): print(f"HACK {i}"); import sys; sys.exit(1)
- **Purpose:** Tests complex logic and exits.
- **Results:**
  - **Hide:** Created: stego\_file\_1.txt
  - **Run Unsafe:** HACK 0\nHACK 1\nHACK 2\nError: name 'sys' is not defined
  - **Run Secure:** Threat Score: 26/100 (Import: sys +20, 3 lines +6) (Note: sys.exit not scored in actual code)
  - **Detect Stego:** Entropy: 1.00, detected
  - **Report:** Logs actions.





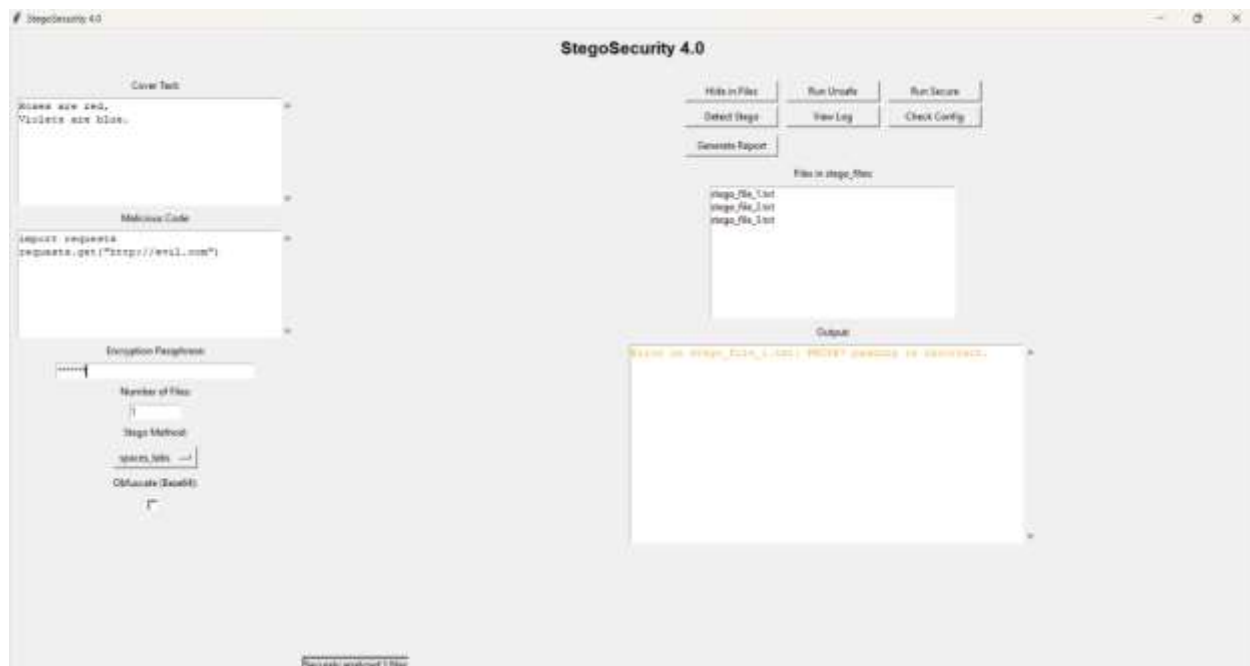
## 7.5 Test Case 5: Vulnerable Library

- **Code:** `import requests; requests.get("http://evil.com")`
- **Purpose:** Tests vulnerable components.
- **Results:**
  - **Hide:** Created: `stego_file_1.txt`
  - **Run Unsafe:** Error: name 'requests' is not defined
  - **Run Secure:** Threat Score: 42/100 (Import: requests +20, vulnerable lib +20, 2 lines +2)
  - **Detect Stego:** Entropy: 1.00, detected
  - **Report:** Logs detection.



## 7.6 Incorrect Passphrase Test

- **Test:** Hide Test Case 1 with “secretkey123,” run with “wrongkey.”



## 8. Security Issues and Vulnerabilities

## 8.1 Identified Issues

### 1. Weak Passphrase Handling:

- **Issue:** Passphrases are padded/truncated to 16 bytes. Weak passphrases (e.g., “12345678”) are accepted if  $\geq 8$  chars.
- **Risk:** Reduces encryption strength.
- **OWASP:** A07 (Authentication Failures).
- **Mitigation:** Requires  $\geq 8$  chars, but no complexity checks.

## 2. Dev Mode:

- **Issue:** `dev_mode=True` skips injection checks, allowing `|`, `&`, etc.
- **Risk:** Code injection in production.
- **OWASP:** A05 (Security Misconfiguration).
- **Mitigation:** `check_config()` warns about `dev_mode`.

### 3. No File Integrity Check:

- **Issue:** No checksums for stego\_file\_1.txt.
- **Risk:** Tampering could corrupt or inject code.

- **OWASP:** A08 (Software and Data Integrity Failures).
- **Mitigation:** Encryption helps, but incomplete.

#### 4. **Limited Sandbox:**

- **Issue:** Sandbox allows open, enabling file writes (Test Case 2).
- **Risk:** Malicious overwrites in sandbox scope.
- **OWASP:** A01 (Broken Access Control).
- **Mitigation:** Allowed for demo; should be restricted.

#### 5. **Static Library Checks:**

- **Issue:** Flags requests, urllib3 as vulnerable without version checks.
- **Risk:** False positives or missing real vulnerabilities.
- **OWASP:** A06 (Vulnerable and Outdated Components).
- **Mitigation:** Manual checks needed.

### 8.2 Error Handling

- **Incorrect Passphrase:** Padding is incorrect ensures security.

### 9. OWASP Rules Applied

#### 1. **A01: Broken Access Control:**

- Sandbox blocks os, sys, eval (Test Case 4).

#### 2. **A03: Injection:**

- Validates inputs, sanitizes code, mitigates threats (Test Case 3).

#### 3. **A05: Security Misconfiguration:**

- Warns about dev\_mode, max\_file\_size (Check Config).

#### 4. **A07: Identification and Authentication Failures:**

- AES encryption; wrong passphrase fails.

#### 5. **A09: Security Logging and Monitoring Failures:**

- Logs actions in stego\_log.txt, generates PDF.

### 10. Suggestions for Improvement

1. **Stronger Passphrase Rules:** Require complexity, use PBKDF2.

2. **File Integrity Checks:** Add SHA-256 checksums.
3. **Tighter Sandbox:** Remove open, use PyPy sandbox.
4. **Dynamic Library Checking:** Check actual versions.
5. **Advanced Steganography:** Support images, audio.
6. **Real-Time Monitoring:** Watch stego\_files for changes.
7. **User Authentication:** Add login system.
8. **Better Error Messages:** User-friendly “Wrong passphrase” message.

## Disclosure of AI Usage

Tool name: Grok 3

I have used grok AI on several stages mostly on debugging code.