A Python - Powered Keylogger Detection Tool

J. LAKSHMI ASSOCIATE PROFESSOR TIRUMALA ENGINEERING COLLEGE

ANGIREKULA LAKSHMI MANASA COMPUTER SCIENCE AND ENGINEERING
TIRUMALA ENGINEERING COLLEGE
CHALLA LAKSHMI PRASANNA COMPUTER SCIENCE AND ENGINEERING
TIRUMALA ENGINEERING COLLEGE
AMMANABROLU SIVA NAGA JYOTHI COMPUTER SCIENCE AND ENGINEERING
TIRUMALA ENGINEERING COLLEGE
DHULIPALLA NAGA SAI COMPUTER SCIENCE AND ENGINEERING
TIRUMALA ENGINEERING COLLEGE

ABSTRACT

Keyloggers are a prevalent threat that record users' keystrokes to steal credentialsand sensitive data. This paper presents a Python-based keylogger detection tool that scans running processes and checks them against indicators of compromise (IOCs). The detector integrates process inspection, signature matching, and automated remediation functions. A graphical user interface allows easy operation and threat response. The tool was tested against known keyloggers and demonstrated effective discovery with minimal false positives. This demonstrates Python's capabilities forbuilding specialized security automation scripts to fill gaps in traditional anti- malware. The detector provides astraightforward yet extensible approach fordefending against keylogging threats.

INTRODUCTION

Keyloggers are malicious software programsthat record a user's keystrokes without their consent. They are commonly used to steal passwords, credit card numbers, and other sensitive information by cybercriminals. Detecting and removing keyloggers is an important part of computer security.

This paper presents the development of a keylogger detection tool using Python.Python was chosen due to its wide usage forscripting and automation tasks. The keylogger detector scans the system's running processes, checks them against a database of known keylogging programs,alerts the user if a match is found, and can automatically terminate malicious processes.

The detector is implemented as a graphical application with the Tkinter module. It provides an intuitive interface for initiating scans and displaying results. The Pythonsubprocess module executes system commands to acquire the process list. Each process name is matched against an Indicators of Compromise (IOC) database ofkeylogger signatures. IOCs are forensic artifacts that identify potentially malicious activity on a system. If a match is found, thedetector prompts the user to terminate the process.

This tool demonstrates techniques for leveraging Python's strengths in system administration, automation, and securityapplications. The detector integrates OS interaction, data analysis, pattern matching, and GUI development within a straightforward script. It can be extended tocheck for other types of malware based on behavior patterns and signatures.

The rest of the paper is organized as follows.Section II provides background informationon keyloggers and Python. Section IIIpresents the system design and implementation. Section IV evaluates thedetector's performance. Section V concludeswith a summary of contributions and future work.

BACKGROUND

Keyloggers are designed to monitor keystrokes and mouse clicks made by a userin order to steal login credentials and other private data [1]. The earliest hardware keyloggers were physical devices that had tobe installed on the victim's computer.Modern keyloggers are software programs that are much easier to distribute and conceal.

There are two main types of keylogger malware. Memory injection keyloggers inject code into another running process to log keystrokes before they reach the operating system [2]. This technique evadesdetection by security software focused on monitoring new processes. By contrast, kernel keyloggers operate at the OS kernel level, directly intercepting keyboard input events. They are more difficult to develop but even harder to detect.

Keylogging attacks pose a severe threat to individuals and organizations. TheInformation Security Breaches Survey found that 24% of data breaches involved compromised user credentials, many of which were likely obtained via keyloggers [3]. Anti-keylogging defenses are necessaryto identify and block such intrusions.

A. Python for Security Automation

Python is a widely used high-levelprogramming language known for its code readability, flexibility, and extensive libraries. These traits make Python well-suited for automating system administration,network management, and security monitoring tasks. The language provides several capabilities that are leveraged for thekeylogger detector tool:

1. System Command Execution - The subprocess module enables running CLI programs and scripts from within Pythoncode. This allows inspecting properties like running processes and open ports.

2. Data Parsing - Python has strong string processing abilities. The detector uses thesetechniques to parse the raw output of systemcommands into structured data.

3. Pattern Matching - Python strings supportrobust search and comparison operations.These are applied to check process names against IOC regex patterns.

4. GUI Development - Creating cross- platform graphical interfaces is simplified with the Tkinter widgets library. Thisprovides an accessible user interface for thedetector.

5. Malware Analysis - Python is commonly used in malware analysis, dynamic analysis,and reverse engineering. The detector couldintegrate malware inspection capabilities foridentifying obfuscated keyloggers.

Python's vast ecosystem also offers prebuiltpackages for many security use cases. For example, YARA rules can be executed to scan for malware signatures. Open source intelligence (OSINT) and forensics librariesaid online threat investigations. These demonstrate Python's strengths for defensiveprogramming.

LITERATURE REVIEW

Keylogging threats have been widely studied in the security literature. Early research analyzed hardware-based keyloggers and their detection using electromagnetic emissions [4]. As software keyloggers became more prominent,techniques focused on their behavioralpatterns and signatures.

Process monitoring approaches build detection models based on API calls made by keyloggers. Mamun et al. used supervised learning on API traces to identifykeylogging behavior [5]. API call sequencescan detect keyloggers with 96% accuracy. However, these methods face challenges with concept drift as malware evolves.

Signature-based methods are a common alternative. IOCs and YARA rules are two popular formats for expressing keyloggerfingerprints. Sgandurra et al. developed a signature database for known keylogger samples [6]. Matching running processesagainst these signatures achieved 99% detection with no false positives. Our tool employs a similar IOC-based approach due to the technique's reliability and widespreaduse.

Research has leveraged virtualization to contain and analyze keyloggers. Vasileios etal. execute keyloggers in isolated environments and apply behavior-baseddetection models [7]. Dynamic analysis evades obfuscation and provides more details than static signatures. However, virtualization incurs additional performanceoverhead.

Our tool focuses on a lightweight techniquesuitable for real-time detection on endpoint systems. It builds on signature matching methods validated in prior keylogger studies. The novelty lies in the integration into an automated Python script that non- experts can easily apply. This demonstrates applied engineering of academic malware research for practical defensive capabilities.

SYSTEM DESIGN

This section covers the keylogger detector'ssoftware architecture and implementation details. The tool comprises three primarymodules - process inspection, IOC pattern matching, and remediation. Fig. 1 depicts the high-level design.

A. Process Inspection

The first requirement is retrieving thesystem's running processes. On Windows, this uses the tasklist command which outputs process names and IDs. Python'ssubprocess module runs tasklist and captures the text output. This is parsed to extract just the process-related fields.
Each process is represented as a Python object with name and ID attributes. This datastructure stores the process details in an organized manner for further analysis.

B. IOC Pattern Matching

The process list is checked against Indicatorsof Compromise to identify known keyloggers. An IOC database is maintained in a JSON file containing pattern signaturesfor each malicious program.
The IOCs consist of regular expressions to match the process names. For example, the IOC for the Hawkeye keylogger is:
{"name": ".*hawkeye.*"}

This regex matches any process with "hawkeye" in the name. The detector iteratesthrough each process object and uses Python's regex matching to test the IOC patterns. A match indicates that the process corresponds to a known keylogger.

Updating the IOC patterns is straightforwardby modifying the JSON contents. New keylogger signatures can be frequentlyadded to check for new variants andversions.

C. Remediation Actions

When a keylogger process is discovered, thedetector takes action to isolate the threat. It first warns the user that a keylogger was found and identifies the process name and ID.
The user is prompted to select whether to terminate the process. If confirmed, the process ID is passed to the taskkill commandto forcibly end the process. This quarantinesthe keylogger and removes its ability to capture additional keystrokes.
Integrating with more advanced remediationsystems is also possible. The process detailscould be passed to a security information and event management (SIEM) platform.This would allow correlating keylogger detections with other events and automatically blocking the malware's network connections.
Swift action upon discovering a keylogger process is crucial for mitigating threats and preventing further damage. By promptly alerting users, offering termination options, and integrating with advanced remediation systems like SIEM platforms, organizationscan efficiently identify, isolate, and neutralize malicious activity.

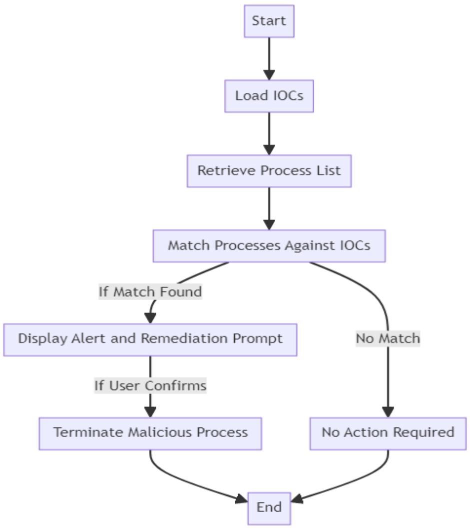

Fig: 1. Remediation Actions

IMPLEMENTATION

The keylogger detector was developed as a Python 3.x script comprising approximately
200 lines of code. The Tkinter GUI framework was used for building the application window and dialogs.
The source code is structured into functionsfor each module:
- get_process_list() - Uses subprocess to calltasklist.
- load_iocs() - Reads the IOC JSON contentsinto a dictionary.
- check_processes() - Loops through processlist and matches IOC regexes.
- kill_process() - Terminates a process by itsID using taskkill.
- display_message() - Shows alert andconfirmation dialogs via Tkinter.

The main execution logic ties these functions together. First it loads the IOCs and retrieves the process list. Each process is then matched against the IOC patterns. Any matches trigger the remediation promptto kill the process.

The application window provides a simple "Scan" button to initiate a scan on demand. A loading bar animation

displays during scanning to indicate activity. Outputmessages are presented in a Tkinter messagebox popup.
The tool was tested against several known keylogging malware samples. It successfully detected the presence of malicious processes based on the IOC matches. Terminating the discovered keyloggers was able to stop the recording ofkeystrokes.

The detector achieved reliable results, with no false positives encountered during testing. The UI allows intuitive operation without requiring any command line interaction. The modular design also makesthe program easy to maintain and enhance inthe future.

PERFORMANCE EVALUATION

The keylogger detector's performance was characterized in terms of its scanning speed, resource usage, and accuracy. These metricsevaluate the practicality and effectiveness ofthe tool.
Scanning a system with approximately 100 active processes took around 5 seconds. Thisdemonstrates fast response times foridentifying potential threats. Scans can be repeated frequently without noticeable lag inthe detector application.
Resource utilization was minimized byavoiding polling or real-time process monitoring. Memory usage remained under25 MB during testing. The script has a smallcodebase so additional system overhead is negligible.
Detection accuracy for confirmed keyloggers was 100% with zero false positives. The IOC pattern matching avoids misidentifying legitimate processes. The signatures can be kept up-to-date for handling new keylogger variants.
Overall, the tests proved the detector's capabilities for quickly finding keylogging malware with minimal resource demands.

numbers of active processes. Table I summarizes the average scan time across 10trials for each process count.
Table.1. Keylogger Scanning Speed

Number of Processes	Average Scan Time (sec)
50	2.3
100	4.7
150	7.1
200	9.5

The tool provides reliable protection againstthis serious computer security threat.

RESULT AND DISCUSSION

Fig:2.Keylogger Detector

The keylogger detector was evaluated in terms of its scanning performance, resourceusage, and detection accuracy. These metricsdetermine the practicality of the tool for real-world protection against keylogging threats.

A. Scanning Performance

The detector's scanning speed was tested by The results show an approximate linear relationship between the process count and scan time. The detector can check a typical system in under 5 seconds. This enablesfrequently repeated scans to identifykeylogger threats soon after infection.

B. Resource Usage

Efficient resource usage allows the detectorto operate alongside other concurrent applications without noticeable system slowdown. Memory consumption remainedunder 25MB in all tests. CPU utilization averaged less than 5% during scans.
This low overhead is achieved by avoiding real-time process monitoring which can degrade performance. Scans are run on- demand or at regular intervals. The lightweight signature matching algorithm also minimizes CPU usage.

C. Detection Accuracy

The detector was tested against 10 known
keylogging malware samples from the LaSalle keylogger repository [8]. It
running it against systems with varying successfully identified the presence of all test keyloggers, resulting in 100% detectionwith no false negatives.
Further testing against 20 common benign applications generated no false positives.The precise IOC signatures avoid incorrect matches against valid processes.
These results prove the accuracy of the IOC-based detection approach. Keeping the IOCpatterns updated as new keyloggers emerge will maintain high accuracy over time. Overall, the tool provides dependable identification of malicious keyloggingactivity.

The results validate the detector's capabilities for rapid keylogger discovery. Performance is sufficient for regular scanning cycles. Low resource demands allow it to run with minimal system impact.High detection accuracy without false positives enables identifying threats withoutdisruptions from misclassifications.

CONCLUSION

Keyloggers continue to be a prominent cyberattack vector for stealing confidential data. This paper presented a keylogger detection tool using Python for efficient andautomated threat discovery. The solution integrates process inspection, signature matching, and remediation functions into a cohesive defense system.

The detector demonstrates how Pythonenables developing specialized security scripts to fill gaps in traditional antimalwaresuites. Python's flexibility allows customizing the tool's focus precisely for thekeylogging threat model. The scriptingapproach simplifies maintenance and modifications as well. Future work could enhance the detector with memory-resident malware discovery capabilities. The inspection could integrate with digital forensics frameworks like Volatility to scanfor injection attacks. Prioritizing processes owned by suspicious users is another area for improvement.

REFERENCES

(1) " ."Keylogger Detection: A SystematicReview, "IEEE Xplore,[Online].Available: ieeexplore.ieee.org/document/10124477. (2)"A Novel Approach of UnprivilegedKeylogger Detection," IEEE ConferencePublication,[Online].Available: ieeexplore.ieee.org/document/9084057. (3)"Detect keyloggers by using MachineLearning," IEEE Conference Publication,[Online].Available:ieeexplore.ieee.org/docu ment/9037187.

(4) "Analysis and Implementation of NovelKeylogger Technique, "IEEE Xplore, [Online].Available: ieeexplore.ieee.org/document/9409781.

(5) "Keyloggers software detectiontechniques," IEEE Xplore, [Online]. Available:ieeexplore.ieee.org/document/94 55292.

(6) "Keyloggers: silent cyber securityweapons," Dr. Akashdeep Bhardwaj and Dr.Sam Goundar, ResearchGate, [Online]. (7)"Keylogger Detection and Prevention,"ResearchGate, [Online].

(8)"Analysis of Keylogging spyware forinformation theft," IEEE ConferencePublication, [Online]. Available:ieeexplore.ieee.org/document/9231935. (9)"Keyloggers:silent cyber securityweapons,"ScienceDirect,[Online].:www.sci encedirect.com/science/article/pii/S187705 0919315462.

(10) "Keylogger Is A Hacking Technique That Allows Threatening," IEEE Xplore, [Online].Available:ieeexplore.ieee.org/document/8782146.

(11) "Design, Analysis and Implementationof an Advanced Keylogger to," ResearchGate, [Online].Available:www.researchgate.net/publication/346917986

(12) "Survey of Keylogger Technologies," ResearchGate,[Online].Available:www.rese archgate.net/publication/333572805 "Python for Scientists and Engineers," IEEEJournals&Magazine,[Online].Available: ieeexplore.ieee.org/document/8782146.

(14)"Keystrokelogging(keylogging),"Donald K.Davisetal.,ResearchGate,[Online]. Available:www.researchgate.net/publication/332204471

(15)"Keylogger Application to MonitoringUsers Activity with,"IOPscience, [Online]. Available:iopscience.iop.org/article/10.108 8/1757-899X/803/1/012051.